

Draft Study Material



JUNIOR SOFTWARE DEVELOPER

(Job Role)

(Qualification Pack: Ref. Id. SSC/Q0508)

Sector: Information Technology-Information Technology Enable Services (IT-ITeS)

(Grade XI)



PSS CENTRAL INSTITUTE OF VOCATIONAL EDUCATION

(a constituent unit of NCERT, under Ministry of Education, Government of India)

Shyamla Hills, Bhopal - 462 002, M.P., India

© PSS Central Institute of Vocational Education, Bhopal 2024

No part of this publication may be reproduced, stored in a retrieval system or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise without the prior permission of the publisher.

PSSCIVE Draft Study Material © Not to be Published

Preface

Vocational Education is a dynamic and evolving field, and ensuring that every student has access to quality learning materials is of paramount importance. The journey of the PSS Central Institute of Vocational Education (PSSCIVE) toward producing comprehensive and inclusive study material is rigorous and time-consuming, requiring thorough research, expert consultation, and publication by the National Council of Educational Research and Training (NCERT). However, the absence of finalized study material should not impede the educational progress of our students. In response to this necessity, we present the draft study material, a provisional yet comprehensive guide, designed to bridge the gap between teaching and learning, until the official version of the study material is made available by the NCERT. The draft study material provides a structured and accessible set of materials for teachers and students to utilize in the interim period. The content is aligned with the prescribed curriculum to ensure that students remain on track with their learning objectives.

The contents of the modules are curated to provide continuity in education and maintain the momentum of teaching-learning in vocational education. It encompasses essential concepts and skills aligned with the curriculum and educational standards. We extend our gratitude to the academicians, vocational educators, subject matter experts, industry experts, academic consultants, and all other people who contributed their expertise and insights to the creation of the draft study material.

Teachers are encouraged to use the draft modules of the study material as a guide and supplement their teaching with additional resources and activities that cater to their students' unique learning styles and needs. Collaboration and feedback are vital; therefore, we welcome suggestions for improvement, especially by the teachers, in improving upon the content of the study material.

This material is copyrighted and should not be printed without the permission of the NCERT-PSSCIVE.

Deepak Paliwal
(Joint Director)
PSSCIVE, Bhopal

Date: 06 September, 2024

STUDY MATERIAL DEVELOPMENT COMMITTEE

Members

Deepak D. Shudhalwar, Professor (CSE), Head, Department of Engineering and Technology, PSSCIVE, NCERT, Bhopal, Madhya Pradesh

Ganesh Kumar Dixit, Assistant Professor in IT-ITeS (Contractual), Department of Engineering and Technology, PSSCIVE, NCERT, Bhopal

Prakash Khanale, Professor and Head, Department of Computer Science, DSM College, Parbhani, Maharashtra

Rizwan Alam, Assistant Professor in IT-ITeS (Contractual), Department of Engineering and Technology, PSSCIVE, NCERT, Bhopal

Member Coordinator

Deepak D. Shudhalwar, Professor (CSE), Head, Department of Engineering and Technology, PSSCIVE, NCERT, Bhopal, Madhya Pradesh

Table of Contents

S. No.	Title	Page No.	
1	Module 1: Software Construction Essentials	1	
	Module Overview	1	
	Learning Outcomes	1	
	Module Structure	1	
	Session 1: Computer System Architecture	2	
	Check Your Progress	13	
	Session 2: Data Representation in Computer	15	
	Check Your Progress	31	
	Session 3: Basic Concepts of Mathematics and Statistics	33	
	Check Your Progress	47	
	Session 4: Problem Solving Methods	48	
	Check Your Progress	65	
	Session 5: Programming Language Concepts	66	
	Check Your Progress	79	
2	Module 2: Operating System and Computer Network	80	
	Module Overview	80	
	Learning Outcomes	80	
	Module Structure	80	
	Session 1: Operating System	81	
	Check Your Progress	100	
	Session 2: Computer Networks	102	
	Check Your Progress	117	
	Session 3: Transmission Media and Network Protocols	119	
	Check Your Progress	129	
	Session 4: Network Security	131	
	Check Your Progress	146	
	3	Module 3: Python Programming	147
		Module Overview	147
Learning Outcomes		148	
Module Structure		148	
Session 1: Python Basics		148	
Check Your Progress		173	
Session 2: Control Structures		176	
Check Your Progress		193	
Session 3: Functions		195	
Check Your Progress		213	

S. No.	Title	Page No.
	Session 4: Strings	216
	Check Your Progress	228
	Session 5: Lists	230
	Check Your Progress	246
	Session 6: Tuples and Dictionaries	246
	Check Your Progress	262
4	Module 4: Data Structure	266
	Module Overview	266
	Learning Outcomes	266
	Module Structure	267
	Session 1: Introduction to Data Structure	267
	Check Your Progress	272
	Session 2: Linear Data Structures	273
	Check Your Progress	292
	Session 3: Non-Linear Data Structures	294
	Check Your Progress	303
5	Answer Key	304

Module 1**Software Construction
Essentials****Module Overview**

In the software development process, it is essential to have the strong foundation of computer science with computational thinking. These fundamentals are referred to the software construction essentials. It will help you to become a software n effective.

This unit deals with fundamentals of computer, which includes computer architecture, internal parts and peripherals, input and output devices, primary and secondary storage devices. This is important to understand the concept of data and its representation in memory. Computer process the instructions in machine code which is represented in the form of binary code 0 and 1. In this unit, you will also understand the data encoding systems, number system and conversions from one number system to another.

Discrete mathematics is a branch of mathematics in which it is possible to perform various mathematical operations on the discrete data. These operations are very much needed when we write programs to perform a specific function or a task. So, the concepts of discrete mathematics such as sets, relations, functions, mathematical logic, group theory, graph theory and statistical methods are covered in this unit.

Software are developed to solve the complex problem and automate the tasks. In real life, many problems occur. It is required to understand the problem first and, take the necessary input into consideration and then take the actions to arrive at the solution. A software developer professional should possess this problem-solving skill. To solve a given problem require we require a set of logical steps. Algorithm and flowchart are the building blocks of programs, which helps to code the program in a programming language. Programming language is a systematic notation that is used to describe the computational process. The programming language paradigm is explained in this Module.

Learning Outcomes

After completing this module, you will be able to:

- Describe the computer system architecture.
- Describe data representation in computer.
- Describe the basic concepts of Mathematics and Statistics.
- Solve the problem using problem solving method.
- Describe programming language concepts.

Module Structure

Session 1: Computer System Architecture

Session 2: Data Representation in Computer

Session 3: Basic Concepts of Mathematics and Statistics

Session 4: Problem Solving Methods

Session 5: Programming Language Concepts

Session 1: Computer System Architecture

Aviral, class 10th student was given a mathematical problem to multiply an eight-digit fractional number with another eight-digit fractional number and divide the result by 67888.9. He took the whole hour to finish the job. When he was asked to do the same using a computer and he finished the same in fraction of seconds. Computer is a powerful electronic device with very high computing capacity. Figure 1.1 illustrate the power of computing.

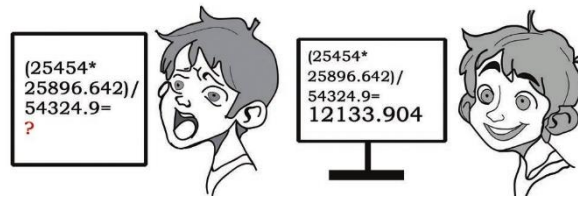


Fig. 1.1: Power of computing

In this chapter you will understand the computer as a system, its various functional units, different types of computers, importance of computers in software development. You will also understand the working of various parts of computer and various input, output and peripheral devices.

Computer

The word computer is derived from the Latin word '*compute*', means to calculate. A computer is an electronic computational device that can perform sequence of arithmetic or logical operations based on computer programs. Computer programs are set of instructions that perform wide range of tasks and calculations at very high speed and accuracy.

All computers, whether they are the smartphone in your hand or powerful servers, operate on the same five basic operational stages as shown in Figure 1.2. These are input, processing, storage, output and communication. Each component of a computer performs one of these functions, but they all work together to work as computer.

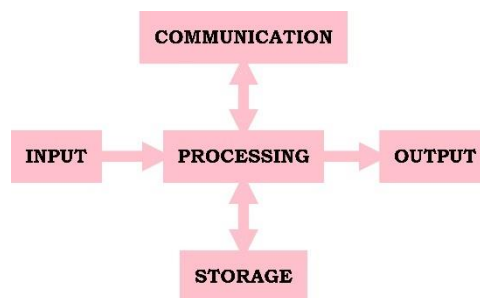


Fig.1.2 The stages of computer processing cycle

1. Input Stage

In the input stage, the data or instructions are entered into the computer through input devices. Data can be in the form of numbers, words, picture, audio, video. The input devices such as keyboard, mouse, touchscreen and microphones are used to enter data in the computer. Keyboard is used to enter numbers and characters. Mouse is used as pointing device. Magnetic Ink Character Readers (MICR) and Optical Character Readers (OCR) are also the input devices. The data entered by these devices is converted into computer understandable binary code of 1s and 0s.

2. Processing Unit

The central processing unit (CPU) process the data and instructions received from input device. The CPU has three units – Memory Unit, Arithmetic Logic Unit and Control Unit as shown in Figure 1.3. Memory unit is linked with CPU. Input devices and output devices are also connected to the processing unit.

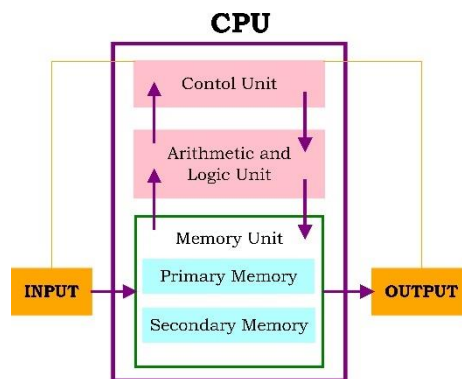


Fig.1.3 Block diagram of computer system

a) Memory Unit – Registers in the memory unit hold the data temporarily until CPU processes it. The memory unit uses a set of pre-programmed instructions to further transmit this data to other parts of the CPU.

b) Arithmetic and Logic Unit (ALU) – ALU performs arithmetic and logical operations. It performs basic mathematical calculations like addition, subtraction, multiplication, division and logical operations such as comparison of data.

c) Control Unit – Control unit coordinates tasks between all components of a computer system. It collects data from input unit and sends it to processing unit. It also sends the processed data to output unit.

3. Output

Output is generated after processing the data as per the instructions. The output generated is sent to the output unit such as monitor. There are many other types of output devices, such as printers, plotters, speakers and projectors.

Activity 1

Practical Activity 1.1. Identify the various parts of computer system

Materials Required

Computer System and peripherals

Procedure

Observe the system computer system along with its peripherals as shown in Figure 1.4. Identify and name the parts and peripherals of the computer system as Monitor, Keyboard, CPU, Mouse, Speakers.



Fig. 1.4. Identify the parts of computer system and peripherals

The role of computer in software development

Any computing device such as computer, laptop, tablet or smartphone, all work as computer. It is possible to use all these devices for software development. The various types of software applications are being used by us in our daily lives. Computer is the basic hardware used for the development and dissemination of software applications. This software is also in the form of web applications which we use in our daily life. It includes, Amazon, Flipkart and other similar online shopping web applications, online banking applications, online reservation systems for booking air and train tickets, hospital management systems, hotel management systems and many more.

Types of computer systems

A computer is a general-purpose computing device which can be programmed to carry out a finite set of arithmetic or logical operations. There are various types of computers. In earlier days computers were classified as Supercomputer, Mainframe, Minicomputer and Microcomputer based on its processing capability and hardware used. Currently we have various types of computers such as *desktop*, *laptop*, *tablet*, *iPad* and *smartphone*, as shown in Figure 1.5.



Fig. 1.5 Different types of computer

COMPUTER HARDWARE

The recent computer hardware technology has made too much advancements in technology. As a result the computers are becoming smaller and more powerful in terms of processing. Figure 1.6 shows the main internal hardware components of computer system that makes the computer more powerful.

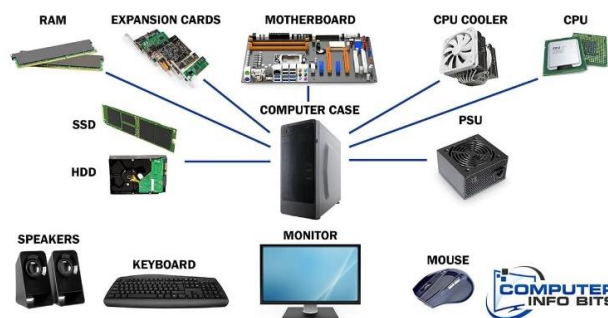


Fig. 1.6: Internal components of computer system

PROCESSING DEVICES

The data received from an input device is processed to generate some output. The motherboard with processor and RAM chips mainly form the processing devices of the computer system. It depends upon your software development requirement which processing device is best for your work.

Motherboard

Motherboard is the main circuit board of the computer system fitted inside the cabinet. All other important components such as processor, memory and hard disk drive are connected with the motherboard. Figure 1.7 shows a typical motherboard of a computer system.

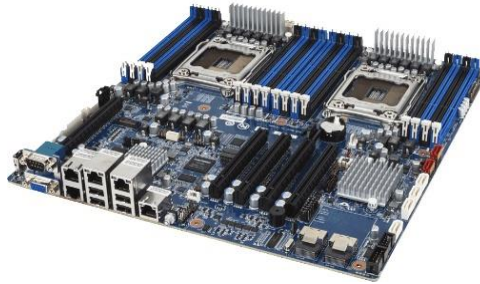


Fig. 1.7: Motherboard

Processor or Central Processing Unit (CPU)

The processor or CPU is the main processing unit mounted on the motherboard. It is an electronic chip of about one-inch square as shown in Figure 1.8. The data and instructions are fetched from the main memory, decoded in machine code and the executed by the processor. A modern processor may contain billions of transistors, that performs computing and control all the other components.



Fig. 1.8: Processor or CPU

The CPU frequency is measured in Gigahertz and it is known as processor's speed. There are many factors to decide the speed of CPU such as generation, clock speed, frequency, cores. The CPU of higher frequency and more cores are more powerful. The best suitable CPU can be selected depending upon the requirement of software development work.

Graphics Processing Unit (GPU)

The GPU is also referred to as the display adapter, graphics card, video adapter, video card, video controller or even gaming card. It is a specialised graphics processing unit. The modern computers come equipped with onboard GPU as shown in Figure 1.9, that is capable of performing the actions needed by the average computer user. However, for graphically intensive tasks, such as graphic design or gaming, a dedicated advanced GPU is required.



Fig. 1.9: A GPU on a PCB board

Main Memory

Main memory is also called primary memory or internal memory. It is directly accessed by the CPU. It temporarily holds the data and instructions before and after execution by the processor. The primary memory is generally made up of semiconductor device. It consists of memory

locations with physical addresses to store and access the contents. RAM and ROM are the two types of primary memory.

(i) Random-access memory (RAM) is volatile type of memory, where stored information is lost if power is shut off. It comes in the form of chip as shown in Figure 1.10. There are various families of RAM, of which DDR5 is the latest that are supported by the modern motherboards. Motherboards are built to support only one type of memory, so you can't mix and match SDRAM, DDR, DDR2, DDR3, DDR4, or DDR5 memory on the same motherboard.

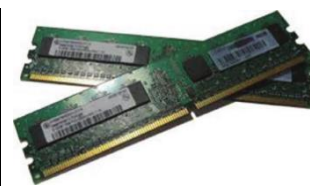


Fig. 1.10: Random-access memory (RAM)

(ii) Read Only Memory (ROM) is non-volatile memory, which means the information stored in it is not lost even if computer power is switched off. It consists of instructions stored during manufacture such as BIOS (Basic Input Output System) or boot program used during the booting process of computer. (Figure 1.11) ROM are further classified as PROM (Programmable Read Only Memory), EPROM (Erasable Programmable Read Only Memory), EEPROM (Electrically Erasable Programmable Read Only Memory).



Fig. 1.11: Read Only Memory (ROM)

(iii) Cache Memory – It is a small-sized type of volatile computer memory that provides high-speed data access to a processor and stores frequently used computer programs, applications and data. It is supplementary memory system that temporarily stores frequently used instructions and data for quick processing by CPU. (Figure 1.12) It is the fastest memory in a computer, and is typically integrated onto the motherboard and directly embedded in the processor or main random access memory (RAM).



Fig. 1.12: Cache Memory

Activity 2

Practical Activity 1.2. Identify the internal and external components of computer system

Materials Required

Paper, pen, pencil, pictures of processor, memory, printer, monitor, keyboard, mouse

Procedure

Step 1. Observe the Figure 1.13, consisting of the main components of a computer system.



Fig. 1.13 Components of a computer system

Step 2. Identify and name each component from the Figure 1.13.

Secondary Memory

It is used to store data permanently for long period. These are generally used to back-up data. The most common secondary memory is the internal hard disk drive that is fixed inside the

computer. It is connected with the motherboard and receives the power from internal power supply of computer. The storage capacity of modern hard disk ranges from 1 TB to 4 TB (terabyte). Some other storage devices used as secondary memory will be discussed in the later section of this chapter.

Activity 3

Practical Activity 1.3. Identify the memory unit and its size in different computing device

Materials Required

Personal Computer System, Laptop, Tablet

Procedure

Step 1. Start the Personal Computer and observe the RAM used in the computer.

Step 2. Start the Laptop, and observe the RAM used in the Laptop computer.

Step 3. Start the Tablet, and observe RAM used in the Tablet.

Record your observations in the following table.

Computing device name	Size of memory used
PC	4 TB
Laptop	8 GB
Tablet	3 GB

Peripheral Devices

Peripheral devices are additionally connected to the computer system from outside. The various input, output and some storage devices are peripheral devices.

INPUT DEVICES

Input devices are essential for any computer to enter data and interact with the computer. Keyboard and mouse are the basic input devices that comes along with the computer. There are variety of input devices used for the various purpose such as, Touch-pad, Trackball, Bar-code reader, Optical Mark Reader (OMR), Optical Character Reader (OCR) and Magnetic Ink Character Recognition (MICR), Scanner and Microphone.

Examples of input devices

Keyboard – The keyboard is basic input device for communicating with a computer system. It is designed to enter text, number and special symbols by striking on the particular key on the keyboard. Modern keyboard use USB port or wireless dongle to connect to the computer externally as shown in Figure 1.14.



Fig. 1.14: Keyboard

Mouse – It is a pointing device that allows to move the cursor on the computer screen, as well as to point and click to select the programs and items. Its main task is to detect the movement when moved on a flat surface and input the information by *left-clicking*, *double clicking*, *right-clicking*, *dragging*, *scrolling* as shown in Figure 1.15.

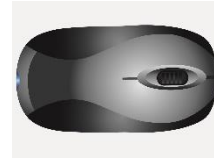


Fig. 1.15: Mouse

Touchscreen – The touch screen is both input and an output device. Touchscreens are used in ATM where limited options are available for selection. The touch screen allows the user to use fingers, or a stylus, to directly press buttons and select options that appear on the screen. It is mostly used in smartphones, tablet, iPad and even laptop or notebook now. (Figure 1.16)

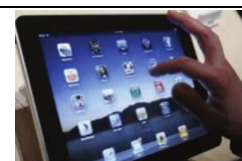


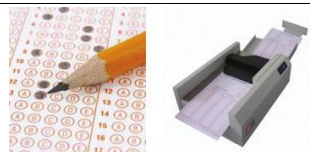


Fig. 1.16: Touchscreen

<p>Scanners – Scanners are used to convert hard copy or printed documents, photographs to soft copy in digital form. The digital data is then stored and manipulated by the computer. (Figure 1.17) Examples of scanners include the traditional flatbed scanner, as well as the more modern mouse scanner.</p>	 <p><i>Fig. 1.17: Scanners</i></p>
<p>Touchpad – The touch-pad is a small square or rectangular input device on a laptop. It has two buttons just like in mouse that perform the same function. The left button is used to select objects and right button is used to bring up a menu. It is used by moving finger across the pad to move the cursor on the screen. It also has a function to scroll up and down. (Figure 1.18)</p>	 <p><i>Fig. 1.18: Touchpad</i></p>
<p>Graphics Tablet – Graphics Tablet is used to create graphics on the computer just as a drawing the graphic on paper. A graphic is drawn with a special pen called as stylus and can be saved on the computer and further edited in graphics software. (Figure 1.19)</p>	 <p><i>Fig. 1.19: Graphics Tablet</i></p>
<p>Microphone – Microphones is a input device that takes sound as input. It accepts the analogue sound signals, which are converted into digital signals by sound card. The digitized sound can be stored and manipulated by the computer. A microphone can be used with digital camera or webcam in online discussions or meetings. (Figure 1.20)</p>	 <p><i>Fig. 1.20: Microphone</i></p>
<p>Digital Camera – A digital camera can be connected to computer through cable or wi-fi. The digital pictures captured can be transferred to computer. Modern digital camera is capable to capture both video and audio. (Figure 1.21)</p>	 <p><i>Fig. 1.21: Digital Camera</i></p>
<p>Web Camera – It is a video capturing input device connected to the monitor via a USB cable or Wi-Fi signal. Laptops and notebooks have a built-in webcam. (Figure 1.22) It is used for online chatting and video conferencing to capture digital photographs and video. Some webcams include a microphone to record sound.</p>	 <p><i>Fig. 1.22: Web Camera</i></p>
<p>Barcode scanner / Reader – Bar code scanner reads the bar code by visible red light reflected through bar code scanner and translates the data into digital information as shown in Figure 1.23. The bar code present on the products holds information of the product such as product id, name and manufacturer. The bar code does not store price of the product. The price is stored in the database which is accessed using product id. After scanning the bar code, the computer can read the information stored on the bar code and access the product details from database.</p>	 <p><i>Fig. 1.23: Barcode scanner</i></p>
<p>Optical Mark Reader (OMR) – The OMR can read the marks made by pen or pencil. OMR shines a light on the form and reflects less light on the pencil mark as shown in Figure 1.24. The selected option is matched with the correct answer in the database and thus the answer sheet is evaluated.</p>	 <p><i>Fig. 1.24: OMR</i></p>

Optical Character Reader (OCR) – The OCR consists of scanner that scans the text on the paper and the in-built software converts the scanned text into digital format. The digital text can be further edited and formatted. Figure 1.25 shows the typical OCR.



Fig. 1.25: OCR

Magnetic Ink Character Recognition (MICR) – MICR devices are mainly used to process cheques in banks. The MICR device reads the account number written in the special ink and converts it to computer understandable form. MICR processes the cheques with speed and accuracy. Figure 1.26 shown the typical MICR.



Fig. 1.26: MICR

OUTPUT DEVICES

An output device is connected to the computer. The output is mostly viewed on computer monitor, or printed through devices like a printer or heard using speakers.

(i) Monitor – The computer monitor or screen is the main output device of computer. It makes it possible for the user to visually interact with data and programs in a quick and easy manner; whether it is a page of text, video on the internet, or 3D elements in a game.

Traditionally, we had CRT (Cathod Ray Tube) monitors, but the modern computer uses the most recent technology of LCD (Liquid Crystal Display) and LED (Light Emitting Diode) display screen. Figure 1.27 shows the different types of monitors.



CRT Monitor



LCD Monitor



LED Monitor

Fig. 1.27: Different Types of monitors

(ii) Speakers

Speakers are the most popular output devices that give sound as output. (Figure 1.28) This makes it possible to listen to music, speak to friends over Skype, or watch movies. Most computers have their own built-in speakers, which are fine for hearing the normal computer sounds. External speakers are plug into audio jack on the computer and preferred to listen to music, movies or games.



Fig. 1.28: Speakers

(iii) Printer

Printer is an output device that accepts electronic data from a computer as an input and generates a hard copy of the data. It is possible to print pages of text, illustrations, diagrams and photos from computer on a paper. A printer is a peripheral device used to make a persistent

human-readable representation of graphics or text on paper. There are various features of printer, such as *speed, paper type and size, printer resolution, memory, ink cartridges and wireless capabilities.*

Types of printer

Printers are broadly categorised as impact printers and non-impact printers. Impact printer contacts the paper, while a non-impact printer prints without contacting the paper.

Impact Printer – An impact printer is a type of printer that works by banging a head or needle against an ink ribbon to make a mark on the paper. Dot-matrix printers, daisy-wheel printers, line printers are examples for impact printer.

Non-impact Printers – Non-impact printers form characters and images without direct physical contact between the printing mechanism and the paper. These printers do not hammer on the paper and hence do not make noise and are faster. Inkjet printers and laser printers are examples of non-impact printer.

Dot-matrix Printer – The dot-matrix printer uses print heads containing 9 pins to 24 pins. (Figure 1.29) The print quality increases with the number of pins used. Patterns of dots are produced by these pins on the paper to form character. Dot-matrix printers are inexpensive and print at speed of 100-600 characters per second.



Fig. 1.29: Dot-matrix Printer

Line Printer – Line printers are fast that use special technology. They can print at the range of 1,200 to 6,000 lines per minute. Drum, chain and band printers are the examples of line printers. (Figure 1.30)



Fig. 1.30: Line Printer

Ink-jet Printers – Ink-jet printers prints on paper by spraying ink from tiny nozzles through electrical field that arranges the charged ink particles into characters at the rate of approximately 250 characters per second. A nozzle for black ink is used to print text, and full color printing is possible with three extra nozzles for the cyan, magenta, and yellow primary colors. A typical ink-jet printer can produce copy with a resolution of at least 300 dots per inch (dpi). (Figure 1.31)



Fig. 1.31: Ink-jet Printer

Laser Printer – A laser printer uses a laser and electrical charge model instead of traditional printing of ink. Laser printers can print high quality text and graphics and medium quality photographs, with typical resolutions of 600 dots per inch or higher. Figure 1.32 shows the typical laser printer. Laser printer repeatedly passes a laser beam back and forth over a negatively charged cylinder called a *drum*. The drum then selectively collects electrically charged powdered ink called as toner, and transfers the image to paper.



Fig. 1.32: Laser Printer

SECONDARY STORAGE DEVICES

Secondary memory also called secondary storage can store large amount of data on permanent and long-term basis. Secondary memory is non-volatile memory and is not directly accessed by the CPU.

Certain parameters should be taken into consideration for selecting the storage device. This includes Storage capacity, Storage speed, Volatility, Reliability and durability. For internal hard drives, a storage speed of 7200 revolutions per minute (RPM) is a good, while the capacity depends on your need. For example, a storage device of 1 TB usually has enough storage space to store software, programs and data files.

Examples of storage devices

The variety of storage devices are available to store data. Internal hard disk drive is the most commonly used because it is fixed inside the computer. The most commonly used storage devices are explained here.

Internal Hard Disk Drive – It comes with every computer is fixed inside the cabinet. It is used to store data as well as to install operating system and software. Modern hard drives have the potential to store up to 12 terabytes (12 TB) of data. (Figure 1.34)



Fig. 1.34: Internal Hard Disk

External Hard Disk Drive – These are portable storage devices used to back up the data. External hard drives are protected by a case that is designed to prevent damage to the drive and is connected to the computer by using a USB cable. The small size and portability of external hard drives ensure that to quickly connect to different computers and therefore, ideal for backing up data from internal hard drive. (Figure 1.35)



Fig. 1.35: External or portable Hard Disk Drive

Solid-State Drives – The SSD is also a hard drive that does not use the traditional mechanical parts of standard hard drives. Instead, the drive consists of interconnected flash memory chips made of silicon. Due to this, SSD functions more like a CPU that contains billions of small transistors; each storing one bit of data. SSDs are many times faster than normal hard drives. Generally, the operating system and software are kept in SSD and data is stored in hard disk to improve the performance of computer. It's capacity ranges from 256 to 1TB. (Figure 1.36)



Fig. 1.36: Solid State Drive

Universal Serial Bus (USB) or Flash drive – A flash drive is a very small portable and re-writable storage device. It is non-volatile in nature. You can connect it to computer via a USB port. A flash drive is the best way to transfer data quickly and efficiently between two computers. A flash drive is also an ideal storage device to store documents which can be easily used on different computers. It is also referred to as a thumb drive, USB stick, memory stick, or flash stick. Flash drives usually have a storage capacity between 4 GB and 512 GB. (Figure 1.37)

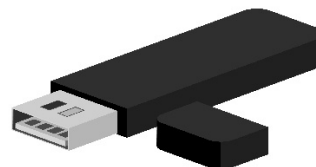


Fig. 1.37: USB Flash Memory

Optical Disk – An optical disc such as compact discs (CDs), digital optical discs (DVDs) and Blu-ray discs (BDs) are used to read and write the data. (Figure 1.38) It is the most popular storage device to store music and movies. One of the major advantages of these drive is that it gives the user an affordable and time-efficient way of producing multiple copies of a disc. However, due to the limited storage space, these are outdated now.



Fig. 1.38: Optical disks (a) CD (b) DVD (c) Blue Ray disk

Memory cards – Memory card is another type of flash memory. They are very small in size of about 1×1 inch with thickness is in millimeters. Memory cards can also be externally connected to the computer systems. They can also be used with cell phones and tablets. A typical storage capacity of memory cards varies from 1 GB to 64 GB. (Figure 1.39)



Fig. 1.39: Memory Cards

Capacity and cost of the most common storage devices

We have discussed some of the most commonly used storage devices. From the above discussion you can easily understand that which device will be suited for a specific task. Table 1.1 gives a summary of the capacity and cost of these devices, as well as for which task they are best suited.

Table 1.1 Capacity and cost of these devices

Storage Device	Storage Capacity	Used for the task
Internal HDD	1 TB to 4 TB	Primary computer storage
External HDD	1 TB to 4 TB	Taking backup of large amount of data
SSD	128 GB to 1024 GB	Improving the speed of operating system.
Flash drive	1 GB to 512 GB	Transferring small amount of data between computers.
Optical drives	CD – 700 MB	Transferring small amount of data quickly to multiple recipients.
	DVD – 4.7 GB	Distributing data to multiple recipients.

Cloud Storage

Cloud storage is the modern storage used to store data including programs, software, music or video files. It allows to store information on the internet with security and easy access using internet. There are various types of cloud storage available for use. Few of them which are commonly used are *Google drive*, *Apple's iCloud*, *One Drive*, and *Dropbox*. It provides specific amount of storage, for example, 5 GB for iCloud and 15 GB for Google Drive at no charge. Additional storage space can be purchased. In the modern world most of the software development platforms are available under cloud. The programs are directly executed under the cloud environment without having the compiler or interpreter with the programmer in their local machine. For example, you can execute python programs directly from cloud. Therefore, cloud storage is one of the preferred storage for software development.

Activity 4

Practical Activity 4. Identify and name the various storage devices

Materials Required

Hard Disk Drive, Pen Drive, Compact Disk

Procedure

Step 1. Observe the given storage devices carefully, identify and record its name as per its shape and size. Record your observation as shown in the following table.




Storage device	Name of the storage device	Storage capacity
 Fig. 1.40 (a)	Hard Disk Drive	1 TB
 Fig. 1.40 (b)	Pen Drive	32 GB
 Fig. 1.40 (c)	Compact Disk	700 Mb



Fig. 1.40 (d)

Digital Veritable Disk (DVD)

4 GB

Activity 5

Practical Activity 5. Identify and connect peripheral devices to the computer

Materials Required

Mouse, Keyboard, Printer, Scanner, External Hard Disk Drive, Pen Drive

Procedure

Step 1. Take a peripheral device.

Step 2. Identify the corresponding socket on the CPU of computer system.

Step 3. Connect the device to the appropriate socket on the CPU of computer system.

Step 4. Ensure the proper connectivity.

Step 5. Check whether the device is properly connected or not.

Check Your Progress

A. Multiple Choice Questions

- RAM is used (a) to store the boot program (b) to store applications (c) to load the operating system (d) both b and c
- Which of the following is the fastest memory in a computer (a) RAM (b) Registers (c) HDD (d) ROM
- Type of printer in which characters or letters are formed without use of any mechanical impact is termed as (a) Page printers (b) Line printers (c) Impact printer (d) Non-impact printer
- The devices that are used for storing program and data for long term are (a) volatile devices (b) non-volatile devices (c) primary memory (d) CPU registers
- Hard copy of the data can be obtained by using a device called as _____ (a) keyboard (b) mouse (c) pen drive (d) printer
- Printers that contact the paper are called as ___ printer (a) non-impact (b) impact (c) global (d) local
- Printers that do not make any contact with the paper while printing data are called as ___ printer (a) non-impact (b) impact (c) global (d) local
- Dot matrix printer is _____ (a) non-impact (b) impact (c) global (d) local
- Laser printer is _____ (a) non-impact (b) impact (c) global (d) local
- Which of the following is the faster printer (a) dot matrix (b) inkjet (c) laser (d) line.
- Which of the following printer incur low cost for printing (a) dot matrix (b) inkjet (c) laser (d) line.
- Which of the following printer speed is measured in characters per second. (a) Inkjet (b) Laser (c) Dot matrix (d) Drum
- GPS stands for (a) Global Positioning System (b) Global Partitioning System (c) Google Positioning System (d) Global Partitioning System
- In modern computers, DVD is replaced by which technology? (a) Flash disk (b) HDD (c) Blu-ray (d) CD
- What is the average capacity of a DVD? (a) 700 MB (b) 125 GB (c) 4-5 GB (d) 2-4 TB
- Which of the following is NOT a storage device? (a) HDD (b) RAM (c) SSD (d) USB stick
- What is average rotational speed of an internal hard drive's? (a) 3600 rpm (b) 7200 rpm (c) 9200 rpm (d) 8400 rpm
- Which of the following is a limitation of an internal hard drive? (a) Data can be retrieved and saved much faster than from DVDs or CDs (b) Lacks portability, as they are fixed inside the computer (c) Permanent storage of data (d) Large capacity of storage space

19. Which of the following is a limitation of an external hard drive? (a) Large capacity of storage space (b) Lacks portability, as they are fixed inside the computer (c) Slightly slower than an internal hard drive (d) Slightly more storage space than an internal hard drive
20. Which of the following is an advantage of SSDs? (a) More durable and compact due to absence of mechanical parts (b) Longer life span than standard hard drives or flash drives (c) Less expensive than standard mechanical hard drives (d) Increased storage capacity due to high cost per gigabyte

B. Fill in the Blanks

1. Computer has different types such as desktop, laptop and _____.
2. The term compute is derived from the Latin word _____.
3. A computer can perform arithmetic and _____ operations.
4. A set of the instructions is called as _____.
5. Central Processing Unit consists of control unit and _____
6. Keyboard and mouse are _____ devices of computer system.
7. Monitor and printer are _____ devices of computer system.
8. ROM means _____.
9. Primary memory is computer memory that is accessed directly by the _____
10. Printers are mainly divided into _____ and _____ printers.

C. True and False

1. UPS provides uninterruptible power supply
2. UPS needs battery for its operation
3. Mouse and keyboard cannot be connected to the USB port
4. Speakers of the computer systems are generally connected to the USB port
5. Modern computer is an analog electronic device
6. Tablet is a type of computer
7. Memory of the computer system is measured in bytes
8. One megabyte means 1 Kilobyte
9. In computer the data is stored in 0s and 1s
10. Cache memory is slower and consumes lot of access time than main memory
11. Cache memory is used to store application programs
12. The instructions stored in ROM can be changed by the user

D. Short Answer Question

1. What is the use of USB port?
2. Draw the connectivity diagram of UPS and battery.
3. List the devices that can be connected to USB port.
4. How ink jet printers are different from laser printers?
5. Draw the diagram for inkjet printers.
6. With suitable example explain the uses of line printer.
7. Draw the diagram of dot matrix printer and explain its working.
8. List the uses of memory card and pen drive in different applications.
9. What is cache memory. State advantages of cache memory?
10. What are the Characteristics of Main Memory?
11. What are the major differences between RAM and ROM?
12. What are the various types of printers?
13. Define digital computer and draw its block diagram?
14. State the different types of computer along with their features.
15. What are the different I/O devices that are commonly used with computer system?

Session 2: Data Representation in Computer

Suppose you have passed class 10 examination and you want to continue your education. Naturally, you will approach to your nearby school. To get an admission you need to fill a form. A sample form is shown in Figure 2.1.

The form is yellow and contains the following fields and sections:

- Top left: School logo and name.
- Form fields: Name, Date of Birth, Telephone Number, Address, Email, Cell, and a field for pasting a photograph.
- Declaration: A text box with a small print disclaimer: "I certify that all information supplied in this application is complete and correct. Misrepresentation of information stated in this application will be considered sufficient grounds both for refusal of admission and expulsion. I understand that upon signing this form, I am agreeing to abide by the terms and conditions, rules, and regulations, and administrative policies of [School Name] School."

Fig. 2.1: School Admission form

In this form you will be entering your name, address, date of birth, telephone number and pasting your photograph. When you fill such information, then actually you are providing your personal data to the school.

In this chapter, you will understand the concept of data and how it is represented in computer memory. Data can be in any form such as number, text, picture, audio, video and so on. This data is stored in encoded form in computer memory. This encoded data is represented in the series of binary digits 0 and 1, also called machine code. You will also understand the data encoding systems, number system and conversions from one number system to another.

Data and Information

The word data is taken from Latin word "*datum*". Data is raw, unorganised numbers, signals, or facts. Data can be in different formats such as number, text, image, audio, video. It becomes useful when it is processed or organised. The processed data is called as information. Data can be measured, collected, reported analysed and visualized. For example, your school might have data on the names, addresses, contact details, as well as the results of every class stored on a computer. It is difficult to interpret such large data of number of students.

Information refers to facts and numbers that have been organised or processed so that they are useful to people. For example, if your teacher want to see your performance as compared to previous class. It is possible to infer this by taking the averages of the two years. In this way, thousands of pages of data will be converted into two numbers that can be compared easily. Similarly, the report you receive at the end of each school year takes all the data that the teachers collected throughout the year and turns that data into a single report that you can use to measure your performance. If data is well organized and processed then it becomes useful. Figure 2.2 shows how data is transformed into information after processing.



Fig. 2.2: Data and information

Activity 1

Practical Activity 2.1. Identify data types for the given data values

Materials Required


Pen, Paper, Photo, Personal details

Procedure

Step 1. Prepare your personal details consisting of Name, Father's Name, Photo and other details as shown in the following table.

Step 2. Identify the data type for each of data value provided by you for your personal details.

Step 3. Record the details as shown in the following table.

Personal Details		
Data Item	Data Value	Data type
Name	Ms. Devanshi Ghosh	Character with text
Photo		Image
Father's Name	Mr. Aryan Ghosh	Character with text
Mother's Name	Mrs. Poorva Ghosh	Character with text
Permanent Address	House Number 215, APJ Kalam Marg, Delhi	Character with text
Present Address	House Number 115, Kasturba Nagar, Bhopal	Character with text
Date of Birth	9 December, 1990	Date with number and text
Qualification	SSC	Character with text
Height	158 CM	Number with text
Weight	51 kg	Number with text
Contact Number	987654321	Number

Activity 2

Practical Activity 2.2. Visualise the data using bar graph, pie chart and line graph

Materials Required

Computer system with spreadsheet software, Paper, Given data

Procedure

Step 1. Record the maximum and minimum temperature in your town on all weekdays as shown in the table below.

Day	Max. Temperature in degree Celsius	Min. Temperature in degree Celsius
Sunday	41.2	29.1
Monday	40.6	29.6
Tuesday	40.8	29.5
Wednesday	41.1	30.0

Thursday	41.2	30.1
Friday	41.0	29.9
Saturday	41.3	30.3

Step 2. Enter the data in the spreadsheet. Create the column chart and visualise the data as shown in Figure 2.3 (a).

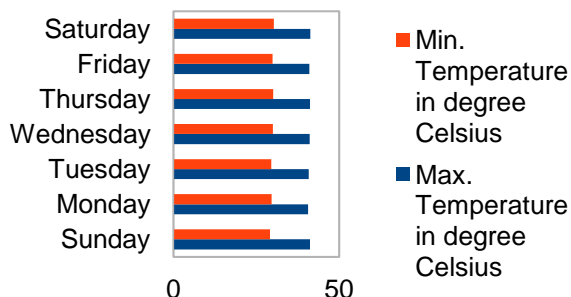


Fig. 2.3 (a) Visualising data in column chart

Step 3. Create the bar chart and visualise the data as shown in Figure 2.3 (b).

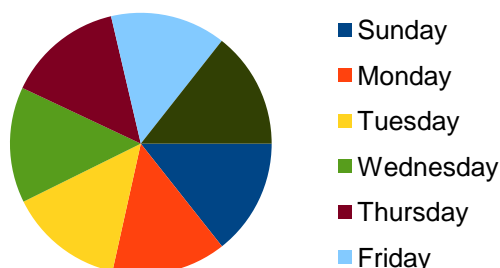


Fig. 2.3 (b) Visualising data in bar chart

Step 4. Create the bar chart and visualise the data as shown in Figure 2.3 (c)

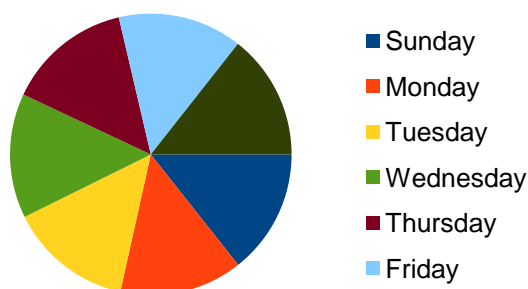


Fig. 2.3 (c) Visualising data in pi chart

Step 5. Record your observations in visualising the data.

Data Representation in Computer

Digital computers process data that is in discrete form. The data entered by user is first converted and transmitted as electrical pulses that can be represented by two unique states ON and OFF. The ON state may be represented by a “1” and the off state by a “0”. The sequence of ON and OFF forms the electrical signals that the computer can understand.

Units to Measure Data Storage in Computer Memory

Computer memory stores data and instructions in the form of machine code. The digital data whether it may be number, text, picture, audio or video is the collection of number of bits. A **binary digit** (bit) represented by “0” or “1” is the smallest unit of data. Data storage capacity is

measured in units such as bits, bytes, kilobytes, Megabytes, Gigabytes, Terabytes and Petabytes. Let us understand how to arrange these different data storage capacities in from the small unit to big unit and to define the relationships between these as well.

Bit	A Binary Digit . (Bit) is the binary digit 0 or 1. It is the basic unit of data or information in digital computers.
Byte	Byte is group of 8 bits used to represent a character. It is the basic unit of measuring memory size in computer.
Nibble	A nibble is half a byte, a group of 4 bits.
Kilobyte	Kilobyte (KB) consists of 1024 (or 2^{10}) bytes. ($1024 = 2^{10}$).
Megabyte	Megabyte (MB) consists of 1024 (or 210) kilobytes = 1048576 (or 220) bytes.
Gigabyte	Gigabyte (GB) is made of 1024 Megabytes. (1024 MB). It is wrong to write 'Gb' as it indicates gigabit.
Terabyte	Terabyte is made of 1024 Gigabytes (1024 GB). This is written as TB.
Petabyte	Petabyte is made of 1024 Terabytes (1024 TB).

Coding Systems

Computer can only process the machine code of data and instructions. So, whether any data that contains numbers, letters, special symbols, sound or pictures is first be converted into machine code in binary form and then it is processed in computer and again the result of the processed data is converted into human readable form that appears on the computer screen.

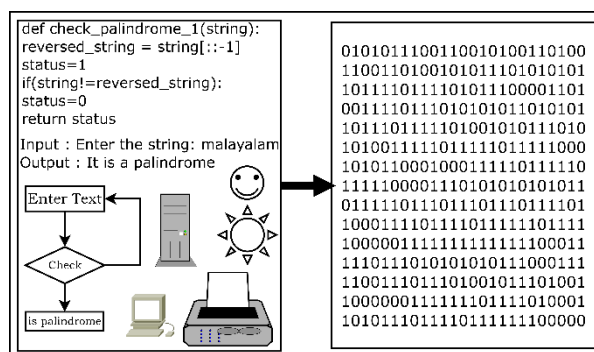


Fig. 2.3: Data representation in computer

Observe that in Figure 2.3 any data entered by the user is converted into different patterns made of 0 and 1. Thus, binary codes are used to store numeric, alphabetic, special character, images and sounds in internal storage devices of computers. The entire program or software can also be represented by binary code.

To understand this, suppose you have to type the letter "A" on computer keyboard. The data supplied in the form of letter "A" is converted into machine code, that goes to video memory through RAM and hence displayed on the screen as "A". The Figure 2.4 shows the process involved in the operation.

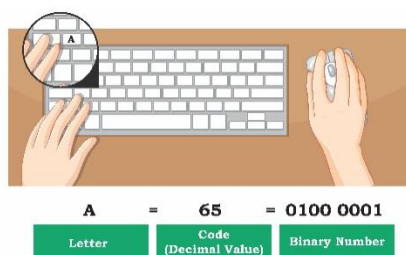


Fig. 2.4: Encoding of data entered using keyboard

When the key 'A' is pressed, it is internally mapped to its machine code "1000001" to understand by the computer. This mechanism of converting data into machine code is called *encoding*. This encoded value is the same for all computer machines irrespective of manufacturer, because all computers follow the standard encoding schemes where each letter, numeral and symbol is encoded or assigned a unique code. Currently used encoding schemes are:

1. Binary Coded Decimal (BCD)
2. American Standards Code for Information Interchange (ASCII)
3. Extended Binary Coded Decimal Interchange Code (EBCDIC)
4. Unicode

1. Binary Coded Decimal (BCD)

BCD stands for Binary Coded Decimal. In this coding system, the decimal values from 0 to 9 are represented in 4 digit binary code. In this system one digit is represented by 4 bits. This 4-bit code used to represent numeric data only. For example, decimal $(9)_{10}$ can be represented using BCD as $(1001)_2$. The weights bits from left to right are 8,4,2,1 respectively and hence also called as 8421 code. This is used only to represent decimal numbers. Sixteen symbols ($2^4 = 16$) can be represented in this system. The table 2.1 shows the BCD codes for decimal values from 0 to 9. A *standard Binary Coded Decimal*, an enhanced format of Binary Coded Decimal, is a 6-bit representation scheme allows to represent non-numeric characters also. This allows $2^6 = 64$ characters to represent. For example, letter A can be represented as $(110001)_2$ using standard Binary Coded Decimal.

Table 2.1: Decimal Numbers and BCD Values

Decimal Value	BCD Value 8421
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Example 2.1, Let us see how a number $(37)_{10}$ represented in BCD.

$$\begin{aligned} &3_{10} \quad 7_{10} \\ &(0011)_2 \quad (0111)_2 \\ (37)_{10} &= (00110111)_2 \end{aligned}$$

Assignment – Write the BCD values for the decimal values given below.

- (i) 302 (ii) 2136 (iii) 17295

In 4-bit BCD only $2^4 = 16$ configurations are possible which is insufficient to represent the various characters. Hence 6-bit BCD code was developed by adding two zone positions with which it is possible to represent $2^6 = 64$ characters.

2. American Standard Code for Information Interchange (ASCII)

The ASCII encoding scheme was developed to standardize the character representation. ASCII coding system uses 7-bit binary digit to represent a character. The total number of different characters that can be encoded by 7-bit ASCII code is $2^7 = 128$. So, 128 characters can be represented using this coding system. However, manufactures have added 8th bit to this coding scheme, which can now provide for 256 characters. It is referred as 8-bit ASCII. Table 2.2 shows

some printable characters for ASCII code. ASCII is able to encode character set of English language only.

Table 2.2 ASCII code for some printable characters

Character	ASCII Code in Decimal	ASCII Code in Binary
space	32	0100000
!	33	0100001
"	34	0100010
#	35	0100011
\$	36	0100100
%	37	0100101
&	38	0100110
'	39	0100111
(40	0101000
)	41	0101001
@	64	1000000
A	65	1000001
B	66	1000010
C	67	1000011
D	68	1000100
E	69	1000101
F	70	1000110
G	71	1000111
H	72	1001000
I	73	1001001
a	97	1100001
b	98	1100010
c	99	1100011
d	100	1100100
e	101	1100101
f	102	1100110
g	103	1100111
h	104	1101000
i	105	1101001

Example 2.2, Encode the word "School" and convert the encoded value into binary values which can be understood by a computer.

First find the ASCII code for various letters appeared in the word "School".

ASCII value of "S" is 83 in decimal and its 7-bit ASCII code = 1010011

ASCII value of "c" is 99 in decimal and its 7-bit ASCII code = 1100011

ASCII value of "h" is 104 in decimal and its 7-bit ASCII code = 1101100

ASCII value of "o" is 111 in decimal and its 7-bit ASCII code = 1101000

ASCII value of "l" is 108 in decimal and its 7-bit ASCII code = 1101111

Replace each alphabet in "School" with its ASCII code value to get its equivalent ASCII code and with 7-bit binary code to get its equivalent binary number as shown in Table 2.3.

Table 2.3 ASCII and Binary values for word "School"

Code	S	c	h	o	o	l
Decimal	83	99	104	111	111	108
Binary	1010011	1100011	1101100	1101000	1101000	1101111

Assignment – Write down the ASCII code of “ICT” in binary numbers.

3. Extended Binary Coded Decimal Interchange Code (EBCDIC)

It is possible to write only 128 characters using ASCII system, but the EBCDIC code system allows the use of 256 characters. Here, one symbol can be written with a binary number which consists of 8 bits. Hence, 256 characters can be represented using this system. This system was used in IBM main frame computers. The table 2.4 shows that there are different EBCDIC codes for the 26 different capital letters and 26 different EBCDIC codes for the 26 simple letters in this system.

Table 2.4: EBCDIC values for English capital and simple letters

Character	Binary code	Hexadecimal code
A	1100 0001	C1
B	1100 0010	C2
C	1100 0100	C3
D	1100 0101	C4
a	1000 0001	81
b	1000 0010	82
c	1000 0011	83
d	1000 0100	84

4. Unicode System

There were many encoding schemes to represent character sets of different languages. But they were not able to communicate with each other, as each of them represented characters in their own ways. ASCII system used 128 characters and EBCDIC system uses 256 characters for data representation. Hence, text created using one encoding scheme was not recognised by another machine using different encoding scheme. These coding systems cannot be used for languages such as Devanagari, Tamil, Telugu, Japanese, Chinese languages as there are more than 256 characters. Hence Unicode system was designed to represent 65536 different symbols of 16 bits ($2^{16} = 65536$). It can represent $2^{16} = 65536$ different symbols of 16 bit.

Unicode **encoding scheme** is used to represent Unicode characters as bits. The Unicode encoding scheme allows to represent each character with a pattern of bits.

It incorporates all the characters of every written language of the world. Unicode system provides a unique number for every character. The Unicode encoding schemes are known as UTF (Unicode Transformation Format). There are various UTF formats such as UTF-8, UTF-16, and UTF-32. It is a super-set of ASCII, and the values 0–128 have the same character as in ASCII. Some of the Unicode encoding schemes are fixed length, and some are variable length. In fixed length, each character is represented using the same number of bits. Variable length means that some characters are represented with fewer bits than others.

The Unicode system are classified to represent characters of all the international languages. The institution which initiated is the International Standard Institution and Unicode Consortium. Unicode is largely used in websites.

UTF-8 is a *variable length* encoding schemes scheme for Unicode. It requires 8 bits to represent a character in UTF-8 scheme.

UTF-16 is a *variable length* encoding scheme for Unicode. It requires 16 bits to represent a character in UTF-8 scheme.

UTF-32 is a *fixed length* Unicode encoding scheme. It requires 32 bits to represent a character in UTF-32 scheme.

Character	UTF-8	UTF-32
H	01001000	00000000 00000000 00000000 01001000
\$	00100100	00000000 00000000 00000000 00100100

Unicode characters for Devanagari script is shown in Table 2.5. Each cell of the table contains a character along with its equivalent hexadecimal value.

Table 2.5 Unicode table for the Devanagari script

U+0900	U+0901	U+0902	U+0903	U+0904	U+0905	U+0906	U+0907	U+0908	U+0909	U+090A	U+090B	U+090C	U+090D	U+090E	U+090F
ँ	ं	।	:	ऌ	ऍ	ऎ	ए	ऐ	ऑ	ऒ	ओ	औ	क	ख	ग
U+0910	U+0911	U+0912	U+0913	U+0914	U+0915	U+0916	U+0917	U+0918	U+0919	U+091A	U+091B	U+091C	U+091D	U+091E	U+091F
ऐ	ऑ	औ	ओ	औ	क	ख	ग	घ	ङ	च	छ	ज	झ	ञ	ट
U+0920	U+0921	U+0922	U+0923	U+0924	U+0925	U+0926	U+0927	U+0928	U+0929	U+092A	U+092B	U+092C	U+092D	U+092E	U+092F
ठ	ड	ढ	ण	त	थ	द	ध	न	न	प	फ	ब	भ	म	य
U+0930	U+0931	U+0932	U+0933	U+0934	U+0935	U+0936	U+0937	U+0938	U+0939	U+093A	U+093B	U+093C	U+093D	U+093E	U+093F
र	ऱ	ल	ळ	ळ	व	श	ष	स	ह	'	।	.	ऽ	।	ि
U+0940	U+0941	U+0942	U+0943	U+0944	U+0945	U+0946	U+0947	U+0948	U+0949	U+094A	U+094B	U+094C	U+094D	U+094E	U+094F
ी	ी	ी	ी	ी	ी	ी	ी	ी	ी	ी	ी	ी	ी	ी	ी
U+0950	U+0951	U+0952	U+0953	U+0954	U+0955	U+0956	U+0957	U+0958	U+0959	U+095A	U+095B	U+095C	U+095D	U+095E	U+095F
ॐ	'	-	'	'	'	'	'	ऋ	ॠ	ऌ	ॡ	ऴ	व	श	ष
U+0960	U+0961	U+0962	U+0963	U+0964	U+0965	U+0966	U+0967	U+0968	U+0969	U+096A	U+096B	U+096C	U+096D	U+096E	U+096F
ऋ	ॠ	ॡ	ॢ	।	॥	०	१	२	३	४	५	६	७	८	९
U+0970	U+0971	U+0972	U+0973	U+0974	U+0975	U+0976	U+0977	U+0978	U+0979	U+097A	U+097B	U+097C	U+097D	U+097E	U+097F
०	.	अँ	अं	आँ	आं	अुँ	अुं		ॠ	ॡ	गुँ	गुं	?	डुँ	डुं

Number of bits used by the different coding systems.

Coding System	Number of Bits Used
Binary Coded Decimal (BCD)	4
American Standard Code for Information Interchange (ASCII)	7
Extended Binary Coded Decimal Interchange Code (EBCDIC)	8
Unicode	8, 16 or 32

Number System

The number system uses well defined symbols called digits. A number system is a method to represent numbers. Number system is basically classified into two types – (i) Non-positional number system and (ii) Positional number system.

Non-Positional Number System – The non-positional number system consists of different symbols that are used to represent numbers. Roman number system is an example of the non-

positional number system i.e. I=1, V=5, X=10, L=50. This number system cannot be used effectively to perform arithmetic operations.

Positional Number System – In positional numbering systems a numeric value is represented through increasing powers of a radix or base, which is the total number of digits present in number system. Every number is represented by a base x , which represents x digits. The base is written after the number as subscript such as $(512)_{10}$. It is a Decimal number as its base is 10.

The number is multiplied by an integer power of x depending on its position and then finds the sum of the weighted digits.

For Example, consider a decimal number $(512.45)_{10}$ which can be represented in equivalent value as:

$$5 \times 10^2 + 1 \times 10^1 + 2 \times 10^0 + 4 \times 10^{-1} + 5 \times 10^{-2}$$

The number system used for data representation in the computer is as given below.

Number System	Base	Number of digits
Binary	2	0, 1
Octal	8	0, 1, 2, 3, 4, 5, 6, 7
Decimal	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Hexadecimal	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Let us discuss each number system.

Decimal Number System

Decimal number system is well known to us from our childhood. It has base 10 and digits are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Normally, the base value of decimal numbers are not mentioned as it is used by default in our daily life. So, if the number is not mentioned with its base then it is assumed as a decimal number with base 10. Examples are: $(123)_{10}$, $(456)_{10}$, $(7890)_{10}$.

Consider a decimal number $(542.76)_{10}$ which can be represented in equivalent value as:

$$5 \times 10^2 + 4 \times 10^1 + 2 \times 10^0 + 7 \times 10^{-1} + 6 \times 10^{-2}$$

	Hundreds	Tens	Units	One-tenth	One-hundredth
Weights	10^2	10^1	10^0	10^{-1}	10^{-2}
Digits	5	4	2	7	6
Values	500	40	2	0.7	0.06

Binary Number System

Digital computer represents the data and information in the binary system. Binary number system consists of two digits 0 (low voltage) and 1 (high voltage). Its base or radix is 2. Each digit or bit in binary number system can be 0 or 1. The positional values are expressed in power of 2.

Example: $(1011)_2$, $(111)_2$, $(100001)_2$

Consider a binary number $(11011)_2$ which can be represented in equivalent value as:

$$1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$

Weights	2^4	2^3	2^2	2^1	2^0
Digits	1	1	0	1	1
Values	16	8	4	2	1

In the binary number 11010, the left most bit 1 is the highest order bit, called as Most Significant Bit (MSB) and the right most bit 0 is the lowest bit, called as Least Significant Bit (LSB).

Octal Number System

The octal number system has digits starting from 0 to 7. The base or radix of this system is 8. The positional values are expressed in power of 8. Any digit in this system is always less than 8. Examples: $(123)_8$, $(236)_8$, $(564)_8$

The number 6418 is not a valid octal number because 8 is not a valid digit in octal system.

Consider Octal number $(234)_8$ which can be represented in equivalent value as:

$$2 \times 8^2 + 3 \times 8^1 + 4 \times 8^0$$

Weights	8^2	8^1	8^0
Digits	2	3	4
Values	64	8	1

Hexadecimal Number System

The hexadecimal number system consists of 16 digits from 0 to 9 and A to F. The letters A to F represent decimal numbers from 10 to 15. That is, 'A' represents 10, 'B' represents 11, 'C' represents 12, 'D' represents 13, 'E' represents 14 and 'F' represents 15. The base or radix of this number system is 16.

Examples: $(A4)_{16}$, $(1AB)_{16}$, $(93E4)_{16}$, $(2FE.8C)_{16}$

Consider a Hexadecimal number $(5AF.D)_{16}$ represented in equivalent value as:

$$5 \times 16^2 + A \times 16^1 + F \times 16^0 + D \times 16^{-1}$$

Weights	16^2	16^1	16^0	16^{-1}
Digits	5	A	F	D
Values	256	16	1	0.0625

Table 2.6: Number system table

Decimal	Binary	Octal	Hexadecimal
Base-10	Base-2	Base-8	Base-16
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Applications of Hexadecimal Number System

In computer, each memory location has a unique address. Usually, size of a memory address is 16-bit or 32-bit. To access 16-bit memory address, a programmer has to use 16 binary bits, which is difficult to deal with. To simplify the address representation, hexadecimal and octal numbers are used. Let us consider a 16-bit memory address 1100000011110001. Using the

hexadecimal notation, this address is mapped to COF1 which is easier to remember. The equivalent octal representation for this 16-bit value is 140361.

Hexadecimal numbers are also used for describing the colours on the webpage. Each colour is made up of three primary colours red, green and blue, popularly called RGB (in short). In most colour maps, each colour is usually chosen from a palette of 16 million colours. Therefore, 24 bits are required for representing each colour (8 bits for Red, 8 bits for Green, 8 bits for Blue component). It is difficult to remember 24-bit binary colour code. Therefore, colour codes are written in hexadecimal form for compact representation.

For example, 24-bit code for RED colour is 11111111,00000000,00000000. The equivalent hexadecimal notation is (FF,00,00), which can be easily remembered and used. Table 2.7 shows examples of some colours represented with decimal, binary, and hexadecimal numbers.

Table 2.7 Colour codes in decimal, binary and hexadecimal numbers

Color Name	Decimal	Binary	Hexadecimal
Black	(0,0,0)	00000000,00000000,00000000	00,00,00
White	(255,255,255)	11111111,11111111,11111111	FF,FF,FF
Yellow	(255,255,0)	11111111,11111111,00000000	FF,FF,00
Grey	(128,128,128)	10000000,10000000,10000000	80, 80, 80

Number Systems Conversion

You have learnt the – decimal, binary, octal and hexadecimal number systems and you also observed from the above discussion that you may be required to convert the number from one number system to another as per the need. So, it is essential to understand how to convert a number from one number system to another number system. Decimal number system is most commonly used by humans, but digital computer understands binary numbers; whereas Octal and hexadecimal number systems are used to simplify the binary representation for us to understand.

Conversion from Decimal to other Number Systems

(a) Decimal to Binary Conversion

The base value of binary system is 2. So, to convert the given decimal number to binary number, divide the given decimal number repeatedly by 2 as per the above steps till the quotient is 0. Record the remainder after each division and finally write the remainders in reverse order in which they are computed.

In Practical Activity 2.3, observe that the binary equivalent of 65 is $(1000001)_2$. Let us now convert a decimal value to its binary representation and verify that the binary equivalent of $(65)_{10}$ is $(1000001)_2$.

Activity 3

Practical Activity 2.3 – Convert the binary equivalent of the decimal number $(65)_{10}$

Step 1. Divide the given decimal number 65 by 2.

Step 2. Write the remainder.

Step 3. Keep on dividing the quotient by the base value 2 and note the remainder till the quotient is zero.

Step 4. Collect the remainders from bottom to top to get the binary equivalent of $(65)_{10}$ $(1000001)_2$.

$$65/2 = 32 \quad (\text{Remainder} = 1)$$

$$32/2 = 16 \quad (\text{Remainder} = 0)$$

$$\begin{array}{l}
 16/2 = 8 \quad (\text{Remainder} = 0) \\
 8/2 = 4 \quad (\text{Remainder} = 0) \\
 4/2 = 2 \quad (\text{Remainder} = 0) \\
 2/2 = 1 \quad (\text{Remainder} = 0) \\
 1/2 = 0 \quad (\text{Remainder} = 1)
 \end{array}$$

Fig. 2.5: Conversion of a decimal number to its equivalent binary number

Example 2.5. Convert $(122)_{10}$ to binary number.

$$\begin{array}{l}
 122/2 = 61 \quad (\text{Remainder} = 0) \\
 61/2 = 30 \quad (\text{Remainder} = 1) \\
 30/2 = 15 \quad (\text{Remainder} = 0) \\
 15/2 = 7 \quad (\text{Remainder} = 1) \\
 7/2 = 3 \quad (\text{Remainder} = 1) \\
 3/2 = 1 \quad (\text{Remainder} = 1) \\
 1/2 = 0 \quad (\text{Remainder} = 1)
 \end{array}$$

Therefore, $(122)_{10} = (1111010)_2$

Assignment – Convert the following decimal numbers in the binary number system.

- (i) $(75)_{10}$
- (iii) $(479)_{10}$
- (iii) $(9843)_{10}$

(b) Decimal to Octal Conversion

The base value of octal number system is 8. So, the given decimal number is repeatedly divided by 8 to obtain its equivalent octal number.

The octal equivalent of letter “A” by using its ASCII code value $(65)_{10}$ is calculated in the Practical Activity 2.4.

Activity 4

Practical Activity 2.4 – Convert the octal equivalent of the decimal number $(65)_{10}$

Step 1. Divide the given decimal number 65 by 8.

Step 2. Write the remainder.

Step 3. Keep on dividing the quotient by the base value 8 and note the remainder till the quotient is zero.

Step 4. Collect the remainders from bottom to top to get the octal equivalent of $(65)_{10} = (100)_8$.

$$\begin{array}{l}
 65/8 = 8 \quad (\text{Remainder} 1) \\
 8/8 = 1 \quad (\text{Remainder} 0) \\
 1/8 = 0 \quad (\text{Remainder} 1)
 \end{array}$$

Fig. 2.6: Conversion of a decimal number to its equivalent octal number

Example 2.6. Convert $(122)_{10}$ to octal number.

$$\begin{array}{l}
 122/8 = 15 \quad (\text{Remainder} 2) \\
 15/8 = 1 \quad (\text{Remainder} 7) \\
 1/8 = 0 \quad (\text{Remainder} 1)
 \end{array}$$

Therefore, $(122)_{10} = (172)_8$

Assignment – Convert the following decimal numbers into octal number.

(i) $(87)_{10}$

(iii) $(847)_{10}$

(iii) $(9834)_{10}$

(c) Decimal to Hexadecimal Conversion

The base value of hexadecimal number system is 16. So, the given decimal number is repeatedly divided by 16 to obtain its equivalent hexadecimal number.

The hexadecimal equivalent of letter “A” by using its ASCII code value $(65)_{10}$ is calculated in the Practical Activity 2.3.

Practical Activity 2.3 – Convert the hexadecimal equivalent of the decimal number $(65)_{10}$

Step 1. Divide the decimal number by 16.

Step 2. Write the remainder.

Step 3. Keep on dividing the quotient by the base value 16 and note the remainder till the quotient is zero.

Step 4. Collect the remainders from bottom to top to get the hexadecimal equivalent.

$$65/16 = 4 \quad (\text{Remainder } 1)$$

$$4/16 = 0 \quad (\text{Remainder } 4)$$

Fig. 2.7: Conversion of a decimal number to its equivalent hexadecimal number

Example 2.7 Convert $(122)_{10}$ to hexadecimal number

Therefore, $(122)_{10} = (7A)_{16}$

$$122/16 = 7 \uparrow (\text{Remainder } 10 \text{ means } A)$$

$$7/16 = 0 \quad \uparrow (\text{Remainder } 7)$$

Assignment – Convert the following decimal numbers into hexadecimal number.

(i) $(84)_{10}$

(ii) $(506)_{10}$

(iii) $(2976)_{10}$

Conversion from other Number Systems to Decimal Number System

We can use the following steps to convert the given number with base value b to its decimal equivalent, where base value b can be 2, 8 and 16 for binary, octal and hexadecimal number system, respectively.

Step 1. Write the position number for each alphanumeric symbol in the given number.

Step 2. Get positional value for each symbol by raising its position number to the base value b symbol in the given number.

Step 3. Multiply each digit with the respective positional value to get a decimal value.

Step 4. Add all these decimal values to get the equivalent decimal number.

(a) Binary Number to Decimal Number

The binary number system has base 2. The positional values are computed in terms of powers of 2. Using the above-mentioned steps let us convert a binary number to its equivalent decimal value as shown below:

Example 2.8. Convert $(1101)_2$ into decimal number.

Digit	1	1	0	1
Position Number	3	2	1	0

Positional Value	2^3	2^2	2^1	2^0
Decimal Number	1×2^3	1×2^2	0×2^1	1×2^0

$$= 8 + 4 + 0 + 1 = (13)_{10}$$

Note: Add the product of positional value and corresponding digit to get decimal number.

(b) Octal Number to Decimal Number

The octal number system has base 8. The positional values are computed in terms of powers of 8. The following example shows how to compute the decimal equivalent of an octal number using base value 8.

Example 2.9. Convert $(3651)_8$ into decimal number.

Digit	3	6	5	1
Position Number	3	2	1	0
Positional Value	8^3	8^2	8^1	8^0
Decimal Number	3×8^3	6×8^2	5×8^1	1×8^0

$$= 3 \times 512 + 6 \times 64 + 5 \times 8 + 1 \times 1$$

$$= 1536 + 384 + 40 + 1$$

$$= (1961)_{10}$$

(c) Hexadecimal Number to Decimal Number

For converting a hexadecimal number into decimal number, use steps given in this section with base value 16 of hexadecimal number system. Use the appropriate decimal value equivalent to alphabet symbol of hexadecimal number in the calculation.

Example 2.10. Convert $(4A5)_{16}$ into decimal number.

Digit	4	A	5
Position Number	2	1	0
Positional Value	16^2	16^1	16^0
Decimal Number	4×16^2	10×16^1	5×16^0

$$= 4 \times 256 + 10 \times 16 + 5 \times 1$$

$$= 1024 + 160 + 5$$

$$= (1189)_{10}$$

Conversion from Binary Number to Octal/Hexadecimal Number and Vice-Versa

A binary number is converted to octal or hexadecimal number by making groups of 3 and 4 bits, respectively, and replacing each group by its equivalent octal/ hexadecimal digit.

(a) Binary Number to Octal Number

Given a binary number, an equivalent octal number represented by 3 bits is computed by grouping 3 bits from right to left and replacing each 3-bit group by the corresponding octal digit. In case number of bits in a binary number is not multiple of 3, then add required number of 0s on most significant position of the binary number.

Example 2.11. Convert $(10101100)_2$ to octal number.

Make group of 3-bits of the given binary number (right to left)	<u>010</u> <u>101</u> <u>100</u>
Write octal number for each 3-bit group	2 5 4

Therefore, $(10101100)_2 = (254)_8$

Why 3 bits in a binary number are grouped together to get octal number?

The base value of octal number system is 8. Convert value 8 in terms of exponent of 2, i.e., $8=2^3$. Hence, three binary digits are sufficient to represent all 8 octal digits. Simply stated, count all possible combinations of three binary digits, which are $2 \times 2 \times 2 = 8$. Therefore, 3 bits are sufficient to represent any octal digit. Hence, 3-bit groups in a binary number are formed to get equivalent octal number.

(b) Octal Number to Binary Number

Each octal digit is an encoding for a 3-digit binary number. Octal number is converted to binary by replacing each octal digit by a group of three binary digits.

Example 2.12. Convert $(675)_8$ to binary number.

Octal digits	6	7	5
Write 3-bits binary value for each digit	110	111	101

Therefore, $(675)_8 = (110111101)_2$

(c) Binary Number to Hexadecimal Number

For a given a binary number, its equivalent hexadecimal number is computed by making a group of 4 binary digits from right to left and substituting each 4-bit group by its corresponding hexadecimal alphanumeric symbol. If required, add 0 bit on the most significant position of the binary number to have number of bits in a binary number as multiple of 4.

Example 2.13. Convert $(0110101100)_2$ to hexadecimal number.

Make group of 4-bits of the given binary number (right to left)	0001 1010 1100
Write hexadecimal symbol for each group	1 A C

Therefore, $(0110101100)_2 = (1AC)_{16}$

Why 4 bits in a binary number are grouped together to get hexadecimal number?

The base value of hexadecimal number system is 16. Write value 16 in terms of exponent of 2 i.e. $16 = 2^4$. Hence, four binary digits are sufficient to represent all 16 hexadecimal symbols.

(c) Hexadecimal Number to Binary Number

Each hexadecimal symbol is an encoding for a 4-digit binary number. Hence, the binary equivalent of a hexadecimal number is obtained by substituting 4-bit binary equivalent of each hexadecimal digit and combining them together (see Table 2.6). *Example 2.14.* Convert $(23D)_{16}$ to binary number.

Hexadecimal digits	2	3	D
Write 4-bit binary value for each digit	0010	0011	1101

Therefore, $(23D)_{16} = (001000111101)_2$

Assignment –

Write binary representation of the following hexadecimal numbers.

- (i) $(F018)_{16}$
- (ii) $(172)_{16}$
- (iii) $(613)_8$

Binary Addition

The addition of two binary numbers can be performed like addition of decimal number. In binary system since there are only two numbers 0 and 1, only four combinations are possible as shown in table below.

Addend1	Addend2	Sum	Carry
---------	---------	-----	-------

0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Example 2.23, Add the two numbers 9 and 5

Deci- mal		Binary	
9		1 0 0 1	
+ 5	+	1 0 1	
<hr/>			
14		1 1 1 0	

Example 2.24, Add 75 and 18 in binary number.

Solution – Binary equivalent of $(75)_{10} = 64+8+2+1 = (1001011)_2$

Binary equivalent of $(18)_{10} = 16+2 = (10010)_2$

Carry →		1	
Addend →			
1	1 0 0 1 0 1 1		
Addend →			
2	1 0 0 1 0		
<hr/>			
Sum →	1 0 1 1 1 0 1		

Example 2.25, Add binary number 1011.011 and 1101.111

Solution –

Carry →	1 1 1 1 1 1	
Addend 1 →	1 0 1 1 . 0 1 1	
Addend 2 →	1 1 0 1 . 1 1 1	
<hr/>		
Sum →	1 1 0 0 1 . 0 1 0	

Assignment –

- 1) Add 10101 and 11011
- 2) Add 1011101 and 1100111

Binary Subtraction

Binary subtraction can also be performed like in the decimal system. First determine whether it is necessary to borrow. When we subtract 1 from 0, it is necessary to borrow 1 from the next left column i.e. from the next higher order position. If the subtrahend (the lower digit) is larger than the minuend (the upper digit), it is necessary to borrow from the column to the left. In binary two is borrowed. Subtract the lower value from the upper value. The possible combinations of subtraction two binary numbers are as shown in table below.

Minuend	Subtrahend	Difference	Barrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Example 2.26, Subtract the number 10 from 14

Decimal		Binary
14		1 1 1 0
- 10	-	1 0 1 0
4		0 1 0 0

Example 2.27. Subtract 18 from 75 in binary number.

Borrow	→	1				
Minuend	→	1	0	0	1	0
Subtrahend	→			1	0	0
Difference	→	0	1	1	1	0

Assignment

- 1) Subtract 01110 from 10101
- 2) Subtract 25 from 35

Check Your Progress






A. Multiple choice questions

1. Which of the following is not represented as data (a) image (b) audio sound (c) video file (d) drama (d)
2. Which of the following is not a data encoding system (a) ASCII (b) EBCDIC (c) Number system (d) Unicode (c)
3. Which of the following code is used by computer (a) Decimal (b) Binary (c) Octal (d) Hex (b)
4. Which of the following data take largest amount of space in memory (a) picture file (b) audio file (c) video file (d) text file (c)
5. Data is processed by (a) Keyboard (b) Memory (c) Processor (d) Monitor (c)
6. Which of the following is fixed length Unicode system (a) UTF-8 (b) UTF-16 (c) UTF-32 (d) UTF-64 (c)
7. Which of the following encoding system has uses highest number of bits (a) BCD) (b) ASCII (c) EBCDIC (d) Unicode (d)
8. Which of the following number system is used to represent color codes (a) binary (b) octal (c) hexadecimal (d) decimal (c)
9. How many bits are used by the Unicode encoding system (a) 8 (b) 16 (c) 32 (d) All of the above (c)
10. Which of the following color uses the same code as 0 to represent a color code in any number system (a) Black (b) White (c) Yellow (d) Grey (a)

B. Fill in the blanks

1. The data is stored in computer in the form of _____.
2. The smallest unit of data is _____.
3. The process of converting data into machine code is called as _____.
4. In positional number system a numeric value is represented through increasing power of _____.
5. How many bits are required to represent a color code ____.
6. The _____ number system also uses alphabets to represent its symbol.
7. Provide the hexadecimal value of the corresponding color codes in the following table

Name of the Colour	Colour	Hexadecimal Value	R	G	B

Dark purple		#871F78	135	31	120
Brick			255	64	0
Sky blue			46	92	153
Green			0	255	0
Yellow			255	238	0

C. True and False

1. Data means facts or set of values.
2. Useful knowledge about data can be obtained without processing it.
3. Data cannot be measured and analysed.
4. Data can be visualised.
5. Data is converted into machine code after processing.
6. BCD system used 4 bits to represent a digit.
7. Only 10 digits can be represented by BCD system.
8. Pie chart, bar graphs and line graph can be used to visualize data.
9. Data generated by the electronic devices such as mobile phone is called as analog data.
10. ASCII code of capital and small alphabets are the same.
11. ASCII encoding system uses 7 bit or 8 bits to represent a character.
12. EBCDIC coding system uses 8 bits to represent 256 characters.
13. All the encoding system have the same code.
14. All the encoding system communicate with each other.
15. It is possible to convert a number from any number system to any other number system.

D. Short Answer Question

1. What is data and information? Explain with example.
2. Classify the following data and information (a) Roll Number (b) Mobile Number (c) School Attendance Record (d) School Report Card (e) Digits (f) Alphabets (g) Video recorded by smartphone (h) Height (I) Weight
3. Write the acronym for the following (i) ASCII (ii) BCD (iii) EBCDIC
4. What is difference between positional and non-positional number system
5. What do you mean by base or radix of number system. What is the base values of binary, octal and hexadecimal number system?
6. Convert the following binary numbers to its decimal, octal and hexadecimal equivalent (i) 110101 (ii) 1011000 (iii) 1110.100 (iv) 101110.1011
7. Convert the following decimal numbers to its equivalent binary, Octal and Hexadecimal (i) 178 (ii) 6253 (iii) 257.25 (iv) 359.125
8. Convert the following octal numbers to its equivalent binary, Decimal and Hexadecimal (i) 531 (ii) 756 (iii) 376.64 (iv) 705.125
9. Convert the following hexadecimal numbers to its equivalent binary, Decimal and Octal (i) BA1 (ii) 79C.D (iii) DAC.34F (iv) DAF.12B
10. Convert the following decimal number to other number systems.

(i) $(54)_{10} = (?)_2$	(ii) $(120)_{10} = (?)_2$	(iii) $(76)_{10} = (?)_8$
(iv) $(889)_{10} = (?)_8$	(v) $(789)_{10} = (?)_{16}$	(vi) $(108)_{10} = (?)_{16}$

Session 3: Basic Concepts of Mathematics and Statistics

Rahul wants to go to Mumbai to visit his relatives. For train ticket reservation, he went to reservation center to book a ticket. He saw that every employee in the reservation center are using the computer to book the ticket, cancel the ticket or to provide the information about the train number and station. He noticed that all the information is being accessed through computer by the railway reservation officials. Till this time, it is confirmed that nothing can be done without processing data using computer. (Figure 3.1) When he fills the form to book the ticket, he observed that all the data provided by him was entered and processed by the computer system only. He then understands the power of computing and decides to understand how such processing is possible with computer machines.



Fig. 3.1: Graphical illustration of railway reservation window

Computer is a digital electronic machine in which the digital data is processed. Digital data means the discrete data, i.e. the data which can be measured in whole numbers only. For example, number of tables, number of chairs, number of fingers are the examples of discrete data which are counted in whole numbers only. Most of the time the real-life data is analogues data i.e. continuous data. Temperature, pressure, volume are the examples of analogues data. Computer cannot process any analogue data. Such data need to be converted into discrete form and then only it can be processed by computer machine.

Discrete mathematics is a branch of mathematics in which it is possible to perform various mathematical operations on the discrete data. These operations are very much needed when we write programs to perform a specific function or a task. In this session you will understand the different concepts of discrete mathematics such as sets, relations, functions, mathematical logic group theory and graph theory. Also, we will describe the statistical methods such as Mean, Mode, Standard Deviation and Variance. These methods are required for the analysis of the discrete data. Figure 3.2 shows the use of mathematics in Computer Science.

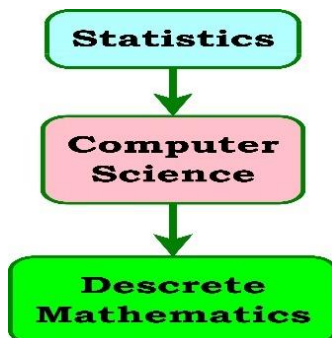


Fig. 3.2: Use of Mathematics in Computer Science

Set Theory

Whenever we want to write a computer program to solve a problem then it is necessary to make the use of logical reasoning to solve the problem. Set theory is an important tool for the logical reasoning. A set is defined as a well-defined collection of objects or things. For example, in the set $A = \{2,4,6,8,10\}$, A is the name of the set and its elements are 2,4,6,8,10. Observe that the set elements are enclosed into braces ($\{ \}$) and they can be numbers, names or symbols. If there are finite number of elements in the set then the set is called as finite set and if there are infinite number of elements in the set then the set is being called as infinite set. The only care that should be taken while writing the set elements that the same element should not be repeated in the set. Set is a very important tool in the programming because it allows us to group the data and describe the data.

<https://www.youtube.com/watch?v=YDGrF2AU3tQ>

Video link on set theory

In the set when the element x is member of the set then it is written as $x \in A$ and when the element x is not member of the set then it is written as $x \notin A$.

If the set consists of finite number of elements, then it is finite set. For example, $A = \{1,3,5,7,9\}$ is a finite set. If the set consists of infinite number of elements, then it is infinite set. For example, $A = \{x \in \mathbb{N}, x > 1\}$ an infinite set. If the number of the elements in the two sets are the same then they are called as equal sets. A set can be called as a universal set if it is a collection of all elements in particular context. For example, the set of all animals on the earth is being called as universal set.

If all the elements of the set Y are in the set X , then the set Y is called as a subset of X and X is called as a super-set of Y .

Set operations

Various operations can be performed on the set, such as intersection, union, difference and symmetric difference.

Union – If there are two sets A , B then the union of these two sets ($A \cup B$) is the set of all the values, that are the member of A or B or both.

For example, Let set $A = \{1,2,4,8\}$ and $B = \{2,5\}$ then the $(A \cup B) = \{1,2,4,5,8\}$

Intersection – If there are two sets A , B then the intersection of these two sets ($A \cap B$) is the set of all the values, that are the member of both A and B .

For example, Let set $A = \{1,2,4,8\}$ and $B = \{2,5\}$ then then $(A \cap B) = \{2\}$

Difference – If there are two sets A , B then difference $A \setminus B$ is the set of all the values of A that are not members of B .

For example, Let set $A = \{1,2,4,8\}$ and $B = \{2,5\}$ then the $A \setminus B = \{1,4,8\}$

$A \setminus B$ is not same as $B \setminus A$.

Symmetric Difference – If A and B are the two sets ($A \Delta B$) is the set of all values, which are in one of the sets, but not in both.

For example, Let set $A = \{1,2,4,8\}$ and $B = \{2,5\}$ then the $(A \Delta B) = \{1,4,5,8\}$

The set operations can also be shown by the graphical diagrams and these diagrams are called as Venn diagrams. The Venn diagrams of these set operations are shown in Figure 3.3.

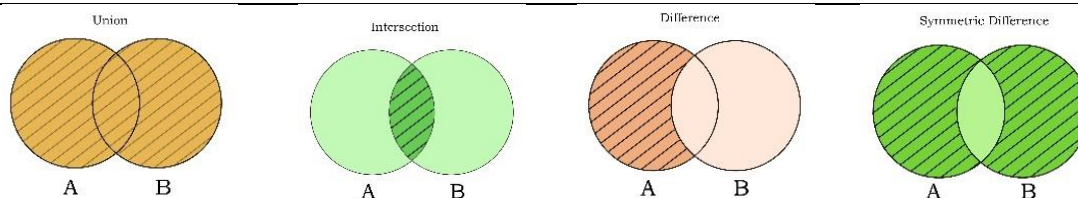


Fig. 3.3 (a) Union Fig. 3.3 (b) Intersection Fig. 3.3 (c) Difference Fig. 3.3 (d) Symmetric Difference

Activities

Practical Activity 3.1. Compute the union and intersection of sets

Suppose set $A = \{1,2,4,18\}$ and $B = \{x : x \text{ is an integer, } 0 < x \leq 5\}$, then compute $(A \cup B)$ and $(A \cap B)$

Solution:

$$(A \cup B) = \{1,2,3,4,5,18\}$$

$$(A \cap B) = \{1,2,4\}$$

Practical Activity 3.2. Compute the union of set

Suppose Set $A = \{1,2,4,18\}$ and $B = \{x \in \mathbb{Z} : 0 < x \leq 5\}$

then compute, $A \setminus B$, $B \setminus A$ and $A \Delta B$

Solution:

$$A \setminus B = \{18\}$$

$$B \setminus A = \{3,5\}$$

$$= \{3,5,18\}$$

Practical Activity 3.3. Draw the Venn diagrams for (i) $B \setminus A$ (ii) $(A \cap B)$

Solution:

(i) $B \setminus A$ Everything in B that is not in A

(ii) $(A \cap B)$: Everything in B that is not in B

Relation and Function

In our daily lives we come across several relations, such as brother and sister, father and so on, teacher and student, and so on. That is, every two human beings can have some kind of relationship among them. In discrete mathematics it is possible to define a relation between the two sets.

If A, B are the two sets, then the relation between A and B is provided through a Cartesian product $A \times B$. The relation between the two sets is used to define the connection between the two sets. A relation is just nothing but a set of ordered pairs. Relations are very useful in computers because they can be used for describing computer languages and compiler grammar. They are also used in data structures such as the traversal of the graph.

Cartesian product

If A, B are the two sets then the Cartesian product $A \times B$ is the set of all ordered pairs of the elements of A and B.

For example, if the set $A = \{1,2\}$ and $B = \{3,4,5\}$ then the Cartesian product $A \times B = \{(1,3), (1,4), (1,5), (2,3), (2,4), (2,5)\}$

That is $A \times B = \{(a, b): a \in A \text{ and } b \in B\}$ and $B \times A = \{(b, a): b \in B \text{ and } a \in A\}$

Observe that $A \times B$ is not equal to $B \times A$.

Cartesian Product is very important tool by using which we can obtain one set from the two sets. This tool is very widely used in matrix applications.

Relation

If A and B are the two sets then the relation R from the set A to B is a subset of $A \times B$.

Suppose the set $A = \{2,4,5,6,7\}$ and set $B = \{2,3\}$ then the relation R from A to B is defined by $R = \{(a, b): a \in A, b \in B \text{ and } a \text{ is divisible by } b\}$. Such relation $R = \{(2,2), (4,2), (6,2), (6,3)\}$.

If the relation $R = \emptyset$, i.e. empty set then the relation R is called as void relation. If the relation $R = A \times B$, then the relation R is called as universal relation.

If the relation R is between the two sets A and B then the set of the first elements of all ordered pairs of R is called as domain and the set of second elements of all ordered pairs of R is called as range.

For example, if $R = \{(2,2), (4,2), (6,2), (6,3)\}$ then the domain of $R = \{2,4,6\}$ then the range of $R = \{2,3\}$.

If the relation R is from set A to B , then the set B is called as co-domain of R . The relation between two sets can be graphically represented by the relation diagram. Figure 3.4 shows the relation R .

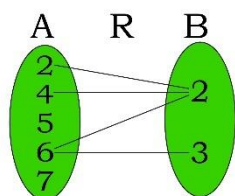


Fig. 3.4 Relation R

https://www.youtube.com/watch?v=0N_IbBifuMk

Video link on Relation and Function

Function

The relation f from the set A to B , where every element of set A has a unique image in the set B is called as the function from A to B .

For example, if the set $A = \{a, b, c, d\}$ and $B = \{1,2,3,4,5\}$ then the relation $f: \{(a,1), (b,2), (c,3), (d,5)\}$ is a function from A to B .

The function is written as $f: A \rightarrow B$ and graphically shown as in Figure 3.5.

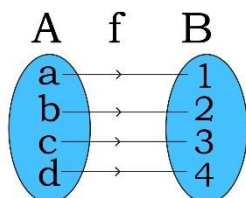
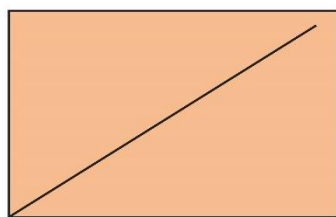


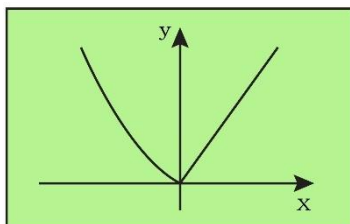
Fig. 3.5 Function

A function can be a variety of types such as identity functions, constant functions, quadratic function and polynomial function as shown in Figure 3.6.



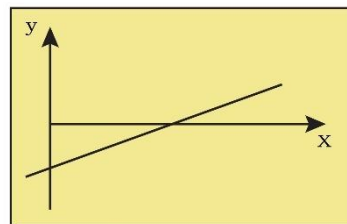
$$f(a) = b = a$$

(a) Identity function



$$y = x^2$$

(b) Quadratic function



$$ax^2 + bx + c = 0$$

(c) Polynomial function

Fig. 3.6 Types of function (a) Identity function (b) Quadratic function (c) Polynomial function

Mathematical Logic

In our daily lives, we always make the use of the logic to perform any action. In mathematics the logic is used to differentiate between valid and invalid arguments. An argument is a set of statements consisting of two parts. (1) set of premises and (2) a conclusion.

For example, if the set of premises consists of two statements such as

Statement 1: If today is Monday then Mr. A gets Rs. 50.

Statement 2: Today is Monday.

Conclusion: Mr. A gets Rs. 50.

These arguments are called as valid arguments and such validity can be checked by using logic.

A simple statement or a proposition is an expression that is either false or true, but not both. That is, every statement will have a logical value of True (T) and False (F).

True (T) and False (F) are called as the truth values of the statement.

When the simple statements are connected together by making the use of the words such as 'Not', 'And', 'Or' 'Implies (if then)', and 'If and only if' then a complex or compound statement is formed. These connecting words Not, Or, implies are called as logical connectives.

<https://www.youtube.com/watch?v=Slab1cep2Bo>

Video link of Mathematical Logic

Table 3.1 shows the different logical connectives along with their symbol and meaning.

Table 3.1 Logical Connectives

Name	Symbol	Connection	Meaning
AND (Conjunction)	\wedge	$P \wedge Q$	$P \wedge Q$ is T if both P and Q are T otherwise F.
OR (Disjunction)	\vee	$P \vee Q$	$P \vee Q$ is T if either P or Q is T or both are T otherwise F.
NOT (Negation)	\neg	$\neg P$	$\neg P$ is the opposite of P. If P is true then negation of P is F and vice versa.
Exclusive OR	\oplus	$P \oplus Q$	Either P or Q but not both if P and Q are different then $P \oplus Q$ will be T otherwise F.
Implication (If then)	\rightarrow	$P \rightarrow Q$	If P happens then Q happens
Double Implication (if and only if)	\leftrightarrow	$P \leftrightarrow Q$	P happens if and only if Q happens

Truth Table

A Truth Table is a table that shows all possible Truth values (T) for a compound statement. The Truth Table for the different logical connectives are as given in the following tables.

p	$\sim p$
---	----------

T	F
F	T

Table 3.2 Truth Table for Negation

p	q	$p \wedge q$
T	T	T
T	F	F
F	T	F
F	F	F

Table 3.3 Truth Table for AND (conjunction)

p	q	$p \vee q$
T	T	T
T	F	T
F	T	T
F	F	F

Table 3.4 Truth Table for OR (conjunction)

p	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

Table 3.5 Truth Table for if then (Implication)

p	q	$p \leftrightarrow q$
T	T	T
T	F	F
F	T	F
F	F	T

Table 3.6 Truth Table for if and only if (Double implication)

p	q	$p \oplus q$
T	T	F
T	F	T
F	T	T
F	F	F

Table 3.7 Truth Table Exclusive OR

Practical Activity 3.4. Prove that $p \vee (p \wedge q) \equiv p$

Solution: Identity can be proved by constructing the truth table as given below.

p	q	$p \wedge q$	$p \vee (p \wedge q)$
T	T	T	T
T	F	F	T
F	T	F	F
F	F	F	F

Table 3.8

Observe that all entries in column 1 and 4 are same hence proof of identity.

Logical Equivalence

The two propositions A and B are said to be logically equivalent if they have the same truth value under any truth assignment. When A and B are logically equivalent then it is written as $A = B$.

Logical equivalence of the proposition can be proved by construction the truth table.

For example, $P \rightarrow Q \equiv \neg Q \rightarrow \neg P$

This logical equivalence can be proved by constructing a truth table as given Table 3.9.

p	q	$p \rightarrow q$	$\neg q$	$\neg p$	$\neg q \rightarrow \neg p$
T	T	T	F	F	T
T	F	F	T	F	F
F	T	T	F	T	T
F	F	T	T	T	T

Table 3.9 Truth Table for $\neg q \rightarrow \neg p$

From the above table it is observed that all the entries in column 3 and 6 are same hence the proof of the proposition.

A proposition or a formula that has the truth value T for each assignment is called as *Tautology*.

A proposition or a formula that has the truth value F for each assignment is called as *Contradiction*.

A proposition or a formula that is neither a tautology nor a contradiction is called as *Contingency*.

Table 3.10 shows the Truth Table of *Tautology Contradiction Contingency*.

p	$\neg p$	$p \vee \neg p$	$p \wedge \neg p$
T	F	T	F
F	T	T	F
<i>Contingency</i>		<i>Tautology</i>	<i>Contradiction</i>

Table 3.10 Truth Table of *Tautology Contradiction Contingency*

GRAPH THEORY

A Graph is a non-linear data that consists of nodes and edges. Nodes are called as the vertices and the edges are called as the lines or arcs. A graph $G = (VE)$, where, V is the set of vertices and E is the set of edges. For example, if the set of the vertices $V = \{0,1,2,3,4\}$ and the set of edges $E = \{01, 12, 23, 34, 04, 14, 13\}$. Such a graph can be graphically represented as shown in the Figure 3.7.

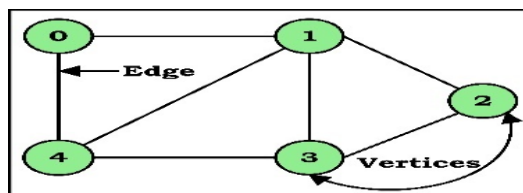


Fig. 3.7 Graph

<https://www.youtube.com/watch?v=Md-ZEzbN-tA>

Video link on Graph Theory

Graphs are very useful to solve many real-life problems. They are used to represent the network such as computer network, circuit network, and telecommunication network. Graphs are used in the social networks such as LinkedIn, Facebook and WhatsApp where every person is represented by a vertex or a node. Figure 3.8 shows the social network.

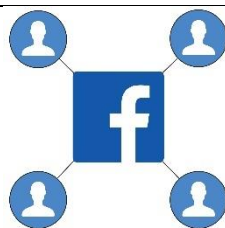


Fig. 3.8 Facebook social network

Each node is a structure that contains the information such as identity ID, name, gender and address. Google map also makes the use of graph to find the shortest distance between the two locations, where each location is represented by a vertex. Figure 3.9 shows representation of Google map.

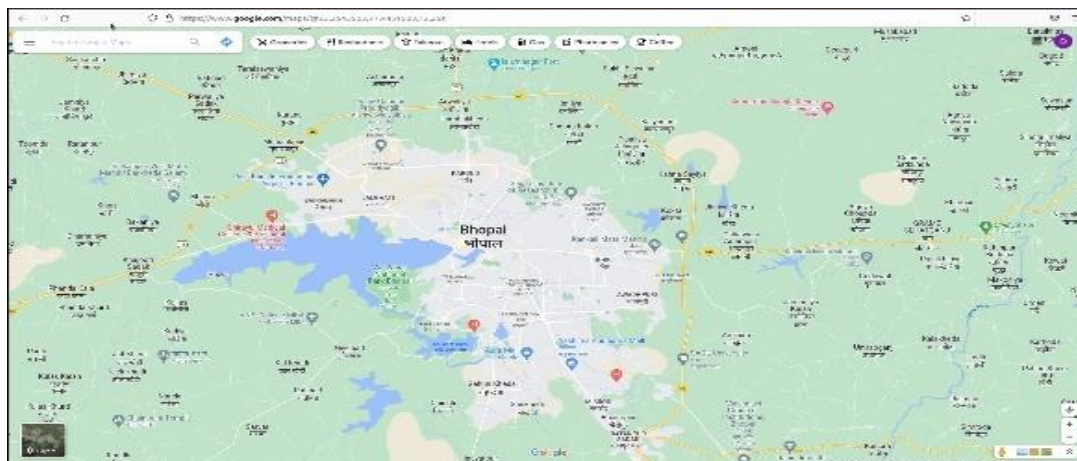


Fig. 3.9 Representation of Google map

Adjacency Matrix

A graph can be represented by the adjacency matrix. The adjacency matrix A for the graph $G = (V, E)$ with n vertices is a $n \times n$ matrix such that $A_{ij} = 1$, if there is an edge from V_i to V_j and $A_{ij} = 0$, if there is no edge.

For example, consider the graph as shown in the Figure 3.10. For this graph, we can obtain the adjacency matrix as given below.

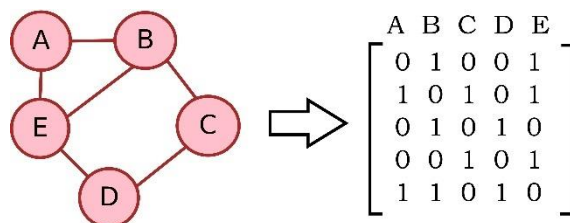


Fig. 3.10 Graph and adjacency matrix

BOOLEAN ALGEBRA

<https://www.youtube.com/watch?v=nIFj-EkVKnw> Video link on Boolean Algebra

In nineteenth century, English mathematician, George Bull, has written the “laws of thoughts”. These *laws of thoughts* are being recognized later as Boolean Algebra. Boolean Algebra are the rules for manipulation of binary values. These rules form the basis of modern digital electronics from where the computers are evolved.

A Boolean variable is a binary variable that can have either a 0 or 1 value. Boolean operations are the logic operations, such as NOT, AND, and OR. Boolean expression is a logic expression that contains Boolean variable and Boolean operations. Boolean Algebra contains the various rules such as AND laws OR laws, NOT laws, Associative laws and Distributive laws.

AND laws – AND laws are the laws of logic AND operation. There are four AND laws.

Law 1 – $A \cdot 0 = 0$

Law 2 – $A \cdot 1 = A$

Law 3 – $A \cdot A = A$

Law 4 – $A \cdot \bar{A} = 0$

OR laws – OR laws are the laws of logic OR operation. There are four OR laws.

Law 1 – $A + 0 = A$

Law 2 – $A + 1 = 1$

Law 3 – $A + A = A$

Law 4 – $A + \bar{A} = 1$

NOT laws – NOT laws are the laws of logic NOT operation. These laws are also called as laws of complementation. There are three NOT laws.

Law 1 – $\bar{\bar{A}} = A$

Law 2 – $\overline{A \cdot B} = \bar{A} + \bar{B}$

Law 3 – $\overline{\bar{A}} = A$

Commutative laws – These laws allow us to change the position of AND and OR variables. There are two Commutative laws.

Law 1 – $A + B = B + A$

Law 2 – $A \cdot B = B \cdot A$

Associative laws – These laws allow us to change the grouping of the variables. There are two Associative laws.

Law 1 – $A + (B + C) = (A + B) + C$

Law 2 – $A \cdot (B \cdot C) = (A \cdot B) \cdot C$

All of the above laws of the Boolean Algebra can be proved by constructing the Truth Table for left hand side and right-hand side of the identity.

Distributive laws – These laws tell us that the factoring or multiplying of different terms in an expression is allowed. There are three Distributive laws.

Law 1 – $A \cdot (B + C) = A \cdot B + A \cdot C$

Law 2 – $A + (B \cdot C) = (A + B) \cdot (A + C)$

Law 3 – $A + \bar{A} \cdot B = A + B$

The Distributive laws can be proved either by constructing the Truth Table or by simplifying the equation by using other laws of Boolean Algebra.

Activity 5

Practical Activity 3.5. Prove following law of Boolean algebra

(a) $A + B = B + A$

(b) $A + \bar{A} \cdot B = A + B$

Solution:

(a) $A + B = B + A$

This law can be proved by constructing truth table as given below

A	B	A + B	B + A
0	0	0	0

0	1	1	1
1	0	1	1
1	1	1	1

Observe that all entries in column 3 and 4 are same, hence proof.

$$(b) A + \bar{A}B = A + B$$

$$\begin{aligned} \text{LHS} &= A + \bar{A}B \\ &= A.1 + \bar{A}B && \text{..... AND Law} \\ &= A. AB + \bar{A}B && \text{..... OR Law} \\ &= A + AB + \bar{A}B && \text{..... Distributive Law} \\ &= A + (A + \bar{A})B && \text{..... Distributive Law} \\ &= A + 1.B && \text{..... OR Law} \\ &= A + B && \text{..... AND Law} \\ &= \text{RHS} \end{aligned}$$

LHS= RHS, hence proof the law.

LHS = RHS, Hence proof of the law

STATISTICAL METHODS

Statistical methods are the mathematical formulae, models and techniques that are used in the analysis of the data. Many times, we need to collect a lot of data related with any given problem. But it becomes very difficult to draw a useful conclusion simply by observing the data. In such a case the statistical methods are useful to draw a fruitful conclusion from the large collection of the data.

There are different terminologies that are being used in the statistics. In this session we will understand some of these terminologies.

1. Measure of central tendency – when we have been provided with a large amount of data, many times we need to measure the central tendency of the data. The central tendency of the data can be measured by the parameters such as *Mean, Median and Mode*.(Figure 3.11)

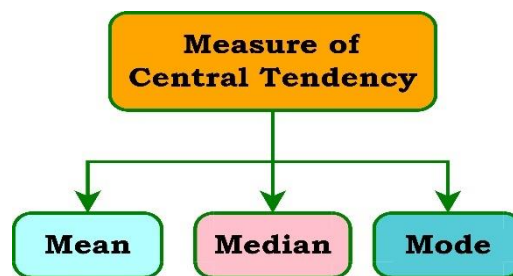


Fig. 3.11 Measure the central tendency

Mean – It is defined as the average value of the data set. If there are n data values given to you then the mean can be computed by using the formula.

$$\bar{X} = (x_1 + x_2 + x_3 + \dots + x_n) / n =$$

For example, the given data set is 1, 4, 4, 6, 10 then its mean is

$$\bar{X} = (1+4+4+6+10)/5 = 25/5 = 5$$

Median – Median of the given data set is the middle number that split the data set into two halves.

Steps to find the Median.

Step 1. Arrange the data in the increasing order.

- Step 2. Determine number of data values in the data set which is equal to n .
 Step 3. If n is odd then median is the middle number.
 Step 4. If n is even, then the median is the average of the two middle numbers.

Activity 6

Practical Activity 3.6. Compute Mean, Median and Mode for the given data set

Given data set is 34, 22, 15, 25, 10

Step 1. The increasing order of the data is 10, 15, 22, 25, 34

Step 2. There are five numbers in the data set i.e., $n = 5$.

Step 3. Therefore, the median is equal to the median is middle number 22.

Given data set 19, 34, 22, 15, 25, 10

Step 1. The increasing order of the data is 10, 15, 19, 22, 25, 34

Step 2. There are six numbers in the data set i.e., $n = 6$.

Step 3. Therefore, there are two middle numbers 19 and 22. Therefore median is equal to $(19+22)/2=20.5$

Observe that mean and median numbers are not from the data set. Mean and median can have only one value. Mean is affected by the extreme values while the median is resistant.

Mode – The most frequent number in the data set is called as mode.

Example, suppose we have been provided with the data set 19, 19, 34, 3, 22, 10, 15, 25, 10, 6. In this data set the number that occurs the most is 10, and therefore, Mode is equal to 10.

If the two different numbers appear same number of times in the given data set, then both of them are called as the mode of the given data set.

Observe that mode is always the number from the data set. It is also possible that mode can have 0 or 1 or more than one value and it is being called as 0 Mode, 1 Mode, 2 Mode, and so on.

The relationship between Mean, Median and Mode can be graphically represented in Figure 3.12.

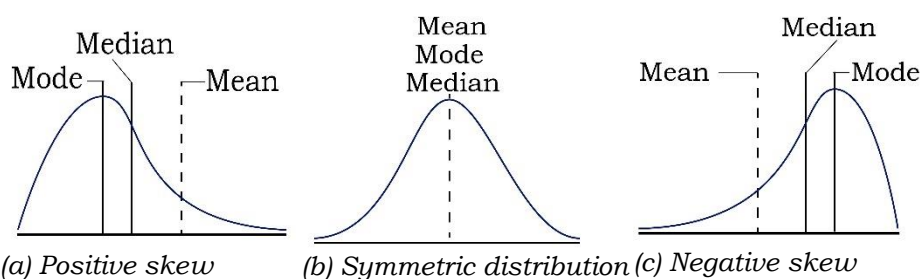


Fig. 3.13 Graphical representation of Mean, Median and Mode

2. Measurement of Deviation – Deviation is defined as the difference between the value x and the population mean (μ).

$$\text{Deviation} = x - \mu$$

Deviation can be measured by the parameters called as variance and standard deviation.

Variance – It is a measure of how far the values of the data set are from the mean, on average. The average of the squared deviation is the population variance.

$$\text{Population variance } \sigma^2 =$$

$$\text{Sample variance } S^2 =$$

Standard deviation – The square root of the variance is called as standard deviation.

The population standard deviation

$$\sigma =$$

Sample standard deviation

$$S =$$

Steps for computing variance and standard deviation.

Step 1. Compute the sample mean \bar{x}

Step 2. Calculate the difference $x_i - \bar{x}$ for each value in the data set.

Step 3. Calculate the squared difference $(x_i - \bar{x})^2$ for each value in the data set.

Step 4. Sum the squared differences $\sum_{i=1}^n (x_i - \bar{x})^2$

Step 5. Divide the sum of squared differences to get the variance S^2

Step 6. Compute the square root of the variance to get the standard deviation S .

Example: Compute the variance and standard deviation of the following data in spreadsheet.

x	$x_i - \bar{x}$	$(x_i - \bar{x})^2$
15	-38	1444
25	-28	784
35	-18	324
45	-8	64
55	2	4
60	7	49
65	12	144
70	17	289
75	22	484
85	32	1024
$\sum x = 530$		$\sum (x_i - \bar{x})^2 = 4610$

Total No of item (N)=10

$$\text{So Mean } (\bar{X}) = \frac{\sum x}{n} = \frac{530}{10} = 53$$

$$\text{Variance } \sigma^2 = \frac{\sum (x_i - \bar{x})^2}{N} = \frac{4610}{10} = 461$$

$$\text{Standard Deviation} = \sqrt{\sigma^2} = \sqrt{461} = 21.47091055$$

Data Visualization

The data mostly appears in the form of numbers or values. Such data is difficult to understand or interpret. The data interpretation can be performed by visualizing the data in the form of charts or graphs. By using the spreadsheet package such as Excel we can visualize the data in the form of bar graph, line graph and pie chart.

Steps to create the charts – Bar chart, Line chart or Pie chart

Step 1. Open the spreadsheet package Microsoft Excel or LibreOffice Calc with a new worksheet.

Step 2. Enter the data in the worksheet at appropriate rows and columns

Step 3. Save the worksheet by proper name.

Step 4. To display the data graphically in the form of bar chart, line chart or pie chart, select the data cells by dragging the mouse.

Step 5. Click on **Insert > Chart** and select the desired chart option from the selection window.

Step 6. Then click on **Finish** button. The data will be displayed graphically as per the selection of bar chart, line chart or pie chart.

Activity 7

Practical Activity 3.7. Represent the given data values graphically in bar chart, line graph and pi chart using spreadsheet software.

Step 1. You have been provided the marks of few students as given below.

Student	Marks
David	74
Hiba	78
Kannan	69
Kushaal	69
Manpreet	62
Pawani	75
Smitha	74
Vibhanshu	81

Draw a bar chart using the spreadsheet software to represent the given data graphically as shown in Figure 3.13.

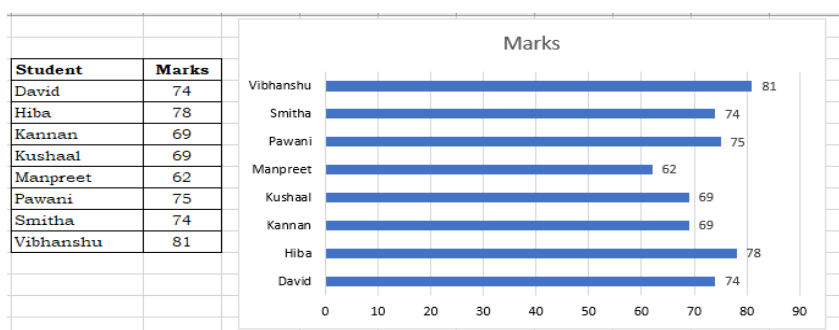


Fig. 3.13: Bar chart drawn for the given data values

Step 2. You have been provided the data of temperature recorded in one week as given below.

Date	Temperature
21/07/21	33.5
22/07/21	34.2
23/07/21	36.4
24/07/21	38.1
25/07/21	35.9
26/07/21	36.8
27/07/21	34.2
28/07/21	39.5
29/07/21	34.8
30/07/21	36.5
31/07/21	37.5

Draw a line chart using the spreadsheet software to represent the given data graphically as shown in Figure 3.14.

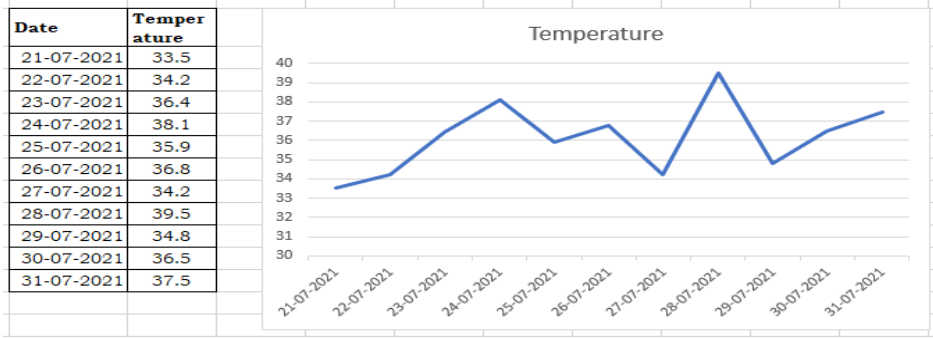


Fig. 3.14: Line chart drawn for the given data values

Step 3. You have been provided the data of temperature recorded in one week as given below.

Company	% in market share
HP	22
Dell	33
Lenovo	13
Asus	15
Acer	17

Draw a pie chart using the spreadsheet software to represent the given data graphically as shown in Figure 3.15.

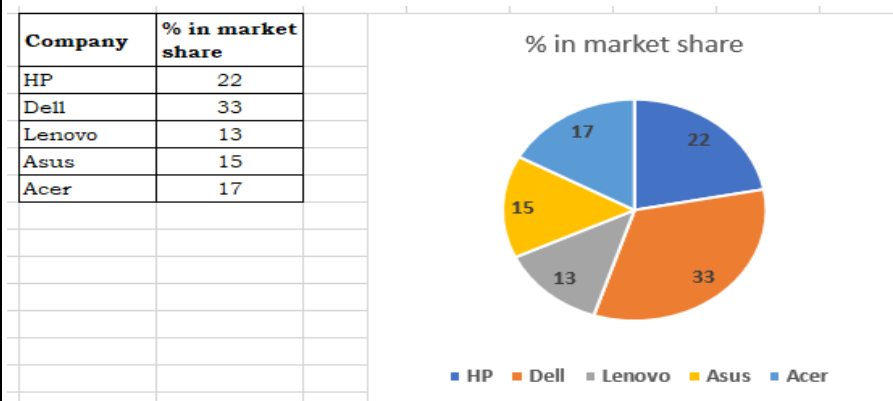


Fig. 3.15: Pie-chart drawn for the given data values

Step 3. You have been provided the data of temperature recorded in one week as given below.

Company	% in market share
HP	22
Dell	33
Lenovo	13
Asus	15
Acer	17

Draw a pie chart using the spreadsheet software to represent the given data graphically as shown in Figure 3.15.

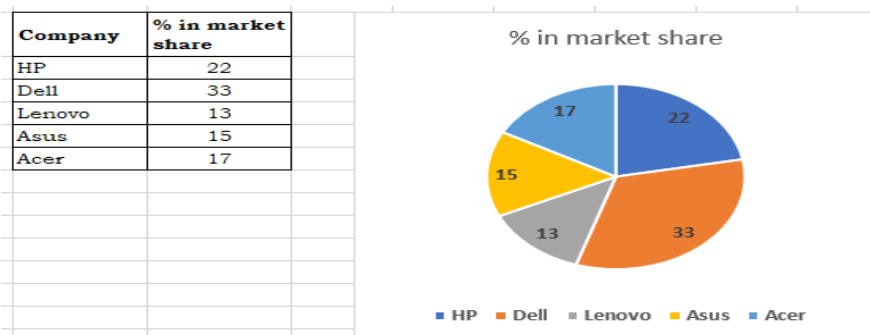


Fig. 3.15: Pie-chart drawn for the given data values

Check Your Progress

A. Multiple choice questions

- Which of the following is not digital data (a) Roll Number (b) Mobile Number (c) Volume (d) Marks (c)
- Which of the following is analogue data (a) Pressure (b) Number of fingers (c) Roll Number (d) Phone number (a)
- Which of the following does not come under discrete mathematics (a) sets (b) relations (c) functions (d) calculations (d)
- Which of the following operations can be performed on the set (a) union (b) intersection (c) symmetric difference (d) all of the above (d)
- Which of the following is used to prove the logical equivalence (a) Truth Table (b) Tautology (c) Contingency (d) Contradiction (a)
- Which of the following is not a boolean operation (a) NOT (b) AND (c) OR (d) ADD (d)
- Symmetric difference of two sets means (a) all the values of one set are not members of other set (b) set of all the values, that are the member of any of the set or both (c) all values which are in one of the sets, but not in both set (d) set of values that are the member of both sets (ans c)
- Union of two sets means (a) all the values of one set are not members of other set (b) set of all the values, that are the member of any of the set or both (c) all values which are in one of the sets, but not in both set (d) set of values that are the member of both sets (ans b)
- Intersection of two sets means (a) all the values of one set are not members of other set (b) set of all the values, that are the member of any of the set or both (c) all values which are in one of the sets, but not in both set (d) set of values that are the member of both sets (ans d)
- Difference of two sets means (a) all the values of one set are not members of other set (b) set of all the values, that are the member of any of the set or both (c) all values which are in one of the sets, but not in both set (d) set of values that are the member of both sets (ans a)

B. Fill in the blanks

- Discrete mathematics performs various _____ operations on _____ data (mathematical, discrete)
- It is necessary to use the _____ to solve the problem. (logical reasoning)
- Symmetric difference is the set of all values which are in one of the set but not in _____ (both)
- In discrete mathematics the relation between two sets is provided through _____ (Cartesian product)
- If the relation is empty set then it is called as _____ relation (void)

6. If there is relation between two sets, then the set of first element of all ordered pairs is called as _____ and set of second element of all ordered pairs is called as _____ (domain, range)
7. The relation between two sets can be graphically represented by the _____ (relation diagram)
8. An argument is a set of statements consisting of _____ and _____ (set of premises, conclusion)
9. A proposition or a formula that has the truth value T for each assignment is called as _____ (Tautology)
10. A proposition or a formula that is neither a tautology nor a contradiction is called as _____ (Contingency)

C. State whether True or False

1. Computer can process only digital data.
2. Infinite set has the limited number of elements.
3. Intersection of two sets contains all the values of two sets.
4. Difference of two sets contains common values of two sets.
5. Union of two sets contains uncommon values of two sets.
6. Cartesian product is used in matrix applications.
7. A logic is used to decide the validity of the argument.
8. The two prepositions A and B are said to be logically equivalent if they have the same truth value under any truth assignment.
9. A proposition or a formula that has the truth value T for each assignment is called as Contradiction.
10. Graph is a non-linear data that consists of nodes and edges.

D. Short answer questions

1. What are the types of functions?
2. Give real life example of mathematical logic.
3. State the application of Graph in real life.
4. What are the different types of Graphs?
5. What is degree of vertex?
6. What are statistical methods?
7. What is mean, mode and standard deviation?
8. What is boolean algebra?
9. State the distributive law.
10. State associative law.

Session 4. Problem Solving Methods

Ram wakes up early in the morning and looked for his mobile phone. As he could not find it, he tried to call on his mobile number from his father's mobile phone. But he still could not locate his mobile, probably the mobile is shut off due to consumption of battery. Thereafter some time he recalled the location of the phone and found that the mobile battery was totally discharged. Thereafter, he recharged the battery of the mobile hand set and tried to take the online session. Then he tried to troubleshoot the problem. He checked his mobile settings, router, also checked for validity of his data plan to resolve the problem. As the data plan had to be recharged, he immediately made the payment and his problem was solved. Thus, we find that in real life whenever we have certain problem then such problem can be resolved by some logical steps. The same has to be followed while solving a given problem using computer. (Figure 4.1)



Fig. 4.1: Illustration of problem of mobile connectivity

4.2 PROBLEM SOLVING

In our real life, many problems occur and we apply the necessary steps to solve the problem. In order to solve the problem, we need to understand the problem first, take the necessary input into consideration and then take the actions to arrive at the solution. In this process you can observe that we require the set of input and processing steps to arrive at the solution. The similar steps are followed in the software development process. A software developer professional should possess this problem-solving skill.

Any real-life problem can be solved by using computer. In order to solve such problem, it is necessary to develop a program for the specific problem. The development process is called as the problem-solving cycle. This problem-solving cycle consists of 5 steps. (Figure 4.2)

1. Analyzing the problems
2. Designing the solution
3. Coding for the solution
4. Testing the solution
5. Documenting the solution



Fig. 4.2: Problem solving cycle

4.2.1. Analysing the problem

It is important to clearly understand a problem before you begin to find the solution for it. If you are not clear as to what is to be solved, you may end up developing a program which may not solve your purpose. Thus, you need to read and analyse the problem statement carefully in order to list the principal components of the problem and decide the core functionalities that your solution should have. By analysing a problem, you would be able to figure out what are the inputs that your program should accept and its probable solution.

This is the first step in the programming process. This step includes.

- (i) Make sure that you have a complete definition of the requirement of the program.
- (ii) Understand the written definition and produce the desire results manually by hand.
- (iii) Define the input facts that are required to produce the desired output.
- (iv) Identify the different sources of those input facts.

(v) Make sure that you have the verified formulae that will be required to produce the desired output.

The Figure 4.3 shows the steps in analyzing the problem.



Fig. 4.3: Problem analysis

In this analysis there are four areas have to be looked into involved. (i) Output (ii) Storage (ii) Input (iv) Process. The order in which the analysis is performed is as given below.

1. The first area to analyze should be the output of the program. This comes from the written definition of the requirements. It consists of the reports or the screens that show the information resulted from program.
2. The second area to look is the storage. The storage requirement can be found from the written definition of the problem. Sometimes there is no output requirement, but always there will be storage requirement.
3. The third area to look into is input. One needs to determine what facts are needed to produce the required information. Also, one has to identify from where the data is coming.
4. The last step in the analysis is the process. Here you need to list the logic that is to be applied to the input data to produce the output information.

4.2.2. Designing the solution

After completion of the analysis of the problem one had a clear understanding of the requirements of the program. In the second step, it is essential to design a solution before writing a program code for a given problem. The solution is represented in natural language and is called an algorithm. There are different strategies available for designing the solution. The easiest way is the modular approach. In the modular approach, the project is divided into small portions called modules, so that one can completely design the logic for each part of the problem. The Figure 4.4 shows the steps in designing the solution of the problem.

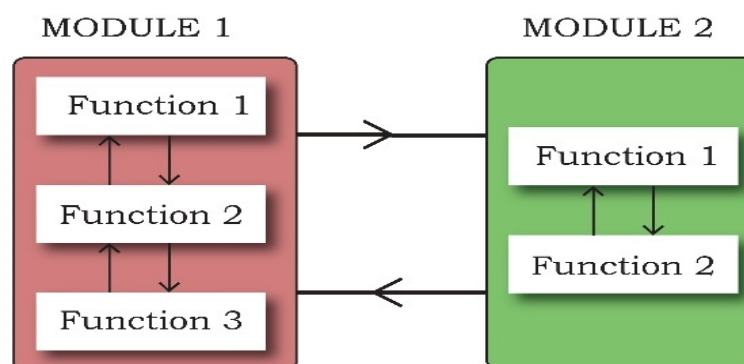


Fig. 4.4: Modular approach for designing the solution of problem

Certain programmers like to design easy part first and then they go for the difficult part. Always remember that all the programs will have the beginning, middle and end. So, some programmers will design the beginning and end first before they go to the middle part of the program.

Many good programmers always start from the middle part, which is real purpose of the program. If you get stuck with your thinking process then you can switch to the easy part of the program for the change of the pace. It will help you to have more thinking about the difficult part of the program and such programmer can find the solution sooner than the others.

The program design tools such as algorithms and flowcharts are used for the appropriate design of the program. The following simple examples depict the process.

Problem 1: Finding the larger number between two numbers

Input: Two numbers

Process: Comparing the two numbers, finding the larger one

Output: Larger number

Problem 2: Finding whether a number is odd or even

Input: Number

Process: Dividing the number by 2

Deciding that the number is even if the remainder = 0

Deciding that the number is odd if the remainder = 1

Output: Indicating whether the number is odd or even

Problem 3: Finding the area of a rectangle

Input: Length and width of the rectangle

Process: Area = Length × Width

Output: Area

4.2.3. Coding for the solution

After designing the solution, it is required to code the algorithm into programming code to generate the desired solution. Different high-level programming languages can be used for writing a program. Coding of the program is the process of translating your program design into appropriate computer programming language that you are working with to solve the problem. The programmer needs to identify the different variables and the data structure that are required in the given program. Program design is the short but the translation of it in a program may be a long process. The length of the program will depend upon the complexity of the problem. Each step of the design needs to be converted into one or more statements of the programming language. While writing such programs the programmer needs to take care of the typing mistakes. It is also necessary that the programs are tested before they are actually completed. Figure 4.5 shows the translation of the design state into statements.



Fig. 4.5: Translation of the design state into statements

4.2.4. Testing and Debugging

Once the coding part is completed, the program should be tested on various parameters. Before testing make sure that the required data files are present in the appropriate drive that you have defined in your program. The program should meet the requirements of the user. It must respond within the expected time. It should generate correct output for all possible inputs.

By using appropriate program environment, you can execute your program. It is very common that your program will not execute correctly in the first attempt only. In the presence of syntactical errors, no output will be obtained. In case the output generated is incorrect, then the program should be checked for logical errors, if any. Make the corrections in the program until it generates required output. Following check points are needed for the corrections in the program.

1. Check for alignment problem with the output information. To fix this problem make the changes in the print statement.
2. Check for the missing information.
3. Check for the zero numeric information.
4. Check for the formulae that are written in the program. i.e. verify computation information.
5. Check for the counter value.
6. If required go back to any step again.
7. Test run the program for the different data sets.

Software industry follows standardised testing methods like unit or component testing, integration testing, system testing, and acceptance testing while developing complex applications. This is to ensure that the software meets all the business and technical requirements and works as expected. The errors or defects found in the testing phases are debugged or rectified and the program is again tested. This continues till all the errors are removed from the program.

Once the software application has been developed, tested and delivered to the user, still problems in terms of functioning can come up and need to be resolved from time to time. The maintenance of the solution, thus, involves fixing the problems faced by the user answering the queries of the user and even serving the request for addition or modification of features.

4.2.5. Documenting the solution

It is equally important to record the details of the coding procedures followed by the software developer and document the solution. This is helpful while revisiting the programs at a later stage. There are two types of documentation required – Programmer documentation and User documentation. Programmer documentation will consist of following items.

1. A copy of the original written specification of the program
2. A written copy of input, output, process and storage analysis.
3. Complete definitions of all files, records and fields used in the program.
4. Mappings of all variable name used in the program to the facts they represent.
5. Copy of the flowchart, pseudo code and other tools that are used in the design of the program.
6. A copy of the source code of the program.
7. A printed copy of the input facts that are used to test the program.
8. A printed copy of all possible output reports or screens.

User documentation is a user manual. Many times, the user manual is written by documentation specialist. For a computer programmer it is very difficult to write a user manual, because they do not know how to communicate with people. The user documentation will consist of the information that is needed to successfully utilize all the capabilities of the program. As shown in Figure 4.6, a user documentation may be in the form of video, a series of steps (Quick Start Guide) – in online or offline mode. Figure 4.6 shows the typical examples of programmer documentation and user documentation.

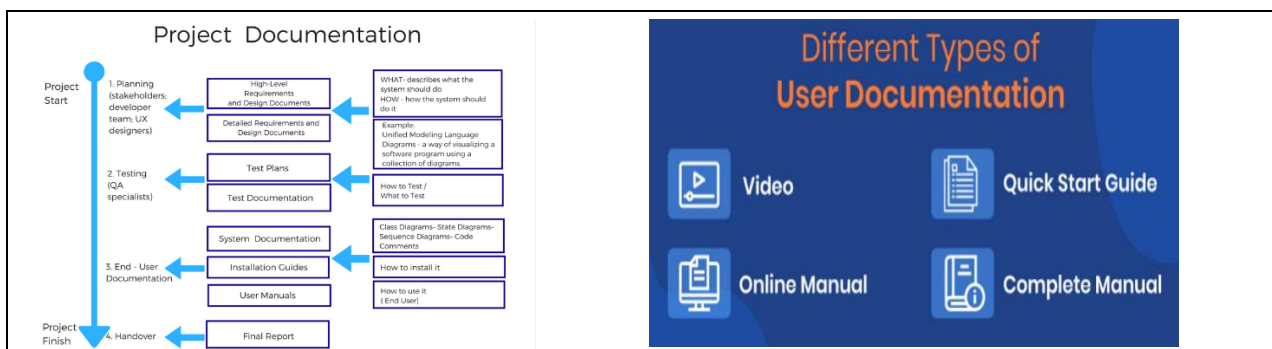


Fig. 4.6: (a) Programmer documentation

Fig. 4.6: (a) Programmer documentation

4.3. ALGORITHMS

In our day-to-day life we perform activities by following certain sequence of steps. Examples of activities include getting ready for school, making breakfast, riding a bicycle, wearing a tie, solving a puzzle and so on. To complete each activity, we follow a sequence of steps.

Let us now consider an example to post a letter. The sequence of steps for this would be:

- Step 1. Writing the letter
- Step 2. Folding the letter
- Step 3. Inserting the letter in an envelope
- Step 4. Writing the address
- Step 5. Sticking the stamp
- Step 6. Posting the letter

Step (1), (2) and (3) in this algorithm should be followed in the given order. Step (4) and (5) can be interchanged.

Thus, there are steps in an algorithm which should be followed in a strict sequential order. Sometimes, if the order of some steps is changed, it does not affect the process and the output is same.

4.3.1. Need of Algorithm

It is required to prepare an algorithm before writing a program. It prepares a road-map of the program to be written, before actually writing the code. It helps the programmer to clearly visualise the instructions to be written. Tasks such as, searching using a search engine, sending a message, finding a word in a document, booking a taxi through an app, performing online banking, playing computer games, all are based on algorithms.

Writing an algorithm is mostly considered as a first step to programming. It is easy to code the program from algorithm in high level programming language. If the algorithm is correct, computer will run the program correctly, every time. So, the purpose of using an algorithm is to increase the reliability, accuracy and efficiency of obtaining solutions.

An algorithm is a set of finite steps which when carried out for given set of initial conditions, produce the corresponding output. In our everyday life we observe that whenever we face certain problem then there exists some method to solve the problem. This method consists of some finite logical steps. Similarly, in order to solve a problem by computer we need to decide the logical steps. Such set of steps is called as algorithm. Every algorithm has the first step as START or BEGIN and the last step as STOP or END.

4.3.2. Characteristics of a good algorithm

There are certain characteristics associated with the algorithm.

1. **Precision** — There should be finite number of steps in algorithm and they are precisely stated or defined. One step should not alter or change the structure of the other steps.

2. **Uniqueness** — Every step must be accurate and complete in itself. The results of each step are uniquely defined and should only depend on the input and/or the result of the preceding steps.
3. **Finiteness** — The algorithm always stops after a finite number of steps. The logic should be capable of handling worst conditions.
4. **Input** — Whenever needed there should be provision to accept the input data.
5. **Output** — After solving the problem the algorithm should produce the corresponding output in a user-friendly format.
6. **Terminate** — An algorithm should terminate in a finite time.

While writing an algorithm, it is required to clearly identify the following:

- The input to be taken from the user,
- Processing or computation to be performed to get the desired result,
- The output desired by the user

An algorithm therefore corresponds to the solution to a problem that is independent of any programming language. Algorithms can be written either by using the natural language like English or one can make the use of pseudo language. When an algorithm is represented graphically, it is called as a flowchart.

Activities

Practical Activity 4.1. Write an algorithm to find the square of a number.

Procedure

Let us first identify the input, process and output:

Input: Number whose square is required

Process: Multiply the number by itself to get its square

Output: Square of the number

Algorithm to find square of a number.

Step 1: Input a number and store it to num

Step 2: Compute $\text{num} * \text{num}$ and store it in square

Step 3: Print square

Practical Activity 4.2. Write the algorithm for swapping of variables.

Suppose you have been given two variables A and B and asked to exchange the value of A and B.

Suppose A is assigned with a value 172 and B is assigned with value 563. Our task is to replace the content of A with the value 563 and the content of B with the value 172.

Procedure

Step 1. Save the original value of "A" in variable 'T', where T is the temporary variable.

Step 2. Assign to 'A' the original value of "B". ($A=B$)

Step 3. Assign to 'B' the original value of 'A' that is stored in T. ($B=T$)

Step 4. Stop

After executing the above four steps we observe that the values of 'A' and 'B' are interchanged. This is called as swapping of variables. This logic is used to solve many sorting and data manipulation problems.

Practical Activity 4.3. Compute the factorial of a number n, where $n \geq 1$.

The factorial of a number n is given by the formula $n! = 1 \times 2 \times 3 \times 4 \times \dots \times n$ where $n \geq 1$.

i.e. $n! = n \times (n-1)!$

Here initial we need to assume that the initial value of the product is equal to 1 and then we go on multiplying this product by 1, 2, 3, ..., until n. this can be achieved by using following steps.

Procedure

Step 1. Read the value of variable n.

Step 2. Set product = 1 and count =1.

Step 3. While count <= n, repeatedly do the following steps:

a. Compute new product by multiplying count to the product.

Product = product x count.

b. Increase the count by 1.

Count = count + 1.

Step 4. Result is the recent final product.

Step 5. Stop.

Assignments

Write an algorithm to find the sum of two numbers.

Write an algorithm to find the Fibonacci sequence till the number accepted from user. Note: Start the Fibonacci sequence from 0 and 1.

4.4 REPRESENTATION OF ALGORITHMS




Using algorithm, the software designers or programmers analyse the problem and identify the logical steps that need to be followed to reach a solution. Once the steps are identified, the need is to write down these steps along with the required input and desired output. There are two common methods of representing an algorithm —flowchart and pseudocode. These methods can be used to represent an algorithm.


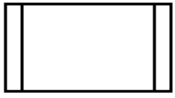
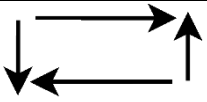

4.4.1. Flowchart

Flowchart is a graphical representation of an algorithm. To express different operations in the flowchart various standard symbols are used. These are graphical symbols. All symbols are connected among themselves in order to show the flow of information and processing. Each symbol represents certain activity. For example, we can have a symbol of start and end, processing, decision, input and output and connectivity. Symbols are connected to each other by using flow lines. These lines have an arrow which indicate the direction of the data flow.

Different symbols as prescribed by American National Standard Institute (ANSI) which are frequently required while drawing flowchart are tabulated in table 4.1.

Table 4.1: Symbols used in the Flowchart

Flowchart Symbol	Symbol Name	Description
	Terminal (Start or Stop)	Terminals are the oval shape symbols used to represent start and stop of the flowchart.
	Process	Process symbol is represented by rectangle and are used to represent process such as Arithmetic operations or Data movements.
	Input/ Output	Input/ Output symbol is represented by parallelogram, used to read input data and display the output.

	Decision	Decision symbol is represented by shape of a diamond, used to check any condition or take decision for which there are two options true or false.
	Predefined process or function	It is used to define function.
	Flow Lines	Flow lines are represented by arrow used to connect symbols used in flowchart and these indicate direction of flow.
	Connector	It is used to connect or join flow lines, useful when the flowchart is designed on multiple pages.

It is essential to follow certain guidelines while preparing the flowcharts. These guidelines are:

1. Standard symbols should be used while drawing flowchart.
2. Ensure that flowchart has START (or BEGIN) and STOP (or END).
3. Flowchart should be neat, clean and easy to follow.
4. There should be no any ambiguity.
5. The usual direction of flowchart is from top to bottom or from left to right.
6. The terminal symbol, that is, START/BEGIN or STOP/END should have only one flow line.
7. Only one flow line should come out from process symbol.
8. Only one flow line should enter a decision symbol, but two or three flow-lines, one for each possible answer, can leave the decision symbol.
9. If the flowchart is lengthy and complex connector symbol should be used to reduce the number of flow lines.
10. Avoid intersection of flow lines.

Example of Flowchart

Flowchart example for finding whether a number is odd or even is shown below Figure 4.7.

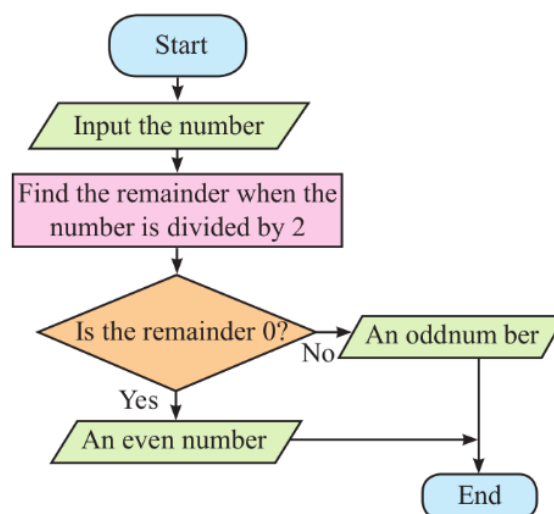


Fig. 4.7: Flowchart for finding whether a number is odd or even

Advantages of using flowchart

Following are the advantages of using flowchart.

1. Flowcharts are independent of any programming language. i.e. they are not related to particular programming language. Hence, they can be implemented using any programming language.
2. Flowcharts are easy to understand. The data flow of the program can be easily understood by the flowchart.
3. Finding errors and recovering them is easy in the flowchart.
4. Whenever the program needs further modification, then through flowchart it is easy to incorporate such modifications.

Practical Activity 4.4. Draw a flowchart for swapping of variables.

Draw the flowchart to interchange the values of two variables as shown in Figure 4.8.

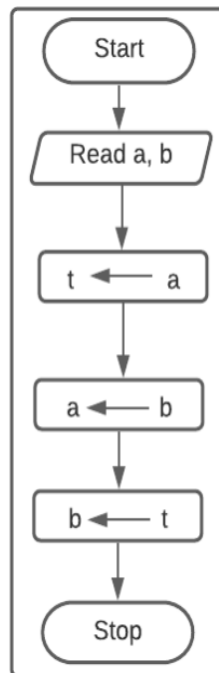


Fig. 4.8: Flowchart to interchange the values of two variables

Note that in the process boxes, the statements can also be written as $t=a$, $a=b$, and $b=t$.

Practical Activity 4.5. Draw a flowchart to compute the factorial of a number n , where $n \geq 1$.

Solution – Factorial of a number n is given by the formula

$$n! = 1 \times 2 \times 3 \times \dots \times n$$

First, we accept the number n whose factorial we have to calculate. As the factorial of a number 1 is 1, assign the product to 1 and count to 1. Compute the Product by multiplying Product to Count. Increment the Count by 1 repeat the steps to compute the Product till the counter reaches the value n . (Figure 4.9)

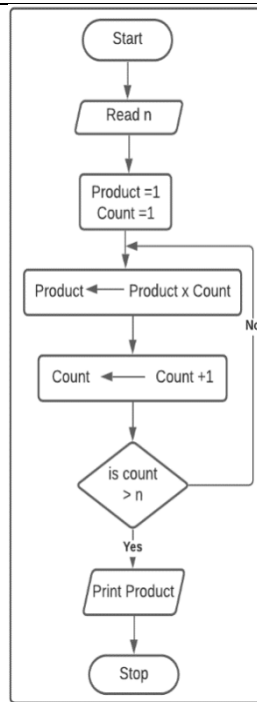


Fig. 4.9: Flowchart to find the factorial of a given number

Practical Activity 4.6. Draw the flowchart to find the largest of three given numbers

Solution – Consider the three numbers a, b, c. Following is the flowchart to find the largest of three numbers given numbers (Figure 4.10).

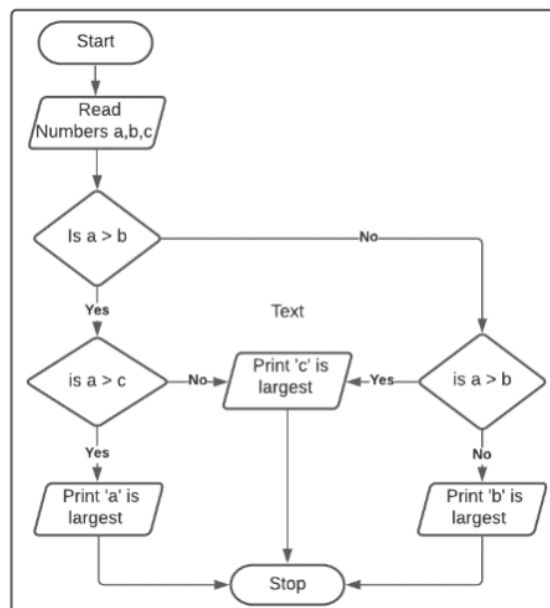


Fig. 4.10: Flowchart to find the largest of three numbers

Practical Activity 4.7. Draw the flowchart to find the largest of set of numbers.

Solution – Following is the flowchart which reads the number and assigns it to largest. It allows the user to enter another number and then it compares the new number with the existing largest number. If the new number is greater than the already stored number (largest) then it assigns the largest with the new number. (Figure 4.11) The loop is executed till the user wishes to enter new numbers.

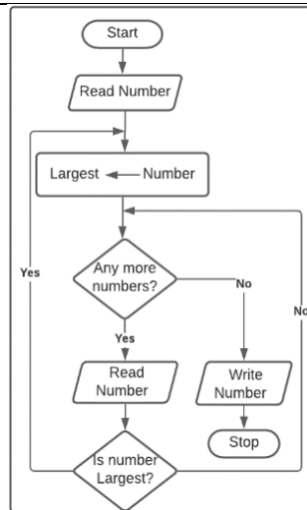


Fig. 4.11 Flowchart to find the largest of set of number

Practical Activity 4.8. Draw the flowchart for generation of n Fibonacci numbers.

Solution – The Fibonacci sequence, also known as Fibonacci numbers, is defined as the sequence of numbers in which each number in the sequence is equal to the sum of two numbers before it. The Fibonacci Sequence is given as:

Fibonacci Sequence = 0, 1, 1, 2, 3, 5, 8, 13, 21,

The sequence ends at the number entered by the user.

We assign the first number as 0 and second number to 1. The following flowchart will generate the Fibonacci numbers for the given number n. (Figure 4.12)

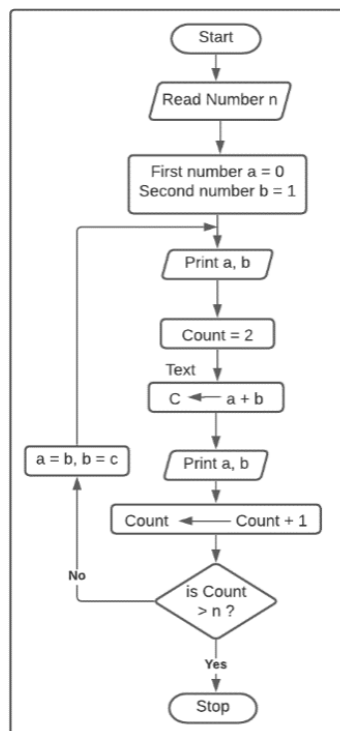


Fig. 4.12 Flowchart for generation of Fibonacci numbers

Assignments

Draw the flowchart to

1. Find the sum of two numbers.
2. Determine whether the given number is prime or not.

3. Arrange the numbers in ascending or descending order.
4. Count non-zero digits in a set of 100 digits.
5. Find the perimeter and area of a rectangle.
6. Calculate the new salary of employee by adding increment of 3% to the basic salary.

4.4.2 Pseudocode

A pseudocode (pronounced Soo-doh-kohd) is another way of representing an algorithm. It is considered as a non-formal language and is a detailed description of instructions that a computer must follow in a particular order. It is intended for human reading and cannot be executed directly by the computer. No specific standard for writing a pseudocode exists. The word “pseudo” means “not real,” so “pseudocode” means “not real code”. Following are some of the frequently used keywords while writing pseudocode:

INPUT
 COMPUTE
 PRINT
 INCREMENT
 DECREMENT
 IF/ELSE
 WHILE
 TRUE/FALSE

Example: Write the pseudocode to display the sum of two numbers entered by user.

Solution – Pseudocode for the sum of two numbers will be:

```
INPUT NUM1
INPUT NUM2
COMPUTE RESULT = NUM1 + NUM2
PRINT RESULT
```

Example: Write the pseudocode to calculate area and perimeter of a rectangle.

Solution – Pseudocode for calculating area and perimeter of a rectangle.

```
INPUT LENGTH
INPUT BREADTH
COMPUTE AREA = LENGTH * BREADTH
PRINT AREA
COMPUTE PERIM = 2 * (LENGTH + BREADTH)
PRINT PERIM
```

Benefits of Pseudocode

A pseudo-code of a program helps in representing the basic functionality of the intended program. By writing the code first in a human readable language, the programmer safeguards against leaving out any important step. Besides, for non-programmers, actual programs are difficult to read and understand, but pseudocode helps them to review the steps to confirm that the proposed implementation is going to achieve the desire output.

4.5 FLOW OF CONTROL

The flow of control depicts the flow of events as represented in the flow chart. The events can flow in a sequence, or on branch based on a decision or even repeat some part for a finite number of times. There are three types of flow control – Sequence, Selection and Iteration as shown in the Figure 4.13.

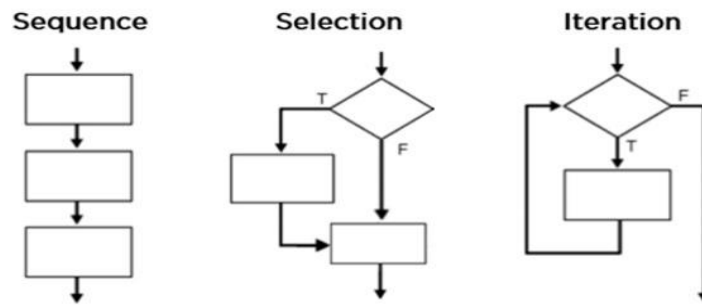


Fig. 4.13 Sequence, selection and loop

4.5.1 Sequence

When all the statements are executed one after another from the beginning to the end of an algorithm in a strict order, it is called a sequence (Figure 4.14).

Example:

1. Climbing up or down step by step when going on a staircase
2. Students who were admitted to class 1 of the school continue studies till class 12



Fig. 4.14: Example of sequence

However, the statements in an algorithm may not always execute in a sequence. Sometimes the problem may require to do some routine tasks in a repeated manner or behave differently depending on the outcomes of previous steps.

4.5.2 Selection

Selection is a situation where step(s) are executed depending on whether a condition of an algorithm is satisfied or not. There are two choices; if the condition is satisfied, one set of instructions is selected otherwise, the other set of instruction is selected.

Examples of selection

1. Admitting a child to Class 1:

If a child is below 6 years as at 1st July of that year

The child cannot be admitted to school

If not

The child can be admitted to school

2. Passing a subject:

If the mark scored is 35 or more

It is a Pass

If not

It is a Fail

3. Buying a book:

If you have money equal to or more than the price of the book

You can buy the book

If not

You cannot buy the book

4. Checking eligibility for voting.

Indian citizen gets the right to vote after completing the age of 18.

If age is greater than or equal to 18,
the person is eligible to vote

If age is less than 18,
the person is not eligible to vote

5. Checking eligibility for marriage.

If you are boy then

age of eligibility for marriage should be more than or equal to 21.

If you are girl then

age of eligibility for marriage should be more than or equal to 18.

Movement of light and heavy vehicle (Figure 4.15).

If you are driving a light vehicles

Go straight

If your are driving a heavy vehicle

Turn left

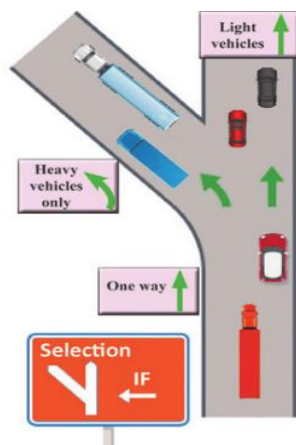


Fig. 4.15: Example of selection

Assignment

- Write down three real life situations which consist of selection.
- Select the most suitable word for the blanks

If you are an Indian citizen

_____ You _____ (have/ do not have) the right to vote.

If not

_____ You _____ (have/ do not have) the right to vote.

4.5.3 Repetition or Iteration

If one or several steps of an algorithm are repeated until a condition is satisfied, it is called repetition or iteration.

Examples

- Let us consider the process of a class teacher marking the attendance register.

- (1) Call the first name on the register
 - (2) Mark “P” if the student is present
 - (3) Mark “A” if the student is absent
 - (4) Call the name of the next student
 - (5) Repeat step (2) or (3) and (4) till the last name of the register is called
2. Let us consider the process of reading a paragraph and calculating the number of words you read.
- (1) Read the first word of the paragraph
 - (2) Number of words = 1
 - (3) Read the next word
 - (4) Add 1 to the number of words
 - (5) Repeat step (3) and (4) till the end of the paragraph
 - (6) After reading the paragraph, indicate the number of words

Assignment –

1. Write down two real life examples with steps that comprise repetition.
2. Fill in the blanks below related to repetition that output 5 times from 5 to 60.
 - a. $n = 5$
 - b. Print n .
 - c. Add 5 to the value of n .
 - d. Repeat step a to c, till the value of n is 60.

4.6. ALTERNATIVE SOLUTION AND EFFICIENCY OF ALGORITHM

There can be more than one solution to a given problem, such solutions are called alternative solutions. Also, there can be more than one approach to solve a given problem. Hence there can have more than one algorithm for a single problem statement.

Example 1. Let us examine the solution to find the perimeter of a rectangle.

Let us analyze the input, process and output related to this problem.

Input: Length and width of the rectangle

Process: Calculating the perimeter

Output: Indicating the perimeter

Let us examine the solution to calculate the perimeter.

Solution 1. Perimeter = length + width + length + width

Solution 2. Perimeter = length \times 2 + width \times 2

Solution 3. Perimeter = (length + width) \times 2

All these three solutions are correct. Think of the difference between all these three solutions and decide which one is more efficient.

Example 2. Consider the problem of finding whether a given number is prime or not. There can be four different ways to write algorithms to check whether a given number is prime or not as given below:

Solution 1. Starting with divisor 2, divide the given number (dividend) and check if there are any factors. Increase the divisor in each iteration and repeat the previous steps as long as divisor < dividend. If there is a factor, then the given number is not prime.

Solution 2. In *Solution 1*, instead of testing all the numbers till the dividend, only test up to half of the given value (dividend) because the divisor cannot be more than half of the dividend

Solution 3. In *Solution 1*, only test up to the square root of the dividend (numbers)

Solution 4. Given a prior list of prime number till 100, divide the given number by each number in the list. If not divisible by any number, then the number is a prime else it is not prime.

All these four methods can check if a given number is prime or not. In programming efficiency is important concern and hence the selection of algorithm depends on the efficiency of the algorithm.

Solution 1, requires large number of calculations hence requires more processing time, as it checks for all the numbers as long as the divisor is less than the number. If the given number is too large, this method will take more time to give the output.

Solution 2, is more efficient than *Solution 1*, as it checks for divisibility till half the number, and thus it reduces the time for computation of the prime number.

Solution 3, is even more efficient as it checks for divisibility till square root of the number, thereby further reducing the time taken.

As *Solution 4*, uses only the prime numbers smaller than the given number for divisibility, it further reduces the calculations. But in this method, we require to store the list of prime numbers first. Thus, it takes additional memory even though it requires lesser calculations.

Hence, algorithms can be compared and analysed on the basis of the amount of processing time they need to run and the amount of memory that is needed to execute the algorithm. These are termed as time complexity and space complexity, respectively. The choice of an algorithm over another is done depending on how efficient they are in terms of processing time required (time complexity) and the memory they utilise (space complexity).

4.7. CODING

Coding refers to the converting the algorithm to any a high-level programming language by following the syntax of that programming language. Syntax is the set of rules or grammar that governs the formulation of the statements in the language, such as spellings, order of words, punctuation.

The programs are written in high-level programming language which are then converted to machine language or low-level language consisting of binary code with 0s and 1s which are understood by the computer. The high-level programming language are close to natural languages and are easier to read, write, and maintain, but are not directly understood by the computer. A wide variety of high-level languages, such as FORTRAN, C, C++, Java, Python, are used by the programmer for software development.

A program written in a high-level language is called source code. This source code is translated into machine language using a compiler or an interpreter, so that it can be understood by the computer.

The selection of programming language depends on the various factors such as requirement of the client, the platform (OS) where the program will run, whether the application would be a desktop application, a mobile application or a web application.

Desktop and mobile applications are generally developed for a particular operating system and for certain hardware whereas the web applications are accessed in different devices using web browsers and may use resources available over cloud.

Besides, programs are developed not only to work on a computer, mobile or a web browser, but it may also be written for embedded systems like digital watch, mp3 players, traffic signals or vehicles, medical equipment and other smart devices. In such cases, we have to look for other specialised programming tools or sometimes write programs in assembly languages.

4.8. PROBLEM DECOMPOSITION

For a complex problem, the solution is not directly derivable. In such cases, we need to decompose it into simpler parts.

Let us look at the Railway reservation system. The complex task of designing a good railway reservation system is seen as designing the different components of the system (booking tickets, cancellation of tickets, train schedule) and then making them work with each other effectively.

The basic idea of solving a complex problem by decomposition is to 'decompose' or break down a complex problem into smaller sub problems.

These sub problems are relatively easier to solve than the original problem. Finally, the sub-problems are combined in a logical way to obtain the solution for the bigger, main problem.

Breaking down a complex problem into sub problems helps to each sub problem can be solved independently and by different programmers. Having different teams working on different sub problems can also be advantageous because specific sub problems can be assigned to teams who are experts in solving such problems.

There are many real-life problems which can be solved using decomposition. Examples include solving problems in mathematics and science, events management in school, weather forecasting, delivery management system, etc.

Once the individual sub problems are solved, it is necessary to test them for their correctness and integrate them to get the complete solution.

Check Your Progress

A. Multiple choice questions

1. Programmer documentation consists of (a) source code (b) flowchart and pseudo code (c) specification of program (d) all of the above
2. Which of the following is flow control statement (a) sequence (b) selection (c) loop (d) all of the above
3. Which of the following is the first step of problem solving cycle? (a) Testing the solution (b) Designing the solution (c) Analysing the problem (d) coding for the solution
4. In flowchart, the decision box is indicated by (a) rectangle (b) diamond shape (c) circle (d) parallelogram
5. Which of the following is not a type of flow control? (a) Iteration (b) Sequence (c) Selection (d) None of the above
6. Raman is driving his car. On a traffic signal, he can move only if the signal turns green. Which of the following is the current flow control? (a) Iteration (b) Sequence (c) Selection (d) None of the above

B. Fill in the blanks

1. The problem solving cycle consists of _____ steps.
2. The solution of problem represented in natural language is called as _____ .
3. In the modular approach, the project is divided into _____ .
4. Coding the program is the process of translating algorithms into _____ .
5. The length of the program will depend upon the _____ of the problem.
6. The tow types of documentations required are _____ and _____
7. The steps are executed depending on whether a condition of algorithm is satisfied or not is _____
8. The set of rules or grammar that govern the formulation of the statements in the language is _____

C. State whether true or false

1. The steps in algorithm should be followed in strict sequential order.
2. The common methods to represent the algorithm are flowchart and pseudo-code.

3. A pseudo-code represents the basic functionality of program.
4. A program written in machine code is called as source code.
5. In modular approach, the complex problem is divided into smaller portions.
6. Testing is not important phase of software development process.
7. A good algorithm is not precise.
8. A pseudo code is intended for human reading only.

D. Short answer questions

1. What are the steps of problem-solving cycle?
2. What is modular approach for designing the solution of problem?
3. What are characteristics of good algorithm?
4. Differentiate between an algorithm and a flowchart?
5. State the guidelines to prepare the flowchart.
6. State the advantages of flowchart.
7. Give the real-life examples of sequence, selection and repetition.
8. What is meant by alternative solution?
9. List the five high level programming languages.
10. Write an algorithm, pseudocode and draw the flowchart for the following statements.
 - a) To display the sum of digits of a number.
 - b) To accept two numbers, divide them and display the quotient and remainder.
 - c) To display the series, 0, 4, 16, 36, 64, ..., n^2
 - d) To display sum of first 10 natural numbers.
 - e) To accept number n and display the average of all numbers.

Session 5. Programming Language Concepts

In real life we always encounter with the active objects such as automatic door, automatic washing machine, automatic dish washer, a vehicle such as scooter or car. Observe that these objects can perform actions. Only they need is to receive some stimulus from outside to perform the action. That is when you approach to automatic door in a hospital or airport then the door automatically recognizes you and gets open. Similarly, the dishes are washed and cleaned by the dish washer once you switch on the dish washer. In programming it is possible to make the use of active objects that can perform an action themselves once they receive the stimulus from the program and such programming environment is been called as an object-oriented programming environment.

Fig. 5.1 Illustration of automatic door

5.1. EVOLUTION OF PROGRAMMING LANGUAGES

5.1.1. Need of a programming language

We know that whenever we want to solve a given problem then we need some set of logical steps. This set of logical steps is called an algorithm. Algorithm can be graphically represented by using the flowchart. Once the algorithm is prepared, the next step is to translate this algorithm into the set of instructions. This set of instruction is called as a program.

A program is a sequence of instructions which is designed to perform a certain task is using the computer. A computer language is needed to provide the instructions.

Whenever we want to write a program then we need to make the use of a programming language. A programming language is a formal language that is used to describe the computation. It is also called as user interface to the computer system. Programming languages are the high-level languages which are English like languages. Programming language is defined as a systematic notation that is used for describing the computational process. Computational process means

the arithmetic and logical operations that we need to perform. The steps for obtaining the solution for a given problem are shown in the Figure 5.2.



Fig. 5.2 Steps for obtaining the solution

5.1.2. Programming Language

The computer is a digital electronic machine that can process only binary data. The language which consists of only such binary data and instructions is being called as a low-level language. But writing every instruction in the form of 0s and 1s is very difficult task. Therefore, the high-level languages are evolved. These high-level languages are English like languages where we make use of English words. These languages are called as the programming languages. There are different types of programming languages (Figure 5.3).

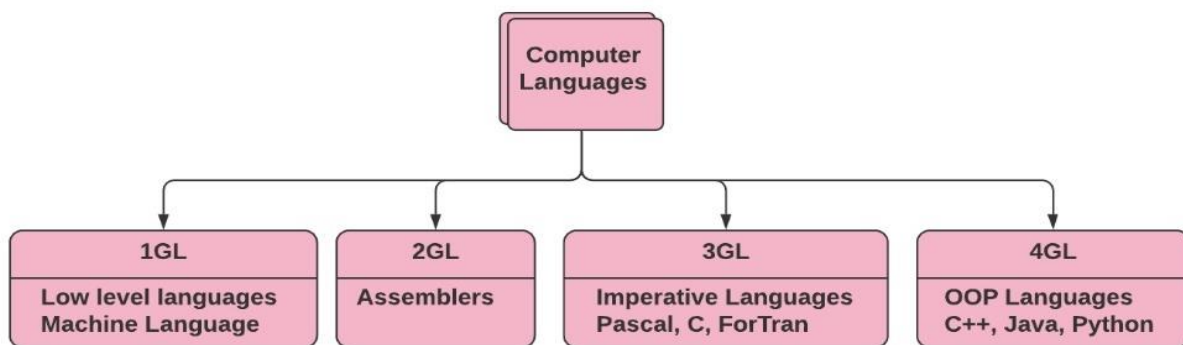


Fig. 5.3 Generation of programming languages

1 GL – First generation languages such as low-level languages or machine language.

2 GL – Second generation languages such as symbolic assemblers.

3 GL – Third generation languages, these are machine independent imperative languages such as high-level languages ForTran, Pascal and C.

4 GL – Fourth generation languages. These are the domain specific object-oriented languages such as C++, Java, Python.

5.2. Low level programming languages

5.2.1. Machine language

This is a language which can be directly understood by the computer. Binary numbers such as 0s and 1s (bits) are used to provide instructions. Hence, the processor could directly run a program written in a machine language. A program written in machine language has the following features;

- Could be executed directly on the machine
- Fast in operation
- No need of language translating programs to translate the program into binary
- Dependency on machines (a program written to one computer may not run on another computer)
- Difficult to understand by humans as it is written using 0 and 1.

5.2.2. Assembly language

Instead of commands written in a machine language using 0 and 1, assembly language is designed to use simple symbols. A program written in assembly language has the following features:

- Operation is comparatively slower than the machine language.
- Assembly language should be translated to instructions using the language translating program called assembler.
- Dependency on machines a program written for one computer cannot be run on another computer.
- The use of symbols makes it simpler to understand.

5.2.3. High-level programming languages

Languages which are designed with simple English words enabling the programmer to understand it easily are called high-level computer languages.

Examples for high level computer languages are FORTRAN, BASIC, COBOL, PASCAL, C.

Features of high-level languages

High level languages (HLL) are the English like languages where we can make the use of English words such as if, then, else, for, next, do, while, break, switch while writing the statements in the program. The languages such as C, C++, Java, Python are high level languages. These languages have following features.

Ease of understanding – Programs written in HLL are easy to understand as they consist of English natural words.

Naturalness – The original or natural meaning of the English words is maintained in the HLL. Therefore, they are very similar that of natural languages.

Portability – Programs written in HLL on one machine can be executed without any modification on another machine and therefore these programs are portable in nature.

Efficiency of use – It is possible that we can perform efficient programming in HLL i.e. we can keep the size of the code small and also the time required for its execution is optimized.

Syntax and Semantics of the High-Level Languages

Syntax

Every HLL has certain grammar. This grammar will tell us how to write program in that programming language. There are grammatical rules associated with every programming language and it is called as the syntax of the programming language. It tell us how to write expression, statements or loop in the program. It will give us a clear and unambiguous description of the different constructs of the programming language. The context free grammar is most widely used for describing the syntax of the programming language. Figure 5.4 shows the typical syntax of assignment expression. Observe that syntax consist of production rules that contain terminal and non-terminal symbols.

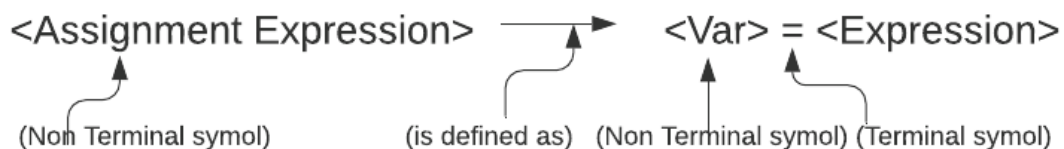


Fig. 5.4 Production rule for assignment expression

Semantics

Every construct of the programming language has certain meaning associated with it. It is called as the semantics of the programming language. It is described by what happens when that construct is executed by the computer. Semantics can be described by variety of way such as operational semantics, axiomatic semantics and denotational semantics. The code below shows the typical operational semantics of the programming language. Observe that the meaning of loop is described in operational semantics.

Code	Operational semantics
For I = first to last do begin end	i = first loop : if i > last goto out i = i + 1 goto loop out :

Programming Constructs

Every programming language offer certain programming constructs that can be used by the programmer for the translation of algorithmic steps into program. These constructs are data types, expressions, statements, control structures, loop structures, procedures, comments and errors.

Data types – In real life the data or facts has different forms. Sometimes the data is in the form of integers and sometimes it is in the form of real numbers. Sometimes the data consists of the characters and string and it can be in the form of images or pictures, audio and video. Every programming language must provide certain constructs so that all such different types of data can be read and processed through a program. Normally, all programming language has different data types such as integer, character, float, array, record, pointer, structure. Figure 5.5 shows commonly used data types in the programming language.

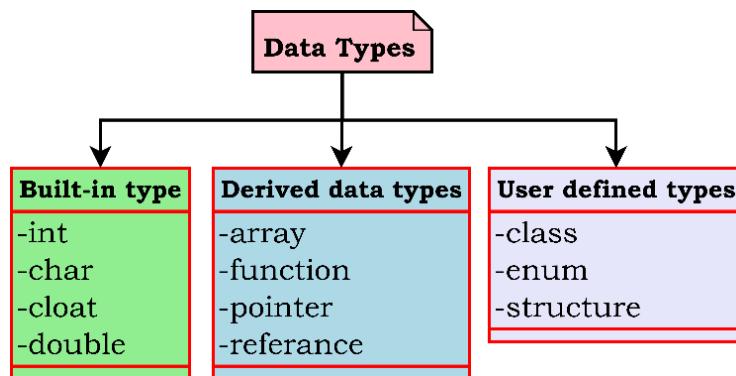


Fig. 5.5 Commonly used data types in the programming language

Observe that data types can be built in data type, derived data type and user defined data type.

Expressions – When variables are connected together with operator symbols then it is called as an expression. It is a sequence of operands and operators. Operators such as arithmetic operators, relational operators, logical operators are used in the programming languages. Arithmetic operators are used to perform arithmetic operations and such expressions is called as arithmetic expression. Relational operations are used to compare the data and such expression is called as relational expression. Logical operators are used to perform the logical operations such as AND, OR, NOT, and such expression are called the logical expression. Figure 5.6 shows the different types of operators used in the programming languages.

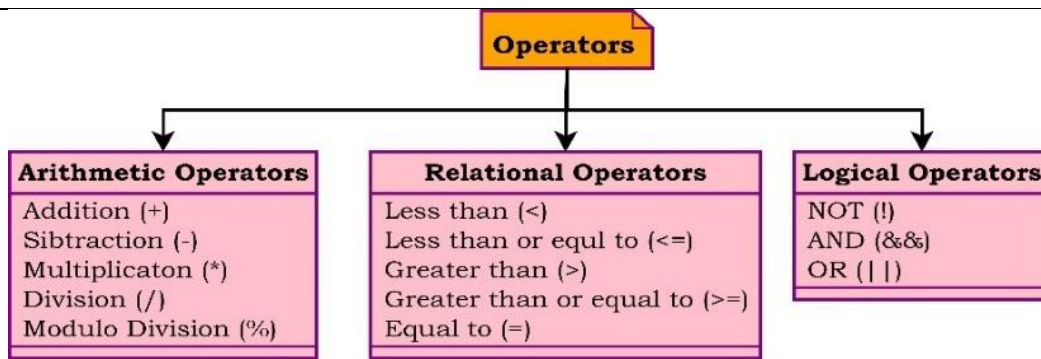


Fig. 5.6 Different types of operators in programming languages

Statements – A statement causes an action to be performed by the program. It translates directly into one or more executable computer instructions. There are different types of statements such as simple statements, compound statements and assignment statements. A simple statement is a single statement. A compound statement is a unit of code that consists of one or more statements. It is also called as a block. A compound statement allows a group of statements to be treated as a single entity. An assignment statement is a statement that is used to assign a value to a variable, i.e. it stores the value in the variable. Figure 5.7 shows the simple statements, compound statements and assignment statements used in the programming language.

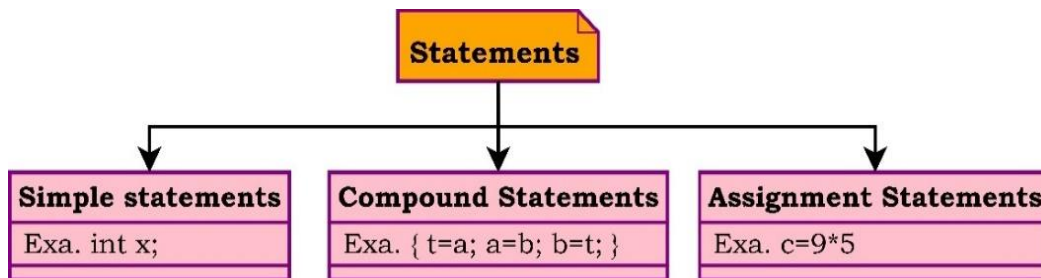


Fig. 5.7 Simple statements, compound statements and assignment statements

Observe that the number of statements in a compound statement are enclosed in braces.

Control structures – Instructions written in the program are executed sequentially one after the other. But many times, we do not need such sequential execution, instead we need to jump from one statement to another statement. This is possible by using control structures in the programming languages. The control structures that are used in the programming languages are “if then”, “if then else” and “switch” statements. In “if then” statement if the condition is true then the then part of the statement is executed. In “if then else” if the condition is false then the else part of the program is executed. Figure 5.8 shows the control flow for if then and else statements.

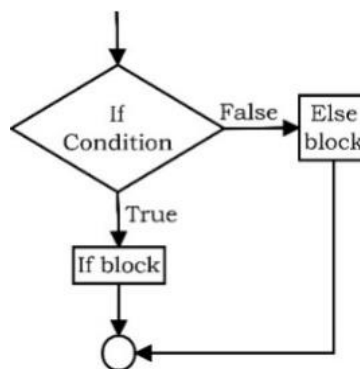


Fig. 5.8 Control flow for if then and else statements

The switch statement allows the multiple way selection in the program. Depending upon the value of the expression different parts of the program are executed as shown in the Figure 5.9.

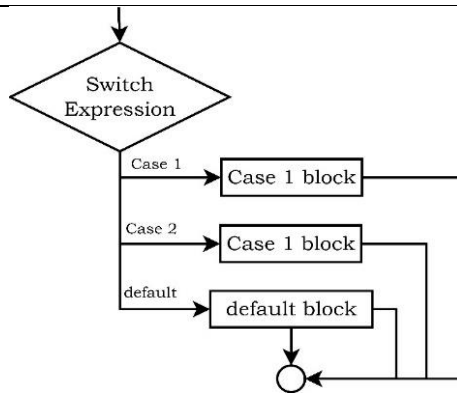


Fig. 5.9 Switch statement

Loop structures – Many times in the program we need to repeatedly execute certain part of the program. Such repetitive execution is possible through the loop control structures or statements. These statements also called as iterative statements. Most of the programming languages provides iterative statements such as *for*, *do while*, *while*. The *for* statement is described in the Figure 5.10.

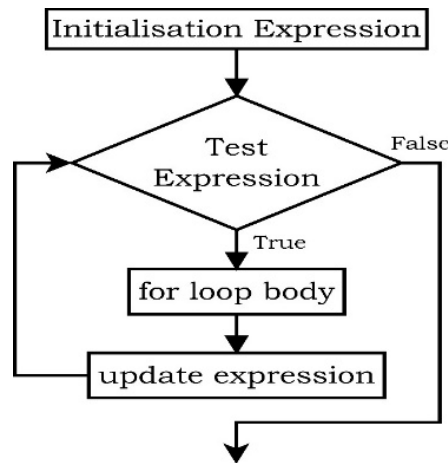


Fig. 5.10: Iterative statements 'for'

Procedure – It is a set of statements that is written to perform the specific task. Many times, procedure is also called as a function. Almost all programming languages has two types of functions (i) User defined functions and (ii) Built in functions. The code for the built-in functions is readily available with the compiler so that it can be directly used in the program. User defined functions are those functions that are defined by the user as per the requirement of the program. Figure 5.12 shows classification of functions.

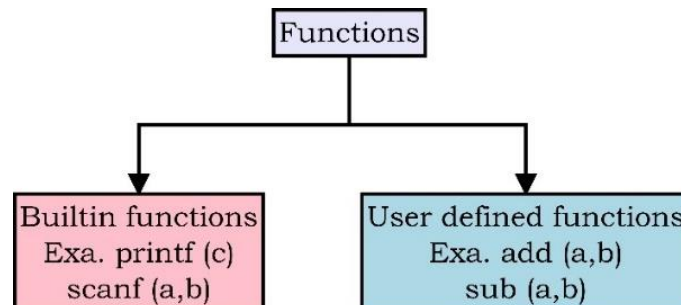


Fig. 5.11: Classification of functions

Because of this utility the programming efforts are saved and the same code can be reused in many applications.

Comments – Whenever a programmer writes a program then the comments are used to describe the purpose of the part of the program. Sometimes programs are very complex and it is difficult

to read and understand them. In such a case the comments are useful for better understanding of the program. Every programming language provides the facility to write the comments as shown below.

```
// This is a single line comment
/* This is a multiple line comment
*/
```

Observe that the comments are not useful for the compiler and therefore comments written in the program are removed by the compiler during execution of the program.

Errors – whenever a programmer write a program then it is necessary to obey the syntactic rules of the programming language. Many times, while writing the program, a programmer may commit certain mistakes and these mistakes are called as errors. Errors can be a variety of types such as algorithmic errors, syntax errors, lexical errors and semantic errors. Compiler can detect syntax errors, lexical errors and semantic errors during compilation process and these errors will be prompted to the user for correction. User need to make appropriate correction in the program and then only these programs are compiled. Algorithmic errors are logical errors and they cannot be detected by the compiler. It may result in wrong output. Figure 5.12 shows the typical errors generated by the compiler.

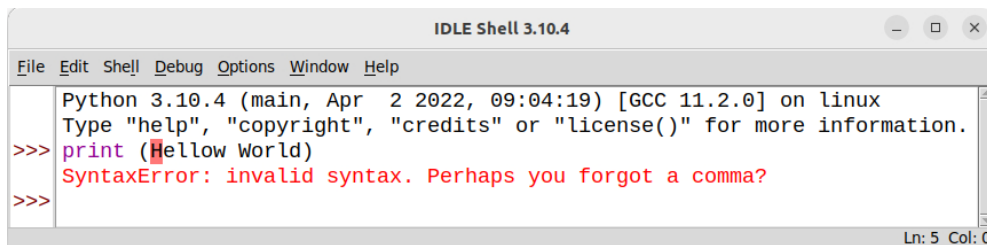


Fig. 5.12: Errors generated by the compiler

LANGUAGE TRANSLATORS – COMPILER AND INTERPRETER

Programs written in any computer language except in machine language (object code) should be translated to machine language instructions before execution.

A program written in assembly language is translated to machine language instructions using a language translator called assembler.

Interpreter and *Compiler* are two types of language translators can be used to translate a program written in a high-level language to machine language instructions. They are:

Programs written in HLL are translated into machine level object code by a software that is called as compiler or interpreter. Both compiler and interpreter will do the same job of translation but their way of doing it is different. The compiler will compile the whole program into machine code in one attempt only. Interpreter will convert a single line of code at a time. A compiler may generate an intermediate machine code but the interpreter never generates an intermediate code. The compilers are faster programs than that of interpreters. Compiler is best for the production environment and interpreter is best for software development environment. Most of the HLL such as C, C++, Java makes the use of compilers for translation. But many other languages Python, PHP, Perl makes the use of interpreter for the translation. Java is a typical example where it makes the use of compiler and interpreter both. Table 5.1 demonstrates the difference between the compiler and interpreter.

Compiler	Interpreter
Compiler scans the whole program code at a time and list out the errors after compilation of the whole program.	Interpreter scans the source code line by line and errors are also shown line by line.
It links various code files into executable	It does not require the linking of files for

program file (.exe)	generation of machine code
It converts the entire program code into object code	It does not convert source code into object code instead it scans it line by line
The compiled codes run comparatively faster.	The interpreted codes run comparatively slower.
It does not require source code for later execution.	It requires source code for later execution.
It generates an output program in the .exe format. A user can run it independently from the originally intended program	It doesn't generate an output program. Meaning, it evaluates the source program every time during individual execution.
C, C++, C# are the examples of the programming language that are using compiler	Python, Ruby, Perl, PHP are the examples of the programming language that are using interpreter

Table 5.1: Difference between Compiler and Interpreter

PROGRAMMING APPROACHES

Programming is a creative task where a computer programmer provides instructions to the computer on how to perform a particular task.

There are different ways to program for obtaining the solution for the given problem. These different ways of writing the program are been called as the programming approaches. Many times they are also called as the computer language paradigms. Different approaches develop solutions to problems using programs using different paradigms. Even though most of the programming languages come under one paradigm type, certain languages show elements related to different paradigms. There are four programming approaches commonly available (i) procedural (ii) object oriented (iii) functional (iv) declarative. Figure 5.13 shows the different programming approaches.

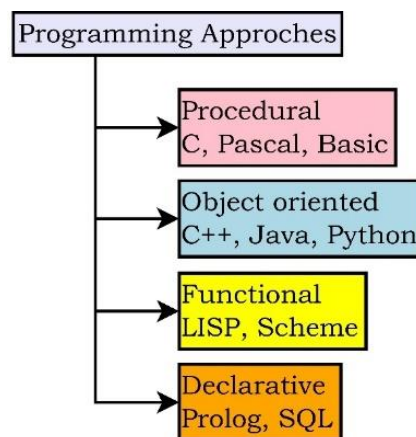


Fig. 5.15: Various programming approaches

Difference between procedural and declarative paradigms

There are two groups of programming languages – imperative and declarative programming languages. An imperative language is a computer programming language which consists of a well-structured set of steps and procedures. This includes statements for problem solving steps. A declarative paradigm develops a structure and elements of computer program by indicating calculations and/or logic without a control flow. This helps to reduce or eliminate side effects. In declarative programming, program is designed to solve problems explaining what you want rather than stating how to solve the problem as in primary programming languages. The program itself does not explain how it is executed. This means the computer is provided only what the

problem and the required solutions are, not how to solve it. The computer finds solutions related to the given problem. This is completely different from procedural paradigms which execute algorithm as explanatory steps. Declarative paradigms related to Artificial Intelligence.

(i) Procedural approach – in this approach the program is a collection of procedures or functions. The program is an active agent that manipulates the passive objects. All the data and variables that are used in this approach are passive objects. For example, a stone, a book, a lamp or a table or chair can be considered as passive objects or entities. A passive object cannot initiate action by itself, but it can receive action from the active object i.e. program.

In procedural approach, program is active that can manipulate the data. For example, a program that prints the content of the file is active but the file itself is passive. To print the file the program can make use of the procedure called as a print. Figure 5.14 shows the concept of procedural approach.

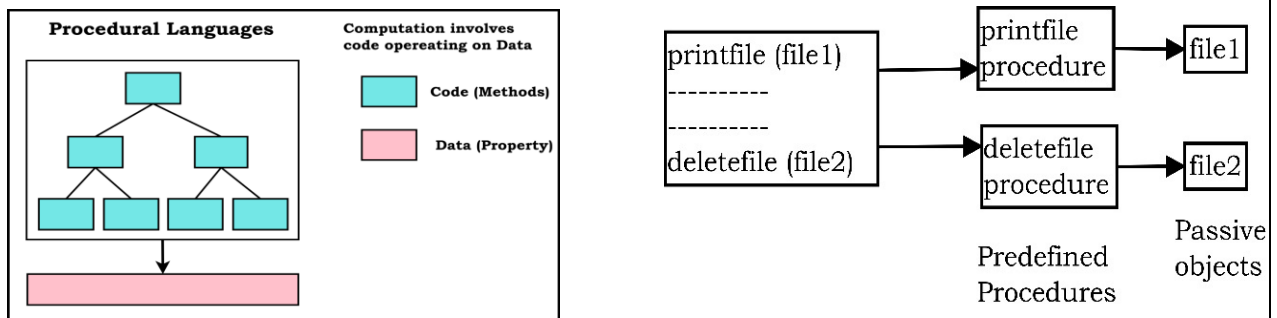


Fig. 5.14: Concept of procedural approach

Observe that many HLL such as ForTran, BASIC, C, Pascal are the procedure-oriented languages. These languages are also called as imperative languages.

A program in the procedure-oriented languages has three parts – (i) a part for the creation of the passive objects or declaration of variables (ii) a set of procedures calls (iii) a set of code for each procedure.

Figure 5.15 shows the contents of the procedure-oriented program. Some procedures may be defined in the language itself and therefore their code need not to be written by the programmer. This code can be combined with the program, so that we can make the use of that procedure. User defined procedures need to be defined by the programmer. And these procedures are called as new procedures.

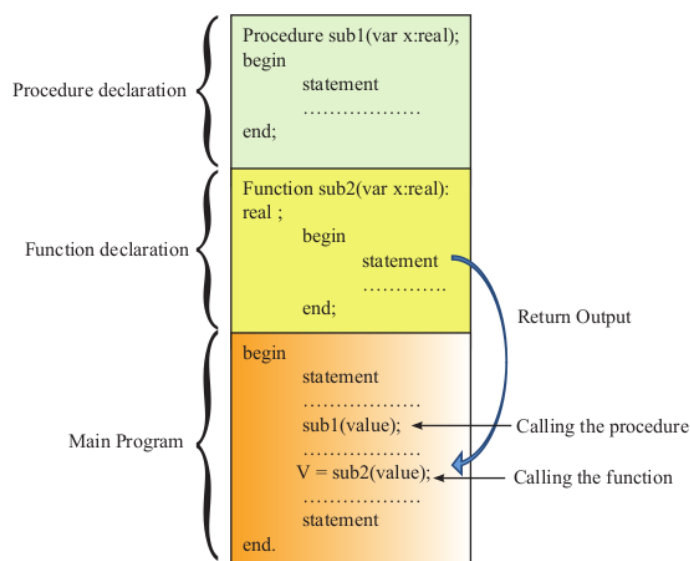


Fig. 5.15: Contents of the procedure-oriented program

(ii) Object oriented approach – In object-oriented approach the program consists of active object instead of passive object i.e. in this case both program and objects are active. In object-oriented programming a program is basically a collection of active objects. In real life we always encounter with the active objects such as automatic door, automatic washing machine, automatic dish washer and a vehicle such as scooter or car. Observe that these objects can perform actions on their own. Only they need is to receive some stimulus from outside to perform the action. That is, when you approach to automatic door in a hospital or airport then the door automatically recognize you and gets open. Similarly, the dishes are washed and cleaned by the dish washer once you switch on the dish washer. In programming it is possible to make the use of active objects that can perform an action themselves once they receive the stimulus from the program and such programming environment is been called as object-oriented programming environment. Figure 5.16 shows the concept of object-oriented programming environment.

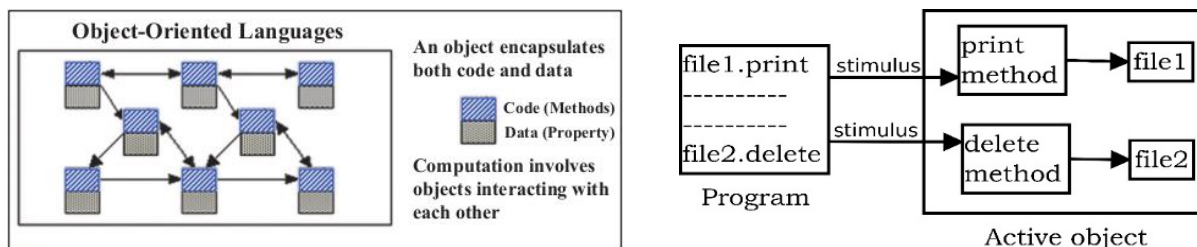


Fig. 5.16: Concept of object-oriented programming environment

Observe that the file in the *Object-Oriented Programming (OOP)* environment will consists of a method of printing the file and therefore whenever a stimulus is sent by the program then the file on its own can gets printed.

Certain commonly used concepts in OOP environment are (i) classes (ii) Inheritance (iii) Polymorphism

(i) Classes – the concept of class is defined as the collection of variables and related methods. Figure 5.17 shows the concept of class in the OOP programming environment. Observe that object variables are the instances of the class. This concept of class provides a kind of encapsulation to data and its related methods or procedures. This is very close to the real life understanding of the objects. Say for example, a teacher can remember a student by not only his name, but also by various parameters such as sincerity and cleverness. Similarly, a dog can be remembered not only by name and color but also how he provides the protection to the family.

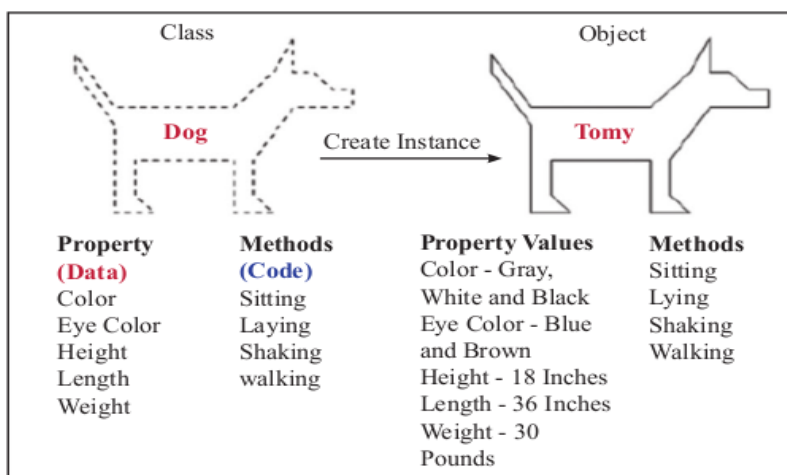


Fig. 5.17: Concept of class in the OOP

(ii) Inheritance – In OOP environment an object can inherit the properties of another object. This is called as inheritance. When a general class is defined then we can define a more specific class that inherits some of the properties of the general class, but at the same time this class will have certain new characteristics. This concept is also taken from the real life. For example,

the next generation of human beings will always carry certain properties of earlier generation such as skin color, height and shape of the face. But, at the same time they have something different which is not there in their earlier generation. In OOP environment when we define a class called as geometrical shape then from this class we can further derive different classes such as rectangles, triangles and circle. Rectangles, triangles and circles are the geometrical shapes with certain additional characteristics. Figure 5.18 shows the inheritance in OOP environment.

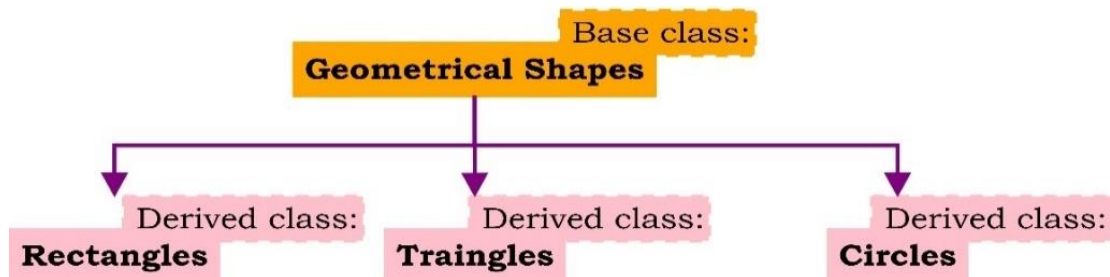


Fig. 5.18 Inheritance in OOP environment

(iii) Polymorphism – ‘Poly’ means ‘many’ and ‘morphism’ means ‘forms’, i.e. ‘Polymorphism’ means ‘many forms’. In OOP environment it is possible to define several operations with the same name that can-do different things. This is called as Polymorphism. For example, it is possible that we can use the same function name, ‘area’ to compute the area of rectangle and to compute the area of triangle. These are the two different operations but the still same name can be used for these two operations. This is called as function overloading. Also, it is possible to use the same operator to perform similar operation for built in data types and user defined data types. That is the operator + can be used for the addition of two integers and also the same operator can be used for the addition of two complex numbers, where complex number is a user defined data type. This is called as operator overloading. Function overloading and operator overloading are the two forms of polymorphism. Figure 5.19 shows the polymorphism in OOP environment.

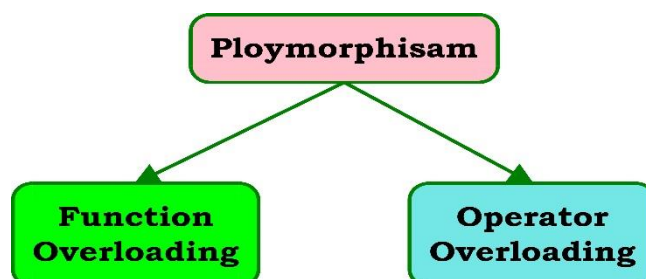


Fig. 5.19 Polymorphism in OOP environment

Programming languages such as C++ and Java are OOP languages. Python can also be called as an OOP language because it provides almost all functionalities of OOP environment.

(iii) Functional Programming – It is an environment where the programs are constructed by applying and composing functions. In functional programming everything is written in the form of functions. Even mathematical expressions are also written in the form of functions. For example, the programming languages like LISP (LISt Processing) and Scheme are functional programming languages. These languages are designed for specific domain. LISP can be used in solving the problems of Artificial Intelligence. Figure 5.20 shows the functional programming environment.

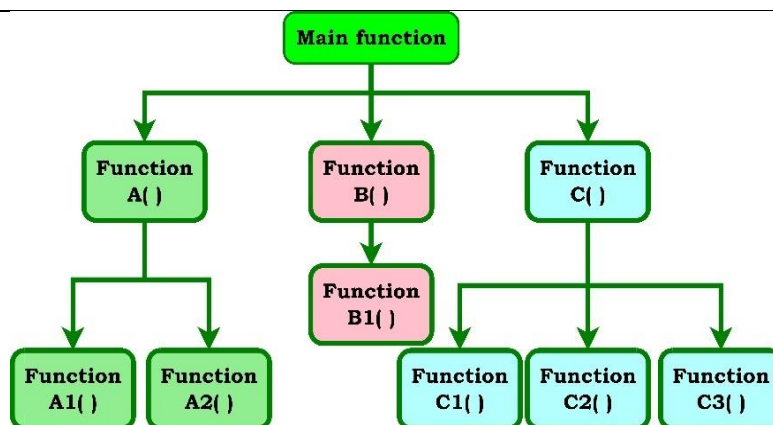


Fig. 5.20 Functional programming environment

(iv) Declarative Programming – It is a programming environment in which the logic of computation is expressed without describing its control flow. It is a method to abstract away the control flow for the logic. This approach is opposite to that of imperative programming. In this case the programs are built in such a way that you want to describe and not on how you want to do. Examples of declarative programming languages are SQL (Structure Query Language) and ProLog (Programming Logic). Python is not a pure declarative language but it has some flexibility that to an extent it can be used as a declarative language. Figure 5.21 shows the declarative programming environment.

Fig. 5.21: Declarative programming environment

Selection of programming language

Whenever a programmer is asked to develop a solution from the problem then it is the responsibility of the programmer to have a right selection of programming language to develop the solution of the problem. The solution of the problem must be such that it is concise, easy to debug, easy to document, easy to expand and easy to fix the problem. In order to achieve this the programmer should consider the following factors for the right selection of programming language. The factors are the (i) targeted platform (ii) elasticity of language (ii) time to production and (iv) performance.

(i) Platform – It is very much necessary to consider first the platform on which the program will run. For example, if the program is developed in C language, and it needs to be run on Windows and Linux then it could require the platform compilers and two different executable files. If the program is developed using Java, then the program can run on any machine provided JVM (Java Virtual Machine) is installed.

(ii) Elasticity of language – the selected language must be elastic in the sense that the new features can be easily added to the existing program. If the programming language do not provide the mechanism to add the new features to existing program then such language must be avoided for the development of the new program.

(ii) Time to production – the time taken to make the program so that it can go live is called as time to production. It is highly dependent upon the size of the code. Always one can find that size of the code for the live projects is huge and the time allotted for the development is very low. In such a case the programmer needs to optimize the resources so that the final program can be delivered within a time frame.

(iv) Performance – It depends upon the platform on which it runs and also it depends upon the programming language used in the program. A programmer can check the performance of the language by considering some similar other projects in which the same language is used for the development.

Python is being considered as the topmost programming language because it can provide all the object-oriented features such as C++ and Java. Also, this language provides readability and the

code development in this language is very easy and fast. The program modularity feature, code re-usability feature provides a high speed of the development. The maintenance of the code in Python is easy. This language increases productivity and save time and effort while working with the language. It is also an open source language which reduces the cost of the development. By using Python, we can create applications, frameworks, tools, websites, web Apps and also solutions for AI based applications. Python has simplified syntax which is not completed and therefore it is just like a natural language. Because of these reasons Python is being widely used in providing the solutions to the different forms to the problem.

Coding environment

It is an environment which provides the comprehensive facilities to the computer programmer for the software development. The coding environment can also be called as IDE (Integrated Development Environment). It normally consists of various tools such as the operating system or cloud environment, source code editor, build animation tools, debugger, necessary compiler or interpreter or both. Net beans and Eclipse are the examples of coding IDE. Sharp develop and lazarus are free and open source IDE.

Operating system – OS the basic software requirement for any computer system. It is the interface between the hardware machine and software.

Cloud environment – Now a days almost all the programs written in various programming languages can be run on internet-based cloud platforms.

Source code editor – It is a text editor program specifically designed for the editing of the source code of the computer programs. This is the fundamental programming tool that is required for the development of the program.

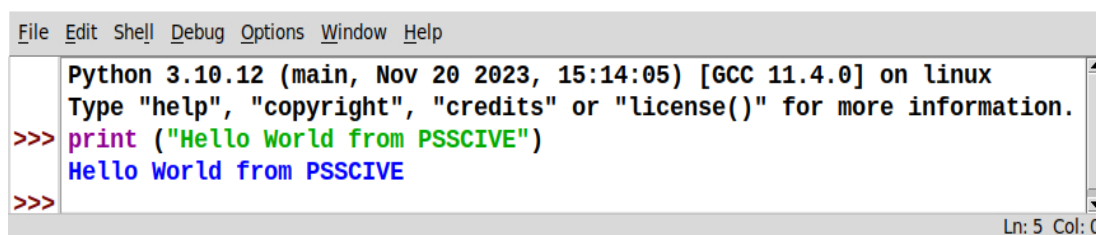
Build animation tools – It the process of automating the creation of software build. It is associated with the other processes such as compiling the source code into binary code, packaging the binary code and running automated test.

Debugger – It is a debugging tool is used to test and debug the program. The main use of the debugger is to run the target program under control conditions that permit the programmer to track its operation and its progress. It also monitors changes in the computer resources such as memory. By using debugger, it is possible to halt the program at any specific point and check the content of memory, CPU and storage devices.

Compiler or interpreter – These are the computer programs that translate the computer code written in programming language into another language such as the object code of machine code. Output of the compiler can be executed by the computer system.

Net beans and Eclipse – These are the examples of IDE environment for Java. Net beans run on Windows, Linux, Mac and Solaris OS. It can be extended to other languages such as PHP, C, C++ and JavaScript. Eclipse can also be extended to the Python language.

Sharp develop and lazarus – it is the free integrated open source coding environment for .Net framework and Python. Lazarus is a cross platform IDE. It is used for Rapid Application development using the free Pascal compiler. It can be used for the development of the program such as by using language such as Object Pascal and Delphi. For Python programming language the most commonly used coding environments are Atom, Code spaces, Visual Studio, Subline Text3, and Vim. Figure 5.22 shows the typical IDE of Python.



```
File Edit Shell Debug Options Window Help
Python 3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>> print ("Hello World from PSSCIVE")
Hello World from PSSCIVE
>>>
```

Fig. 5.22: Typical IDE of Python

Check Your Progress

A. Multiple choice questions

1. Which of the following is the third-generation language (a) assembly language (b) machine code (c) ForTran (d) Python
2. Which of the following is the fourth-generation language (a) C (b) B (c) assembly language (d) Python
3. Which of the following concept is not used in OOP environment (a) classes (b) Inheritance (c) Polymorphism (d) procedure
4. Which of the following programming language is highly flexible and rich in development of variety of applications (a) C (b) C++ (c) Python (d) Fortran
5. Which of the following operators are used in programming languages (a) arithmetic operators (b) relational operators (c) logical operators (d) all of the above
6. Problem decomposition means (a) solving a problem (b) breaking down a problem into smaller sub programs (c) increasing the complexity of problem (d) combining different tasks into one bigger task
7. Source code is the programming code written in (a) machine code (b) high level language (c) low level language (d) binary code
8. Which of the following is the control flow used in programming (a) sequence (b) selection (c) iteration (d) all of the above
9. Which of the following are the factors for right selection of programming language (a) targeted platform (b) elasticity of language (c) time to production (d) performance (e) all of the above
10. Which of the following are the commonly used concepts in OOP environment (a) classes (b) Inheritance (c) Polymorphism (d) procedures

B. Fill in the blanks

1. Programming language is a systematic notation used to describe the _____ process. (computational)
2. Computational process involves _____ and _____ operations (arithmetic, logical)
3. Variables connected with operator forms _____ (expression).
4. The coding environment is called as _____ (Integrated Development Environment)
5. Which programming language is used to solve the problems of Artificial Intelligence _____

C. State whether True or False

1. A program is a sequence of instructions which is designed to perform a certain task.
2. High level programming languages are the first-generation languages.
3. Assembly language is the second-generation languages.
4. Python can be used as a declarative language.
5. It is possible to check the content of memory, CPU and storage devices.

D. Short answer questions

1. What are the features of low-level language?
2. What are the features of high-level language?
3. What is syntax and semantics of the high-level programming languages?
4. What are the factors for selection of programming language?
5. What is the difference between procedural and declarative approach?
6. What is the difference between compiler and interpreter?
7. List the programming languages that uses compiler, interpreter and both.
8. Give the examples of declarative programming languages.
9. Explain the inheritance with real life examples.
10. What is polymorphism?

Module 2**Operating System and
Computer Network****Module Overview**

Today, we are all living in the digital world, where electronic devices have become an important part of our day-to-day life. All computing devices, such as computers, smartphones, tablets, and motor vehicles run on operating systems. These devices perform the actions as instructed by a user. These instructions are executed by the operating system. An operating system acts as an interface between user and computing devices.



An Operating system (OS) is important software for every computing device. An OS is an integrated set of programs that directs and manages the components and resources of a computer system, including main memory, CPU and the peripheral devices. The OS is somewhat like a house keeper as it organizes and maintains the functioning of various devices. The task of an operating system is to manage the hardware optimally to achieve the best possible performance. This is accomplished by the operating system's controlling and coordinating resources such as CPU, and memory. The hardware provides computing power and the operating system provides interface between a computer's hardware and the applications that run on it.

In the recent age, computing devices and their uses have grown rapidly and widely throughout the world. The applications and influence of computing devices can be seen in various sectors, including education, healthcare, transportation, and communication sector.

It is essential to understand the functioning of operating systems for software developers. This unit provides the basic knowledge of operating system that is required for software development. You will learn about what is an operating system, structure of operating system, the functions of an operating system, different types of operating systems.

Learning Outcomes

After completing this module, you will be able to:

- Describe the basic concepts of operating system
- Describe the basic concept of computer networks
- Describe the transmission media and network protocols
- Describe the features of network security

Module Structure

Session 1. Operating System

Session 2. Computer Networks

Session 3. Transmission Media and Network Protocols

Session 4. Network Security

Session 1. Operating System

An operating system (OS) is software that runs on your computer. It is responsible for managing software applications programs and computer hardware resources. It acts as a bridge between hardware, software, and the user for easy interaction to complete the task effectively. It allows you to communicate with the computer hardware, execute application programs and act as an intermediary between a user of a computer and the computer hardware. It manages the basic tasks such as memory management, process management, file management, input-output management, and also controls the peripheral devices.

Let us understand it with an analogy. (Figure 1.1) When you go to a doctor for a check-up, he examines you and write diagnosis in medical term that you cannot understand. But he makes you understand the problem in your language and prescribes the medicines. These medicines are as per your interaction with the doctor. It will not suit any other patient with similar symptoms.

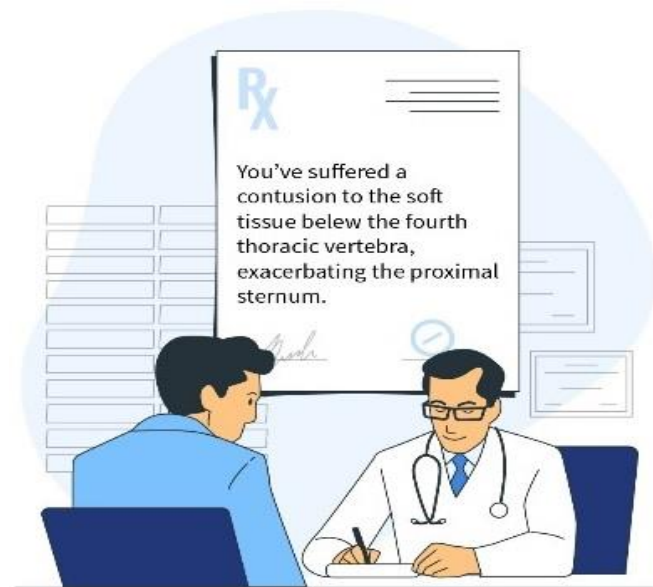


Fig. 1.1: Analogy of operating system

In this chapter you will understand the role of operating system, structure of operating system, and functions of operating systems. You will be able to use operating systems to perform the basic tasks.

1.1 Overview of Operating System

It is necessary to have at least one operating system installed in the computer to run the software and carry out various tasks. When you start a computer, Power-On Self-Test (POST) routine check is done to check the working of peripheral devices. Thereafter, the operating system, that is stored in Read Only Memory (ROM) is loaded on the primary memory. It remains in the memory till your system is shut down.

The operating system of a computing device provides a platform to perform various activities on computer and run different applications. Many times, several different programs run simultaneously on your device and they all need to access your computer's central processing unit (CPU), memory, and storage. The operating system coordinates all of this to make sure each

program runs properly. It is essential for the security and stability of the computer, as it protect the computer from malicious threats. It also allows to access hardware.

There are various operating systems designed by various companies for various types of computers.

Desktop Operating systems – These are designed for use on personal computers. They contain all the utilities and applications that the users might need. They are usually updated with the latest software versions to provide a better user experience.

Mobile Operating systems – These are designed for use on cellular phones and other portable devices. They contain only the essential software and minimalistic utilities. The advantage of using a mobile OS is that it is easier to update and maintain.

Server Operating systems – They run applications such as email servers, file sharing servers, and web servers. These are updated with the latest software versions, versioning to support multiple users, and security features.

It also possible for developer to develop their own operating system. Following are some of the well established operating systems that are being used on various types of computing devices.

Microsoft Windows – GUI based OS for Personal Computers.

Apple macOS – for Apple’s personal computers and workstations.

Google's Android OS – for smartphones/tablets/smartwatches.

Apple iOS – for Apple’s for iPhones, iPads, and iPods.

Linux – for Personal Computers, Workstations

1.1.1 Working Environment of a computer system

The working environment of a computer system consists of four layers – hardware, operating system, application programs and users. As shown in Figure 1.2, the applications interact with both the user and the operating system. The operating system interacts with applications and hardware. It controls and manages hardware resources from both applications and hardware layers.

The user works on installed applications and it can interact with hardware only through OS.



Fig. 1.2 Layers in computer system

1.1.2 Structure of an operating system

The *kernel* and *shell* are the two important programs of an operating system. The kernel interacts with hardware as well as with the shell. The user interacts with the kernel through shell. Any command given by the user first goes to the shell. The shell interprets it and sends it to the kernel. The kernel processes the request and displays the result on the screen. Figure 1.3 shows that the kernel is close to the hardware while shell can interact with kernel and user.



Fig. 1.3 Structure of operating system

1.1.3 Kernel – Kernel is the core of an operating system that interacts with hardware and manages system resources. It acts as a bridge between the software and hardware of the computer. Kernel is the first program that is loaded after the bootloader, and it remains in the memory until the operating system is shut-down. It offers secure access to the computer hardware and manages the system resources for various programs. It manages low-level tasks such as disk management, task management, and memory management. It also decides when and how long a certain application uses specific hardware.

1.1.4 Shell – Shell provides an interface for users of an operating system to access the services of a kernel. It is the interface between kernel and user. Shells are of two categories – Command Line Interface (CLI) and Graphical User Interface (GUI). The primary purpose of the shell is to invoke or launch another program.

Graphical user interface (GUI) – It offers graphical icons, and visual indicators to fully represent the information and actions available to a user.

Command line interface (CLI) – In CLI, the textual commands are executed on the command prompt. It usually returns output to the user in the form of text lines on the CLI. CLIs are generally used by programmers.

1.2 Functions of Operating System

Operating system provides various services to the user as well as to the application programs. It acts as a resource manager. For example, Windows users can access the Microsoft Store to download and install apps, while macOS users can access the App Store to download apps. For example, Windows users can access the Microsoft Store to download and install apps, while macOS users can access the App Store to download apps. There are four primary resources of the computer – processor or CPU, primary memory or RAM, device and file system. Operating system primarily performs management of these resources. It allocates the resources when required, helps to execute the process and de-allocate the resources after completion of the given task. The allocation of resources such as CPU, Memory and Disk to several processes running on the computer at that particular time can be viewed in operating system. In Windows, open task manager by pressing *Ctrl+Alt+Del* keys to view the allocation of resources as shown in Figure 1.4 (b). In Ubuntu Linux it can be seen in System Monitor as shown in Figure 1.4 (b).

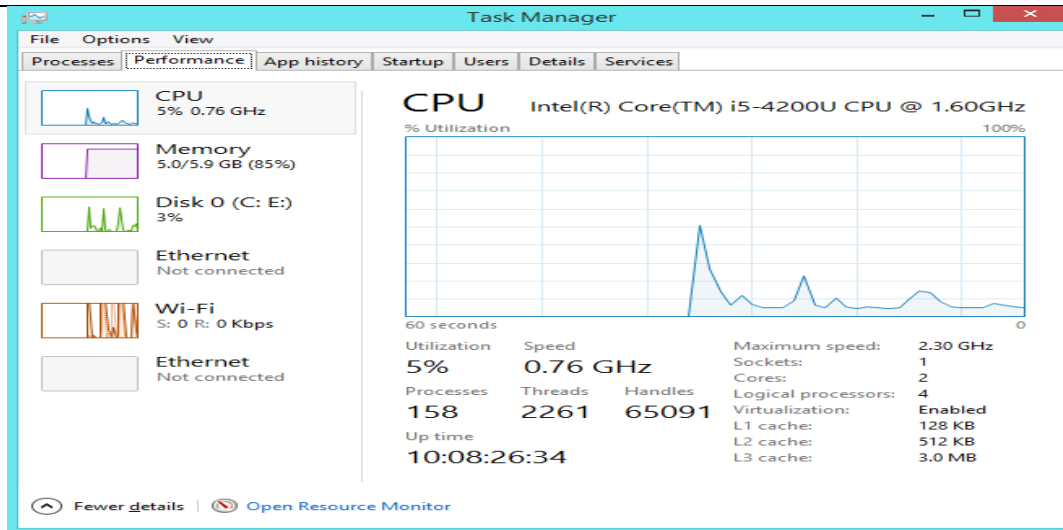


Fig. 1.4 (a) Allocation of resources as seen in Windows Task Manager

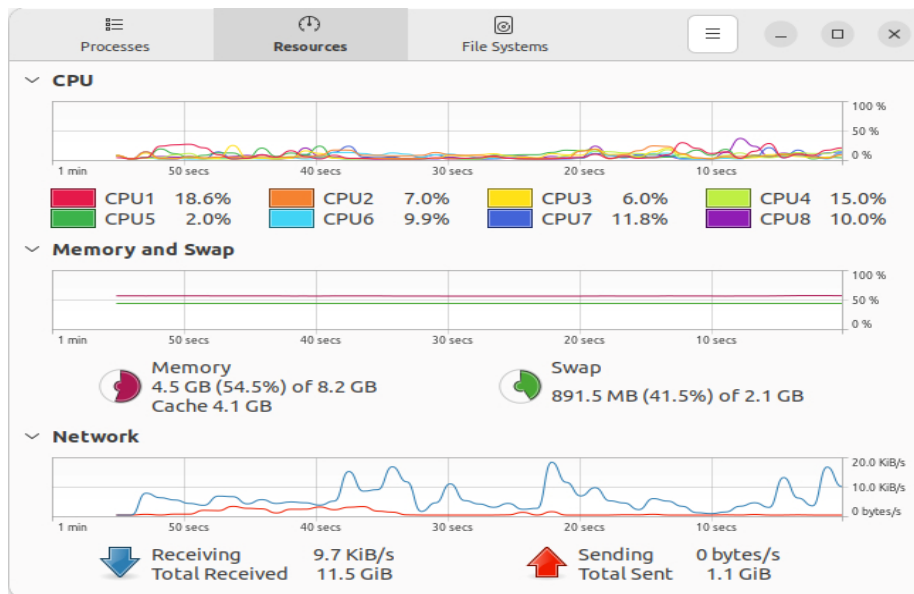


Fig. 1.4 (b) Allocation of resources as seen in Ubuntu Linux System Monitor

Several functions performed by operating system are discussed below.

1.2.1. Processor Management

Processor is an important component of computer system and it should be optimally utilised to execute various processes. It is responsible for execution of process. The operating system determines the amount of processing time allocated to each job. It also keeps tracks of the process status, which is run by the program known as the traffic controller.

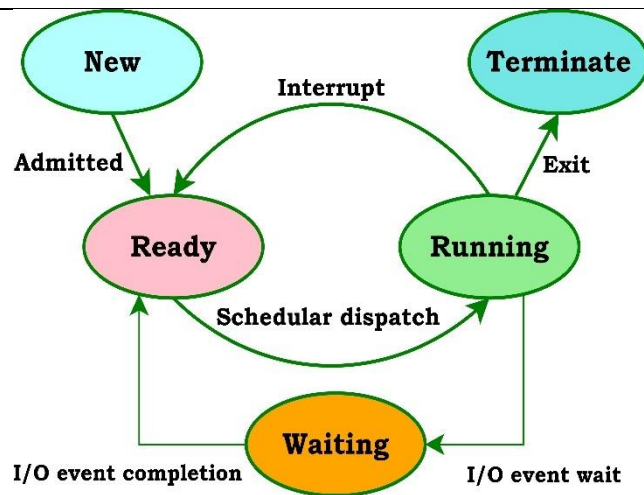


Fig. 1.5 States of process in scheduling

The Figure 1.5 depicts the life cycle of a process. When a new process is to be executed, it stays in the queue until it is ready to be dispatched. Once it is dispatched, it is in the running phase. If the other process has a higher priority than the existing one, then the process can be interrupted by another process in the queue. In case the existing process requires something like an Input from the user, it can be blocked temporarily until the event has occurred. Once the process is completed, it will be terminated.

The operating system allocates the processor to the process, and de-allocates the processor after the process is completed. When more than one process runs on the system the OS decides which process gets the processor, when and for how much time. This is known as processor or CPU Scheduling.

1.2.2. Memory Management

Memory management refers to the management of main memory. Main memory is directly accessed by the CPU and is allocated to execute the current program. After completing the execution of the process, the main memory should be released so that it is available for another process. As number of processes are executed by the processor at a time, it is required to manage the memory. In multiprocessing, it distributes the memory to various processes and the CPU decides which process will get memory, when will it get and how much time will it get. The operating system allocates the memory to the process for running as shown in Figure 1.6.

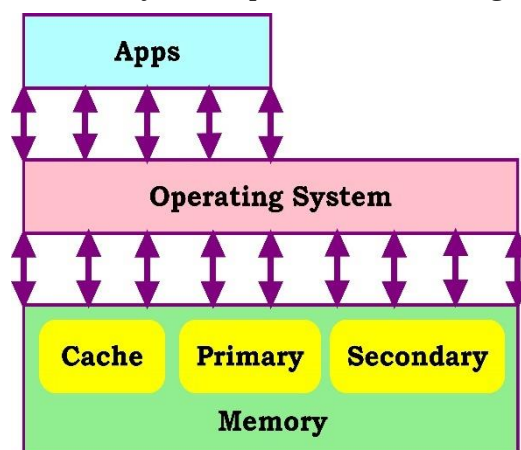


Fig. 1.6 Memory management

1.2.3. Device Management

Operating systems provide essential functions for managing devices connected to a computer. Operating system keeps track of all the devices attached to computer. The device can be a keyboard, mouse, printer, or any other devices attached to computer. It helps to communicate

with peripheral devices via their respective driver software as shown in Figure 1.7. It determines which process gets which device, and for how long. For example, to print a document, the operating system decides whether the printer has to be allocated or not and for how long. It also de-allocates devices when not in use. The program responsible for this task is known as Input/Output (I/O) controller.

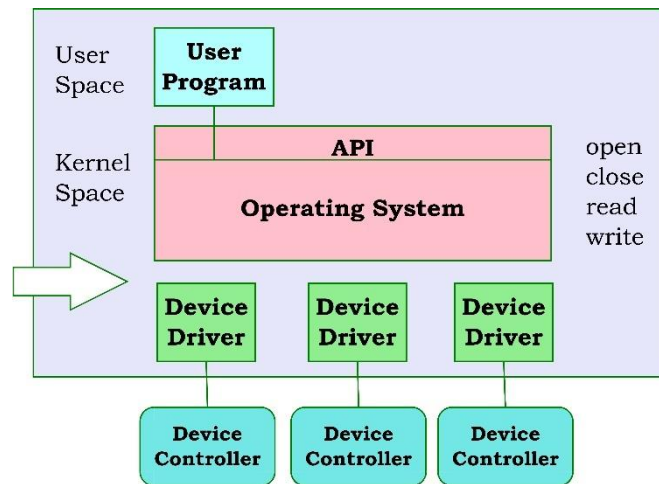


Fig. 1.7 Device management

1.2.4. File Management

A file is a collection of information stored in the memory of computer system. File management refers to the management of file system in secondary memory. The operating system performs the tasks such as creating, editing, updating, deleting, files and directories, mapping files into secondary storage and backing up files on non-volatile media. It keeps track of file system i.e. information, location, uses, status of files. It helps to store the file in folders and locate these files in the computer system, when required. It makes the process easy to share the files among the various user. It helps to manage the files according to their types and uses. Figure 1.8 shows the general hierarchy of file system. The root directory is present at the highest level in the hierarchical structure. It includes all the subdirectories in which the files are stored. Subdirectory is a directory inside another directory.

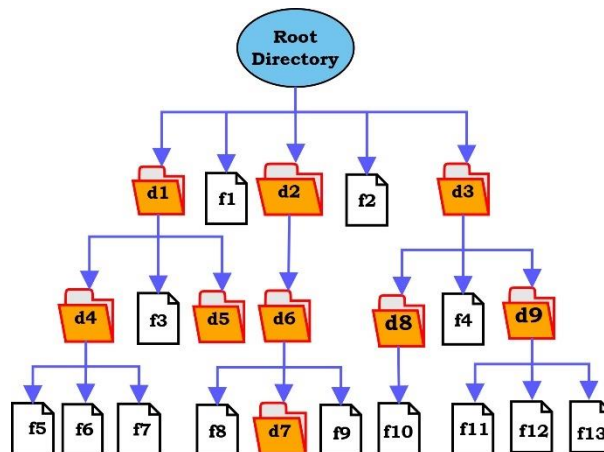


Fig. 1.8 File Management

1.2.5. I/O Management

The input/output process in computer is controlled by the Operating System. Figure 1.9 shows, that when the user instructs the system to play a music file using the music application. The OS first reads the audio file from the disk, which is the storage device to retrieve the input and then send the player to play. This produces the output on output device, i.e. the speakers. This is how, the I/O management function is performed by the operating system.

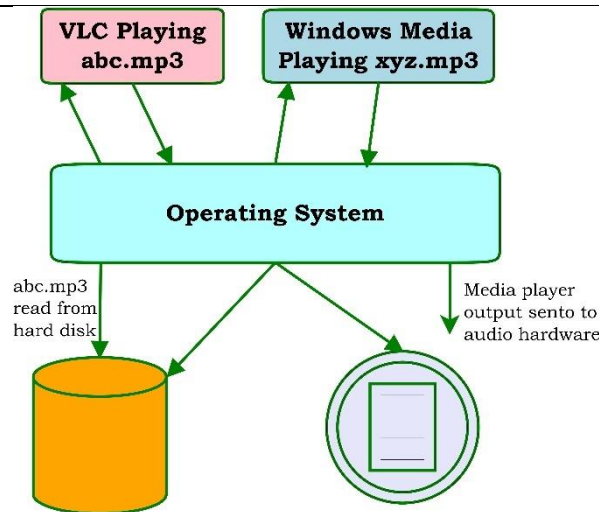


Fig. 1.9 I/O management

1.2.1. Secondary Storage Management

The primary storage of device is easily accessible and secondary storage can be connected and accessed using an operating system. The operating system store files and programs in a directory, which can be accessed quickly by the main memory.

Storage management helps to optimize the storage devices and improve the performance of the data storage resources and also protects data integrity on media. It is one of the important functions of operating system. The creation of files, and directories, reading and writing of data from files and directories, as well as copying the contents of files and directories from one location to another are all included in storage management.

When a process is to be executed then it is taken from secondary memory to main memory (RAM). But since main memory is limited the process is taken out after execution of some instructions and again taken into the RAM for execution of the remaining instructions. This process is called swapping. Thus it make the RAM free for other processes.

Swap-in means removing a program from the hard disk and putting it back in the RAM. Swap-out means removing a program from the RAM and putting it into the hard disk. Figure 1.10 shows the swap in and swap out of the two processes.

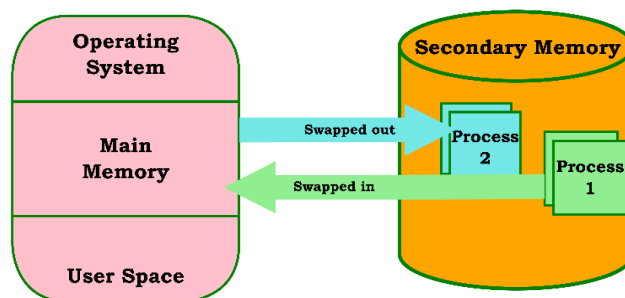


Fig. 1.10 Secondary storage management

1.2.7. Security

Security of data and system is an important concern while working on the computer. Computer viruses can be harmful and result in loss of data or crashing of computer. The OS has a number of built-in tools to protect against security threats, including the use of virus scanning utilities and setting up a firewall to block suspicious network activity. An OS also allows to set up a password so that only authenticated user can use that computer. It further extends to defending external I/O devices, including modems and network adapters, from invalid access attempts and to recording all such connections for detection of break-ins.

1.2.8. Command Interpretation and User interface

A user input provides the commands to computer in English language, but a computer understands only machine language or binary language of 0 's and 1 's. The operating system manages the interpretation of the commands to process the task.

Operating systems provides the user interface to interact the user with computer system. The user interface can be command-line interface (CLI) or graphical user interface (GUI). The CLI uses commands on the command prompt as shown in Figure 1.11 (a) and GUI uses the graphical icons as shown in Figure 1.11 (b) that can be operated using mouse. The actions such as clicking, selecting menu items can be performed in GUI.

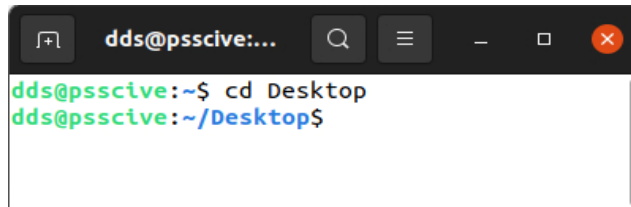


Fig. 1.11 (a) Command-line interface (CLI)

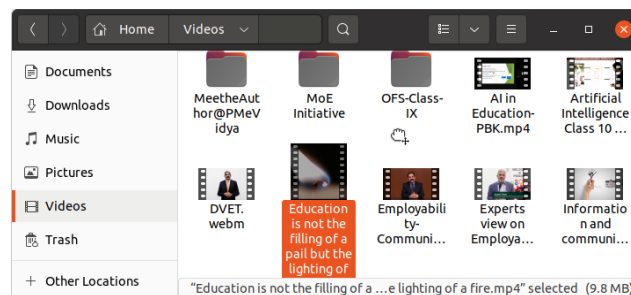


Fig. 1.11 (b) Graphical user interface (GUI)

1.2.9. Networking

Operating systems play a major role in networking. A network operating system such as Windows Server, Unix and Linux Server can be installed in the server computer and it can serve the connected client computers and peripherals as shown in Figure 1.12. A server computer is one that manages all the devices and resources on the network. In such a system, a large number of independent and physically separated computers can connect through a single communication channel and this is supported by the operating system.

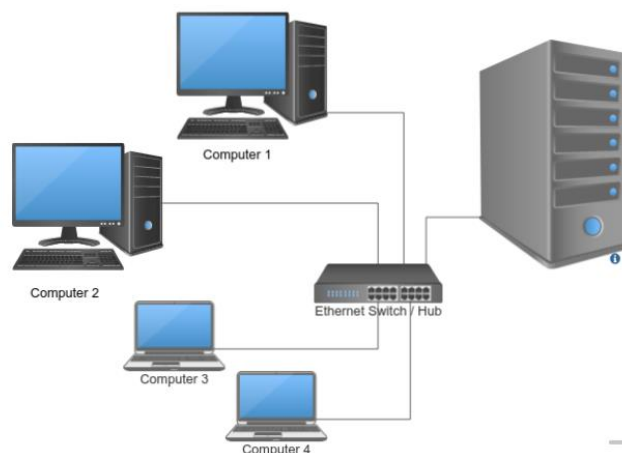


Fig. 1.12 Networking

1.2.10. Communications

There are many circumstances in which one process needs to exchange information with another process. Such communication may occur between processes that are executing on the same computer or between processes that are executing on different computer systems tied together

by a computer network. Communications may be implemented via shared memory or through message passing modules, in which packets of information are moved between processes by the operating system.

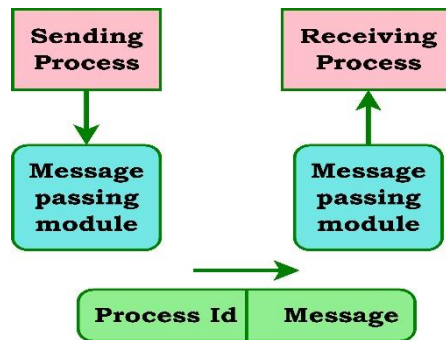


Fig. 1.13 Communication

Figure 1.13 shows that; two processes can communicate with each other. The sending process sends across a message via the message passing module which goes as a packet that contains Process ID and Message and is received by the receiving process.

1.2.11. Job Scheduling

An operating system keeps track of resources and time used by various jobs and users. It is easy to schedule a single task but when multiple tasks are taken into consideration, an operating system determines the order in which tasks are to be allocated resources. The priority of the jobs depends on various criteria such as order of arrival and priority status. Figure 1.14 shows how the jobs in the queue are being executed.

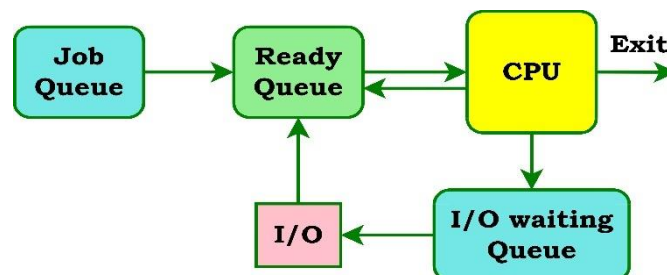


Fig. 1.14 Job Scheduling

1.2.12. Deadlock Detection and Prevention

A deadlock is a situation that can occur in an operating system when two or more processes, each waiting to release a resource. The system becomes unresponsive until one of the processes is killed. When a deadlock occurs, computer freeze up and even difficult to restart. This can cause you to lose important work or data and in some cases, may even damage your computer. It can be resolved by terminating one of the process. One of the important task of operating systems is to handle such deadlocks and resolve the issue.

Detecting deadlocks is one of the most important steps in preventing them. It is shown in resource-allocation graph in Figure 1.15.

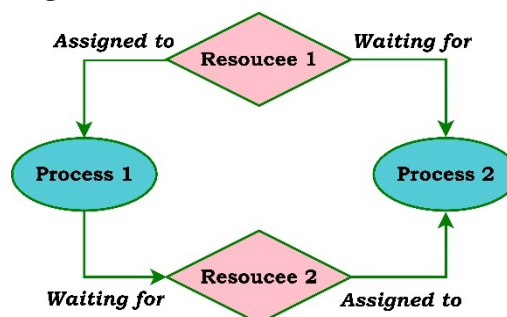


Fig. 1.15 Deadlock detection in Resource allocation graph

This graph checks if there is a cycle in the Resource Allocation Graph and each resource in the cycle provides only one instance. If there is a cycle in this graph then the processes will be in a deadlock state.

1.3 Types of Operating Systems

Operating system can be classified into various types on the basis of several criteria, viz. Number of users working simultaneously, number of simultaneously active programs, number of processors in the computer system. Modern OS such as Windows and Linux that come for single user called as Desktop OS. Windows Server and Cent OS Server OS which comes for server provides services to multiple users called as Network or Server OS. OS can be open source or proprietary. Windows is proprietary OS developed by Microsoft corporation, while Linux is an open source OS developed by open source foundation. The mobile devices such as iPad, Tablet and smartphones have mobile OS such as Android and iOS. While considering various other criteria, there are several types of operating systems.

Operating system are also classified as 32-bit and 64-bit. There are two types of processors exists in computer – 32-bit and 64-bit processors. 32-bit OS can address maximum of 2^{32} memory addresses and 64-bit OS can address 2^{64} memory addresses. All modern operating systems are 64 bits OS.

1.3.1. Single-User operating system

It is intended to be used by a single user in a single computer at a time. Figure 1.16 shows the process flow in single user operating system. It can only run one program or application at a time. Since it is being used by a single user at a time, the processor is not fully utilised.

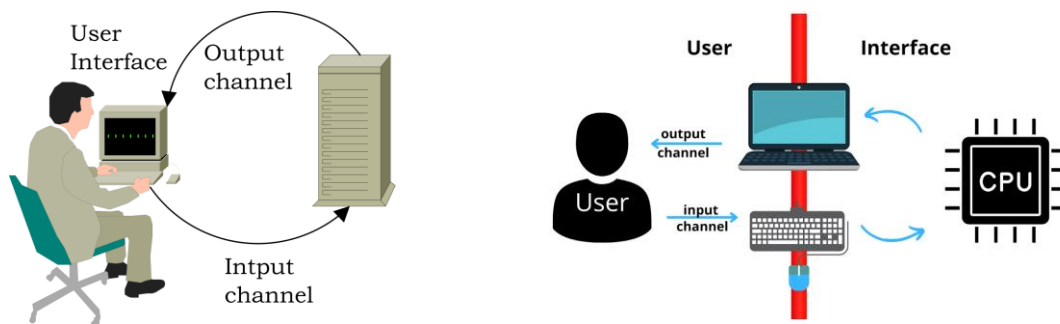


Fig. 1.16 Process flow in single user operating system

A single user operating system can be single tasking or multitasking.

a. Single-User Single-Tasking operating system

The single-user operating system allows a single user to perform only one task at a time. Any one task such as executing the command, editing a document or printing a document can be performed at a time. It occupies less space in memory. MS-DOS is an example of single user OS. Its interface screen is shown in Figure 1.17.

```
Microsoft(R) MS-DOS(R) Version 6.30
(C)Copyright Microsoft Corp 1981-1995.
A:\>ver /r

MS-DOS Version 6.30
Revision A
DOS is in low memory
A:\>
```

Fig. 1.17 MS DOS single user, single tasking operating system

b. Single-User Multi-Tasking operating system

It is designed for a single user to perform multiple tasks simultaneously. For example, when a user is working on a document, one can simultaneously play music, open file manager and print the document, all at the same time.

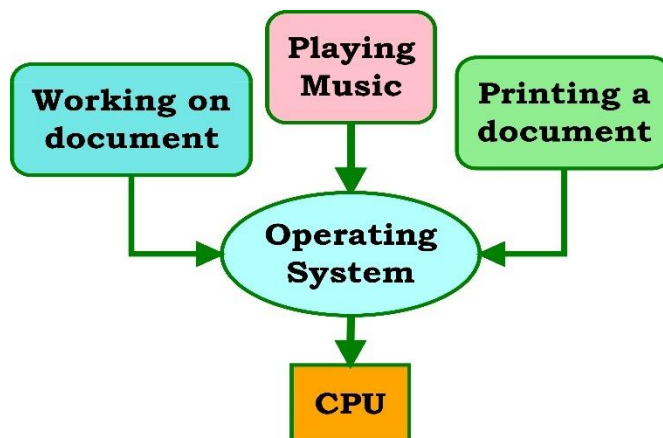


Fig. 1.18 (a) Multitasking operating system

It is possible to perform multiple tasks simultaneously. To achieve this the processor time is divided amongst different tasks. Windows, Linux and Mac OS are the examples of single user multitasking OS. Figure 1.18 shows the multiple tasks are being performed by the user in Ubuntu Linux.

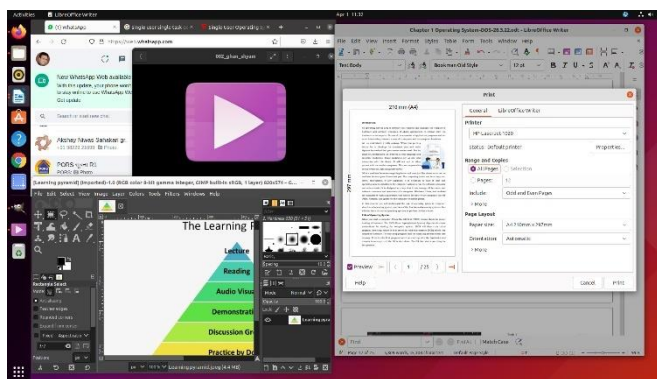


Fig. 1.18 (b) Multitasking in Ubuntu Linux

1.3.2. Multiuser Operating System

It allows multiple users to access a computer system simultaneously. It is used in networking where data and applications are accessed by multiple users at the same time. Time-sharing systems can be classified as multi-user systems as they enable a multiple user to access a computer through time sharing. The server OS, such as Windows Server, Linux Cent OS and Unix are examples of multiuser OS. They are also multitasking OS. Figure 1.19 shows one such multiuser OS.

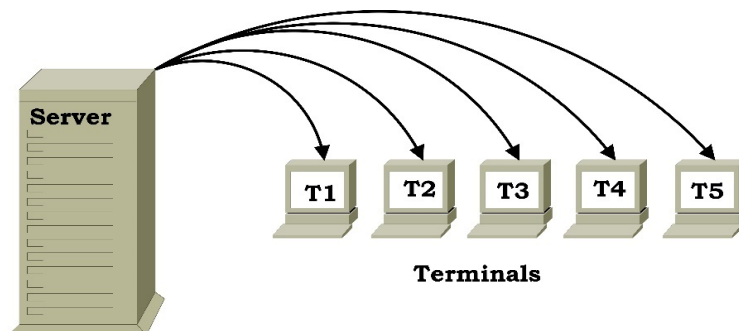


Fig. 1.19 Multiuser operating system

1.3.3. Batch operating systems

Batch operating system is used to fully utilise the power of processor in early days. These types of operating systems are not used nowadays. An operator collects the jobs and groups them together into a batch as shown in Figure 1.20. It automatically keeps executing one job after another in a batch which is taken care by batch monitor. The batch monitor accepts batch initiation commands from the operator. It processes each job in the batch, and then moves to the next batch and executes each job in batch till termination of last batch. To speed up the processing, the similar jobs are batched together and run on the computer as a group. Batch operating system is appropriate for lengthy and time-consuming tasks. It is used for tasks such as managing payroll systems, data entry and bank statements.

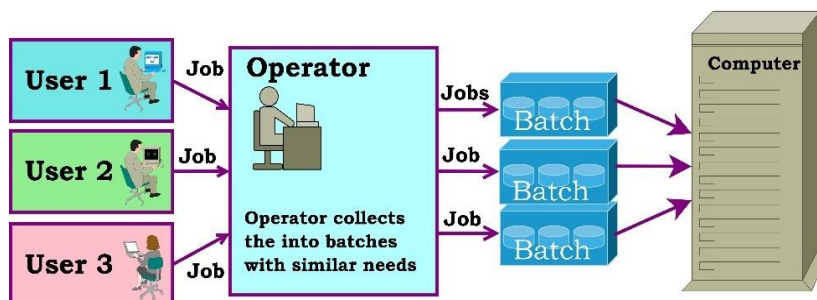


Fig. 1.20 Batch operating systems

1.3.4. Multiprogramming Operating System

The batch processing system tries to reduce the CPU idle time through executing jobs in batches. But still the CPU is idle during IO operation. Multiprogramming operating system eliminates the CPU idle time by providing multiple computational tasks. The CPU performs other jobs until the IO operation is being carried out for the current job. The supervisor manages all the activities simultaneously. The multiprogramming operating system can run multiple processes on a single processor.

The different programs to be executed are kept in the ready queue. The CPU execute one by one process. If one process gets blocked then other processes from the ready queue are assigned to the CPU. In this way CPU can be used optimally. Figure 1.21 shows that, different processes are there in RAM(main memory). Some processes are waiting for the CPU, and process 2, which was previously executing, is now doing I/O operations. So CPU shifted to execute process 1. Thus, the multiprogramming operating system manages all the activities simultaneously.

UNIX, VMS, and Windows-NT are the examples of multiprogramming operating systems.

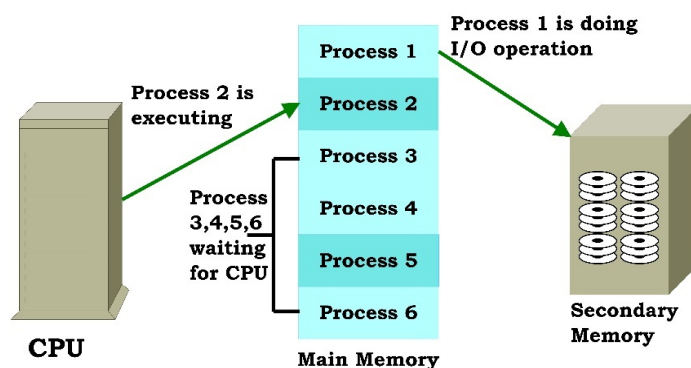


Fig. 1.21: Multiprogramming operating system

1.3.5. Time-Sharing Operating System

Time sharing is a logical extension of multiprogramming and multitasking operating system. It is more complex than multi-programming OS. It provides a shared user interface with multiple users logged in simultaneously. It allows the user to perform multiple tasks at a time. But, the switching between the tasks is very fast due to which the user feels that all tasks are running at the same time. This type of operating system is most commonly used in businesses, especially those that involve a high number of simultaneous users. The CPU is multiplexed rapidly among

several programs. Time sharing system reduces the CPU ideal time. The operating system performs time sharing through CPU scheduling and multiprogramming. It is latest advancement in the computer science, accepted worldwide, at increasing rate. Figure 1.22 shows the working of time-sharing system. The process of user 6 is in the active state, user 5 is in a ready state while user 1, 2, 3, and 4 are in a waiting state.

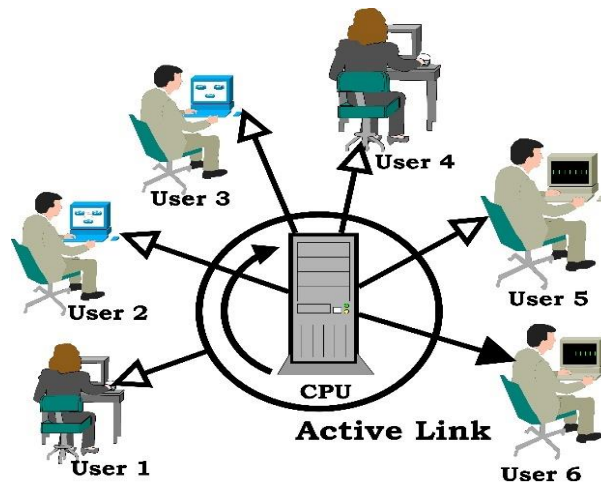


Fig. 1.22: Time sharing operating system

1.3.1. Multiprocessing or Parallel Operating System

There are several complex applications, where processing of multiple processes is required to be performed by multiple processors. A multiprocessing operating system uses more than one processor. For example, to perform weather forecasting, huge set of parameters are required to be processed through the computer system. A single CPU will require lot of time to perform such task. Therefore, multiprocessing or parallel OS is required in such applications. Windows, Linux and UNIX support for multiprocessing. Figure 1.23 shows multiprocessing with three CPU. OS manages three different processes running on different processors. Other processes D, and E are kept in queue by OS. This is because there is no processor available for their execution. This is how OS manages multiple processes in the computer system.

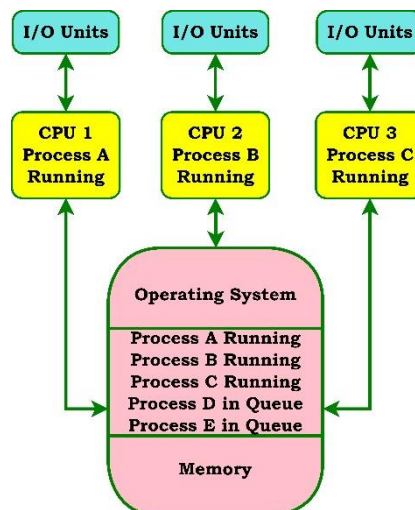


Fig. 1.23 Multiprocessing operating system

1.3.7. Distributed Operating System

This operating system is designed to operate on a network of computers. Distributed systems are usually used to distribute software applications and data. Distributed systems are also used to manage the resources of multiple computers. Several computers are connected through a single communication channel as shown in Figure 1.24. Every system has its own processor

and memory. Resources like disk, computer, CPU, network interface, nodes are shared among different computer at different locations. It increases data availability in the entire system.

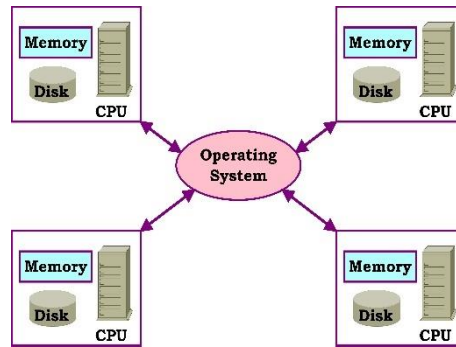


Fig. 1.24 Distributed operating system

These processors communicate through high-speed buses or telephone lines. These individual systems that connect through a single channel are considered as a single unit. The distributed operating system manages group of independent computers and makes them appear to be a single computer. Amoeba and LOCUS are some examples of distributed operating systems.

1.3.8. Client/ Server Network Operating System

In client/server system, two or more computer systems are linked through a telecommunications network. Clients are the computers that use the network to access services provided by the server. Servers are the computers that provide the services to the network. The server and client computers must have certain software installed to connect to each other securely over a network connection. A network operating system, similar to distributed operating system is installed on server computer, which provides the capability to manage data, users, groups, security, applications, and other networking functions. The servers host the applications or services for users while clients use these applications. It allows shared access of files, printers, security, applications, and other networking functions over a small network of computers like LAN. In 1983, Novell NetWare was released as the first network operating system. Figure 1.25 shows how the client systems are connected to the server in network OS.

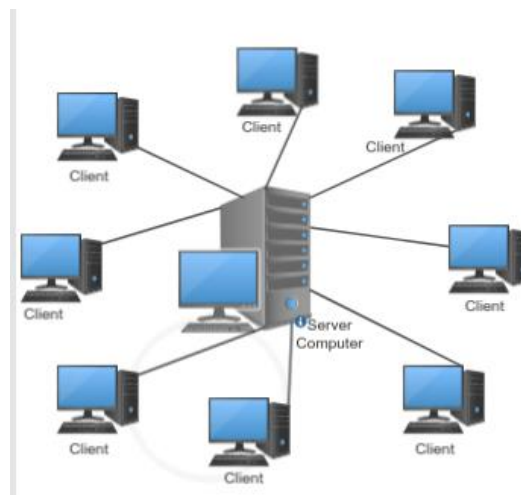


Fig. 1.25 Client/ Server Network operating system

The other examples of network operating system are Microsoft Windows Server 2019, UNIX, Linux, Mac OS X, Novell NetWare, and BSD.

1.3.9. Real time operating system

It is a multitasking operating system that aims to execute real-time applications quickly. The time interval required to process and respond to inputs is very small. Consider the example of a reservation system. The system should perform the given task in a fixed time otherwise; it results in a system failure. The real time operating systems are used when there are time requirements

are very strict like missile launching, air traffic control systems, robots. Windows CE, OS-9, Symbian and LynxOS are some of the commonly known real-time operating systems.

1.3.10. Embedded Operating System

This type of operating system is designed to be compact so that it is efficient for embedding in small machines with less autonomy, such as PDAs. The hardware running an embedded operating system is usually very limited in resources. It is designed to perform a specific task without much user intervention. It is usually 'bundled' together into a single executable image. It does not generally load and execute individual applications at the user's request. ATMs and Satellite Navigation systems are examples of embedded operating systems. (Figure 1.26) *Windows CE, FreeBSD and Minix 3 are some examples of embedded operating systems.*



Fig. 1.26 Embedded operating systems

1.3.11. Mobile Operating System

This is a special type of operating system that is designed to control mobile devices such as tablets and smartphone. Its design supports wireless communication and mobile applications. It allows to install the apps, that are same as application software installed on a computer. It has built-in support for mobile multimedia formats. Blackberry OS, Google's Android and Apple's iOS are some of the most commonly used mobile operating systems.

1.4 Examples of Operating System

Operating systems are normally preloaded on the computer that you purchase. But it is possible to upgrade or install operating system on any computer. Some of the common desktop operating systems are — Microsoft Windows, Mac OS and Linux. Apple iOS and Google Android are commonly used for mobile operating systems.

1.4.1 Microsoft Windows is a graphical user interface (GUI) operating system. In this GUI system all the programs or commands of operating system are available in the form of icons, buttons and menus. Whenever we want to execute any command or program, the corresponding icon needs to be clicked. There are various versions of Microsoft Windows. Windows 11 version released in October, 2021 is the latest version. Windows 10, released in 2015 is the most commonly used today. The earlier versions are Windows 8 released in 2010 and Windows 7 released in 2009. Microsoft Windows is one of the most popular single user operating system. Figure 1.27 shows typical desktop of Microsoft Windows 10 operating system.

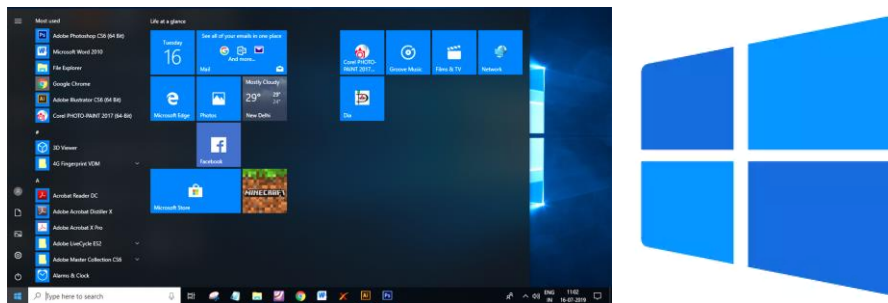


Fig. 1.27 Windows 10 Desktop

1.4.2 Linux is a family of open source operating systems. GNU/Linux is the base of many open-source Oss. GNU/Linux is a set of programs/utilities and a kernel that many open-source OSs share. It means that this OS can be modified and distributed by anyone. Windows and Mac OS are proprietary, means they can be modified only its company. It is necessary to purchase a user license of proprietary software to use it on computer system. Linux is a freeware, you need not to pay any cost and you can use it for free on your computer.



GNU/Linux is the base for many open source operating systems called "*distributions*" or "*distros*" in Linux. They are based on the same kernel and set of utilities with different "flavors" of Linux. For example, we have Debian, Mint, Ubuntu, Fedora, Suse, Red hat Enterprise Linux as the different distribution names of the Linux.

Linux is also available in the form of GUI. Every program in the Linux OS is displayed in the form of icon, button or graphics. By clicking on the icon or button we can execute that appropriate program. Figure 1.28 shows the Ubuntu Linux desktop.

Ubuntu is released every six months, with long-term support (LTS) releases every two years. As of now, Ubuntu 23.04 (Lunar Lobster) is released on Thursday April 20, 2023. Ubuntu releases are given code names using an adjective and an animal with the same first letter. For example, Ubuntu 22.10 ("Kinetic Kudu"), 22.04 ("Jammy Jellyfish"), Ubuntu 21.10 (Impish Indri). Ubuntu 21.04 (Hirsute Hippo) are some of the versions released in recent years.



Fig. 1.28 Ubuntu Linux desktop

1.4.3 MacOS (earlier called OS X) is an operating system created by Apple. It's software and hardware are not compatible with Windows computers. It comes pre-installed on Macintosh computers. The first version of it was released in 1984 and it was the first OS for personal computers to come with a built-in GUI. The GUI of Mac OS is different from that of Microsoft Windows, as shown in Figure 1.29.



Fig. 1.29 Mac OS screen

All the commands and screen programs available in Mac OS are displayed in the form of icons or buttons. By clicking appropriate buttons, the program can be executed. There are various versions of Mac OS. Most recent version of Mac OS is OS X, which is pronounced as OS 10. As of now, the current version of macOS is **Ventura 13.2. 1**, released in February 2023. The earlier versions are macOS 12 (Monterey) released on October 25, 2021, macOS 11 (Big Sur) released on November 19, 2020, macOS 10.15 (Catalina) released on 7 October 2019

1.4.4 OS for Mobile Devices

The OS designed for desktop and laptop computers are not suitable for mobile devices such as smart phones and tablet computers. Any mobile operating system facilitates users to run other apps/software on the mobile, tablets, etc. Basically, these are special type of operating system which are designed for smart phones, tablets, smart watches, etc. These are the mixture of computer operating system with some other essentially required features for mobile device. These are comparatively very light and user friendly. The OS Apple iOS and Google android are designed for mobile devices are most commonly.



1.4.4.1 Apple iOS – This OS is commonly used in Apple mobile phones and iPads. In this, the programs or commands are displayed in the form of icons or graphical buttons. The screens of an iPhone and iPad are shown in Figure 1.30 (a) and (b).



Fig. 1.30 (a) iPhone screen



Fig. 1.30 (b) iPad screen

These are not fully featured OSs and therefore they are unable to run all the software that can be run on desktop or laptop computers. But still these OS are powerful and we can do a lot of things on mobile devices, such as browsing the web, watching movies, listening to music, playing games and many other things that we normally do on desktop.

As of now the current version of iOS 16 is iOS 11.3.1, released in February 2023. The earlier versions are iOS 15.4.1 released on 31 Mar 2022, iOS 14 released in 2020, iOS 13 released in 2019, iOS 12 released in 2018. Apple automatically updates OS on iPhones and iPads whenever the new version is released. Apple iOS is said to be the most secure OS on the mobile devices, as only a few viruses can affect Apple iOS.

1.4.4.2 Android OS – Google Android is another OS used for mobile devices, smartphones and tablet computers. It is specifically designed for mobile devices. It also has a graphical user interface (GUI) where the most of the programs are available in the form of icons or graphical buttons. Whenever we click on any buttons, the corresponding program gets executed. The screen of Android OS on mobile phones and tablets is shown in Figure 1.31 (a) and (b).



Fig. 1.31 (a) Android screen on mobile



Fig. 1.31 (b) Android screen on tablets

Android OS is one of the popular operating system in the world as many companies, such as Samsung, Google and even Indian mobile companies make use of this OS for their mobile devices. Android is designed to be flexible and adaptable to a wide variety of devices, from smartphones to digital cameras, watches, TVs, and cars. By using these devices, we can perform many functions, such as telephone call, browsing a website, identifying location by using maps, bank transactions and many other daily routine tasks. We can maintain a calendar, notes and presentations by using such mobile.

The Android started its version from Android 1.0 released in September 23, 2008, and the latest version of Google Android as of now is Android 15 (Vanilla Ice Cream), released in 2022. The previous versions were Android 14 (Upside Down Cake) released in March 8, 2023, Android 13 (Tiramisu) released in August 15, 2022, Android 12L (Snow Cone v2) released in March 7, 2022.

Assignment

Find out the various versions of operating system Windows, Linux, Mac OS, Apple iOS, and Google Android. Compare their distinguishing features.

Activity

Practical Activity 1.1. Perform the various tasks in any or all the operating system

Materials Required

Desktop or Laptop computer with Windows or Linux or Mac operating system,
Tablet or iPad with iOS operating system,

Smartphone with Android operating system,

Antivirus software

Printer, Scanner

Procedure

Step 1. Start the computer and wait till it loads the installed operating system. Login using user name and password.

Step 2. Identify and name the operating system installed in the computer.

Step 3. Identify the version of the operating system installed in the computing device.

Step 4. Identify and name the various components of the desktop of the operating system.

Step 5. Give the purpose of each components of the desktop as follows.

Step 1. Identify and name the applications installed in the operating system.

Step 7. Open the application and close the application.

Step 8. Open the word processing application and type a letter. Simultaneously, open the application to play the song. Observe the multitasking feature that you can work on multiple applications at a time.

Step 9. Save the letter with the name “Letter” on the desktop. Locate the file recently saved on the desktop. Note the extension name provided to the Letter.

Step 10. Now create a folder with the name “Myfolder” on the desktop. Observe the folder created on the desktop.

Step 11. Move the file “Letter” to the folder “Myfolder”.

Step 12. Perform the basic task such as create a file, save the file, change the name of the file, copy the file to the folder, change the name of the “Myfolder” to your name,

Step 13. Operating system provides some default folders such as *Downloads* to store the downloaded files from the internet, *Documents* to store the documents, *Pictures* to store the picture files, *Videos* to store the video files. Access these default folders from File Explorer.

Step 14. Search the required files in the File Explorer. Type initial name of a file or folder to search. The search results will appear above the search box as shown in Figure 1.32.

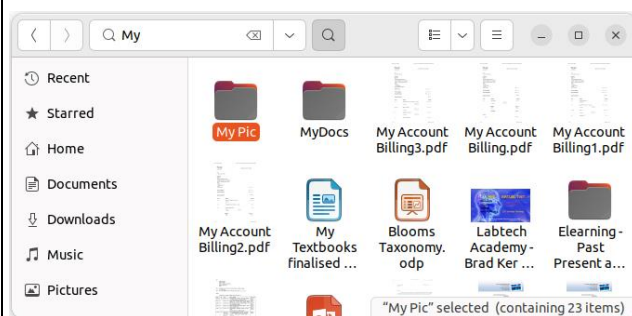


Fig. 1.32 Searching files in File Explorer

Step 15. Access the setting of various components of operating system and observe its parameters.

Step 11. In Windows 10, operating system, click on the Start button and the Settings icon.

Step 17. The Windows Setting window will open as shown in Figure 1.33. Observe the layout for various settings.

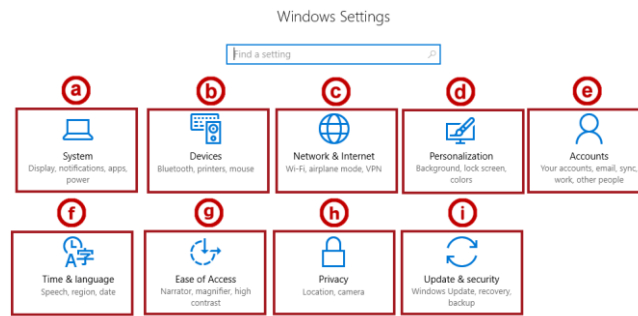


Fig. 1.33 Windows Settings

- Click on the **System** icon to view its current settings. Try to change the system setting and observe the change.
- Click on the **Devices** icon to view its settings. Add and manage external devices such as printers.
- Click on the **Network & Internet** icon to view its settings. Manage network and internet settings.
- Click on the **Personalization** icon to view its settings. Personalize the desktop by changing the desktop appearance, apply themes, change the lock screen and observe the change.
- Click on the **Accounts** icon and view information about your user account on the computer.
- Click on the **Time & language** icon and view the present time zone and language. Change time zone and language and observe the change.
- Click on the **Ease of Access** icon to view and manage computer accessibility options.
- Click on the **Privacy** icon to view computer privacy options.
- Click on the **Update & security** icon to View windows update status and backup/recovery options.

Step 17. To install a antivirus software on computer, follow the steps of its installation.

Step 18. Download the latest version of Python and install it on your computer.

Step 19. Download the latest version of MySQL and install it on your computer.

Step 20. To uninstall the software, follow the steps to uninstalling a software.

Step 21. To install a printer, scanner or any other peripheral devices, follow the steps to install it using its device driver software.

Step 22. Perform all of the above operations on other operating systems.

Check Your Progress

A. Multiple choice questions

- Which of the following is the core component of operating system (a) applications (b) kernel (c) shell (d) user interface
- Which of the following is not a network operating system (a) Windows 10 (b) Windows Server (c) Unix (d) Linux Server
- Which of the following is open source operating system (a) Windows (b) Mac (c) Unix (d) Linux
- Operating system are classified on the basis of (a) number of users working simultaneously (b) number of simultaneously active programs (c) number of processors in computer (d) All of the above

5. Which of the following is not a multitasking operating system (a) Windows (b) Mac (c) MS DOS (d) Linux
6. In which type of operating system, the CPU time is distributed among all the running processes (a) Multiprogramming OS (b) Time sharing OS (c) Distributed OS (d) Network OS
7. Which of the following operating system uses number of CPU to process the complex tasks (a) Multiprogramming OS (b) Multiprocessing OS (c) Distributed OS (d) Network OS
8. Which operating system reads and reacts in terms of actual time? (a) Time sharing system (b) Quick response system (c) Real time system (d) Batch system
9. Which of the following operating system cannot be used in mobile devices (a) Windows (b) Android (c) Apple iOS (d) None of the above
10. Which of the following operating system is not used in desktop computers (a) Windows (b) Linux (c) Android (d) Mac

B. Fill in the blanks

1. The user can interact with computer through _____ (operating system)
2. The _____ and _____ are two important programs of an operating system. (shell, kernel)
3. Time sharing is extension of _____ and _____ operating system.
4. Multiprocessing operating system uses multiple _____
5. The _____ is the main workspace to view and manage files.
6. The keyboard shortcut Ctrl+X and Ctrl+V is used for _____
7. The file deleted once can be recovered from the _____ back to its original location.
8. Folder may contain _____ and _____
9. For installation of any software, you need to arrange the _____ file of that software.
10. User Account Control acts as a _____ from running any software that can modify the Windows system to protect itself.

C. State whether True or False

1. Kernel interact with hardware and shell interact with kernel and user.
2. Shell is an interface between kernel and user.
3. Peripheral devices communicate computer with their respective drivers.
4. Memory management function of operating system manages secondary storage memory.
5. In multiprocessing, memory management allocates equal memory to each process.
6. Time sharing operating system serves the purpose of both multiprogramming and multitasking operating system.
7. Batch operating systems is appropriate for executing the complex tasks.
8. Multiuser operating system can have the accounts of multiple users.
9. Networking operating system allows to share files, printers, and applications.
10. In distributed operating system, memory is distributed among several CPU.

D. Short answer questions

1. List the functions of operating system.
2. What is process scheduling?
3. What are the tasks performed by file management?
4. What are the tasks performed by secondary storage management?
5. What is job scheduling?
6. Differentiate between CLI and CUI.
7. What is the difference between Distributed OS and Network OS?
8. What is the difference between hard real time OS and soft real time OS?
9. What are the common tasks performed in Windows 10?
10. What are various operating systems for mobile devices?

Session 2: Computer Networks

2.1 Introduction to Computer Networks

We are living in a connected world. Information is being produced, exchanged, and traced across the globe in real time. In the digital world, everyone and everything is interconnected through one way or the other. A group of two or more similar things or people interconnected with each other is called network. In family network, family member shares their resources and information. A *telephone network* is an infrastructure for sending voice signals from one telephone to another. A *social network* is a group of people connected through social media such as Facebook, Tweeter and WhatsApp. Figure 2.1 shows an example of social network.

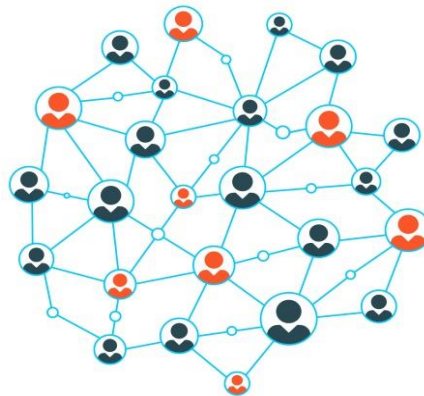


Fig. 2.1 Interconnection forming a social network

A computer network is an interconnection among two or more computers or computing devices. Such interconnection allows computers to share data and resources among each other. Each device in a network that can receive, create, store or send data to different network routes is called a node. Apart from computers, networks include networking devices like switch, router, and modem. Networking devices are used to connect multiple computers in different settings. A basic network has been shown in Figure 2.2, connecting few computers.

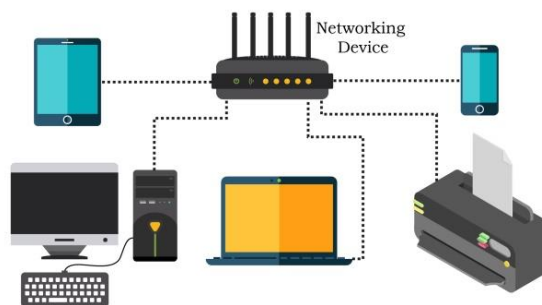


Fig. 2.2: Computer network

The network size may vary from small to large depending on the number of computers it connects. A computer network can include different types of hosts, also called nodes, like server, desktop, and laptop.

For communication, data in a network is divided into smaller chunks called *packets*. These packets are carried over a network. Devices in a network can be connected either through wired media like cables or wireless media like air.

Interconnectivity of computing devices in a network allows to exchange information simultaneously with many parties through email, websites, audio and video calls. Network allows sharing of resources. For example, a printer can be made available to multiple computers through a network; a networked storage can be accessed by multiple computers. People often connect their devices through hotspot, thus forming a small personal network.

2.2 Types of Networks

There are various types of computer networks connected through wired or wireless media. Computer networks are broadly categorised as LAN, WAN and MAN based on the geographical area covered and data transfer rate. There are many other types of network such as Wireless Local Area Network (WLAN), Personal Area Network (PAN), Campus Area Network (CAN), SAN (storage area network), VPN (virtual private network).

2.2.1 Local Area Network (LAN)

It is a network that connects computing devices such as computers, laptop, printers and scanners in a limited geographical area such as home, school, laboratory, office buildings, allowing them to share data and resources. The connectivity is done by wires such as Ethernet cables, fibre optics, or wireless Wi-Fi. A typical Local Area Network (LAN) is shown in Figure 2.3.

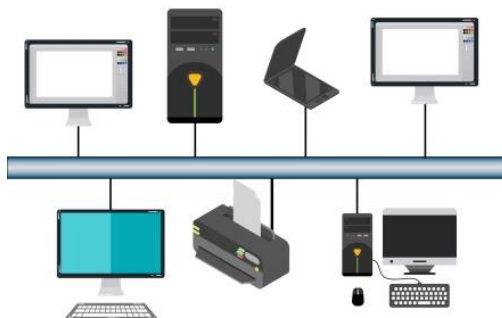


Fig. 2.3: Local Area Network

LAN is comparatively secure as only authentic users in the network can access other computers or shared resources. Users can print documents using a connected printer, upload/download documents and software to and from the local server. Such LANs provide the short range communication with the high speed data transfer rates. These types of networks can be extended up to 1 km. Data transfer in LAN is quite high, and usually varies from 10 Mbps (Megabits per second), called Ethernet to 1000 Mbps, called Gigabit Ethernet. Ethernet is a set of rules that decides how computers and other devices connect with each other through cables in LAN.

LAN that connect computers without wires, using radio frequencies or light is called as wireless LAN or WLAN. It usually connects wireless devices that are very close to each other. WLAN communicates with a wired LAN to access its resources.

2.2.2 Personal Area Network (PAN)

It is a network formed by connecting a personal computing device as shown in Figure 2.4. All these devices lie within an approximate range of 10 metres. A personal area network may be wired or wireless. For example, a mobile phone connected to the laptop through USB forms a wired PAN while two smartphones communicating with each other through Bluetooth or wi-fi hotspot technology form a wireless PAN or WPAN.



Fig. 2.4 Personal Area Network

2.2.3 Metropolitan Area Network (MAN)

Metropolitan Area Network (MAN) is an extended form of LAN which covers a larger geographical area like a city or town. Data transfer rate in MAN also ranges in Mbps, but it is considerably less as compared to LAN. MAN combines multiple LAN through a networking device called *Bridge*. A MAN usually is managed by network provider or local and state governments. Cable TV network or cable based broadband internet services are examples of MAN. This kind of network can be extended up to 30-40 km. Sometimes, many LANs are connected together to form MAN, as shown in Figure 2.5.

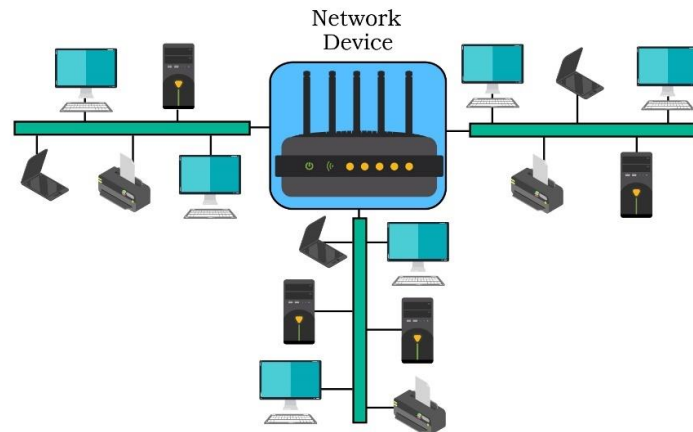


Fig. 2.5 Metropolitan Area Network

2.2.4 Wide Area Network (WAN)

Wide Area Network connects computers and other LANs and MANs, which are spread across large geographic area across the country or world, using communications channel wired or wireless media. A WAN can be one large network or can consist of two or more LANs connected together. The Internet is the world largest WAN that connects billions of computers, smartphones and millions of LANs from different continents.

For example, a company with locations in two different cities would normally set up a LAN in each building and then connect them together in a WAN as shown in Figure 2.6. WAN can be public or private. Bridge, Router and Gateway are the devices used in WAN. The *wireless WAN (WWAN)* uses wireless technology.

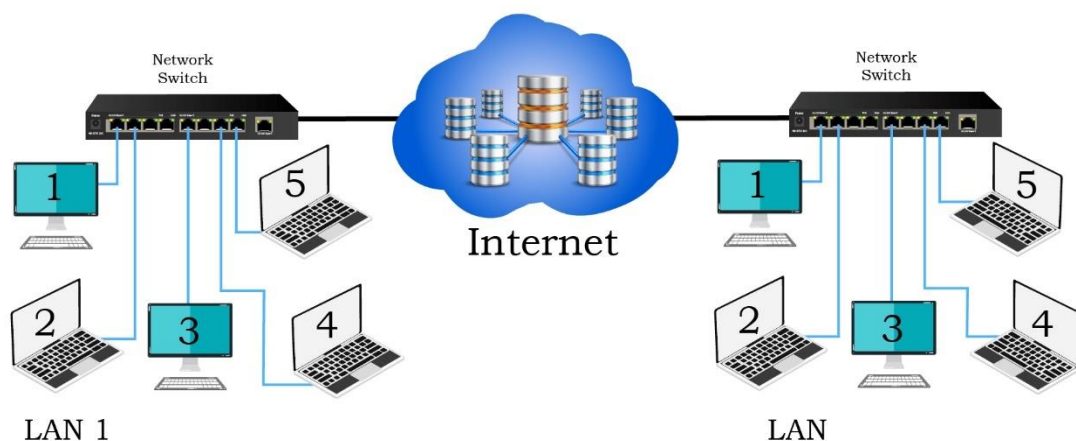


Fig. 2.6 Wide Area Network

Activity 1

Practical Activity 2.1 – Identify the types of network and draw the labelled diagram as shown in the following table.

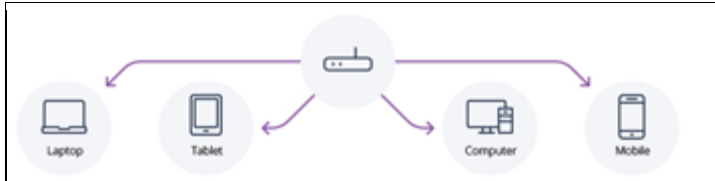


Fig. 2.7(a) Type of network:



Fig. 2.7 (b) Type of network:

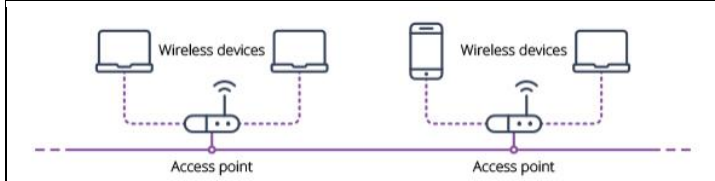


Fig. 2.7 (c) Type of network:

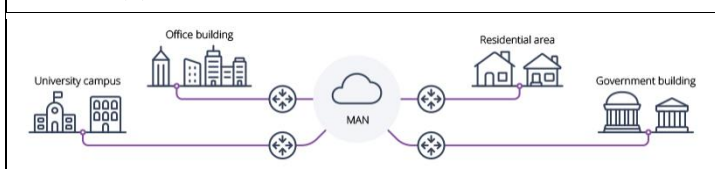


Fig. 2.7 (d) Type of network:

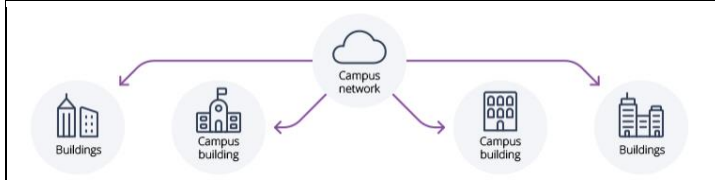


Fig. 2.7 (e) Type of network:

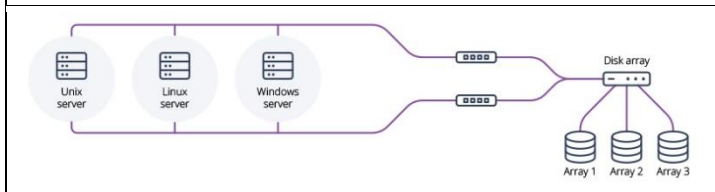


Fig. 2.7 (f) Type of network:

2.3 Network Devices

The network devices such as Modem, Hub, Switch, Repeater, Router, Gateway are required to configure networks data through different transmission media. Let us explore them in detail.

2.3.1 Modem

Modem stands for ‘*MODulator DEModulator*’. It refers to a device used for conversion between analog signals and digital bits. Digital computers store and process data in binary digits of 0s and 1s. However, to transmit data from a sender to a receiver, or while browsing the internet, digital data are converted to an analog signal and the medium carries the signal to the receiver. There are modems connected to both the source and destination nodes. The modem at the sender’s end acts as a modulator that converts the digital data into analog signals. The modem

at the receiver's end acts as a demodulator that converts the analog signals into digital data for the destination node to understand. Figure 2.8 shows connectivity using a modem.

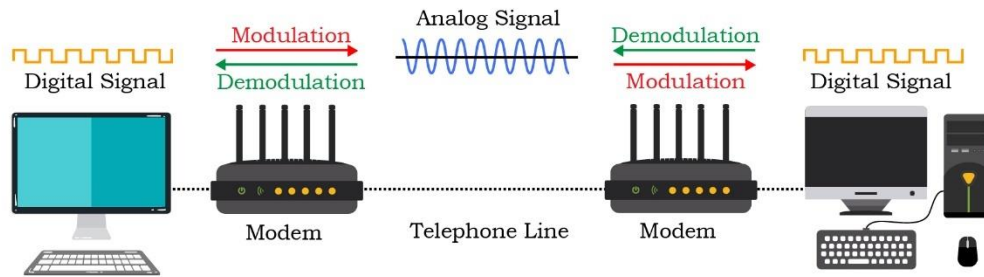


Fig. 2.8: Use of modem

2.3.2 Ethernet Card

Ethernet card, also known as Network Interface Card (NIC) is a network adapter used to set up a wired network. It acts as an interface between computer and the network. It allows to communicate with other networking device. The Ethernet cable connects the computer to the network through NIC. Ethernet cards can support data transfer between 10 Mbps and 1 Gbps (1000 Mbps). Each NIC card has its own unique physical address known as MAC address, which helps to uniquely identify the computer on the network. It has a 48-bit address and is represented as 12-digit hex-decimal number. The main components of NIC are PCI interface, PCI slot, RJ-45 slot, Ethernet processor and RAM/ROM socket. The typical NIC is shown in the Figure 2.9.

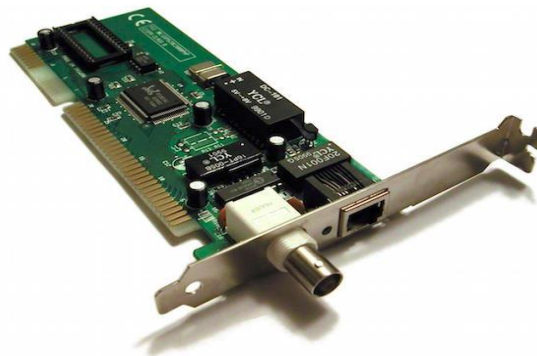


Fig. 2.9 Network Interface Card

2.3.3 RJ45 connector

RJ 45 or Registered Jack-45 is an eight-pin connector as shown in Figure 2.10 that is used exclusively with Ethernet cables for networking. It is a standard networking interface that can be seen at the end of all network cables. Basically, it is a small plastic plug that fits into RJ-45 jacks of the Ethernet cards present in computer.



Fig. 2.10 RJ 45 connector

2.3.4 Repeater

Data are carried in the form of signals over the cable. These signals can travel a specified distance (usually about 100 m). Signals lose their strength beyond this limit and become weak. In such conditions, original signals need to be regenerated.

A repeater is an analog device that works with signals on the cables to which it is connected. The weak signal appearing on the cable is regenerated and put back on the cable by a repeater.

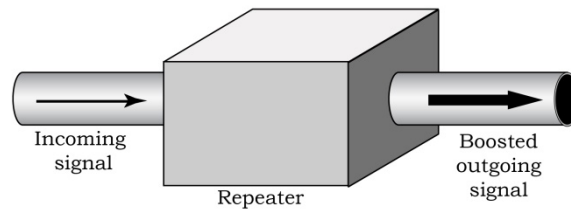


Fig. 2.11 Repeaters

Repeaters do not amplify the signal. When the signal becomes weak, they copy the signal bit by bit and regenerate it at the original strength. Repeater has two ports as shown in Figure 2.11, so it cannot be used to connect more than two devices.

2.3.5 Hub

An Ethernet hub as shown in Figure 2.12 is a network device used to connect different devices through wires. It has many ports, so also called as multi-port repeater. It accepts a signal in one port and rebroadcast out to all of its other ports. It is used to communicate with various network hosts for data transfer. Data arriving on any of the lines are sent out on all the others. The limitation of Hub is that if data from two devices come at the same time, they will collide.



Fig. 2.12 Network hub

2.3.6 Switch

A switch is a commonly used networking device to connect multiple devices in LAN. It is like a hub that is used to connect multiple computers or communicating devices as shown in Figure 2.13.



Fig. 2.13 Network switch

When data arrives, the switch extracts the destination address from the data packet and looks it up in a table to see where to send the packet. Thus, it sends signals to only selected devices instead of sending to all to avoid the wastage of bandwidth. It checks the MAC addresses of the connected devices to determine the correct port. The switch can store MAC address of a device.

A switch can forward multiple packets at the same time. It does not forward the signals which are noisy or corrupted. It drops such signals and asks the sender to resend it. Each switch port

has full-speed operations. So, for 100 Mbps switch, each device operates at 100 Mbps. Ethernet switches are common in offices to connect multiple devices thus creating LANs or to access the Internet.

2.3.7 Router

A router as shown in Figure 2.14 is a network device that route data around the network. The shortest path to transmit data from source to destination is called as route and this process is called as routing. It is used to connect two or more completely different network available in the different regions. It can receive the data, analyse it and transmit it to other networks. A router connects a local area network to the internet. Compared to a hub or a switch, a router has advanced capabilities as it can analyse the data being carried over a network, decide or alter how it is packaged, and send it to another network of different type. For example, data has been divided into packets of a certain size. Suppose these packets are to be carried over a different type of network which cannot handle bigger packets. In such a case, the data is to be repackaged as smaller packets and then sent over the network by a router.



Fig. 2.14 Router

A router can be wired or wireless. A wireless router can provide Wi-Fi access to smartphones and other mobile devices. Usually, such routers also contain some ports to provide wired Internet access. These days, home Wi-Fi routers perform the dual task of a router and a modem or switch. These routers connect to incoming broadband lines, from ISP (Internet Service Provider), and convert them to digital data for computing devices to process.

2.3.8 Gateway

Gateway is an intelligent device used for connecting networks with different types of architectures using different protocols together as shown in Figure 2.15. It is a key access point that acts as a “gate” between an organisation’s network and the outside world of the Internet.

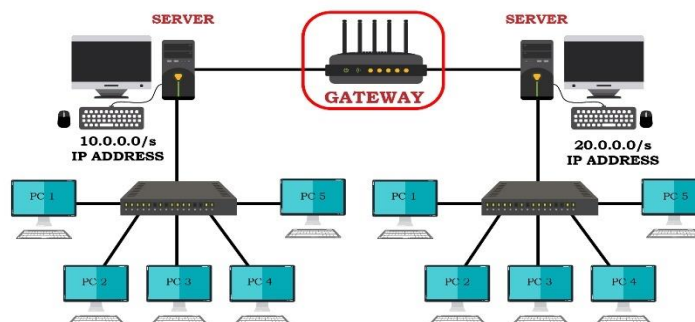


Fig. 2.15 Network gateway

Gateway serves as the entry and exit point of a network, as all data coming in or going out of a network must first pass through the gateway in order to use routing paths. Besides routing data packets, gateways also maintain information about the host network's internal connection paths and the identified paths of other remote networks. If a node from one network wants to communicate with a node of a foreign network, it will pass the data packet to the gateway, which then routes it to the destination using the best possible route.

Activity 2

Practical Activity 2.2 – Visit a computer network centre of any college laboratory or organisation. Identify and name the various networking devices installed in the network as shown in the following table. (Figure 2.16 (a) to Figure 2.16 (h)).



Fig. 2.16 Network devices

2.4 Networking Topologies

We have already discussed that a number of computing devices are connected together to form a Local Area Network (LAN), and interconnections among millions of LANs forms the Internet. A network node is a device that can send, receive, store, or forward data. A network link connects nodes and may be either cabled or wireless links. The arrangement of computers and other peripherals in a network is called its topology. *Network topology* is the way a *network* is arranged. A topology type provides the basis for building a successful network. There are a number of topologies but the most common are Bus, Ring, Star, Mesh and Tree.

2.4.1 Bus Topology

In bus topology, each communicating device connects through transmission medium, known as bus. In this topology, a single continuous coaxial cable called as backbone is shared among the nodes, which makes it cheaper and easier to maintain. Both ends of this cable are terminated through the terminators. The *BNC plug* and *BNC T connector* are used to connect the computer to backbone cable as shown in Figure 2.12. Data sent from a node are passed on to the bus and hence are transmitted to the length of the bus in both directions. That means, data can be received by any of the nodes connected to the bus.

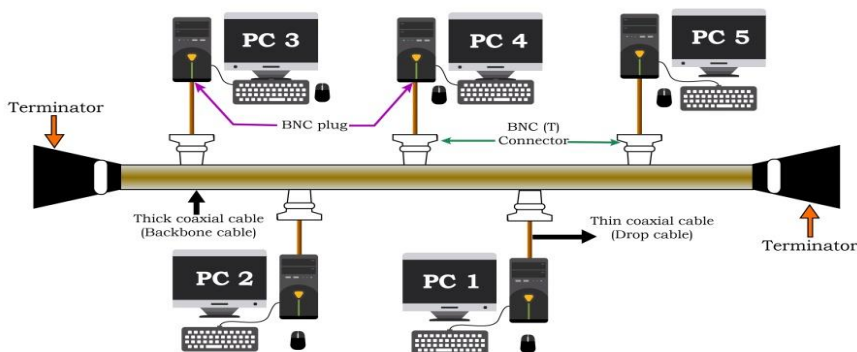


Fig. 2.17: Bus topology

Each packet that is placed on the bus contains the address of the destination node and is transmitted in both directions along the bus. The message is delivered to only that destination node to which it is addressed. A computer waits until the bus is free before it can transmit a message.

Bus topology works best with limited number of nodes. Currently this topology is no longer used. The concept that this topology uses to transmit the data is also used in the other topologies.

2.4.2 Ring Topology

In a *ring topology*, each node is connected to two other devices, one each on either side, as shown in Figure 2.18. The nodes connected with each other thus forms a ring. The link in a ring topology is unidirectional. Thus, data can be transmitted in one direction only either clockwise or counter clockwise.

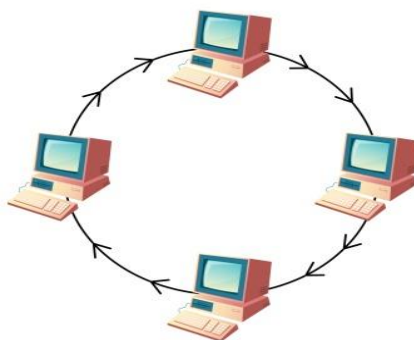


Fig. 2.18 Ring topology

When data signals pass from one computer to the next, each computer regenerates the signals. Since the signals are regenerated on each passing computer, the quality of the signals remains constant throughout the ring. Adjacent pairs are connected directly, non-adjacent pairs are connected indirectly through multiple nodes. A failure in any cable or device breaks the loop and can break down the entire network.

2.4.4 Star Topology

In *star topology*, each communicating device is connected to a central node, which is a networking device like a hub or a switch, as shown in Figure 2.19. Each device in the network uses separate twisted pair cable to connect to the switch.

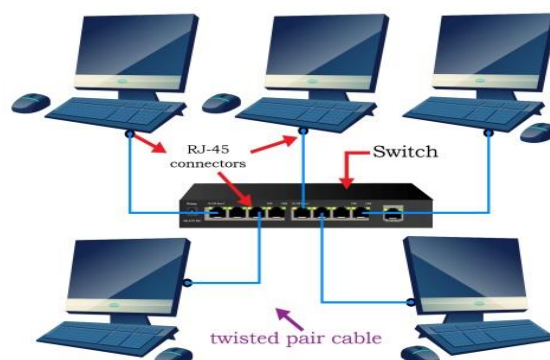


Fig. 2.19 Star topology

Star topology is considered very effective, efficient and fast as each device is directly connected with the central device. Although disturbance in one device will not affect the rest of the network, any failure in a central networking device may lead to the failure of complete network.

The central node can be either a broadcasting device means data will be transmitted to all the nodes in the network, or a unicast device means the node can identify the destination and forward data to that node only.

2.4.5 Mesh Topology

In *mesh topology*, each communicating device is connected with every other device in the network using multiple paths as shown in Figure 2.20. Such a network can handle large amount of traffic since multiple nodes can transmit data simultaneously. Also, such networks are more reliable in the sense that even if a node gets down, it does not cause any break in the transmission of data between other nodes. This topology is also more secure as compared to other topologies because each cable between two nodes carries different data. However, wiring is complex and

cabling cost is high in creating such networks and there are many redundant or unutilised connections.

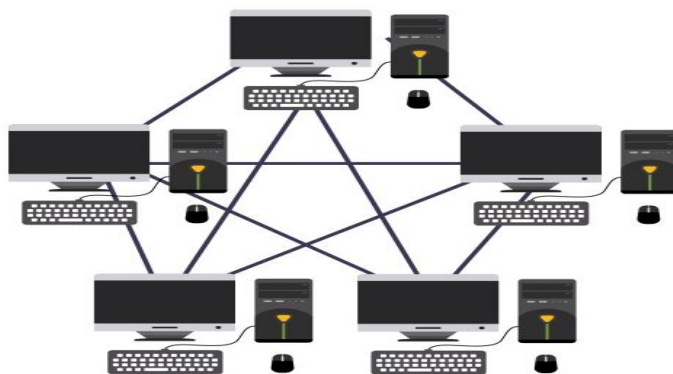


Fig. 2.20: Mesh topology

2.4.5 Tree or Hybrid Topology

It is a hierarchical topology, in which there are multiple branches and each branch can have one or more basic topologies like star, ring and bus. Such topologies are usually realised in WANs where multiple LANs are connected. Those LANs may be in the form of a ring, bus or star. In Figure 2.21, a hybrid topology is shown connecting 4-star topologies in a bus.

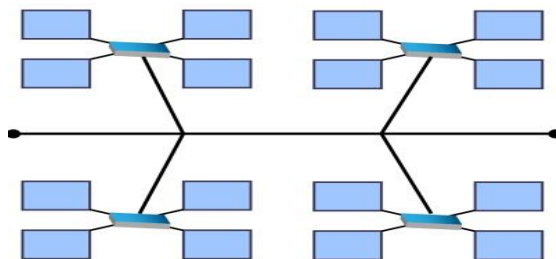


Fig. 2.21: Hybrid topology

In this type of network, data transmitted from source first reaches the centralised device and from there the data passes through every branch where each branch can have links for more nodes.

Activity 3

Practical Activity 2.3 – Identify and name types of network topology shown in Figure 2.22 (a) to (f).

Fig. 2.22 (a)	Fig. 2.22 (b)	Fig. 2.22 (c)	Fig. 2.22 (d)	Fig. 2.22 (e)	Fig. 2.22 (f)

2.5 Computer Network Architecture

Networks are usually configured to share resources using *network architecture*. It defines how computers are organized in the network and the tasks assigned to computers. The design of computers, devices, and media in a network is called as network architecture. It is categorized as *peer-to-peer (P2P)* and *client/server*.

2.5.1 Peer-to-peer network

It is the simplest form of a network. In peer-to-peer (P2P) architecture, two or more computers are connected as “peers,” without going through a separate server computer as shown in Figure 2.23. Each machine has equal power, privileges and resources. However, each computer can be configured to share only some of its resources and prevent access to other resources. There is no centralized management or security. Every computer can communicate directly with every other computer. Any user who wants access to resources on another computer should have an account on that specific computer. This model is used in very small networks at home or small office.

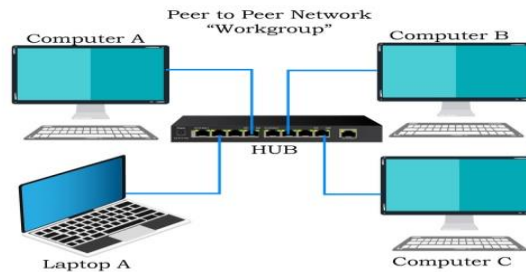


Fig. 2.23 Peer to Peer (P2P) network model

A peer-to-peer network usually uses a hub so that messages can be broadcast to all the computers on the network.

2.5.2 Client-server network

In a *client/server network*, there are clients (workstations) as well as one or more server. A central server or group of servers manage resources and deliver services to large number of user machines in the network called *clients*. The servers are usually powerful computers with more memory, storage space, and fast network connections. The clients are regular PCs. The server software runs on server hardware, and client software is used on client computers that connect to servers. The client request services from the server. A server controls access to hardware, software, and other resources on the network. It also provides a centralized storage area for programs and data. The clients in the network communicate with other clients through the server. In Figure 2.24, there are clients Computer A, Computer B, Laptop A and so on.

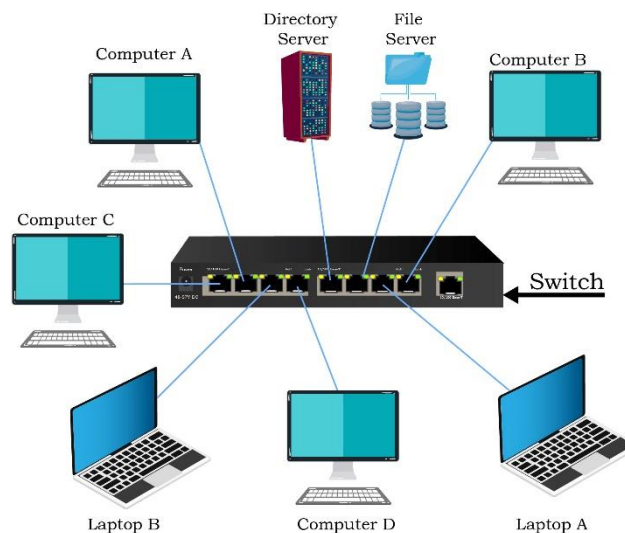


Fig. 2.24: Client/Server network model

Each user needs a user ID and password to access the network. Clients on a client/server network do not share their resources directly with each other. They use the server as an intermediary. Clients and servers communicate through switches or routers. A dedicated server performs a specific task. There can be multiple servers to perform multiple tasks. For example, a file server stores and manages files, a print server manages printers and documents to be

1. Static – It is the permanent internet address that provides a simple and reliable way for communication. The example of a Static IP Address is DNS Server (Domain Name Service). Static IP address is less secure than Dynamic IP address due to its assignment.

2. Dynamic – Internet Service Provider (ISP) provides an IP address from the range of available IP addresses to the device. ISP provides different IP addresses at different times, but from the same available range of IP addresses. IP address changes at every time we connect to the internet. Such IP address is termed as Dynamic IP address. An example of a Dynamic IP Address is DHCP server (Dynamic Host Configuration Protocol).

2.6.4 Types of IP Address

There are two types of IP Address Version currently exists – IPV4 and IPV6.

IPV4 – The initial IP address called version 4 (IPv4 in short), is a 32-bit numeric address, written as four numbers separated by periods, where each number is the decimal (base-10) representation for an 8-bit binary (base-2) number and each can take any value from 0 - 255. An example of IPv4 address is: 192:168:0:178.

IPV6 – With more and more devices getting connected to the Internet, it was realised that the 32-bit IP address will not be sufficient as it offers just fewer than 4.3 billion unique addresses. Thus, a 128 bits IP address, called IP version 6 (IPv6 in short) was proposed. An IPv6 address is represented by eight groups of hexadecimal (base-16) numbers separated by colons. Example of IPv6 address is: 2001:CDBA:0000:0000:0000:0000:3257:9652

2.8 Network Commands

There are various network commands used while working on the computer connected in the network. These commands are used at user level for operating the network based computer as well for system administration purpose. It is essential to know at least some of these commands for a user working on computer connected in the network. These commands are used on the command prompt. So the user has to be cautious and should not execute the commands without knowing about them. For example, the purpose of few basic commands are given below.

1. hostname – Each computer has a unique address to communicate in the network. The “hostname” command displays the name of your computer. *hostname -i*, displays the IP address of your machine.

2. ping Command – Ping stands for “*packet internet groper*”. It used to check the connectivity between two systems.

3. ifconfig Command – The ifconfig command lists the network interfaces attached to the PC along with other statistics.

The **inet** parameter shows the **IPv4** address of the network interface while **inet6** points to the IPv6 address.

4. IP Command – A other way you can view interface statistics is using the ip address command as shown.

ip addr show: It shows IPv4 or IPv6 address on a device

5. ip route Command – The ip route command prints out the routing table of your PC.

6. nslookup Command – The nslookup stands for name server lookup, is a network utility command used to obtain information about internet servers. It display the IP address by providing the hostname and hostname by providing the IP address.

Assignment: Under the supervision of teachers, execute the above mentioned network commands and note the output displayed.

Activities

Practical Activity 2.4 – To find the MAC address and IP address of computing device in Windows OS

Material Required

Computing devices connected to network with OS Windows 10

Procedure

Step 1. Open the command prompt from Windows. In the search box, type **cmd**, and Press the **Enter** key. A command window displays.

Step 2. Type **getmac** command on command prompt. The device MAC address will be displayed. The Physical Address as shown in Figure 2.27 is your device MAC address.

```

C:\Users\acer>getmac
Physical Address      Transport Name
=====
94-C6-91-C8-91-9D   \Device\Tcpip_{574908D9-0B86-4F62-9C1F-9EE66AAA8445}
C:\Users\acer>

```

Fig. 2.27 Finding MAC address using getmac command

Step 3. To fetch the IP address of the device, enter the command **ipconfig** on the command prompt as shown below. IPv4 Address shown in Figure 2.28 is the IP address of your device.

```

Microsoft Windows [Version 10.0.19044.1645]
(c) Microsoft Corporation. All rights reserved.
C:\Users\acer>ipconfig
Windows IP Configuration

Ethernet adapter Ethernet:

    Connection-specific DNS Suffix  . : 
    Link-local IPv6 Address . . . . . : fe80::c56f:f591:9483:e66%11
    IPv4 Address. . . . . : 172.30.12.172
    Subnet Mask . . . . . : 255.255.254.0
    Default Gateway . . . . . : fe80::c6e9:aff:fe48:3fd5%11
                                fe80::c6e9:aff:fee7:b22b%11
                                172.30.12.249
C:\Users\acer>

```

Fig. 2.28 Finding IP address using ipconfig command

Step 4. To see detailed description about Host Name, DNS, Ethernet adapter, Physical Address, IP, DHCP Server, give command as **ipconfig /all**.

Step 5. It is possible to see all these details in GUI mode using Control Panel. Explore the way to view these details using Control Panel.

Practical Activity 2.5 – To find the MAC address and IP address in Linux

Material Required

Computing devices connected to network in Linux OS

Procedure

Step 1. Open a terminal or console window.

Step 2. Enter the command “**ifconfig**”. It will display details of all network connections of your machine.

Step 3. Use the **grep** command to filter the MAC address of your machine. It will display the MAC address as shown in Figure 2.29.

```

dds@psscive: ~
dds@psscive:~$ ifconfig | grep ether
ether e4:e7:49:9b:24:7a txqueuelen 1000 (Ethernet)
dds@psscive:~$ █

```

Fig. 2.29 Finding MAC address using ifconfig command

It is possible to find the MAC address and IP address from GUI through internal network configuration.

To check for the internal IP address, default gateway, MAC address and DNS server settings on Ubuntu Linux, first open Settings and click on Network menu and hit the gear wheel of the desired network interface. The details will be displayed as shown in Figure 2.30. The Hardware Address is the MAC address.

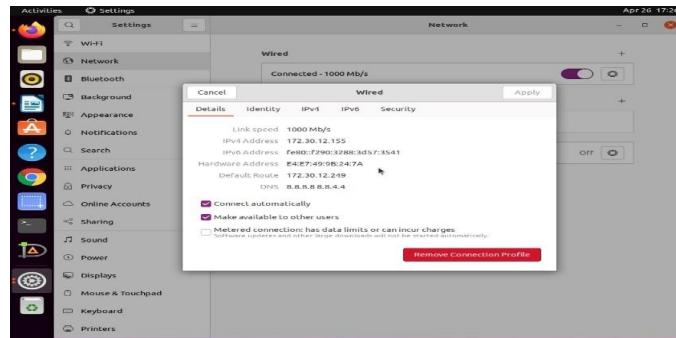


Fig. 2.30 Finding MAC address in Ubuntu Linux GUI

SUMMARY

- A computer network is an interconnection among two or more computers or computing devices.
- A computer network allows computers to share data and resources among each other.
- Networking devices are used to connect multiple computers in different settings.
- In a communication network, each device that is a part of a network and that can receive, create, store or send data to different network routes is called a node.
- Based on the geographical area and data transfer rate, computer networks are broadly categorised into LAN (Local Area Network), MAN (Metropolitan Area Network) and WAN (Wide Area Network).
- LAN is a network that connects a variety of nodes placed at a limited distance ranging from a single room, a floor, an office or a campus having one or more buildings in the same premises.
- Ethernet is a set of rules that decides how computers and other devices connect with each other through cables in a LAN.
- Metropolitan Area Network (MAN) is an extended form of LAN which covers a larger geographical area like a city or a town.
- Cable TV network or cable based broadband internet services are examples of MAN.
- Wide Area Network (WAN) connects computers and other LANs and MANs, which are spread across different geographical locations of a country or in different countries or continents.
- The Internet is the largest WAN that connects billions of computers, smartphones and millions of LANs from different continents.
- Modem stands for 'MODulator DEModulator', is a device used for conversion between electric signals and digital bits.

- Ethernet card, also known as Network Interface Card (NIC card in short) is a network adaptor used to set up a wired network.
- Each NIC has a MAC address, which helps in uniquely identifying the computer on the network.
- A repeater is an analog device that regenerates the signals on the cables to which it is connected.
- A switch is a networking device used to connect multiple computers or communicating devices.
- A router is a network device that can receive the data, analyse it and transmit it to other networks.
- Gateway serves as the entry and exit point of a network, as all data coming in or going out of a network must first pass through the gateway in order to use routing paths.
- The arrangement of computers and other peripherals in a network is called its topology.
- Common network topologies are Mesh, Ring, Bus, Star and Tree.
- In mesh topology each communicating device is connected with every other device in the network.
- In ring topology, each node is connected to two other devices, one each on either side.
- In bus topology, a single backbone wire called bus is shared among the nodes, which makes it cheaper and easy to maintain.
- In star topology, each communicating device is connected to a central networking device like a hub or a switch.
- In tree or hybrid topology, there are multiple branches and each branch can have one or more basic topologies like star, ring and bus.
- The MAC address, also known as the physical or hardware address, is a unique permanent value associated with a network adapter called a NIC.
- It is used to physically identify a machine on the network.
- IP address, also known as Internet Protocol address, is a unique address that can be used to uniquely identify each node in a network.
- Unlike MAC address, IP address can change if a node is removed from one network and connected to another network.

CHECK YOUR PROGRESS

A. Multiple choice questions

1. What physical topology would you use to create your peer-to-peer network where all the workstations are connected to a single switch? (a) Bus (b) Tree (c) Star (d) Cube
2. A computer network is an interconnection among (a) two computers (b) three computers (c) four computers (d) two or more computers
3. Which of the following is not a networking devices (a) switch (b) router (c) modem (d) cable
4. In computer network, data is communicated through _____ (a) hub (b) switch (c) packets (d) router
5. Which of the following network can be constructed without cable (a) LAN (b) WAN (c) PAN (d) CAN
6. Which of the following is not the main components of Network Interface Card (a) PCI interface (b) PCI slot (c) RJ-45 slot (d) Ethernet socket.
7. In which of the following network topology the link is unidirectional (a) Bus topology (b) Ring topology (c) Star topology (d) Mesh topology

8. Which of the following network topology is most effective, efficient and fast (a) Bus topology (b) Ring topology (c) Star topology (d) Mesh topology
9. Which of the following network topology is has multiple paths (a) Bus topology (b) Ring topology (c) Star topology (d) Mesh topology
10. Which of the following command is used to find the MAC address in Linux (a) getmac (b) ifconfig (c) ipconfig (d) ping

B. Fill in the blanks

1. Each device in a network is called as _____.
2. Data transfer in LAN varies from 10 Mbps called _____ to 1000 Mbps, called _____
3. The devices in PAN lie within an approximate range of _____
4. MAN combines multiple LAN through a networking device _____
5. Modem stands for _____ and _____.
6. The modem at the sender's end acts as a modulator that converts the _____ into _____.
7. Ethernet card has unique physical address known as _____
8. In client/server network, clients and servers communicate through _____ or _____
9. MAC address is located on _____ card which cannot be change.
10. In client/server network, there can be _____ servers.

C. State whether True or False

1. In LAN, only authentic users in the network can access the shared resources.
2. A personal area network is wireless.
3. The WWAN uses wireless technology.
4. The modem at the receiver's end converts digital data into analog signals.
5. Ethernet card is used in wireless network.
6. Repeater can be used to connect more than two devices.
7. Hub is also called as multiport repeater.
8. A switch can forward multiple packets at the same time.
9. Gateway is used to connect two or more different network in different regions.
10. Router is used for connecting networks with different types of architectures.

D. Short answer questions

1. What are the basics of computer networking?
2. What are the three main components of a network?
3. What are the types of computer networks?
4. Why do we need networking devices?
5. What are the main devices of a computer network?
6. Give the 3 examples of a computer network?
7. What is the most common type of computer network?
8. What is the primary difference between peer-to-peer and client-server architectures?
9. Give the examples of network devices?
10. What is MAC address and IP address of device?

Session 3: Transmission Media and Network Protocols

Abhinaw when goes to the school, he finds that there was a new comer in his class. Abhinaw started talking with in Marathi language. The new comer was not aware of Marathi language as he was from West Bengal. Abhinaw asked him his name and the new comer could not understand what he is asking for. So, the communication between Abhinaw and newcomer was not taking place because of the languages. Communication between the two people takes place only when both of them can understand a common language. When Abhinaw started speaking in English the newcomer was able to understand him and he replied accordingly. In the similar way the communication between the two computers over a network will takes place only when they follow a certain set of predefined communication rules. These communication rules are called as protocols.

Fig. 3.1 Illustration

In this session we are going to discuss the wireless technologies and different network protocols such as HTTP, FTP, IP, PPP, SMTP and POP.

3.1 Wireless Technologies

In order to transfer the message from one place to another place we require certain medium. The message can be transferred through cables or wires. It is also possible that we can communicate between two computers without using any wire or cable. Such communication technologies are called as wireless technologies. Wireless technologies make the use of radio frequencies and infrared waves.

3.1.1 Wireless Transmission Media

In wireless communication technology, information travels in the form of electromagnetic signals through air. Electromagnetic spectrum of frequency ranging from 3 KHz to 900 THz is available for wireless communication as shown in Figure 3.2. Wireless technologies allow communication between two or more devices in short to long distance without requiring any physical media. There are many types of wireless communication technologies such as Bluetooth, WiFi, WiMax.

The electromagnetic spectrum range (3KHz to 900THz) can be divided into 4 categories – Radio waves, Microwaves, Infrared waves and Visible or Light waves, according to their frequency ranges. Electromagnetic Wave Spectrum of these, three are useful for wireless communication.

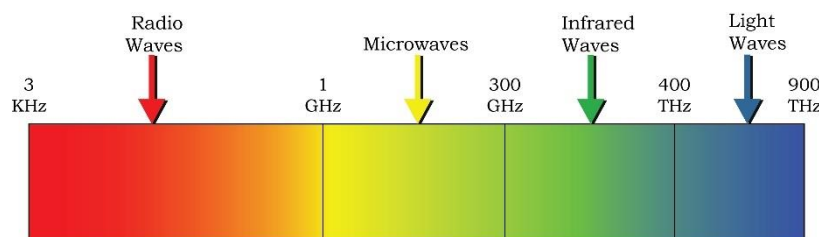


Fig. 3.2: Electromagnetic waves spectrum

Types of Wireless Data Transmission

Radio Waves Transmission – These are electromagnetic waves that travel at the speed of light in vacuum. It is using the radio frequency modulation for data transmission in wireless communication. Radio waves have the lowest frequency and highest wavelength in the spectrum. The radio waves have frequency range from 3 KHz to 1 GHz. These waves are easy to generate and these can travel along long distances. These waves are omni directional in nature which means that they can travel in all the directions. They are widely used for the communication between both indoor and outdoor because they have the property that they can penetrate through the walls very easily. Radio waves are generally used for transmitting sound, images that include both voice signal and television signals. Figure 3.3 shows that antenna broadcasting

the radio signal are received by receiver antenna of Radio. Radio waves are used for directing the movement of ships and aircraft with the help of radio compass or radio time signals.

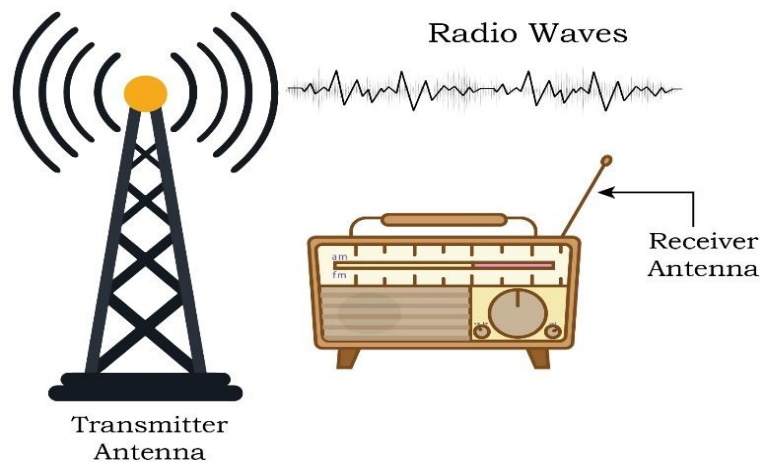


Fig. 3.3: Antenna broadcasting the radio signals to Radio station

Microwave Transmission – Microwaves are electromagnetic waves which have frequency range between 1 GHz to 300 GHz. These can travel along long distances. These are unidirectional in nature which means that they can travel only in straight line. At very high frequency that cannot penetrate into walls. It needs line-of-sight propagation i.e. both communicating antenna must be in the direction of each other. Microwave transmission is used for long-distance telecommunication, where installation of physical transmission media is not possible and line-of-sight transmission is available. These waves are usually used for one to one communication between sender and receiver, such as cellular phones, satellite networks, and wireless LAN. It provides very large information-carrying capacity. Figure 3.4 shows Microwaves used in mobile tower for signal transmission

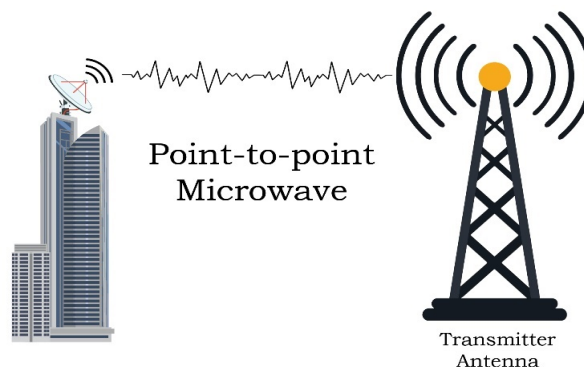


Fig. 3.4: Microwaves Transmission in mobile tower for signal transmission

Infrared Transmission – These are very high frequency electromagnetic waves of frequency range 300GHz – 400THz. They cannot penetrate solid objects such as walls. They also use line-of-sight of propagation. It is used for short-distance point-to-point communication such as mobile-to-mobile, mobile-to-printer, remote-control-to-TV, and Bluetooth-enabled devices to other devices like mouse and keyboard. The remote control uses infrared light waves to control electronic appliances as shown in Figure 3.5.



Fig. 3.5: Infrared light wave used in television remote

3.1.2 Types of Wireless Communication Technologies

The wireless communication technology has become an integral part of our life. It allows us to communicate even in remote areas. The devices used for wireless communication are cordless telephones, mobile phones, GPS units, wireless computer parts, and peripherals. Wireless communication technology is categorized into different types depending on the distance of communication, the range of data, and the type of devices used. There are many wireless communication technologies such as Radio and Television Broadcasting, Satellite communication, Cellular Communication, Global Positioning System, WiFi and Bluetooth. Some of them are discussed here. (Figure 3.6)



Fig. 3.6: Types of communication technologies

Bluetooth – Bluetooth uses radio waves in the frequency range of 2.402 to 2.480 GHz. It is a short-range wireless technology that can be used to connect mobile-phones, mouse, headphones, keyboards, and computer wirelessly over a short distance. One can print documents with Bluetooth-enabled printers without a physical connection. All these Bluetooth-enabled devices have a low-cost transceiver chip. This chip uses the unlicensed frequency band of 2.4 GHz to transmit and receive data. These devices can send data within a range of 10 meters with a speed of 1 - 2 Mbps.

In Bluetooth technology, the communicating devices within a range of 10 meters build a personal area network called piconet. The devices in a piconet work in a master-slave configuration. A master device can communicate with up to 7 active slave devices at the same time.

Bluetooth technology allows up to 255 devices to build a network. Out of them, 8 devices can communicate at the same time and remaining devices can be inactive, waiting for a response command from the master device. (Figure 3.7)

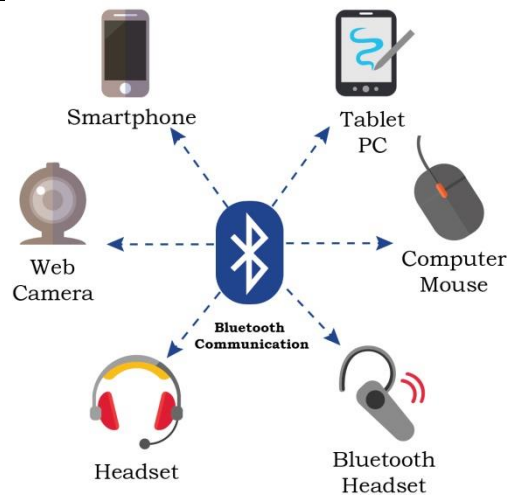


Fig. 3.7 Bluetooth technology

Wi-Fi – Wi-Fi is low-cost wireless communication technology commonly used in home networking. A WiFi setup consists of a wireless router which serves a communication hub, linking portable device with an internet connection. This network facilitates the connection of devices in close proximity to a router. Wi-Fi is a form of low-power wireless communication used by many electronic devices such as laptops and smartphones. Wi-Fi networks need to be secured with passwords so not to be accessed by others. (Figure 3.8)



Fig. 3.8 Wi-fi communication technology

Wireless LAN – This is another way of wireless communication. Wireless LAN is a local area network (LAN), and it is a popular way to connect to the Internet. The international organisation IEEE assigns numbers to each different standards of LAN. The wireless LAN is number as 802.11, and it is popularly known as Wi-Fi. These networks consist of communicating devices such as laptops and mobile phones, as well as the network device called APs (access points) which is installed in buildings or floors (Figure 3.9). An access point is a device that is used to create a wireless local area network, by connecting to a wired router, switch, or hub. The APs are connected to a wired network, and all the devices communicate or access the Internet through an access point.

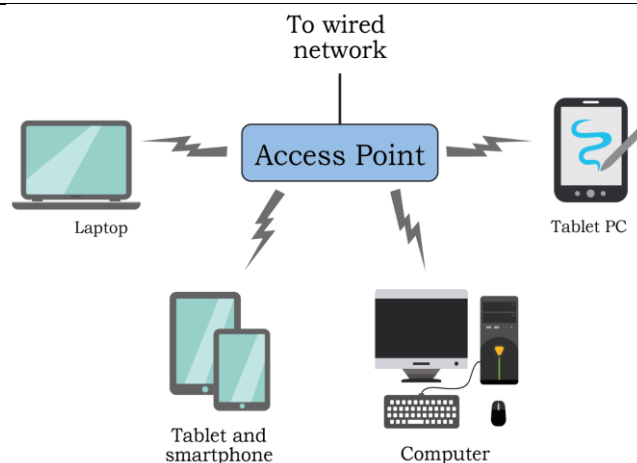


Fig. 3.9: Access point creating a wireless LAN

Wi-Fi gives users the flexibility to move around within the network area while being connected to the network. Following are some of the benefits of WLAN:

- Wireless connections can be used to extend or replace an existing wired infrastructure
- Resulted in increased access for mobile devices
- Provides easy access to the Internet in public places

3.2 Mobile Telecommunication Technologies

Today the mobile phone network is the most used network in the world. The ability to be connected to the network on-the-go makes it very convenient to communicate with people via call or instant messages. It is also handy to access the Internet using the mobile phone network through wireless connection. Besides, the Internet of Things (IoT) is letting us control and communicate with other smart devices as well.

The architecture of the mobile network has rapidly evolved over the last few decades. The different landmark achievements in mobile communication technologies are classified as different generations. They are identified as 1G, 2G, 3G, 4G, and 5G. Let us briefly discuss the mobile telecommunication generations.

The first generation (1G) mobile network system came around 1982. It was used to transmit only voice calls. The analog signals were used to carry voices between the caller and receiver.

The second generation (2G) mobile network system came around 1991. Instead of analog signals, voice calls were transmitted in digital form thus providing improved call quality. This increased capacity allowed more people to talk simultaneously, and led to improved security as the signals could be encrypted. It also enabled an additional service to send SMS and MMS (Multimedia messages).

The third generation (3G) mobile network technology was developed during late 90s, but it was introduced commercially around 2001. It offered both digital voice and data services. 3G provided Internet access via the same radio towers that provide voice service to the mobile phone. It facilitated greater voice and data capacity.

Therefore, more simultaneous calls could happen in the same frequency range and also a significantly faster data transfer speed.

Demand for faster data is always increasing and thus 4G mobile networks were developed and now 5G networks have also come into being. 4G is much faster than 3G and this has revolutionised the field of telecommunication by bringing the wireless experience to a new level altogether. 4G systems support interactive multimedia, voice, video, wireless internet and other broadband services. Technologically, 4G is very different compared to 3G.

The fifth generation or 5G is currently under development. It is expected to be a milestone development for the success of IoT and Machine to Machine (M2M) communications. Machine to

machine (M2M) is direct communication between devices — wired and wireless. 5G is expected to allow data transfer in Gbps, which is much faster than 4G. It is expected to be able to support all the devices of the future such as connected vehicles and the Internet of Things.

3.3 Computer network models

Computer network models facilitate establishing a connection between the sender and receiver for smoothly transmitting the data. There are two most popular computer network models – OSI Model and TCP/IP (Transport Control Protocol/Internet Protocol) Model. OSI is a reference model that describes the functions of a networking system. TCP/IP is the communication protocol suite to connect network devices to the Internet.

Both the models are based upon layered architecture. Each model has different layers with a specific function. This helps in identifying issues if any failure occurs. Protocols are defined in a layer-wise way. Both models convert the raw data into packets and help them reach their destination node. The Physical and Data Link layers of the OSI model correspond to the Network Access layer of the TCP/IP model. The Application, Presentation, and Sessions layers of the OSI model correspond to the Application layer of the TCP/IP model. Both models have the network layer and the transport layer.

OSI Model

OSI is a conceptual model that defines network communication. It is developed by ISO (International Organization for Standardization) in 1984. It consists of 7 layers where each layer performs a specific function.

The OSI model has the following 7 layers. (Figure 3.10)

1. **Physical Layer** – It is the lowest layer of the OSI model that is responsible for the physical data connection between the devices.
2. **Data Link** – It ensures error-free node-to-node delivery of the message.
3. **Network Layer** – The data gets its address in this layer. It also received routing instructions from moving from source to destination in the network.
4. **Transport Layer** – It is responsible for reliable message delivery across different points on the network on its way to its destination.
5. **Session Layer** – This layer helps with establishing, managing, and terminating sessions.
6. **Presentation Layer** – It is responsible for encrypting and decrypting the data and converting it into a form that is accessible by the application layer.
7. **Application Layer** – In this layer, an application (for example, an internet browser) receives the data and a user can then interact with it.

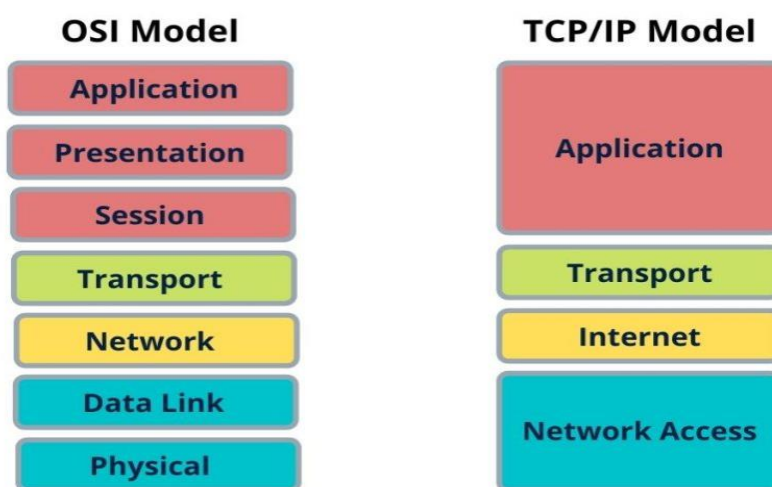


Fig. 3.10 Network models – OSI and TCP/IP

TCP/IP Model

The TCP/IP (Transmission Control Protocol/Internet Protocol) model was introduced before the OSI model. This model helps users understand how a specific computer should be connected to the internet and how data can transmit between them. TCP/IP is used more compared to the OSI model for providing communication between computers over the internet.

The following are the four layers of the TCP/IP model.

1. **Network Access Layer** – It is the lowest layer of the TCP/IP Model that helps in the transmission of data between two devices on the same network. This layer is a combination of the Physical and Data Link layer of the OSI model.
2. **Internet Layer** – Helps in moving packets from destination to source. This layer corresponds to the Network layer of the OSI Model.
3. **Transport Layer** – Responsible for reliable and error-free process to process message delivery. It is parallel to the Network layer of the OSI Model.
4. **Application Layer** – This layer provides access to network resources. When the Application, Presentation, and Sessions layers of the OSI model are combined, they perform similar functions as the Application layer of the TCP/IP model.

3.4 Network Protocol

In communication, Protocol is a set of standard rules that the communicating parties — the sender, the receiver, and all other intermediate devices need to follow. The protocol identifies the rules, syntax, semantics, and synchronization of communication and feasible error managing methods. A Network Protocol is composed of rules, procedures, and types that describe communication among a couple of devices over the network. The sender and receiver can be parts of different networks, placed at different geographic locations. Besides, the data transfer rates in different networks can vary, requiring data to be sent in different formats.

3.4.1 Need for Protocols

We need protocols for different reasons such as flow control, access control and addressing. Flow control is required when the sender and receiver have different speeds of sending and receiving the data. Figure 3.10 shows that Computer A is sending data at the speed of 1024 Mbps and Computer B is receiving data at the speed of 512 Mbps. In this case, Computer B must be able to inform computer A about the speed mismatch so that computer A can adjust its data transmission rate. Otherwise some data will be lost, as shown in Figure 3.11.

Access control is required to decide which nodes in a communication channel will access the link shared among them at a particular instant of time. Otherwise, the transmitted data packets will collide if computers are sending data simultaneously through the same link resulting in the loss or corruption of data.

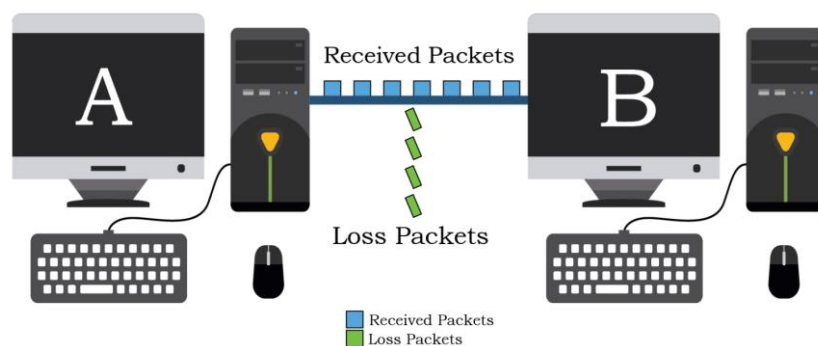


Fig. 3.11: Speed mismatch between two computers can result into loss of data

Protocols also define:

- how computers identify one another on a network.
- the form to which the data should be converted for transit.

- how to decide whether the data received is for that node or to be forwarded to another node.
- ensuring that all the data have reached the destination without any loss.
- how to rearrange the packets and process them at the destination.

If all the rules or protocols of a communication network are defined at one place, it becomes complex to ensure that communicating parties follow the guidelines. In this section, we will briefly talk about some of the protocols required in communication.

3.5 Transmission Control Protocol (TCP)/Internet Protocol (IP)

TCP/IP stands for Transmission Control Protocol/ Internet Protocol. It is a set of standardised rules and procedures, used for interconnecting various network devices over the internet by defining how the data should be transmitted, routed, broken into packets, addressed, and received at the destination. The TCP defines how applications can create communication channels across a network. IP defines the way each packet is addressed and routed to ensure it reaches the correct destination.

TCP uses a connection-oriented service. It decides a path and data units are delivered via the decided path. When the TCP conversation is finished, the session is terminated. TCP ensures that the message or data is broken into smaller chunks, called IP packets. Each of these packets are routed (transmitted) through the Internet, along a path from one router to the next, until it reaches the specified destination. TCP guarantees the delivery of packets on the designated IP address. It is also responsible for ordering the packets so that they are delivered in sequence.

The IP protocol ensures that each computer or node connected to the Internet is assigned an IP address, which is used to identify each node independently. It can be considered to be the adhesive that holds the whole Internet together.

There are many redundant connection paths in the Internet, with backbones and ISPs connecting to each other in multiple locations. So, there are many possible paths between two hosts. Hence, two packets of the same message can take two different routes depending on congestion and other factors in different possible routes. When all the packets finally reach the destination machine, they are reassembled into the original message at the receiver's end.

The commonly used TCP/IP protocol includes – HTTP, HTTPS, FTP, SMTP, PPP, Telnet.

3.5.1 HyperText Transfer Protocol (HTTP)

HTTP stands for HyperText Transfer Protocol. It is the primary protocol used to access the World Wide Web. Tim Berners-Lee led the development of HTTP at CERN in 1989 in collaboration with Internet Engineering Task Force (IETF) and the World Wide Web Consortium (W3C).

HTTP is a request-response (also called client-server) protocol that runs over TCP. The common use of HTTP is between a web browser (client) and a web server (server). A client sends a request to the server through a web browser to view specified information. After receiving a request, the server sends specified information to the client. It is a non-secure transmission.

HTTP facilitates access of hypertext from the World Wide Web by defining how information are formatted and transmitted, and how the Web servers and browsers should respond to various commands.

A web page is written using a mark-up language like HTML and is stored on a web server for access via its URL. Once a user opens a web browser and types in the URL of the intended web page, a logical communication link between the user machine (client) and the web server is created using HTTP.

For example, whenever we enter the URL <http://www.psscive.ac.in> in a browser, it sends HTTP request to the web-server where psscive.ac.in is hosted. The HTTP response from the web-server fetches and sends the requested web-page, which is displayed on browser. (Figure 3.12)



Fig. 3.12: Hyper Text Transfer Protocol

3.4.2 HTTPS

HTTPS stands for HyperText Transfer Protocol Secure. HTTPS establishes a connection between the client and the server for data transmission. It is a secure transmission. The client mainly uses this HTTPS to send private information like credit card details, online transactions, to the server across the internet connection. (Figure 3.13)

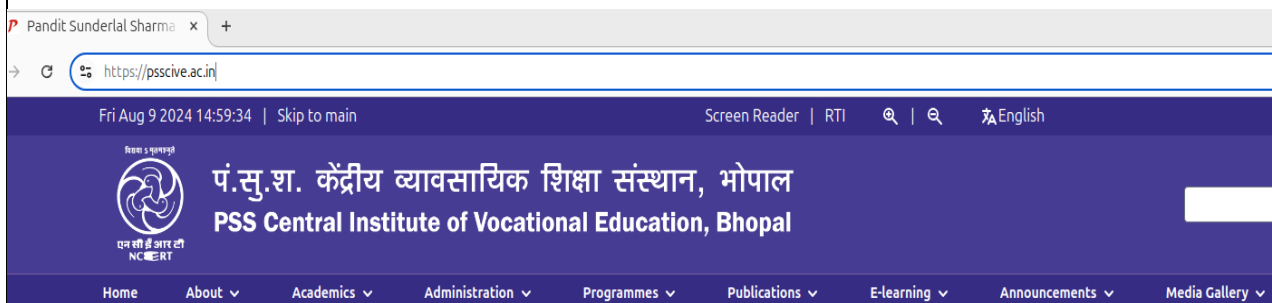


Fig. 3.13: Hyper Text Transfer Protocol Secure

3.4.3 File Transfer Protocol (FTP)

File Transfer Protocol (FTP) is the protocol used for transferring files from one machine to another. It is a standard mechanism provided by TCP/IP. Transforming files from one system to another seems very simple, but some problems need to be dealt with before transforming files. (Figure 3.14) For example, two systems may use a different file name convention, may have different directory structures, may have a different way of representing data. All these problems are resolved by FTP.

FTP also works on a client-server model. When a user requests for a file transfer with another system, FTP sets up a connection between the two nodes for accessing the file. Optionally, the user can authenticate using user ID and password. The user then specifies the file name and location of the desired file. After that, another connection sets up and the file transfer happens directly between the two machines. However, some servers provide FTP logins without authentication for accessing files.

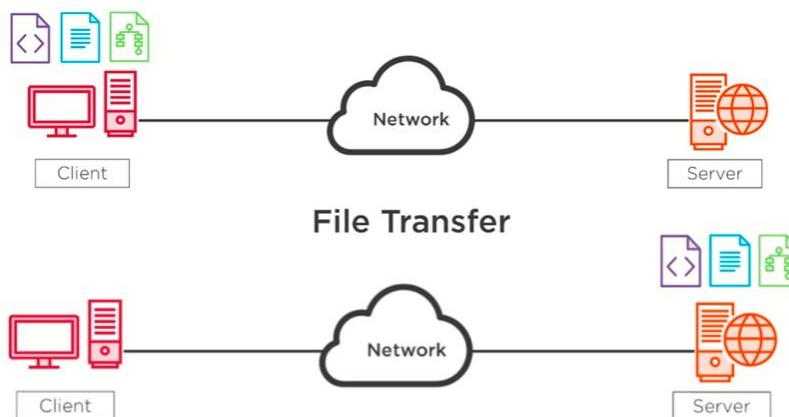


Fig. 3.14 File transfer using FTP

3.4.4 Point to Point Protocol (PPP)

PPP is a communication protocol which establishes a dedicated and direct connection between two communicating devices. This protocol defines how two devices will authenticate each other and establish a direct link between them to exchange data. For example, two routers with direct connection communicate using PPP. The Internet users who connect their home computers to the server of an Internet Service Provider (ISP) through a modem also use PPP.

The communicating devices should have duplex modes for using this protocol. This protocol maintains data integrity ensuring that the packets arrive in order. It intimates the sender about damage or lost packets and asks to resend it.

3.4.5 Simple Mail Transfer Protocol (SMTP)

SMTP is a protocol used for email services. It is a Push protocol that is used to send an email. After that, POP (post office protocol) or IMAP (internet message access protocol) protocols retrieve emails on the receiver end. It uses information written on the message header (like an envelope on a letter sent by post), and is not concerned with the content of the email message. Each email header contains email addresses of recipients. The email containing header and body are entered into a queue of outgoing mails. (Figure 3.15)

SMTP simply defines how data or commands transfer from client to server or server to client. It is used two times between the sender and sender's mail server and between two mail servers. To transfer mails, SMTP uses three phases, i.e. connection establishment, mail transfer and connection termination. The commands, which are used to send data from client to server and responses, which is used to send data from server to client.

HELO – This command is used to identify the user and full domain name, which is transmitted only once per session.

MAIL – This command is used to initiate a message transfer.

RCPT – This command comes after MAIL and is used to identify the recipient's fully qualified name. For multiple recipients, RCPT is used for each of the recipients.

DATA – This command is used to send data one line after the other.

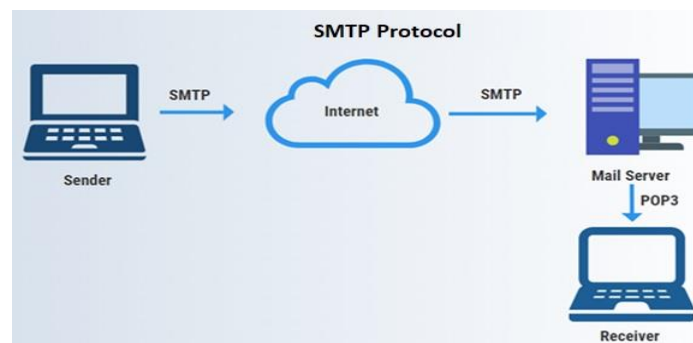


Fig. 3.15 SMTP Protocol

The SMTP sender program takes mails from the outgoing queue and transmits them to the destination(s). When the SMTP sender successfully delivers a particular mail to one or more destinations, it removes the corresponding receiver's email address from the mail's destination list. When that mail is delivered to all the recipients, it is removed from the outgoing queue. The SMTP receiver program accepts each mail that has arrived and places it in the appropriate user mailbox.

SUMMARY

- Data communication refers to the exchange of data between two or more networked or connected devices like laptops, PC, printers, routers etc.
- Sender, receiver, messages, channel and protocols are major components of data communication.

- In data communication, transmission media are the links that carry messages between two or more communicating devices. These are broadly classified into guided and unguided media.
- In guided transmission, there is a physical link made of wire/cable through which data in terms of signals are propagated between the nodes.
- These are usually metallic cable, fibre-optic cable, etc. They are also known as wired media.
- In unguided transmission, data travels in air in terms of electromagnetic waves using an antenna. They are also known as wireless media.
- The capacity of channels is measured bandwidth. The unit of bandwidth is Hertz.
- Communication can be done in three different modes — simplex, half-duplex, and full-duplex communication.
- Switching techniques are alternative to dedicated lines whereby data is routed through various nodes in a network. It forms a temporary route for the data to be transmitted. Two commonly used switching techniques are – circuit switching and packet switching.
- Electromagnetic spectrum of frequency ranging from 3 KHz to 900 THz is available for wireless communication. This spectrum range (3KHz to 900THz) can be divided into four categories- Radio waves, Microwaves, Infrared waves and Visible or Light waves, according to their frequency ranges.
- Bluetooth is a short-range wireless technology that can be used to connect mobile-phones, mouse, headphones, keyboards, computers, etc. wirelessly over a short distance.
- Based on the architecture of the mobile network, mobile communication technologies are classified into different generations identified as 1G, 2G, 3G, 4G, and 5G.
- In communication, protocol is a set of standard rules that the communicating parties — the sender, the receiver, and all other intermediate devices need to follow. Flow control, access control, addressing, etc. are examples of protocol.
- HTTP stands for HyperText Transfer Protocol. It is the primary protocol used to access the World Wide Web, which was developed by Tim Berners-Lee at CERN in 1989.
- File Transfer Protocol (FTP) is the protocol used for transferring files from one machine to another. Like HTTP, FTP also works on a client-server model.
- Point-to-Point protocol (PPP) defines how two devices will authenticate each other and establish a direct link between them to exchange data.
- TCP/IP stands for Transmission Control Protocol/ Internet Protocol. It is a set of standardised rules that uses a client-server model of communication in which a user or machine (a client) requests a service by a server in the network.

CHECK YOUR PROGRESS

A. Multiple choice questions

1. The frequency band of Bluetooth radio is around _____ (a) 2.1 GHz (b) 2.2 GHz (c) 2.3 GHz (d) 2.4 GHz
2. What are the benefits of Bluetooth technology? (a) Cable replacement, ease of file sharing (b) Internet connectivity (c) Low-cost technology (d) All of the above
3. The Bluetooth technologies used in _____ (a) wireless keyboard (b) wireless mouse
4. What was the range of Bluetooth? (a) Only 10m (b) More than 10m (c) Less than 10m (d) None of the above
5. Which of the following is not a wireless transmission media (a) Radio waves transmission (b) Microwave transmission (c) Infrared transmission (d) fibre option transmission

6. Which of the following is not a wireless communication technology (a) Bluetooth (b) WiFi (c) WiMax (d) WAN
7. Which of the following is example of infrared transmission (a) mobile-to-mobile (b) mobile-to-printer (c) remote-control-to-TV (d) All of the above
8. The protocols is required for (a) flow control (b) access control (c) addressing (d) All of the above
9. Which of the following protocol is used to access the web pages (a) HTTP (b) FTP (c) PPP (d) SMTP
10. Which of the following protocol is used for email services (a) HTTP (b) FTP (c) PPP (d) SMTP

B. Fill in the blanks

1. Electromagnetic spectrum of frequency ranging from _____ is available for wireless communication.
2. The spectrum used by Bluetooth starts from _____ and ends at _____.
3. Bluetooth technology allows up to _____ devices to build a network.
4. The power consumption in Bluetooth is _____.
5. The data speed of Bluetooth is around _____.
6. Microwave transmission is used for _____ telecommunication
7. In Bluetooth technology, the communicating devices within a range of 10 meters build a personal area network called _____.
8. Bluetooth technology allows up to _____ devices to build a network.
9. IP protocol ensures each node connected to the Internet is assigned _____
10. Radio waves have the _____ frequency and _____ wavelength in the spectrum.
11. Radio waves are used for transmitting _____ signal and _____ signals.
12. Infrared transmission is used for _____ telecommunication.
13. The TV remote control uses _____ to control TV.
14. Wi-Fi is _____ wireless communication technology used in home networking.
15. HTTP is the _____ used to access the World Wide Web.

C. State whether true or false

1. Infrared is used for short-distance point-to-point communication.
2. The devices in a piconet work in a master-slave configuration.
3. In Bluetooth technology 7 devices can communicate at the same time.
4. Wi-Fi is low-cost and low-power wireless communication technology.
5. TCP/IP is a set of standardised rules that uses a client-server model of communication.
6. SMTP defines how data or commands transfer from client to server and vice versa.
7. FTP protocol used to make a dedicated and direct connection between two communicating devices.
8. HTTPS is used for unsecure transmission.
9. TCP uses a connectionless oriented service
10. It is not possible to communicate between two computers without using any wire or cable.

D. Short answer questions

1. What is the advantage and disadvantage of using Bluetooth technology?
2. What is HyperText Transfer Protocol (HTTP)? How it works?
3. What is File Transfer Protocol (FTP)? How it works?
4. What is Point to Point Protocol (PPP)? How it works?
5. Differentiate between SMTP and TCP/IP.
6. What is protocol and its need in computer network?
7. What is meaning of wireless communication?
8. Differentiate between radio wave and microwave transmission.
9. Write down common wireless devices to which you are using in your daily life.
10. What do you understand by mobile telecommunication technologies?

Session 4. Network Security

4.1 THREATS AND PREVENTION

Being alone is the most ideal situation for an individual in terms of security. It applies to computers as well. A computer with no link to an external device or computer is free from the security threats. However, it is not an ideal solution for a human being or a computer to stay alone in order to mitigate any security threats, as the world at present is on its way to become fully connected. This connectedness of various devices and computers has brought into our focus the various network threats and its prevention. Network security is concerned with protection of our device as well as data from illegitimate access or misuse. Threats include all the ways in which one can exploit any vulnerability or weakness in a network or communication system to cause harm or damage one's reputation. In network security, threat prevention refers to policies and tools that protect the network. With an increasing threat such as malware and ransomware from email spam and phishing attacks, advanced threat prevention requires an integrated, multi-layered approach to security. This may include tools for intrusion threat detection and prevention, advanced malware protection, and additional endpoint security threat prevention.

4.2 NETWORK SECURITY

All modern-day companies require a network of computers, servers, printers, switches, access points, and routers to operate. These devices and applications provide a huge benefit to the organisation. But on the other hand, they also represent a risk. It is necessary to protect a network. Network security helps to protect proprietary information from attack. Network security is the protection of networking infrastructure from unauthorized access, misuse, or theft. It involves creating a secure infrastructure for devices, applications, users, and applications to work in a secure manner.

Cyber Attack

Cyber-attacks are performed with malicious intent when a threat actor attempts to exploit a vulnerability or weakness in a system or individuals of an organization. These attacks threaten to steal, alter, destroy, disable or gain access to or make use of an unauthorized asset. (Figure 4.1)

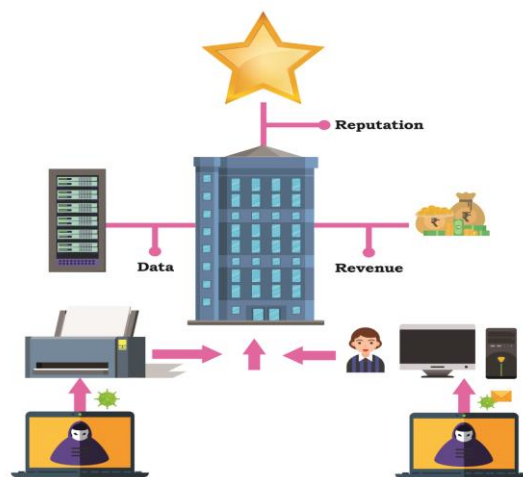


Fig. 4.1 Protection from cyber attacks

There are different types of cyber-attacks such as, Malware attacks, Network security attacks, and Wireless security attacks. Let us discuss these attacks.

4.3 MALWARE ATTACKS

Malware is a short term used for MALicious softWARE. It is any software developed with an intention to damage hardware devices, steal data, or cause any other trouble to the user. Various

types of malware have been created from time-to-time, and large-scale damages have been inflicted. Many of these malware programs have been identified and counter measures have been initiated. However, different types of malware keep on coming on regular basis that compromise the security of computer systems and cause intangible damages. Besides, each year, malware incur financial damages worth billions of dollars worldwide. Figure 4.2 shows the different types of malware such as Viruses, Worms, Ransomware, Trojans, and Spyware.

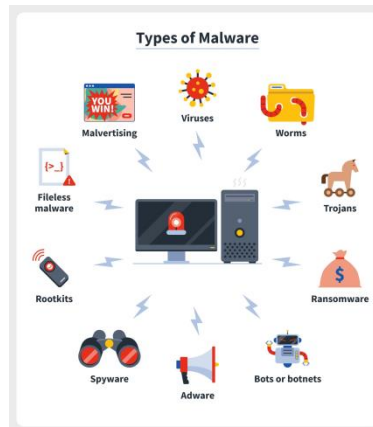


Fig. 4.2: Types of Malware

4.3.1 Virus

A computer virus is a piece of software code created to perform malicious activities and hamper resources of a computer system like CPU time, memory, personal files, or sensitive information. A computer virus infects other computer systems that it comes into contact with by copying or inserting its code into the computer programs or software (executable files). A virus remains dormant on a system and is activated as soon as the infected file is opened or executed by a user. Most viruses self-replicate without the knowledge of the user. These viruses can be spread from one system to another via email, instant messaging, website downloads, removable media (USB), and network connections. Some of the most common intentions or motives behind viruses include stealing passwords or data, corrupting files, spamming the user's email contacts, and even taking control of the user's machine. Some file types are more susceptible to virus infections – .doc/docx, .exe, .html, .xls/.xlsx, .zip. Some well-known viruses include CryptoLocker, ILOVEYOU, MyDoom, Sasser and Netsky, Slammer, Stuxnet.

4.3.2 Worms

Worm is also a malware that incurs unexpected or damaging behaviour on an infected computer system. Worms are standalone programs that are capable of working on its own. It does not require to host program to insert its code. This makes it different from virus. Also, a virus replicates when a user opens or execute the infected file, while a worm replicates on its own and can spread to other computers through the network. Worms are commonly used against email servers, web servers, and database servers. Some prominent examples of worms include Storm Worm, Sobig, MSBlast, Code Red, Nimda, Morris Worm.

4.3.3 Ransomware

It is a type of malware that targets user data. It either blocks the user from accessing their own data or threatens to publish the personal data online and demands ransom payment against the same. Some ransomware simply blocks the access to the data while others encrypt data making it very difficult to access. In May 2017, a ransomware WannaCry infected almost 200,000 computers across 150 countries. It worked by encrypting data and demanding ransom payments in the Bitcoin cryptocurrency. It literally made its victims "cry" and hence the name. Crypto-Malware is a type of ransomware that encrypts user files and requires payment within a time frame and often through a digital currency like Bitcoin. (Figure 4.3)



Fig. 4.3: Ransomware

4.3.4 Trojan

Since the ancient Greeks could not infiltrate the city of Troy using traditional warfare methods, they gifted the king of Troy with a big wooden horse with hidden soldiers inside and eventually defeated them. Borrowing the concept, a Trojan is a malware, that looks like a legitimate software and once it tricks a user into installing it, it acts pretty much like a virus or worm. When activated, Trojans can allow threat actors to spy on you, steal your sensitive data, and gain backdoor access to your system. However, a Trojan does not self-replicate or infect other files, it spreads through user interaction such as opening an email attachment or downloading and executing a file from the Internet. (Figure 4.4)

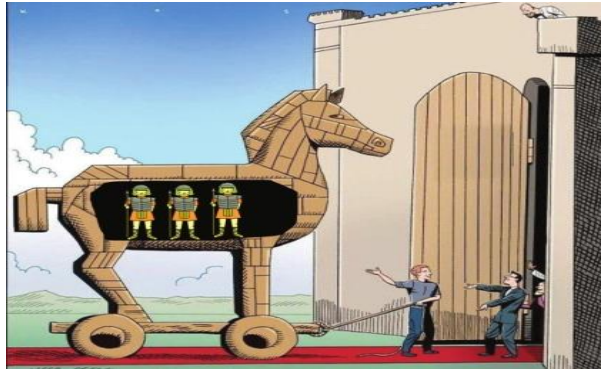


Fig. 4.4: Trojan horse

4.3.5 Spyware

It is a type of malware that spies on a person or an organisation by gathering information about them, without the knowledge of the user. It records and sends the collected information to an external entity without consent or knowledge of the user. Spyware usually tracks internet usage data and sells them to advertisers. They can also be used to track and capture credit card or bank account information, login and password information or user's personal identity.

4.3.6 Adware

An Adware is a malware that is created to generate revenue for its developer. An adware displays online advertisements using pop-ups, web pages, or installation screens. Once an adware has infected a substantial number of computer systems, it generates revenue either by displaying advertisements or using "pay per click" mechanism to charge its clients against the number of clicks on their displayed ads. Adware is usually annoying, but harmless. However, it often paves way for other malware by displaying unsafe links as advertisements.

4.3.7 Keyloggers

A keylogger can either be malware or hardware. The main purpose of this malware is to record the keys pressed by a user on the keyboard. A keylogger makes logs of daily keyboard usage and may send it to an external entity as well. In this way, very sensitive and personal information like passwords, emails, and private conversations can be revealed to an external entity without

the knowledge of the user. One strategy to avoid the threat of password leaks by keyloggers is to use a virtual keyboard while signing into your online accounts from an unknown computer.

A) Online Virtual Keyboard Vs On-Screen Keyboard

The names “on-screen” and “virtual” keyboard refer to any software-based keyboard and are sometimes used interchangeably. But, there exists a notable difference between “on-screen” and “online virtual” keyboards. Both types of keyboards may look the same, but the difference is in terms of the layout or ordering of the keys. The on-screen keyboard of an operating system uses a fixed QWERTY key layout (Figure 4.5), which can be exploited by sophisticated keylogger software. However, an online virtual keyboard randomises the key layout every time it is used (Figure 4.6), thereby making it very difficult for a keylogger software to know or record the key(s) pressed by the user.

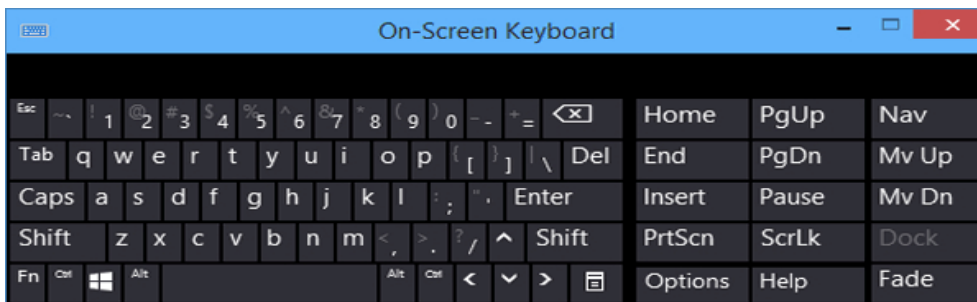


Fig. 4.5: A QWERTY keyboard layout

~	!	@	#	\$	%	^	&	*	()	-	+		
`	6	8	3	1	5	4	0	2	7	9	-	=		
r	w	e	q	t	y	u	p	i	o	{	}			
s	f	a	g	d	j	k	l	h	[]	\	/		
v	x	c	z	b	n	m	<	>	;	:	'	"		
CAPS LOCK						CLEAR						,	?	.

Fig. 4.6: Online virtual keyboard

4.3.8 Modes of Malware distribution

A malware once designed, can take many routes to reach your computer. Some of the common distribution channels for malware are:

Downloaded from the Internet – Most of the time, malware is unintentionally downloaded into the hard drive of a computer by the user. Of course, the malware designers are smart enough to disguise their malware, but we should be very careful while downloading files from the Internet (especially those highlighted as free stuff).

Spam Email – We often receive an unsolicited email with embedded hyperlinks or attachment files. These links or attached files can be malware.

Removable Storage Devices – Often, the replicating malware targets the removable storage media like pen drives, SSD cards, music players, mobile phones, etc. and infect them with malware that gets transferred to other systems that they are plugged into.

Network Propagation – Some malware like Worms have the ability to propagate from one computer to another through a network connection.

4.3.9 Combating Malware

Common signs of some malware infection include the following:

- frequent pop-up windows prompting you to visit

- some website and/or download some software;
- changes to the default homepage of your web browser;
- mass emails being sent from your email account;
- unusually slow computer with frequent crashes;
- unknown programs start-up as you turn on your computer;
- programs opening and closing automatically;
- sudden lack of storage space, random messages, sounds, or music start to appear;
- programs or files appear or disappear without your knowledge.

Malware exists and continues to evolve, and so is the mechanism to combat them. As the saying goes that prevention is better than cure, we list some preventive measures against the malware discussed earlier.

- ✓ Using antivirus, anti-malware, and other related software and updating them on a regular basis.
- ✓ Configure your browser security settings
- ✓ Always check for a lock button in the address bar while making payments.
- ✓ Never use pirated or unlicensed software. Instead go for Free and Open Source Software (FOSS).
- ✓ Applying software updates and patches released by its manufacturers.
- ✓ Taking a regular backup of important data.
- ✓ Enforcing firewall protection in the network.
- ✓ Avoid entering sensitive (passwords, pins) or personal information on unknown or public computers.
- ✓ Avoid entering sensitive data on an unknown network (like Wi-Fi in a public place) using your own computer also.
- ✓ Avoid clicking on links or downloading attachments from unsolicited emails.
- ✓ Scan any removable storage device with an antivirus software before transferring data to and from it.
- ✓ Never share your online account or banking password/pins with anyone.
- ✓ Remove all the programs that you don't recognise from your system.
- ✓ Do not install an anti-spyware or antivirus program presented to you in a pop-up or ad.
- ✓ Use the pop-up window's 'X' icon located on the top-right of the popup to close the ad instead of clicking on the 'close' button in the pop-up. If you notice an installation has been started, cancel immediately to avoid further damage.

4.4 ANTIVIRUS

Antivirus is a software, also known as anti-malware. Initially, antivirus software was developed to detect and remove viruses only and hence the name anti-virus. However, with time it has evolved and now comes bundled with the prevention, detection, and removal of a wide range of malware.

4.5 SPAM

Spam is a broad term and applies to various digital platforms like messaging, forums, chatting, emailing, advertisement, etc. However, the widely recognised form is email spam. Depending on their requirements, organisations or individuals buy or create a mailing list (list of email addresses) and repeatedly send advertisement links and invitation emails to a large number of users. This creates unnecessary junk in the inbox of the receiver's email and often tricks a user into buying something or downloading a paid software or malware.

Nowadays, email services like Gmail, Hotmail, etc. have an automatic spam detection algorithm that filters emails and makes things easier for the end users. A user can also mark an undetected unsolicited email as “spam”, thereby ensuring that such type of email is not delivered into the inbox as normal email in future.

4.6 HTTP V/S HTTPS

Both the HTTP (Hyper Text Transfer Protocol) and its variant HTTPS (Hyper Text Transfer Protocol Secure) are a set of rules (protocol) that govern how data can be transmitted over the WWW (World Wide Web). In other words, they provide rules for the client web browser and servers to communicate.

HTTP sends information over the network as it is. It does not scramble the data to be transmitted, leaving it vulnerable to attacks from hackers. Hence, HTTP is sufficient for websites with public information sharing like news portals, blogs, etc. However, when it comes to dealing with personal information, banking credentials and passwords, we need to communicate data more securely over the network using HTTPS. HTTPS encrypts the data before transmission. At the receiver end, it decrypts to recover the original data. The HTTPS based websites require SSL Digital Certificate.

4.7 NETWORK SECURITY ATTACKS

Network security attacks have become more common in recent years in part because small and mid-sized businesses are not making investments into securing their systems fast enough. As a result, hackers target businesses because their systems are often easier to compromise. Other reasons include a rise in hacktivism, bring your own device (BYOD) use, and cloud-based applications.

Types of network security attacks includes, Denial of Service (DoS), Distributed Denial of Service (DDoS), Buffer Overflow Attacks, Ping Attacks, SYN Flood, DNS Amplification, Back Door, Spoofing, Smurf Attack, TCP/IP Hijacking, Man In The Middle Attacks, Replay Attacks, DNS Poisoning, ARP Poisoning, Domain Kiting, Typosquatting, Client Side Attacks, Watering Hole Attacks, Zero Day Attacks. Some of them are discussed here.

4.7.1. Denial of Service (DoS)

In a Denial of Service (DoS) attack a malicious threat attacker overloads a server with data preventing valid request coming from real clients on the network. The DoS attack floods the victim resource with traffic, making the resource appear busy. A DoS attack can also be performed on entire networks because the attack is targeted at the central router or firewall. (Figure 4.7) As a result, network bandwidth is compromised, and denies access to all systems on that network. A DoS attack on website will flood it with a very large number of network packets by using different IP addresses and overload the web server and will not be able to provide service to a legitimate user. The users will think that the website is not working, causing damage to the victim’s organisation. Same way, DoS attacks can be done on resources like email servers, network storage, disrupting connection between two machines. If a DoS attack makes a server crash, the server or resource can be restarted to recover from the attack. However, a flooding attack is difficult to recover from, as there can be some genuine legitimate requests in it as well.

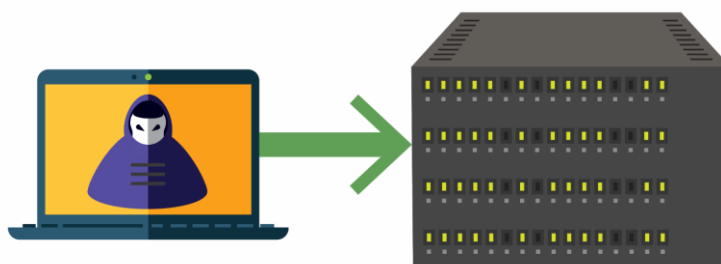


Fig. 4.7: Denial of Service (DoS)

To prevent a denial of service,

- Buy more bandwidth.
- Build redundancy into your infrastructure.
- Deploy anti-DoS hardware and software modules.
- Deploy a DoS protection appliance.
- Protect your DNS servers.

4.7.2. Distributed Denial of Service (DDoS)

A variant of DoS, known as Distributed Denial of Service (DDoS) is an attack on a system that is launched from multiple sources and is intended to make a computer's resources or services unavailable. DDoS attacks typically include sustained, abnormally high network traffic. The attacker installs a malicious software known as Bot on the Zombie machines, which gives it control over these machines. Depending upon the requirement and availability, the attacker activates a network of these Zombie computers known as Bot-Net to carry out the DDoS attack. While as a simple DoS attack may be countered by blocking requests or network packets from a single source, DDoS is very difficult to resolve, as the attack is carried from multiple distributed locations. (Figure 4.8)



Fig. 4.8: Distributed Denial of Service (DoS)

To prevent a distributed denial of service attack,

- Develop a denial of service response plan.
- Secure the network infrastructure.
- Filter routers at the edge of network to spot and drop DDoS connections.
- Blackholing the site that is being DDoS'd, thereby directing all traffic to an invalid address.

4.7.3. Buffer Overflows

In this attack, the attacker overwrites certain memory areas of the computers within the network with code that will be executed later when the buffer overflow occurs as programming error. Once the malicious code is executed, an attacker can initiate a DoS attack or gain access to the network. A buffer overflow itself doesn't cause damage, it does expose a vulnerability. (Figure 4.9)

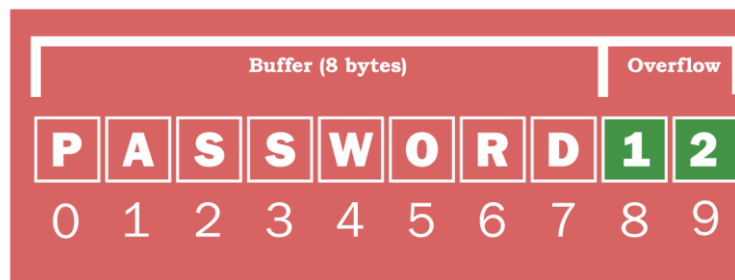


Fig. 4.9: Buffer Overflows

To prevent a buffer overflow attack,

- Perform routine code auditing (automated or manual).
- Provide training including bounds checking, use of unsafe functions, and group standards.
- Use compiler tools such as StackShield, StackGuard, and Libsafe.
- Use safe functions such as strncpy instead of strcpy, strncpy instead of strcpy, etc
- Patch web and application servers regularly and be aware of bug reports relating to applications upon which your code is dependent.
- Periodically scan application with one or more of the commonly available scanners that look for buffer overflow flaws in your server products and your custom web applications.

4.7.4. Ping Attacks

A ping is used to check connectivity between a source and a destination device by way of ICMP echo-requests and echo-reply messages. A Ping Attack floods the target device with requests packets. The destination device is forced to respond with an equal number of reply packets and cannot keep up with the volume of requests. This causes the target to become inaccessible to normal traffic and unresponsive to normal ping requests. (Figure 4.10)

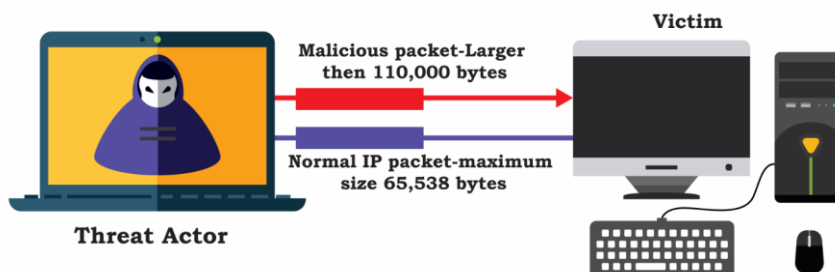


Fig. 4.10: A ping attack

To prevent a Ping attack,

- Configure firewall to block ICMP pings from entering your network at the perimeter.
- Add filters to tell your router to detect and drop malformed data packets or those coming from suspicious sources.
- Look for spoofed packets that do not originate from within your network, also known as egress filtering.
- Install network monitoring software to alert for traffic patterns that are not ordinary.
- Scan network for open ports on a regular basis that is outside of your baseline.

4.7.6. DNS Amplification

A Domain Name Server (DNS) Amplification attack is a popular form of Distributed Denial of Service (DDoS), in which attackers use publicly accessible open DNS servers to flood a target system with DNS response traffic. (Figure 4.11) The primary technique consists of an attacker sending a DNS name lookup request to an open DNS server with the source address spoofed to be the target's address. When the DNS server sends the DNS record response, it is sent instead to the target. Attackers will typically submit a request for as much zone information as possible to maximize the amplification effect.

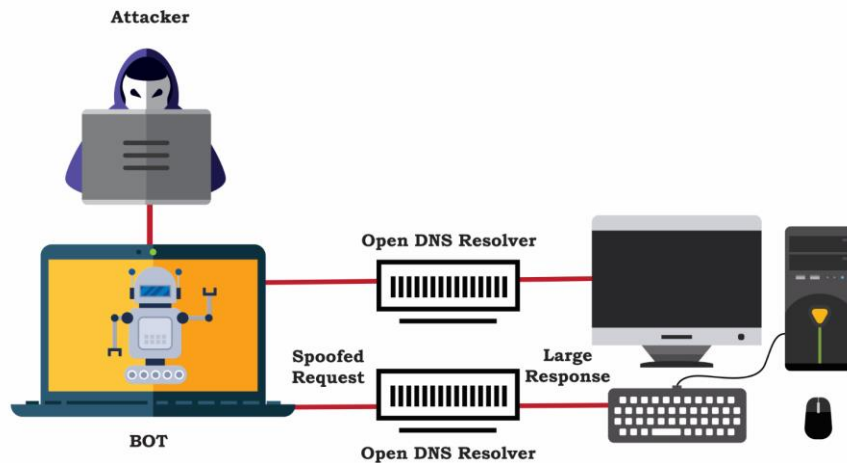


Fig. 4.11: Domain Name Server (DNS) Amplification attack

To prevent DNS Amplification attacks,

- Implement Source IP Verification on network device.
- Disable Recursion on Authoritative Name Servers.
- Limit Recursion to Authorized Clients.
- Implement Response Rate Limiting (RRL) setting on DNS Server.

4.7.7. Back Door

A backdoor is a malware type that negates normal authentication procedures to access a system. As a result, remote access is granted to resources within an application, such as databases and file servers, giving perpetrators the ability to remotely issue system commands and update malware. Backdoor installation is achieved by taking advantage of vulnerable components in a web application. Once installed, detection is difficult as files tend to be highly obfuscated. (Figure 4.12)



Fig. 4.12: Backdoor malware attack

To prevent Back Door Attacks,

- Use an Anti-virus solution.
- Implement a network monitoring tool.
- Implement a solution to detect untrusted software on endpoints.
- Ensure that every device is protected by a host firewall.

4.7.8. Spoofing Attack

A spoofing attack occurs when a malicious party impersonates another device or user on a network in order to launch attacks against network hosts, steal data, spread malware or bypass access controls. There are several different types of spoofing attacks that malicious parties can use to accomplish this. Some of the most common methods include IP address spoofing attacks, ARP spoofing attacks and DNS server spoofing attacks. (Figure 4.13)

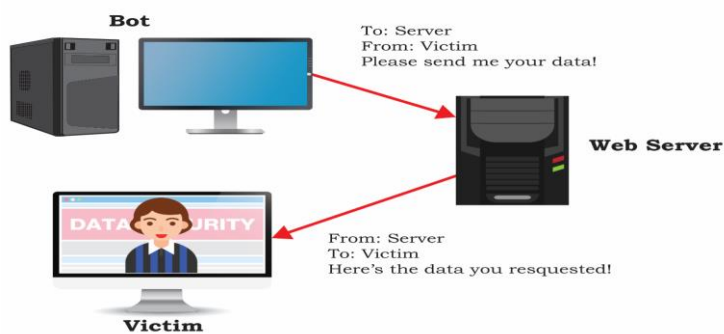


Fig. 4.13: Spoofing attack

To prevent a Spoofing Attack,

- Packet filtering.
- Avoid trust relationships with unknown entities.
- Implement a spoofing detection software.
- Enable cryptographic network protocols, such as Transport Layer Security (TLS), Secure Shell (SSH), HTTP Secure (HTTPS).

4.7.10. Man In The Middle Attacks

A Man-in-the-Middle (MitM) attack is when an attacker intercepts communication between two parties either to secretly eavesdrop or modify traffic traveling between the two. Attackers might use MitM attacks to steal login credentials or personal information, spy on the victim, or sabotage communications or corrupt data. MitM attacks consist of sitting between the connection of two parties and either observing or manipulating traffic. This could be through interfering with legitimate networks or creating fake networks that the attacker controls. Compromised traffic is then stripped of any encryption in order to steal, change or reroute that traffic to the attacker's destination of choice. (Figure 4.14)

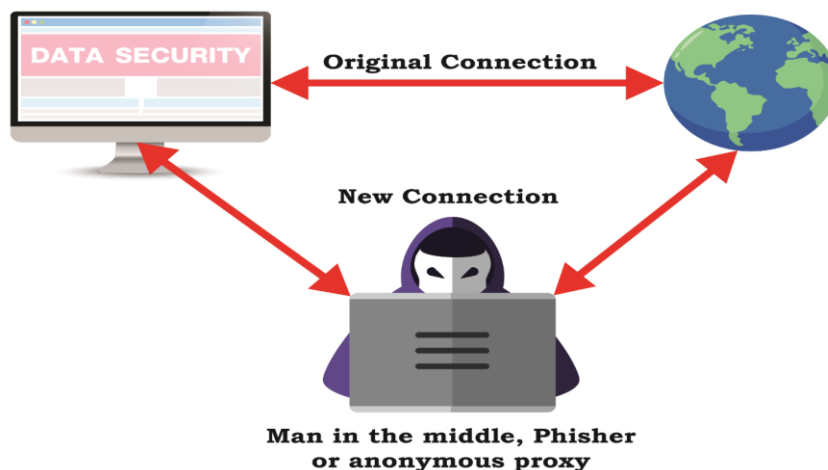


Fig. 4.14: Man-in-the-Middle (MitM) attack

To prevent a MITM Attack,

- Incorporate the latest version of encryption protocols such as TLS.1.3 on infrastructure assets.
- Train staff not to use open public Wi-Fi or Wi-Fi offerings at public places where possible.
- Train staff to recognize browser warnings from sites or connections that may not be legitimate.
- Implement VPNs to help ensure secure connections.

- Implement multi-factor authentication

4.7.11. Bluetooth Vulnerabilities

Several attack methods target Bluetooth devices specifically. These include:

Bluejacking Bluetooth attacks – This is the practice of sending unsolicited messages to nearby Bluetooth devices. Bluejacking messages are typically text, but can also be images or sounds. Bluejacking is relatively harmless but does cause some confusion when users start receiving messages.

Bluesnarfing Bluetooth attacks – Any unauthorized access to or theft of information from a Bluetooth connection is bluesnarfing. A bluesnarfing attack can access information, such as email, contact lists, calendars, and text messages.

Bluebugging Bluetooth attacks – Bluebugging attacks allow an attacker to take over a mobile phone. Attackers can listen in on phone conversations, enable call forwarding, send messages, and more.

To prevent Bluetooth Vulnerability Attacks,

- Enable the “find my device” service on your phone through a trustworthy entity like Apple or Google so you have a way of using their technologies to find and remotely lock your phone if you lose it.
- Avoid the use of Bluetooth to communicate sensitive information like passwords.
- Do not leave your Bluetooth in “discoverable” mode when you’re pairing a new peripheral with your phone or laptop.
- Turn Bluetooth off when you’re not using it.

4.7.12 Evil Twin

In this attack a hacker sets up a fake Wi-Fi network that looks like a legitimate access point to steal victims’ login credentials or other sensitive information. The attack can be performed as a man-in-the-middle (MITM) attack. Because the hacker owns the equipment being used, the victim will have no idea that the hacker might be intercepting things like bank transactions.

An evil twin access point can also be used in a phishing scam. A victim will connect to the evil twin. It will prompt to enter sensitive data, such as their login details. After getting these details the hacker simply disconnects the victim and show that the server is temporarily unavailable.

To prevent Evil Twin Attacks,

- Do not log into any accounts on public Wi-Fi.
- Avoid connecting to Wi-Fi hotspots that say ‘Unsecure,’ even if it has a familiar name.
- Use 2-factor-authentication for all your sensitive accounts. Learn to recognize social engineering attacks, phishing, and spoofed URLs.
- Visiting only HTTPS websites, especially when on open networks.
- Use a VPN whenever you connect to a public hotspot.

4.7.13 Packet Sniffing and Eavesdropping

An eavesdropping attack, also known as a sniffing or snooping attack, is a theft of information as it is transmitted over a network by a computer, smartphone, or another connected device. The attack takes advantage of unsecured network communications to access data as it is being sent or received by its user. An eavesdropping attack can be difficult to detect because the network transmissions will appear to be operating normally.

To be successful, an eavesdropping attack requires a weakened connection between a client and a server that the attacker can exploit to reroute network traffic. The attacker installs network monitoring software, the “*packet sniffer*,” on a computer or a server to intercept data as it is transmitted.

To prevent Packet Sniffing and Eavesdropping,

- Use a personal firewall.
- Keep antivirus software updated.
- Use a virtual private network (VPN).
- Use a strong password and changing it frequently.
- Ensure smartphone is running the most up to date version.

4.9 FIREWALL

A firewall is a network security device designed to protect a trusted private network from unauthorised access or traffic originating from an untrusted outside network such as Internet or different sections of the same network, to which it is connected (Figure 4.18). Firewall can be implemented in software, hardware or both. As discussed earlier, a malware like worm has the capability to move across the networks and infect other computers. The firewall acts as the first barrier against malware.

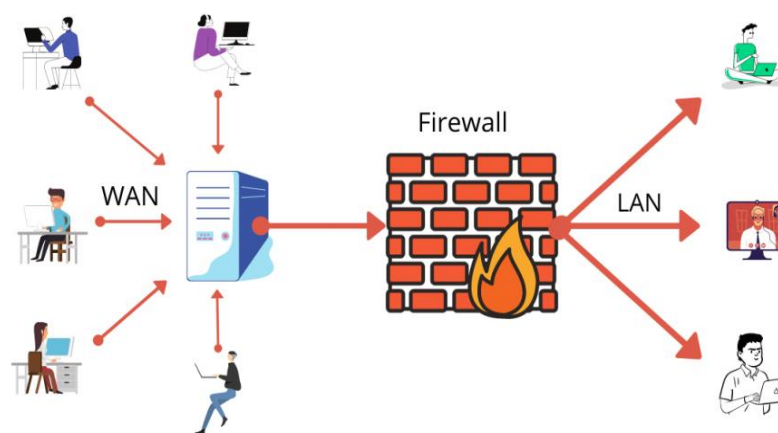


Fig. 4.15: A firewall between two networks

A firewall acts as a network filter and based on the predefined security rules, it continuously monitors and controls the incoming and outgoing traffic. As an example, a rule can be set in the firewall of a school LAN, that a student cannot access data from the finance server, while the school accountant can access the finance server.

4.4.1 Types of Firewall

There are different types of firewalls. Some of the popular firewall types includes Network firewall, Web application firewall, Hardware-based, Software-based, Cloud-based, Personal computer (Windows, macOS) firewall and Mobile firewall. They are mostly categorized under two types – network-based and host-based.

Network Firewall – If the firewall is placed between two or more networks and monitors the network traffic between different networks, it is termed as Network Firewall. They filter all the traffic coming and going across a network. It is usually installed at the network edge and acts as the first layer of protection to block any unwanted traffic.

Host-based Firewall – If the firewall is placed on a computer and monitors the network traffic to and from that computer, it is called a host-based firewall. They are installed on different network nodes, controlling each outgoing and incoming packet or byte. The firewall consists of an application suite installed on a server or computer. Host-based firewalls can protect the individual host against unauthorized access and attacks.

4.10 COOKIES

The term "cookie" was derived from the term "magic cookie" used by Unix programmers to indicate a packet of data that a program receives and sends it back unchanged. A computer cookie is a small file or data packet, which is stored by a website on the client's computer. A cookie is edited

only by the website that created it, the client's computer acts as a host to store the cookie. Cookies are used by the websites to store browsing information of the user. For example, while going through an e-commerce website, when a user adds items to cart, the website usually uses cookies to record the items in the cart. A cookie can also be used to store other user-centric information like login credentials, language preference, search queries, recently viewed web pages, music choice, favorite cuisine, etc., that helps in enhancing the user experience and making browsing time more productive. Depending upon their task, there are different types of cookies. Session cookies keep track of the current session and even terminate the session when there is a time-out (banking website). So, if you accidentally left your e-banking page open, it will automatically close after the time-out. Similarly, authentication cookies are used by a website to check if the user is previously logged in (authenticated) or not. This way, you don't need to login again and again while visiting different web pages or links of the same website. You might have also noticed that certain information like your Name, Address, Contact, D.O.B, etc. automatically fills up while filling an online form. This auto-fill feature is also implemented by websites using cookies.

4.10.1 Threats due to Cookies

Usually, cookies are used for enhancing the user's browsing experience and do not infect your computer with malware. However, some malware might disguise as cookies e.g. "supercookies". There is another type of cookie known as "Zombie cookie" that gets recreated after being deleted. Some third-party cookies might share user data without the consent of the user for advertising or tracking purposes. As a common example, if you search for a particular item using your search engine, a third-party cookie will display advertisements showing similar items on other websites that you visit later. So, one should be careful while granting permission to any websites to create and store cookies on the user computer.

4.11 HACKERS AND CRACKERS

Hackers and crackers are people having a thorough knowledge of the computer systems, system software (operating system), computer networks, and programming. They use this knowledge to find loopholes and vulnerabilities in computer systems or computer networks and gain access to unauthorised information. In simple terms, a hacker is a person that is skilled enough to hack or take control of a computer system. Depending on the intent, there are different types of hackers.

4.11.1 White Hats: Ethical Hacker

If a hacker uses its knowledge to find and help in fixing the security flaws in the system, its termed as White Hat hacker. These are the hackers with good intentions. They are actually security experts. Organisations hire ethical or white hat hackers to check and fix their systems for potential security threats and loopholes. Technically, white hats work against black hats.

4.11.2 Black Hats: Crackers

If hackers use their knowledge unethically to break the law and disrupt security by exploiting the flaws and loopholes in a system, then they are called black hat hackers.

4.11.3 Grey Hats

The distinction between different hackers is not always clear. There exists a grey area in between, which represents the class of hackers that are neutral, they hack systems by exploiting its vulnerabilities, but they don't do so for monetary or political gains. The grey hats take system security as a challenge and just hack systems for the fun of it.

4.12 SECURITY ISSUES AND THREATS IN EMAIL COMMUNICATION

E-mail is one of the main modes of communication today, but it is also not free from threats. The threats in Email communication are as follows.

Eavesdropping – E-mail messages are communicated through Internet. So, it is very easy for someone to track or capture your message and read it.

Identity Theft – If proper security protocols are not followed, someone may steal or capture your username/ password and used to read your email messages and send email messages from your account without your knowledge.

Message Modification – Anyone having administrative rights on any of SMTP server can captures your message, read and also alter your message contents if it is not encrypted.

False Messages – Sender’s name can easily be fabricated so it is very easy to send message that pretends to be send by someone else.

Unprotected Backups – Messages generally stored in plain Text on SMTP server and also backups can be created. Even if you delete the message they can be residing on the severs/ backup-servers for years. So, anyone who accesses these servers can also access or read your message.

Repudiation – As it is known that email messages can easily be forged so anyone sending you some message can later on deny regarding sending of message and it is very difficult to prove it. This has implications corresponding to emails use as contracts in business communications.

Email spoofing – Sometime email that pretends to be received from an authentic source but in actual it is send from somewhere else.

Email Spamming – Spam or junk mail refers to sending of email to number of persons for any advertisement purpose or for some malicious intent. To send spam often lists are created by searching data from Internet, or by stealing mailing list from the internet.

Email bombing – E-mail “bombing” refers to sending identical mail repeatedly by abusers to a particular address/user.

Sending threats – Sometimes false statements are also forwarded to third parties or users to injure the reputation of some particular person. It is called as Defamation; a communication is not considered defamatory unless it is forwarded to someone other than the target.

Email frauds – Email Fraud is intentional deception made for some personal or monetary gain.

Emails used as tools to spread malicious software – Emails are also used as tools to spread viruses, worms and other malicious software. They are attached to emails as attachment, when you click on them they attack your computer or browser.

Phishing- It is also most common attack through email, defined to steal confidential information like passwords, ATM pin and other bank credentials. It works as some email coming to you that pretends to be from some trusted source you know like your bank. These emails entice you to click on some link present in email or to open some attachment or respond to some message and that click directed to you their site in actual but it appears like your trusted website of bank and ask to fill some confidential information like passwords which is actually stolen from you and use for any malicious intent later on.

SUMMARY

- Malware is a software developed with an intention to damage computer hardware, software, steal data, or cause any other trouble to a user.
- A virus is a piece of software code created to perform malicious activities and hamper resources of a computer system.
- The Worm is also a malware that incurs unexpected or damaging behaviour on an infected computer system.
- Worms are standalone programs that are capable of working on its own.
- Ransomware is a type of malware that targets user data.
- Ransomware either blocks the user from accessing their own data or threatens to publish their personal data online and demands ransom payment against the same.
- Trojan is a malware, that looks like a legitimate software and once it tricks a user into installing it, it acts pretty much like a virus or a worm.

- Spyware records and sends the collected information to an external entity without the consent or knowledge of a user.
- An adware displays unwanted online advertisements using pop-ups, web pages, or installation screens.
- A keylogger makes logs of daily keyboard usage and may send it to an external entity as well.
- The on-screen keyboard is an application software that uses a fixed QWERTY key layout.
- Online virtual keyboard is a web-based or a standalone software with a randomised key layout every time it is used.
- A malware can take many routes to reach your computer, which include: Downloaded from the Internet, Spam Email, using infected Removable Storage Devices, and network propagation.
- An antivirus software is used to detect and remove viruses and hence the name anti-virus.
- Antiviruses now come bundled with the prevention, detection, and removal of a wide range of malware.
- Some of the prominent methods of malware identification used by an antivirus include: Signature-based detection, Sandbox detection, Heuristics.
- Any unwanted data, information, email, advertisement, etc. is called Spam.
- HTTP (Hyper Text Transfer Protocol) and HTTPS (Hyper Text Transfer Protocol Secure) are a set of rules or protocol that govern how data can be transmitted over the World Wide Web.
- Firewall is a network security system designed to protect a trusted private network from unauthorised access or traffic originating from an untrusted external network.
- There are two basic types of firewalls — Network Firewall and Host-based Firewall.
- A computer cookie is a small file or data packet, which is stored by a website on the client's computer.
- Cookies are used by the websites to store browsing information of the user.
- Hackers/Crackers find loopholes and vulnerabilities in computer systems or computer networks and gain access to unauthorised information.
- If a hacker uses its knowledge to find and help in fixing the security flaws in the system, its termed as White Hat hacker.
- If hackers use their knowledge unethically to break the law and disrupt security by exploiting the flaws and loopholes in a system, then they are called black hat hackers.
- The grey hats take system security as a challenge and just hack systems for the fun of it.
- The Denial of Service (DoS) attack floods the victim resource with traffic, making the resource appear busy.
- Distributed Denial of Service (DDoS) is an attack, where the flooded requests come from compromised computer (Zombies) systems distributed across the globe or over a very large area.
- Network Intrusion refers to any unauthorised activity on a computer network.
- Snooping is the process of secret capture and analysis of network traffic by malicious users.
- Eavesdropping is an unauthorised real-time interception or monitoring of private communication between two entities over a network.

CHECK YOUR PROGRESS**A. Multiple choice questions**

1. Which of the following comes under the cyber-attacks (a) Malware attacks (b) Network security attacks (c) Wireless security attacks (d) All the above
2. A computer virus is (a) Hardware (b) Software (c) Bacteria (d) Freeware
3. Who is known by the name of crackers in the context of computer security? (a) Black Hat Hackers (b) White Hat Hackers (c) Elite Hackers (d) Script Kiddie
4. In computing _____ is a network security system that monitors and controls incoming and outgoing network traffic based on predetermined security rules. (a) Spyware (b) Cookie (c) Spam (d) Firewall
5. What is the system designed to prevent unauthorized access to or from a private network called as? (a) Computer scan (b) Digital scan (c) Biotech (d) Firewall
6. What does DDoS stand for? (a) Data Denial-of-Service (b) Distributed Denial-of-Service (c) Distributed Data of Server (d) Distribution of Data Service
7. Which of the following is malicious software that, on execution, runs its own code and modifies other computer programs? (a) Virus (b) Spam (c) Spyware (d) Adware
8. Which of the following is not a type of hacker? (a) White Hat (b) Black Hat (c) Green Hat (d) Grey Hat
9. Which of the following is not a type of firewall? (a) Hardware-based, (b) Software-based, (c) Cloud-based, (d) Python based
10. Which of the following usually observe each activity on the internet of the victim, gather all information in the background, and send it to someone else? (a) Spyware (b) Malware (c) Adware (d) Trojan

B. Fill in the blanks

1. Network security is the protection of networking infrastructure from _____ access.
2. Malware is a short term used for _____ software.
3. A computer virus is a piece of software code created to perform _____ activities and _____ resources of a computer system
4. Most viruses _____ without the knowledge of the user.
5. Worms are commonly used against _____ servers, _____ servers, and _____ servers.
6. Ransomware are simply type of malware that targets _____.
7. Spyware records and sends the collected information to an _____ without consent or knowledge of the user.
8. A keylogger makes logs of daily keyboard usage and send it to an _____.
9. Antivirus software was developed to _____ and _____ viruses.
10. Cookies are used by the websites to store _____ of the user.

C. State whether true or false

1. Security threat prevention refers to policies and tools that protect the network.
2. Network security helps to protect proprietary information from attack.
3. A computer virus is a hardware used to hamper resources of a computer system.
4. Crypto-Malware is a type of ransomware that encrypts files and requires payment through a digital currency like Bitcoin.
5. Trojans does not allow threat actors to spy and steal your sensitive data.
6. An Adware is a malware that is used to generate revenue for customers.

7. Malware are distributed via spam email to user's computer.
8. Antivirus cannot be used to detect malware.
9. Email services like Gmail, Hotmail etc. have an automatic spam detection algorithm.
10. Network Intrusion refers to authorised activity on a computer network.

D. Short answer questions

1. Why is a computer considered to be safe if it is not connected to a network or Internet?
2. What is a computer virus? Name some computer viruses that were popular in recent years.
3. How is a computer worm different from a virus?
4. How is Ransomware used to extract money from users?
5. How did a Trojan get its name?
6. How does an adware generate revenue for its creator?
7. Briefly explain two threats that may arise due to a keylogger installed on a computer.
8. How is a Virtual Keyboard safer than On Screen Keyboard?
9. List common signs of malware infection and explain different modes of malware distribution.
10. List some preventive measures against malware infection.
11. What are the risks associated with HTTP? How can we resolve these risks by using HTTPS?
12. List one advantage and disadvantage of using Cookies.
13. Write a short note on White, Black, and Grey Hat Hackers.
14. Differentiate between DoS and DDoS attack.
15. How is Snooping different from Eavesdropping?

Module 3

Python Programming

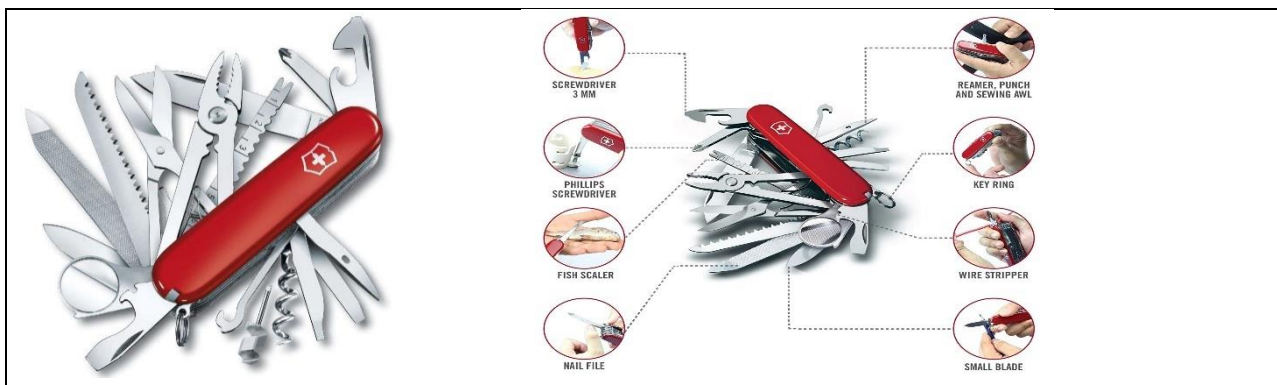
Module Overview

Python is a very simple programming language, developed by **Guido van Rossum** in 1989. It is named after the comedy television show Monty Python's Flying Circus and not after the Python snake.

Python is a dynamic, interpreted language. It is platform independent, free and open source, easy to learn. There are no type declarations of variables, parameters, functions, or methods in source code. This makes the code short and flexible that leads to faster execution.

Python is a general-purpose language, which can be used in various domains including: web applications, big data applications, data science, desktop software, mobile applications, Machine learning and AI.

In our daily life we may require different tools to perform different tasks such as scissor for cutting paper, screw driver for tightening of screws, removing the hook, scaling the fish or making use of the scissors. If you can get one tool to perform plenty of task then that will be the most preferred tool. Swiss knife is such a multi-functional tool having various tools such as blade, opener, screwdriver, reamer and others one in all. In the same way Python is considered as a multifunctional programming language. Also, Python is an object-oriented language as every entity in Python is considered as an object, close to real world.



Learning Outcomes

After completing this module, you will be able to:

- Install the Python IDE and learn to code and execute program
- Code and execute simple programs in Python
- Code and execute programs using control structures in Python
- Code and execute programs using functions in Python
- Code and execute python programs using Strings in Python
- Code and execute python programs using List in Python
- Code and execute python programs using Tuples and Dictionary in Python

Module Structure

Session 1: Python Basics

Session 2: Control Structures

Session 3: Functions

Session 4: Strings

Session 5: Lists

Session 6: Tuples and Dictionaries

Session 7: Python Basics

Session 1: Python Basics

Python is a high-level programming language that makes it easy to learn. It doesn't require us to understand the details of the computer to develop programs efficiently. It is the robust programming language used in many domains from web applications, data analysis, data science, machine learning, and AI. It is important to understand the syntax and semantics to learn any programming language.

In this chapter you will understand the basics of Python programming, its features, and installation of Python. You will also learn about *keywords*, *identifiers*, *variables*, *data types*, *operators* and *statements* in Python is also explained. You will be able to run the sample code in Python interpreter as well as write the programs in IDLE execute them run to get the output.

1.1 Features of Python programming

Python is an interpreted, high-level, general-purpose programming language. It allows solving complex problems in less time and fewer lines of code. It is simple and easy-to-learn. Python can be used to build all types of applications ranging from small and simple scripts to complex

machine learning algorithms. Python source code is available under the GNU General Public License (GPL). The Python programming language first released in 1991 as Python 0.9.0. Python 3 is the latest, most user-friendly, secure, and most powerful. Python 3.1.8 is the newest major release of the Python programming language, used in this book. It will keep on releasing the latest versions subsequently with time. Python comes with following features.

Readable and Easy to Learn – Python is a very readable language. Python programs are easy to understand as they have a clearly defined syntax and relatively simple structure. It is an interpreted language, as Python programs are executed by an interpreter.

Case Sensitive – Python is case-sensitive language, means, the variables “NUMBER” and “number” are different in Python. Python uses indentation for blocks and nested blocks.

Free and Open Source – Python is a free and open source programming language. This means you can download it for free and use it in your application.

Cross platform – Python is portable and platform independent. It can run on various operating systems such as Mac, Windows, Linux, Unix and any hardware platforms. This makes it a cross platform and portable language.

Large standard library – Python comes with a large standard library of predefined functions that has some handy codes and functions which can be used while writing code in Python. Python is also helpful in web development. Many popular web services and applications are built using Python.

Supports exception handling – Python supports exception handling which means we can write less error prone code and can test various scenarios that can cause an exception later on.

Advanced features – Supports generators and list comprehensions. We will cover these features later.

Automatic memory management – Python supports automatic memory management which means the memory is cleared and freed automatically.

Some of the main uses of Python are:

1. Used on a server to create web applications.
2. Handle big data and perform complex mathematical calculations.
3. Connect to database systems, read and modify files.

Python also has a rich set of libraries that makes it popular among software developers. It has been widely used in development of software applications for various domains. Let us start working with Python Interpreter.

1.2 Working with Python interpreter

To code and execute a Python program, you need to install Python interpreter. Python interpreter is available for various operating systems such as Windows, Linux and Mac. It is also available online in cloud environment. Let us see how to install Python interpreter on Windows 10 platform in Practical Activity 1.1.

Activity 1

Practical Activity 1.1. Install the Python compiler (binaries)

Resources required

A Desktop or Laptop computer with operating system installed, Internet Connectivity

Procedure

Step 1. Open the official website of Python <https://www.python.org/> in the web browser. Move to the Download for Windows section as shown in Figure 1.1.

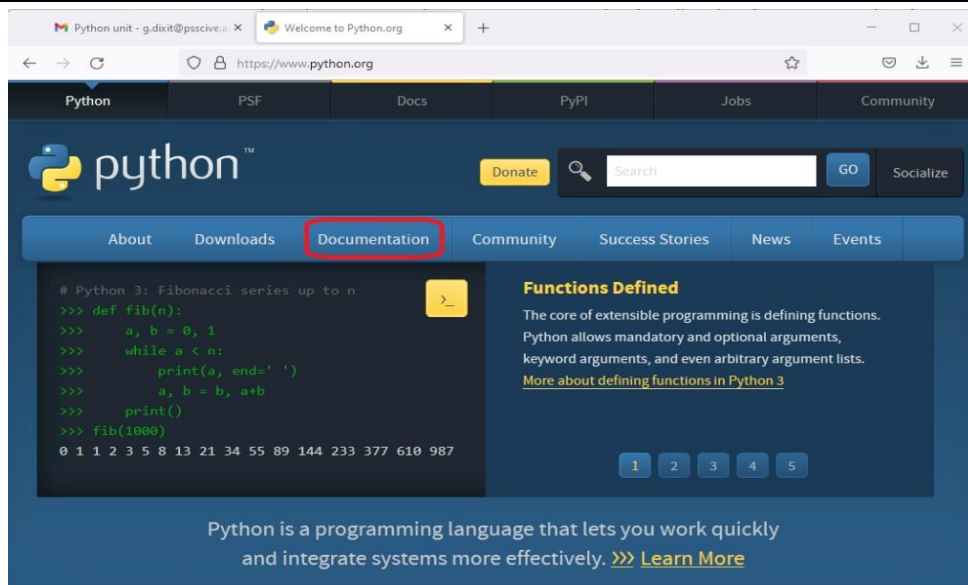


Fig. 1.1 Official website of Python

Step 2. Choose the latest version of Python 3.10 release.

Step 3. Download the installer. Once it is downloaded, run the Python installer.

Step 4. Check the **Install launcher for all users** check box. Further, you may check the **Add Python 3.10** to path check box to include the interpreter in the execution path as shown in figure 1.2(a).

Step 5. Select **Customize installation**.

Step 6. After selecting the Advanced options, click **Install** to start installation.

Step 7. Once the installation is over, a window will appear as shown in Figure 1.2(b) showing the **Python Setup Successful**.

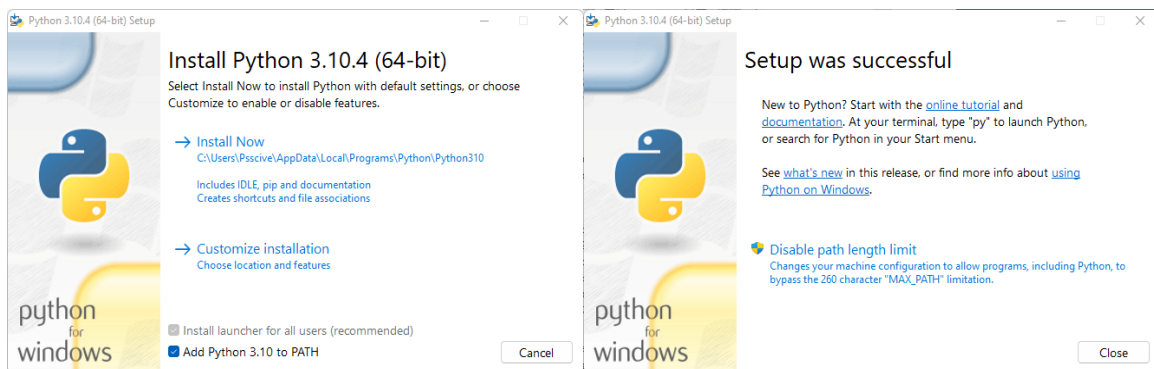


Fig. 1.2 (a) Installing Python setup (b) Setup successful

This shows the successful installation of Python 3.10 under Windows 1. You can verify if the Python installation is successful either through the command line or through the IDLE app that gets installed along with the installation. Search for the command prompt and type “Python” as shown in Figure 1.3. You can see that Python 3.10 is successfully installed.

```

C:\Users\Psscive>python
Python 3.10.4 (tags/v3.10.4:9d38120, Mar 23 2022, 23:13:41)
[MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more in
formation.
>>>

```

Fig. 1.3 Python window after installation

After installing Python, you can start developing Python programs. An alternate way to reach python is to search for “Python” in the start menu and clicking on IDLE (Python 3.10 64-bit). You can start coding in Python using the **Integrated Development and Learning Environment (IDLE)** as shown in Figure 1.4.

```

Python 3.9.2 (tags/v3.9.2:1a79785, Feb 19 2021, 13:44:55)
[MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for mo
re information.
>>> |

```

Fig. 1.4 Python IDE

IDLE is Python’s Integrated Development and Learning Environment. It is coded in 100% pure Python, using the tkinter GUI toolkit. It is cross-platform i.e. it works mostly same on Windows, Linux, and MacOS. Python shell window (interactive interpreter) with colorizing of code input, output, and error messages.

It is multi-window text editor with multiple undo feature, smart indent, call tips, auto completion, and other features. It can be used as debugger with persistent breakpoints, stepping, and viewing of data in process.

In Figure 1.4, the symbol >>> is the Python prompt, which indicates that the interpreter is ready to take instructions. You can type commands or statements on this prompt to execute using a Python interpreter.

IDLE allows changing the font name, style and size as shown in Figure 1.5. You can change these using **Options**→ **Configure IDLE**, from **Fonts/Tab**, choose Font Face, Size of your own choice to work with IDLE.

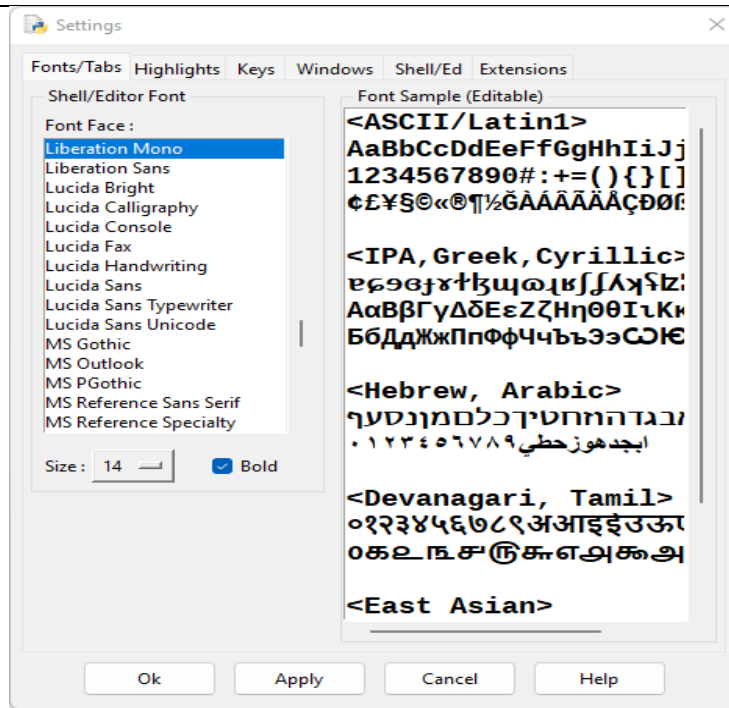


Fig. 1.5 Python IDLE configuration

1.3 Python Interpreter Mode

Python interpreter can be used in two modes – **Interactive mode** and **Script mode**. Interactive mode allows execution of individual statement instantaneously. Whereas, the script mode allows writing more than one instruction in a file called Python source code file that can be executed.

1.3.1 Interactive Mode

To work in the interactive mode, simply type Python statement on the `>>>` prompt directly and press **Enter** key. The interpreter executes the statement and displays the result(s), as shown in Figure 1.5. Working in the interactive mode is convenient for testing a single line code for instant execution. But in the interactive mode, we cannot save the programming code for future use and we have to retype the statements to run them again.

1.3.2 Script Mode

In the script mode, you have to write a Python program in the editor and save the file with **.py** extension. Then use the interpreter to execute it by entering the file name on the python prompt (`>>>`). By default, the Python program is saved in the Python installation folder. Let us see steps to run Python interpreter in script mode

Step 1. Create a new file by clicking on **File > New File** menu in IDLE as shown in Figure 1.6.

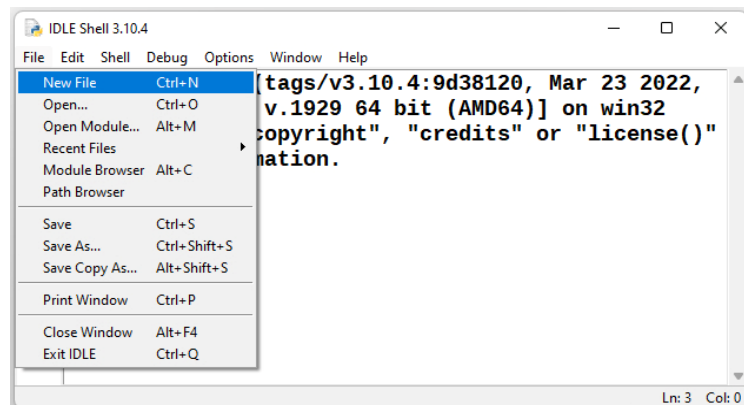
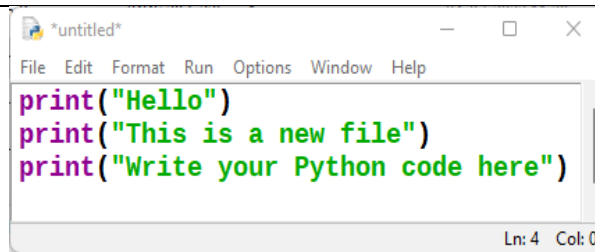


Fig. 1.6 Creating a new file in Python

Step 2. A new file will open in editor. Enter the Python code as shown in Figure 1.7.



```

print("Hello")
print("This is a new file")
print("Write your Python code here")

```

Fig. 1.7 Coding a Python program

Step 3. To save the file, click on **File > Save** menu option. The **Save as** window will open as shown in Figure 1.8. Select the folder in which you want to save the file. Type the file name and click on Save button as shown in Figure 1.8.

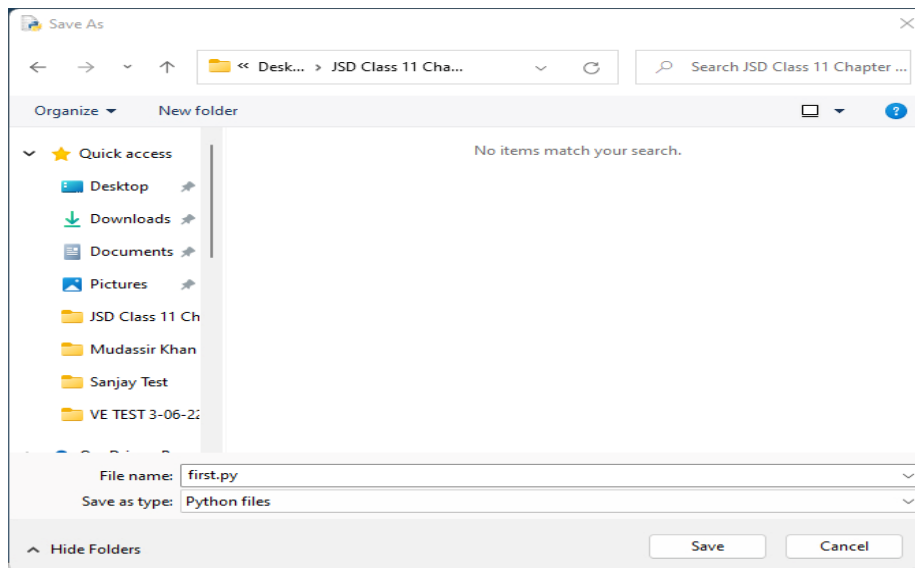


Fig. 1.8 Saving a Python program

Step 4. After saving the file, click **Run Module** from the Run menu as shown in Figure 1.9. You can also press **F5 key** to run the program.

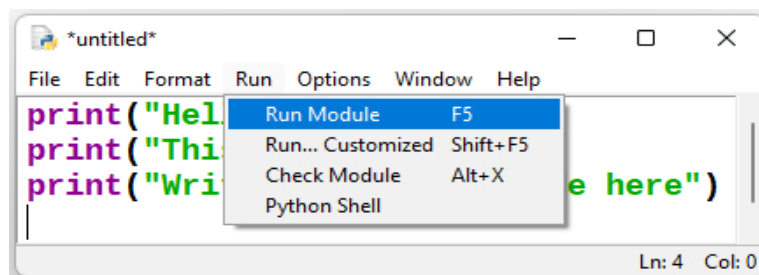


Fig. 1.9 Running a Python program

Step 5. The output appears on shell as shown in Figure 1.10.



```

Python 3.10.4 (tags/v3.10.4:9d38120, Mar 23 2022,
23:13:41) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()"
>>>
= RESTART: C:/Users/Psscive/Desktop/JSD Class 11
Chapter 10/first.py
Hello
This is new file
Write your python code here
>>>

```

Fig. 1.10 Showing output of Python program

Now you are familiar with both mode of Python Interpreter, let us see the general structure of Python Program.

1.4 Structure of a Python program

In general, the interpreter reads and executes the Python statements line by line i.e. sequentially, however there are some statements that can alter this behavior like conditional statements.

1.5. Keywords

Keywords are reserved words. Each keyword has a specific meaning to the Python interpreter. The keyword can be used in program for the purpose it has been defined. As Python is case sensitive, keywords must be written exactly as given in Table 1.1.

Table 1.1 Python keywords

false	class	finally	is	return
none	continue	for	lambda	try
true	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

1.6. Identifiers

In programming languages, identifiers are names used to identify a variable, function, or other entities in a program. The rules for naming an identifier in Python are as follows:

1. The name should begin with uppercase or lowercase alphabet or an underscore sign (_). This may be followed by any combination of characters a–z, A–Z, 0–9 or underscore (_). Thus, an identifier cannot start with a digit.
2. It can be of any length. However, it is preferred to keep it short and meaningful.
3. It should not be a keyword or reserved word.
4. We cannot use special symbols like !, @, #, \$, %, in name of an identifier.

For example, to find the average of marks obtained by a student in three subjects, choose the identifiers as marks1, marks2, marks3 and avg rather than a, b, c, or A, B, C.

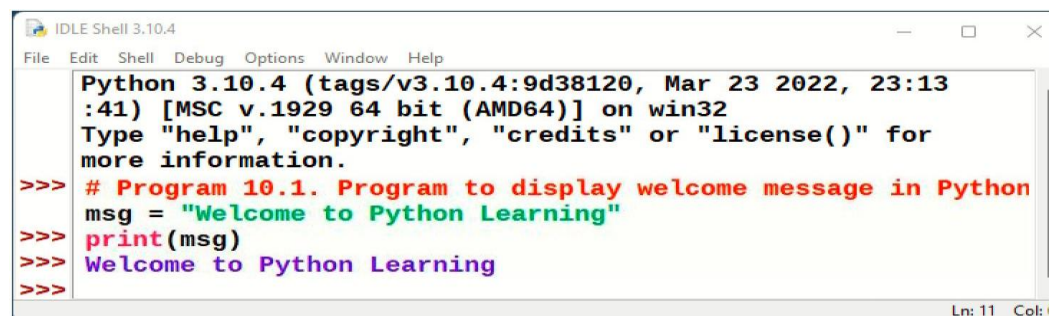
$avg = (marks1 + marks2 + marks3)/3$

Similarly, to calculate the area of a rectangle, use identifier names, such as area, length, breadth instead of single alphabets as identifiers for clarity and more readability.

$area = length * breadth$

Program 1.1. Write a program to display welcome message in Python.

The program coded in Python IDLE and its output is shown below.



```

Python 3.10.4 (tags/v3.10.4:9d38120, Mar 23 2022, 23:13:41) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> # Program 10.1. Program to display welcome message in Python
>>> msg = "Welcome to Python Learning"
>>> print(msg)
>>> Welcome to Python Learning
>>>

```

1.7. Variables

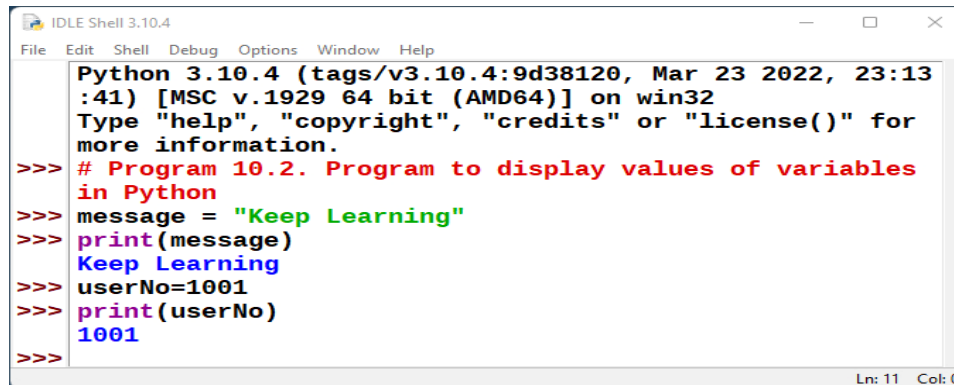
A variable in a program is uniquely identified by a name (identifier). Variable in Python refers to an object — an item or element that is stored in the memory. Value of a variable can be a string

(e.g., 'b', 'Global Citizen'), numeric (e.g., 345) or any combination of alphanumeric characters (CD67). In Python an assignment statement is used to create new variables and assign specific values to them.

```
gender = 'M'
message = "Keep Smiling"
price = 987.9
```

Program 1.2. Write a program to display values of variables in Python.

The program coded in Python IDLE and its output is shown below.



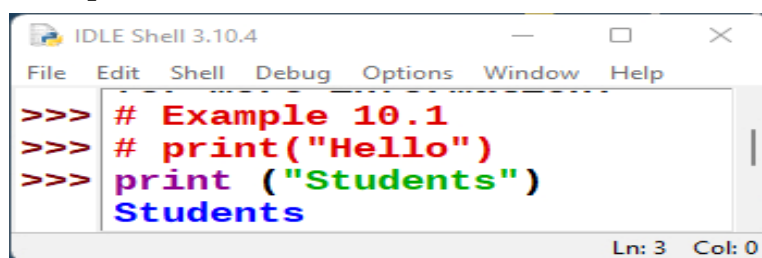
```
Python 3.10.4 (tags/v3.10.4:9d38120, Mar 23 2022, 23:13:41) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> # Program 10.2. Program to display values of variables in Python
>>> message = "Keep Learning"
>>> print(message)
Keep Learning
>>> userNo=1001
>>> print(userNo)
1001
>>>
```

1.8. Comments

Comments are used to add remarks or note in the source code. Comments are not executed by interpreter. They are added to understand the source code for others. They are used primarily to document the meaning and purpose of source code and its input and output requirements, so as to remember later how it functions and how to use it.

For large and complex software, several programmers are working in teams and sometimes, a programmer has to work on the program written by other programmer. In such situations, documentations in the form of comments proves to be useful to understand the logic of program. In Python, a comment starts with # (hash sign). Everything following the # till the end of that line is treated as a comment and the interpreter simply ignores it while executing the statement. The example 1.1 shows the first print statement is commented. So it will not print the word "Hello".

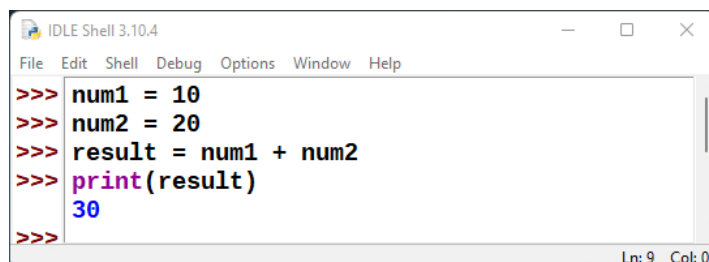
Example 1.1



```
>>> # Example 10.1
>>> # print("Hello")
>>> print ("Students")
Students
Ln: 3 Col: 0
```

Program 1.3. Write a Python program to find the sum of two numbers.

The following program illustrates to find the sum of two numbers.



```
>>> num1 = 10
>>> num2 = 20
>>> result = num1 + num2
>>> print(result)
30
>>>
Ln: 9 Col: 0
```

1.9. Data Types

Every value belongs to a specific data type in Python. Data type identifies the type of data values a variable can hold and the operations that can be performed on that data. Figure 1.11 enlists the data types available in Python.

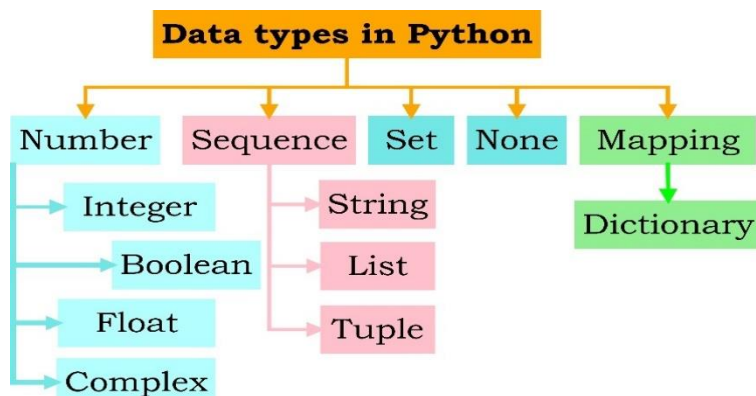


Fig. 1.11: Different data types in Python

1.9.1 Number

Number data type stores numerical values only. It is further classified into three different types: int, float and complex. Boolean data type (bool) is a subtype of integer. It is a unique data type, consisting of two constants, True and False. Boolean True value is non-zero, non-null and non-empty. Boolean False is the value zero.

Table 1.2 Numeric data types

Type/ Class	Description	Examples
int	integer numbers	-12, -3, 0, 125, 2
float	real or floating-point numbers	-2.04, 4.0, 14.23
complex	complex numbers	3 + 4i, 2 - 2i
bool	Boolean values	

Let us execute few statements in interactive mode of Python IDLE to determine the data type of the variable using built-in function type ().

Example 1.2

```

IDLE Shell 3.10.4
File Edit Shell Debug Options Window Help
>>> var1 = 10
>>> type(var1)
<class 'int'>
>>> var2 = 10.5
>>> type(var2)
<class 'float'>
>>> var3 = True
>>> type(var3)
<class 'bool'>
>>> var4 = 2+3j
>>> type(var4)
<class 'complex'>
Ln: 31 Col: 0
  
```

Variables of simple data types like integer, float, boolean that hold single value. But such variables are not useful to hold a long list of information, such as months in a year, student names in a class, names and numbers in a phone book or the list of artifacts in a museum. For this, Python provides other data types like *tuples*, *lists*, *dictionaries* and *sets*.

1.9.2 Sequence

A Python sequence is an ordered collection of items, where each item is indexed by an integer. The three types of sequence data types available in Python are *Strings*, *Lists* and *Tuples*. We will learn about

each of them in detail in later chapters. A brief introduction to these data types is as follows:

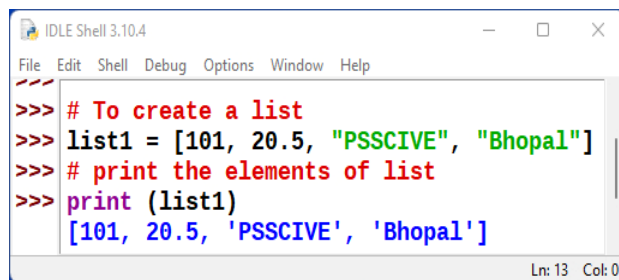
String – String is a group of characters. These characters may be alphabets, digits or special characters including spaces. String values are enclosed either in single quotation marks (e.g., 'Hello') or in double quotation marks (e.g., "Hello"). The quotes are not a part of the string; they are used to mark the beginning and end of the string for the interpreter. For example,

```
>>> str1 = 'Hello Friend'
>>> str2 = "452"
```

It is not possible to perform numerical operations on strings, even when the string contains a numeric value, as in str2.

List – List is a sequence of items separated by commas and the items are enclosed in square brackets [].

Example 1.3



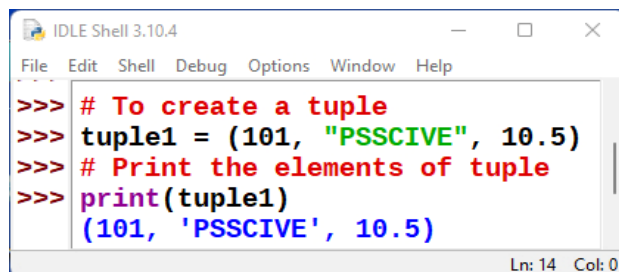
```

IDLE Shell 3.10.4
File Edit Shell Debug Options Window Help
>>> # To create a list
>>> list1 = [101, 20.5, "PSSCIVE", "Bhopal"]
>>> # print the elements of list
>>> print(list1)
[101, 20.5, 'PSSCIVE', 'Bhopal']
Ln: 13 Col: 0

```

Tuple – Tuple is a sequence of items separated by commas and items are enclosed in parenthesis (). This is unlike list, where values are enclosed in brackets [].

Example 1.4



```

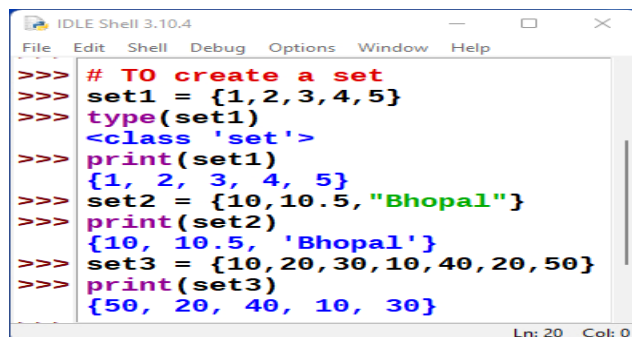
IDLE Shell 3.10.4
File Edit Shell Debug Options Window Help
>>> # To create a tuple
>>> tuple1 = (101, "PSSCIVE", 10.5)
>>> # Print the elements of tuple
>>> print(tuple1)
(101, 'PSSCIVE', 10.5)
Ln: 14 Col: 0

```

1.9.3 Set

Set is an unordered collection of items separated by commas and the items are enclosed in curly brackets {}. A set is similar to list, except that it cannot have duplicate entries. Once created, elements of a set cannot be changed.

Example 1.5



```

IDLE Shell 3.10.4
File Edit Shell Debug Options Window Help
>>> # TO create a set
>>> set1 = {1,2,3,4,5}
>>> type(set1)
<class 'set'>
>>> print(set1)
{1, 2, 3, 4, 5}
>>> set2 = {10,10.5, "Bhopal"}
>>> print(set2)
{10, 10.5, 'Bhopal'}
>>> set3 = {10, 20, 30, 10, 40, 20, 50}
>>> print(set3)
{50, 20, 40, 10, 30}
Ln: 20 Col: 0

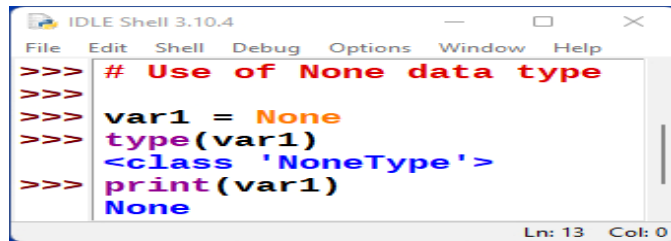
```

In the above example, set1 is a collection of 5 integers. The set2 is collection of different data types of elements. A set does not allow duplicate values, that you may observe for set3. Here, all the duplicate values have been removed from set3.

1.9.4 None

None is a special data type with a single value. It is used to signify the absence of value in a situation. None supports no special operations, and it is neither False nor 0 (zero).

Example 1.6



```

IDLE Shell 3.10.4
File Edit Shell Debug Options Window Help
>>> # Use of None data type
>>>
>>> var1 = None
>>> type(var1)
<class 'NoneType'>
>>> print(var1)
None
Ln: 13 Col: 0

```

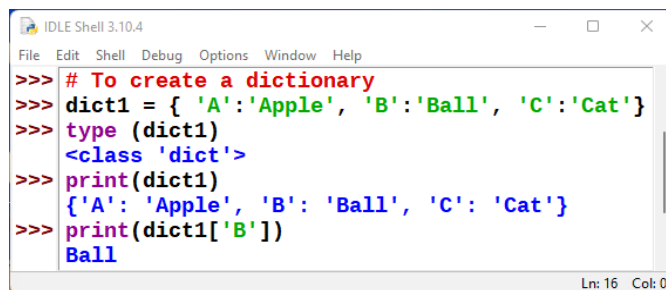
1.9.5 Mapping

Mapping is an unordered data type in Python. Currently, there is only one standard mapping data type in Python called dictionary.

Dictionary

Dictionary in Python holds data items in key-value pairs. Items in a dictionary are enclosed in curly brackets {}. Dictionaries permit faster access to data. Every key is separated from its value using a colon (:) sign. The key: value pairs of a dictionary can be accessed using the key. The keys are usually strings and their values can be any data type. In order to access any value in the dictionary, we have to specify its key in square brackets [] along with the name of the dictionary.

Example 1.7



```

IDLE Shell 3.10.4
File Edit Shell Debug Options Window Help
>>> # To create a dictionary
>>> dict1 = { 'A': 'Apple', 'B': 'Ball', 'C': 'Cat' }
>>> type(dict1)
<class 'dict'>
>>> print(dict1)
{'A': 'Apple', 'B': 'Ball', 'C': 'Cat'}
>>> print(dict1['B'])
Ball
Ln: 16 Col: 0

```

1.10 Mutable and Immutable Data Types

Sometimes we may require to change or update the values of certain variables used in a program. However, for certain data types, Python does not allow to change the values in the same memory location where they are created once a variable of that type has been created and values are assigned.

Variables whose values can be changed after they are created and assigned, at the same memory location where they were created, are called mutable. Variables whose values cannot be changed after they are created and assigned are called immutable. When an attempt is made to update the value of an immutable variable, the old variable is destroyed and a new variable is created by the same name in memory. Python data types can be classified into mutable and immutable as shown in Figure 1.12.

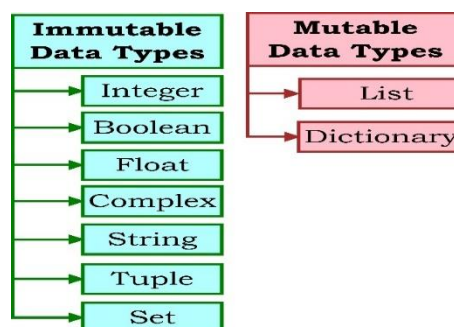


Fig. 1.12: Classification of data types

Let us now see what happens when an attempt is made to update the value of a variable. `>>> num1 = 300` This statement will create an object with value 300 and the object is referenced by the identifier `num1` as shown in Figure 1.13.

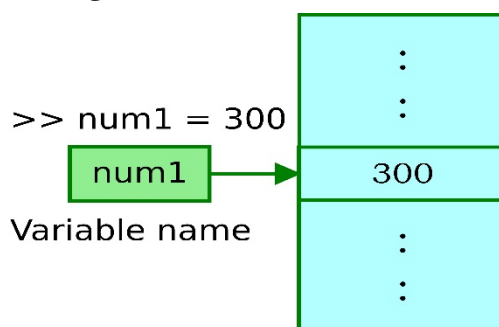


Fig. 1.13: Object and its identifier

The statement `num2 = num1` will make `num2` refer to the value 300, also being referred by `num1`, and stored at memory location number, say 1000. So, `num1` shares the referenced location with `num2` as shown in Figure 1.14.

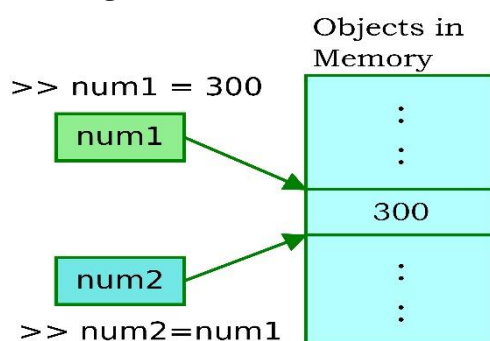


Fig. 1.14: Variables with same value have same identifier

In this manner Python makes the assignment effective by copying only the reference, and not the data:

```
>>> num1 = num2 + 100
```

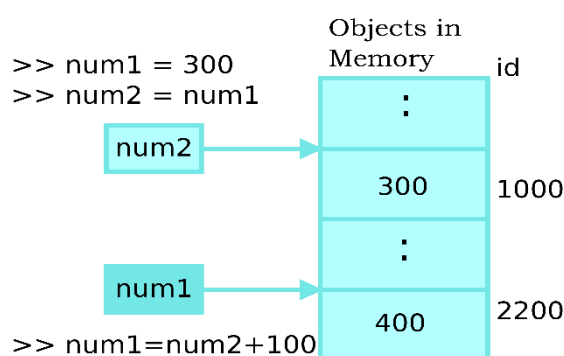


Fig. 1.15: Variables with different values have different identifiers

This statement `num1 = num2 + 100` links the variable `num1` to a new object stored at memory location number say 2200 having a value 400. As `num1` is an integer, which is an immutable type, it is rebuilt, as shown in Figure 1.15.

1.11 Usage of Python Data Types

Each data type has some specific properties. Appropriate use of data type is depending on the situation. It is preferred to use *lists* for a simple iterative collection of data that may go for frequent modifications. For example, if we store the names of students of a class in a list, then it is easy to update the list when some new students join or some leave the course. *Tuples* are used when we do not need any change in the data. For example, names of months in a year.

When we need uniqueness of elements and to avoid duplicity it is preferable to use *sets*, for example, list of artifacts in a museum. If our data is being constantly modified or we need a fast lookup based on a custom key or we need a logical association between the key: value pair, it is advised to use *dictionaries*. A mobile phone book is a good application of dictionary.

1.12 OPERATORS

An operator is used to perform specific mathematical or logical operation on values are called operands. For example, in the expression “ $10 + num$ ”, the value 10, and the variable *num* are operands and the + (plus) sign is an operator. Python supports several types of operators as discussed below.

1.12.1 Arithmetic Operators

Python supports arithmetic operators (+, -, *, /) that are used to perform the four basic arithmetic operations as well as modulus division (%), floor division (//) and exponentiation (**).

Addition operator (+) : Adds the two numeric values on either side of the operator. This operator can also be used to concatenate two strings on either side of the operator.

Subtraction operator (-) : Subtracts the operand on the right from the operand on the left.

Multiplication operator (*) : Multiplies the two numeric values on both side of the operator. Repeats the item on left of the operator if first operand is a string and second operand is an integer value.

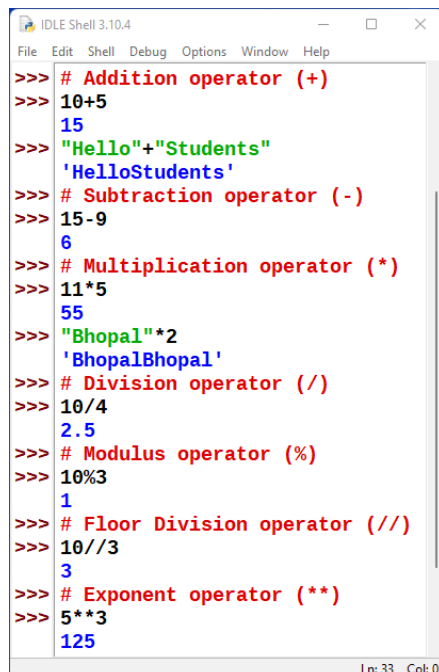
Division operator (/) : Divides the operand on the left by the operand on the right and returns a float value as the quotient.

Floor Division Operator (//) : Divides the operand on the left by the operand on the right and returns an integer value as the quotient

Modulus operator (%) : Divides the operand on the left by the operand on the right and returns the remainder.

Exponent operator ()** : Performs exponential (power) calculation on operands. That is, raise the operand on the left to the power of the operand on the right.

The following code as shown in Figure 1.16 demonstrates the use of *Arithmetic Operators* in Python.



```

IDLE Shell 3.10.4
File Edit Shell Debug Options Window Help
>>> # Addition operator (+)
>>> 10+5
15
>>> "Hello"+"Students"
'HelloStudents'
>>> # Subtraction operator (-)
>>> 15-9
6
>>> # Multiplication operator (*)
>>> 11*5
55
>>> "Bhopal"*2
'BhopalBhopal'
>>> # Division operator (/)
>>> 10/4
2.5
>>> # Modulus operator (%)
>>> 10%3
1
>>> # Floor Division operator (//)
>>> 10//3
3
>>> # Exponent operator (**)
>>> 5**3
125
Ln: 33 Col: 0

```

Fig. 1.16 Demonstration of Arithmetic Operators

1.12.2 Relational Operators

Relational operator compares the values of the operands on its either side and determines the relationship among them.

Equals to (==) : If the values of two operands are equal, then the condition is True, otherwise it is False.

Not equal to (!=) : If values of two operands are not equal, then condition is True, otherwise it is False.

Greater than (>) : If the value of the left-side operand is greater than the value of the right-side operand, then condition is True, otherwise it is False.

Less than (<) : If the value of the left-side operand is less than the value of the right-side operand, then condition is True, otherwise it is False.

Greater than or equal to (>=) : If the value of the left-side operand is greater than or equal to the value of the right-side.

Less than or equal to (<=) : If the value of the left operand is less than or equal to the value of the right operand, then is True otherwise it is False.

Assume the Python variables num1 = 10, num2 = 20, num3 = 10, str1 = "Hello", str2 = "Students" and str3 = "Hello".

The following code as shown in Figure 1.17 demonstrates the use of **Relational Operators** in Python.

```

IDLE Shell 3.10.4
File Edit Shell Debug Options Window Help
>>> # Relational operators
>>> num1 = 10
>>> num2 = 20
>>> num3 = 10
>>> str1 = "Hello"
>>> str2 = "Students"
>>> str3 = "Hello"
>>> # Relational operator Equals to (==)
>>> num1 == num2
False
>>> num1 == num3
True
>>> str1 == str3
True
>>> # Relational operator Not equal to (!=)
>>> num1 != num2
True
>>> num1 != num3
False
>>> # Relational operator Greater than (>)
>>> num1 > num2
False
>>> str2 > str1
True
>>> # Relational operator Less than (<)
>>> num1 < num2
True
>>> str1 < str2
True
>>> # Relational operator Greater than or equal to (>=)
>>> num1 >= num2
False
>>> str1 >= str2
False
>>> # Relational operator Less than or equal to (<=)
>>> num2 <= num3
False
>>> str2 <= str3
False
Ln: 44 Col: 0

```

Fig. 1.17 Demonstration of Relational Operators

1.12.3 Assignment Operators

Assignment operator assigns or changes the value of the variable on its left. Table 1.3 shows various assignment operators available in python

Table 1.3 Assignment operators.

=	Assigns value from right-side operand to left-side operand.
+=	It adds the value of right-side operand to the left-side operand and assigns the result to the left-side operand. In other words, x += y is same as x = x + y.

-=	It subtracts the value of right-side operand from the left-side operand and assigns the result to left-side operand. In other words, $x-=y$ is same as $x=x-y$.
=	It multiplies the value of right-side operand with the value of left-side operand and assigns the result to left-side operand. In other words, $x=y$ is same as $x=x*y$.
/=	It performs modulus operation using two operands and assigns the results to left-side operand. In other words, $x%=y$ is same as $x = x \% y$.
%=	It performs exponential (power) calculation on operators and assigns value to the left-side operand. In other words, $x**=y$ is same as $x = x ** y$.
**=	

The code as shown in Figure 1.18 demonstrates the use of **Assignment Operators** in Python.

```

IDLE Shell 3.10.4
File Edit Shell Debug Options
Window Help
>>> # Assignment operator =
>>> num1 = 10
>>> num2 = num1
>>> num2
10
>>> city = "My City"
>>> city
'My City'
>>> # Assignment operator +=
>>> num1 = 10
>>> num2 = 5
>>> num1 += num2
>>> num1
15
>>> str1 = "My"
>>> str2 = "City"
>>> str1 += str2
>>> str1
'MyCity'
>>> # Assignment operator -=
>>> num1 = 10
>>> num2 = 4
>>> num1 -= num2
>>> num1
6
>>> # Assignment operator *=
>>> num1 = 10
>>> num2 = 4
>>> num1 *= num2
>>> num1
40
>>> # Assignment operator /=
>>> num1 = 11
>>> num2 = 4
>>> num1 /= num2
>>> num1
2.75
>>> # Assignment operator %=
>>> num1 = 11
>>> num2 = 4
>>> num1 %= num2
>>> num1
3
>>> # Assignment operator //=
>>> num1 = 11
>>> num2 = 4
>>> num1 //= num2
>>> num1
2
>>> # Assignment operator **=
>>> num1 = 3
>>> num2 = 4
>>> num1 **= num2
>>> num1
81
Ln: 58 Col: 0

```

Fig. 1.18 Demonstration of Assignment operators

1.12.4 Logical Operators

There are three logical operators – “**and, or, not**” supported by Python. These are to be written in lower case only. The logical operator evaluates to either **True** or **False** based on the logical operands on either side. Every value is logically either True or False. By default, all values are

True except None, False, 0 (zero), empty collections "", (), [], {}, and few other special values. So, if we say num1 = 10, num2 = -5, then both num1 and num2 are logically True. If we have num3 = 0, then num3 is logically False.

Logical AND (and) - If both the conditions are True, then expression becomes True.

Logical OR (or) - If any of the two condition is True, then expression becomes True.

Logical NOT (not) - It is used to reverse the logical state of its operand.

The code as shown in Figure 1.19 demonstrates the use of **Logical Operators** in Python.

```

Python 3.10.4 (main, Apr 22 2022, 09:04:19) [GCC 11.2.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>> # Logical operators in Python
>>> True and True
True
>>> True and False
False
>>> True or True
True
>>> True or False
True
>>> False or False
False
>>> num1 = 10
>>> num2 = -5
>>> num3 = 0
>>> bool(num1 and num2)
True
>>> bool(num1 or num2)
True
>>> bool(num1 or num3)
True
>>> bool(num1)
True
>>> bool(not num1)
False
>>>
Ln: 27 Col: 0
  
```

Fig. 1.19 Demonstration of Logical operators

1.12.5 Identity Operators

Identity operators are used to determine whether the value of a variable is of a certain type or not. Identity operators can also be used to determine whether two variables are referring to the same object or not. There are two identity operators explained in table 1.4.

Table 1.4 Identity operators in Python

Operator	Description	Example
is	Evaluates True if the variables on either side of the operator point towards the same memory location and False otherwise. var1 is var2 results to True if id(var1) is equal to id(var2).	<pre> >>> num1 = 10 >>> num2 = 5 >>> num3 = num1 >>> num2 is num1 False >>> num3 is num1 True </pre>

is not	Evaluates to False if the variables on either side of the operator point to the same memory location and True otherwise. var1 is not var2 results to True if id(var1) is not equal to id(var2).	<pre>>>> num1 = 10 >>> num2 = 5 >>> num3 = num1 >>> num2 is not num1 True >>> num3 is not num1 False</pre>
--------	---	---

1.12.6 Membership Operators

Membership operators are used to check if a value is a member of the given sequence or not. There are two membership operators explained in table 1.5.

Table 1.5 Membership operators in Python

Operator	Description	Example (Try in Lab)
in	Returns True if the variable/value is found in the specified sequence and False otherwise.	<pre>>>> a = [10,20,30,40,50] >>> 10 in a True >>> 25 in a False</pre>
not in	Returns True if the variable/value is not found in the specified sequence and False otherwise.	<pre>>>> a = [10,20,30,40,50] >>> 10 not in a False >>> 25 not in a True</pre>

1.12.7 Bitwise Operators

Bitwise operators are the only operators which works on equivalent binary value of the integer operands. First of all, integer operands are converted into binary then respected operator works on bit by bit, hence the name is bitwise operators. The result is also converted into decimal format.

The code snippet demonstrates in table 1.6 shows the use of various bitwise operators in Python.

Table 1.6 Bitwise operators in Python

Operator	Description	Example (Try in Lab)
Bitwise & (AND)	Returns 1 if both the bits are 1 else 0. Here x=10 and y=12. x = 1 0 1 0 (10 in decimal) y = 1 1 0 0 (12 in decimal) z = 1 0 0 0 (8 in decimal)	<pre>>>> x = 10 >>> y = 12 >>> z = x & y >>> z 8</pre>
Bitwise (OR)	Returns 1 if either of the bit is 1 else 0. x = 1 0 1 0 (10 in decimal) y = 1 1 0 0 (12 in decimal) z = 1 1 1 0 (14 in decimal)	<pre>>>> x = 10 >>> y = 12 >>> z = x y >>> z 14</pre>
Bitwise ~ (NOT)	Returns one's complement of the number. a = 1010 (Binary) z = ~a = ~ 1010 = - (1010 + 1) = - 11	<pre>>>> x = 10 >>> z = ~x; >>> z -11</pre>

Bitwise ^ (XOR)	Returns 1 if one of the bits is 1 and the other is 0 else returns false. x = 1 0 1 0 (10 in decimal) y = 1 1 0 0 (12 in decimal) z = 0 1 1 0 (6 in decimal)	<pre>>>> x = 10 >>> y = 12 >>> z = x ^ y >>> z 6</pre>
Bitwise >>(RIGHT)	Shifts the bits of the number to the right and fills 0 on voids left (fills 1 in the case of a negative number) as a result. Similar effect as of dividing the number with some power of two.	<pre>>>> x = 10 >>> z = x >> 2 >>> z 2</pre>
Bitwise <<(LEFT)	Shifts the bits of the number to the left and fills 0 on voids right as a result. Similar effect as of multiplying the number with some power of two.	<pre>>>> x = 10 >>> z = x << 2 >>> z 40</pre>

1.13 Expressions

An expression is defined as a combination of constants, variables, and operators. An expression always evaluates to a value. A value or a standalone variable is also considered as an expression but a standalone operator is not an expression. Some examples of valid expressions are given below.

- 100
- num
- num - 20.4
- 3.0 + 3.14
- 23/3 - 5 * 7(14 - 2)
- "Global" + "Citizen"

1.13.1 Precedence of Operators

Evaluation of the expression is based on precedence of operators. When an expression contains different types of operators, precedence determines which operator should be applied first. Higher precedence operator is evaluated before the lower precedence operator. Most of the operators studied till now are binary operators with two operands. The unary operators need only one operand, and they have a higher precedence than the binary operators. The minus (-) as well as + (plus) operators can act as both unary and binary operators, but “not” is a unary logical operator.

#Depth is using - (minus) as unary operator

Value = -Depth

#not is a unary operator, negates True print (not (True))

The table 1.7 lists precedence of all operators from highest to lowest.

Table 1.7 Precedence of all operators in Python

Order of Precedence	Operators	Description
1	**	Exponentiation (raised to the power)
2	~, +, -	Complement, unary plus and unary minus
3	*, /, %, //	Multiply, divide, modulo and floor division
4	+, -	Addition and subtraction
5	<=, <, >, >=	Relational operators
6	==, !=	Equality operators

7	=, %=, /=, //, -=, +=, *=, **=	Assignment operators
8	is, is not	Identity operators
9	in, not in	Membership operators
10	not, or, and	Logical operators

Parenthesis can be used to override the precedence of operators. The expression within () is evaluated first. For operators with equal precedence, the expression is evaluated from left to right.

Example 1.8.

How will Python evaluate the following expression? $20 + 30 * 40$

Solution:

= 20 + (30 * 40) #Step 1

#precedence of * is more than that of +

= 20 + 1200 #Step 2

= 1220 #Step 3

Example 1.9.

How will Python evaluate the following expression? $20 - 30 + 40$

Solution:

The two operators (-) and (+) have equal precedence. Thus, the first operator, i.e., subtraction is applied before the second operator, i.e., addition (left to right).

= (20 - 30) + 40 #Step 1

= - 10 + 40 #Step 2

= 30 #Step 3

Example 1.1.

How will Python evaluate the following expression? $(20 + 30) * 40$

Solution:

= (20 + 30) * 40 # Step 1

#using parenthesis (), we have forced precedence of + to be more than that of *

= 50 * 40 # Step 2

= 2000 # Step 3

Example 1.11.

How will the following expression be evaluated in Python? $15.0 / 4 + (8 + 3.0)$

Solution:

15.0 / 4 + (8.0 + 3.0) #Step 1

15.0 / 4.0 + 11.0 #Step 2

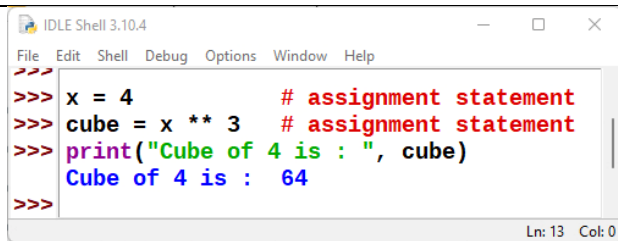
3.75 + 11.0 #Step 3

14.75 #Step 4

1.14 STATEMENT

In Python, a statement is a unit of code that the Python interpreter can execute. Example 1.12 illustrate the statements in Python.

Example 1.12



```

IDLE Shell 3.10.4
File Edit Shell Debug Options Window Help
>>> x = 4          # assignment statement
>>> cube = x ** 3  # assignment statement
>>> print("Cube of 4 is : ", cube)
Cube of 4 is : 64
>>>
Ln: 13 Col: 0

```

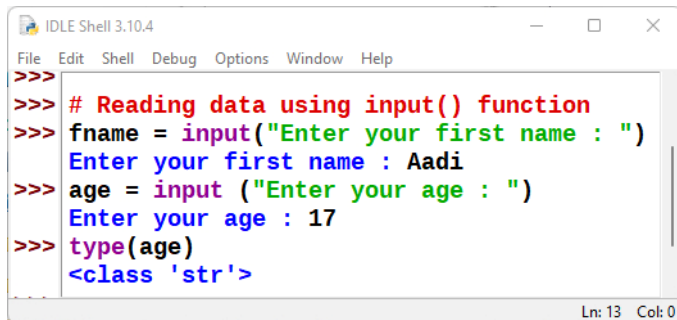
1.15 INPUT AND OUTPUT STATEMENTS

Sometimes, a program needs to interact with the user to receive data which is processed to give the desired output. In Python, the `input ()` function prompts the user to enter data. It accepts all user input as string. The user may enter a number or a string but the `input ()` function treats them as strings only. The syntax for input statement is:

`input ([Prompt])`

Prompt is the string we may like to display on the screen prior to taking input, and it is optional. When a prompt is specified, first it is displayed on the screen after which the user can enter data. The `input()` takes exactly what is typed from the keyboard, converts it into a string and assigns it to the variable on left-hand side of the assignment operator (`=`). Entering data for the `input` function is terminated by pressing the enter key.

Example 1.13



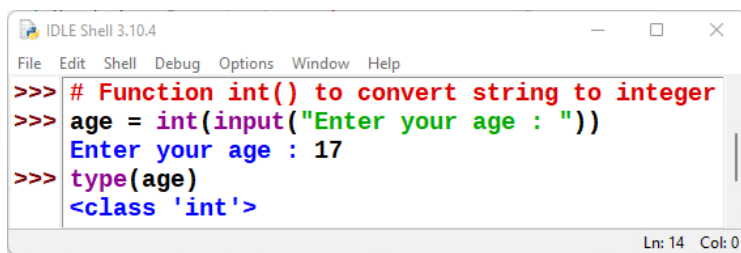
```

IDLE Shell 3.10.4
File Edit Shell Debug Options Window Help
>>> # Reading data using input() function
>>> fname = input("Enter your first name : ")
Enter your first name : Aadi
>>> age = input("Enter your age : ")
Enter your age : 17
>>> type(age)
<class 'str'>
Ln: 13 Col: 0

```

In Example 1.13, the variable `fname` will get the string 'Aadi', entered by the user. Similarly, the variable `age` will get the string '17'. We can typecast or change the datatype of the string data accepted from user to an appropriate numeric value. The code in Example 1.14 convert the accepted string to an integer. If the user enters any non-numeric value, an error will be generated.

Example 1.14



```

IDLE Shell 3.10.4
File Edit Shell Debug Options Window Help
>>> # Function int() to convert string to integer
>>> age = int(input("Enter your age : "))
Enter your age : 17
>>> type(age)
<class 'int'>
Ln: 14 Col: 0

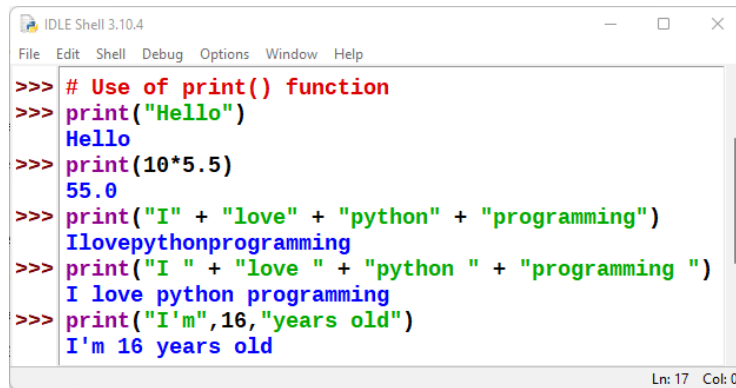
```

Python uses the `print ()` function to output data to standard output device — the screen. The function `print ()` evaluates the expression before displaying it on the screen. The `print ()` display the output one one line and then moves to the next line for subsequent output. The syntax for `print ()` is:

`print (value [, ..., sep = ' ', end = '\n'])`

`sep`: The optional parameter `sep` is a separator between the output values. We can use a character, integer or a string as a separator. The default separator is space.

`end`: This is also optional and it allows us to specify any string to be appended after the last value. The default is a new line.

Example 1.15


```

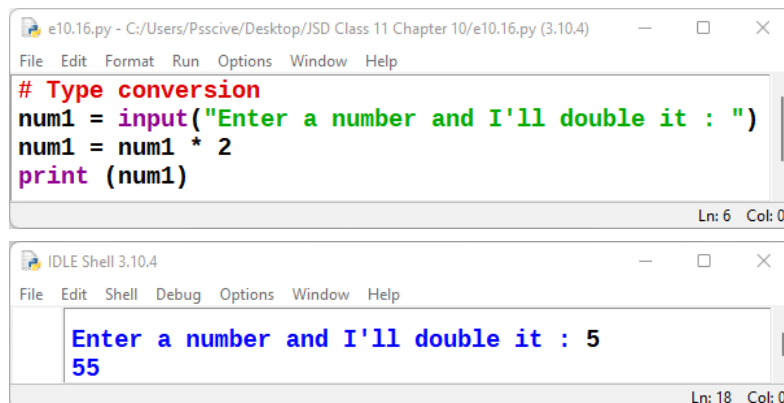
IDLE Shell 3.10.4
File Edit Shell Debug Options Window Help
>>> # Use of print() function
>>> print("Hello")
Hello
>>> print(10*5.5)
55.0
>>> print("I" + "love" + "python" + "programming")
Ilovepythonprogramming
>>> print("I " + "love " + "python " + "programming ")
I love python programming
>>> print("I'm",16, "years old")
I'm 16 years old
Ln: 17 Col: 0

```

In *Example 1.15*, the third print function is concatenating strings, and we use + (plus) between two strings to concatenate them. The fourth print function also appears to be concatenating strings but uses commas (,) between strings. Actually, here we are passing multiple arguments, separated by commas to the print function. As arguments can be of different types, hence the print function accepts integer (16) along with strings here. But in case the print statement has values of different types and '+' is used instead of comma, it will generate an error as discussed in the next section under explicit conversion.

1.16 TYPE CONVERSION

Type conversion is used to convert the data type of any variable that is assigned or read using input statement in Python. Let us take an example to understand it in better way. Consider the following example.

Example 1.16.


```

e10.16.py - C:/Users/Psscive/Desktop/JSD Class 11 Chapter 10/e10.16.py (3.10.4)
File Edit Format Run Options Window Help
# Type conversion
num1 = input("Enter a number and I'll double it : ")
num1 = num1 * 2
print (num1)
Ln: 6 Col: 0

IDLE Shell 3.10.4
File Edit Shell Debug Options Window Help
Enter a number and I'll double it : 5
55
Ln: 18 Col: 0

```

The program was expected to display double the value of the number received and store in variable num1. So, if a user enters 5 and expects the program to display 10 as the output, but the program displays 55 in output. This is because the value returned by the input function is a string ("5") by default. As a result, in statement **num1 = num1 * 2**, num1 has string value and * acts as repetition operator which results in output as "55".

To get 10 as output, we need to convert the data type of the value entered by the user to integer. Thus, we need to modify the program as follows.

Example 1.17

The screenshot shows two windows from the Python IDE. The top window, titled 'e10.17.py', contains the following code:

```
# Type conversion from string to integer
num1 = input("Enter a number and I'll double it : ")
num1 = int(num1) # Converting string to integer
num1 = num1*2
print(num1)
```

The bottom window, titled 'IDLE Shell 3.10.4', shows the execution output:

```
Enter a number and I'll double it : 5
10
>>>
```

Now, the program displays the expected output as 10 as data type of num1 has been changed to integer in statement `num1= int (num1)`. This is called type conversion of the variable.

Let us now understand what is type conversion and how it works. As and when required, we can change the data type of a variable in Python from one type to another. Such data type conversion can happen in two ways, either explicitly (forced) when the programmer specifies for the interpreter to convert a data type to another type; or implicitly, when the interpreter understands such a need by itself and does the type conversion automatically.

1.16.1 Explicit Conversion

Explicit conversion, also called as type casting happens when data type conversion takes place because the programmer forced it in the program. The general form of an explicit data type conversion is:

(new_data_type) (expression)

With explicit type conversion, there is a risk of loss of information since we are forcing an expression to be of a specific type. For example, converting a floating value of `x = 20.67` into an integer type, i.e., `int (x)` will discard the fractional part `.67`. Following are some of the functions in Python that are used for explicitly converting an expression or a variable to a different type.

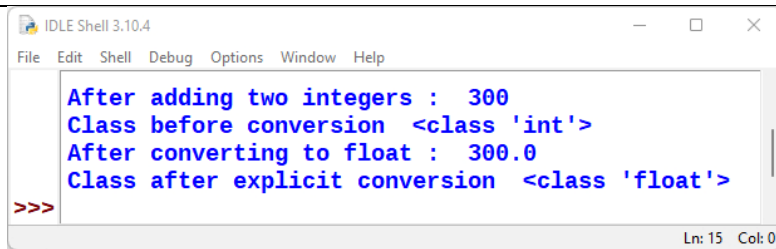
Table 1.8 Explicit type conversion functions in Python

Function	Description
<code>int (x)</code>	Converts x to an integer
<code>float (x)</code>	Converts x to a floating-point number
<code>str (x)</code>	Converts x to a string representation
<code>chr (x)</code>	Converts x to a character

Program 1.4. Program of explicit type conversion from int to float.

The screenshot shows a Python IDE window titled '*p10.4.py' with the following code:

```
# Program for explicit type conversion from int to float
num1 = 100
num2 = 200
num3 = num1 + num2
print("After adding two integers : ", num3)
print("Class before conversion ", type(num3))
num4= float(num1+ num2) # Explicit type conversion to float
print("After converting to float : ", num4)
print("Class after explicit conversion ", type(num4))
```

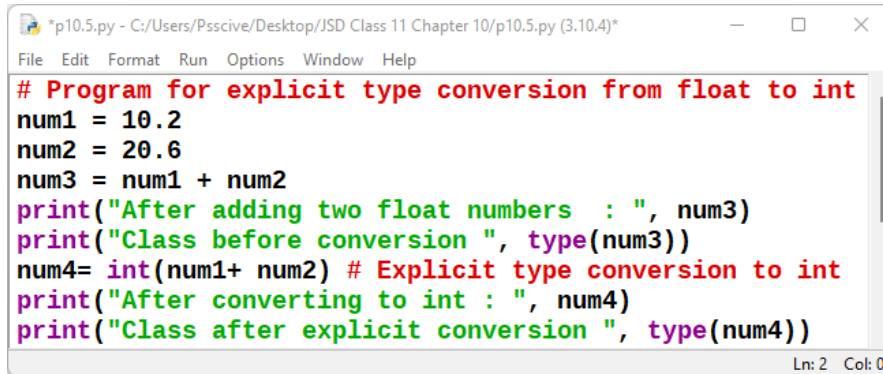


```

IDLE Shell 3.10.4
File Edit Shell Debug Options Window Help
After adding two integers : 300
Class before conversion <class 'int'>
After converting to float : 300.0
Class after explicit conversion <class 'float'>
>>>
Ln: 15 Col: 0

```

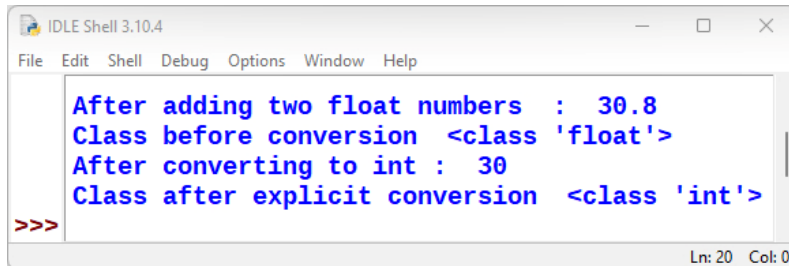
Program 1.5. Program of explicit type conversion from float to int.



```

*p10.5.py - C:/Users/Psscive/Desktop/JSD Class 11 Chapter 10/p10.5.py (3.10.4)*
File Edit Format Run Options Window Help
# Program for explicit type conversion from float to int
num1 = 10.2
num2 = 20.6
num3 = num1 + num2
print("After adding two float numbers : ", num3)
print("Class before conversion ", type(num3))
num4= int(num1+ num2) # Explicit type conversion to int
print("After converting to int : ", num4)
print("Class after explicit conversion ", type(num4))
Ln: 2 Col: 0

```

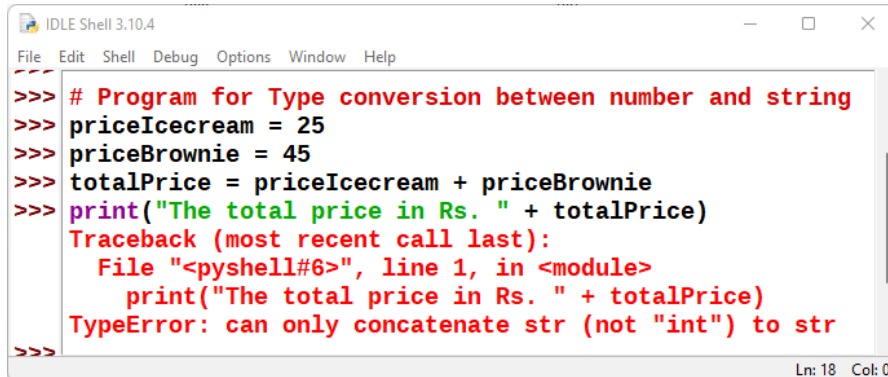


```

IDLE Shell 3.10.4
File Edit Shell Debug Options Window Help
After adding two float numbers : 30.8
Class before conversion <class 'float'>
After converting to int : 30
Class after explicit conversion <class 'int'>
>>>
Ln: 20 Col: 0

```

Program 1.6. Example of type conversion between numbers and strings.



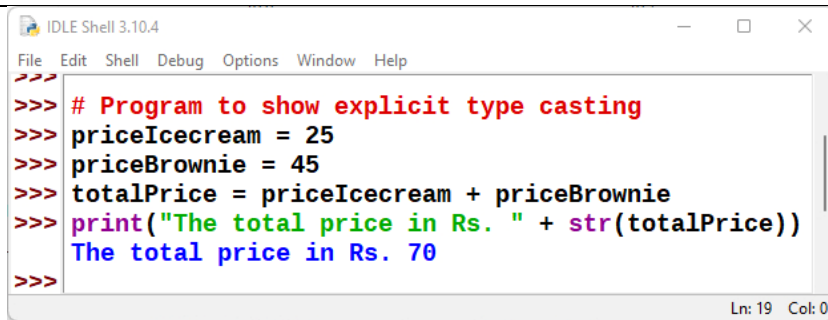
```

IDLE Shell 3.10.4
File Edit Shell Debug Options Window Help
>>> # Program for Type conversion between number and string
>>> priceIcecream = 25
>>> priceBrownie = 45
>>> totalPrice = priceIcecream + priceBrownie
>>> print("The total price in Rs. " + totalPrice)
Traceback (most recent call last):
  File "<pyshell#6>", line 1, in <module>
    print("The total price in Rs. " + totalPrice)
TypeError: can only concatenate str (not "int") to str
>>>
Ln: 18 Col: 0

```

On execution, Program 1.6 gives an error as shown in output of the program, informing that the interpreter cannot convert an integer value to string implicitly. It may appear quite intuitive that the program should convert the integer value to a string depending upon the usage. However, the interpreter may not decide on its own when to convert as there is a risk of loss of information. Python provides the mechanism of the explicit type conversion so that one can clearly state the desired outcome. Program 1.7 works perfectly using explicit type casting:

Program 1.7. Program to show explicit type casting.



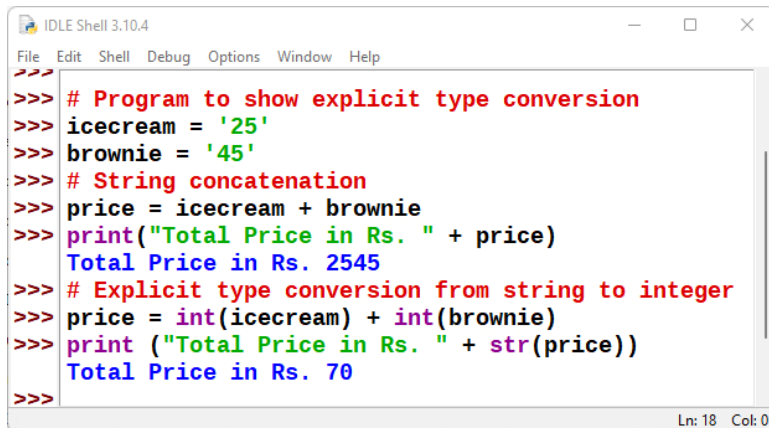
```

IDLE Shell 3.10.4
File Edit Shell Debug Options Window Help
>>> # Program to show explicit type casting
>>> priceIcecream = 25
>>> priceBrownie = 45
>>> totalPrice = priceIcecream + priceBrownie
>>> print("The total price in Rs. " + str(totalPrice))
The total price in Rs. 70
>>>
Ln: 19 Col: 0

```

Similarly, type casting is needed to convert float to string. In Python, one can convert string to integer or float values whenever required.

Program 1.8. Program to show explicit type conversion.



```

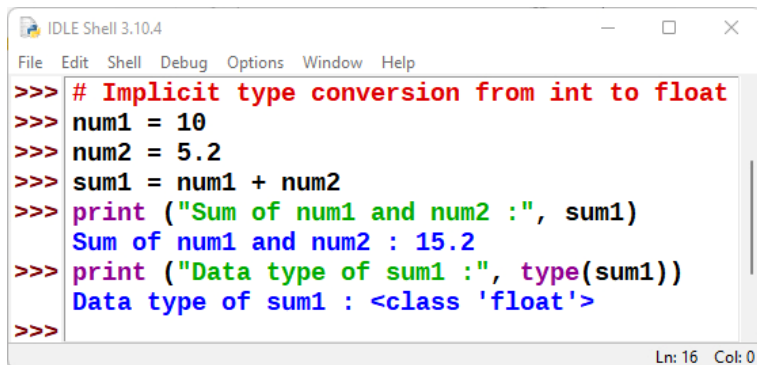
IDLE Shell 3.10.4
File Edit Shell Debug Options Window Help
>>> # Program to show explicit type conversion
>>> icecream = '25'
>>> brownie = '45'
>>> # String concatenation
>>> price = icecream + brownie
>>> print("Total Price in Rs. " + price)
Total Price in Rs. 2545
>>> # Explicit type conversion from string to integer
>>> price = int(icecream) + int(brownie)
>>> print("Total Price in Rs. " + str(price))
Total Price in Rs. 70
>>>
Ln: 18 Col: 0

```

1.16.2 Implicit Conversion

Implicit conversion, also known as coercion, happens when data type conversion is done automatically by Python and is not instructed by the programmer.

Program 1.9. Program to show implicit conversion from int to float.



```

IDLE Shell 3.10.4
File Edit Shell Debug Options Window Help
>>> # Implicit type conversion from int to float
>>> num1 = 10
>>> num2 = 5.2
>>> sum1 = num1 + num2
>>> print("Sum of num1 and num2 :", sum1)
Sum of num1 and num2 : 15.2
>>> print("Data type of sum1 :", type(sum1))
Data type of sum1 : <class 'float'>
>>>
Ln: 16 Col: 0

```

In the above program, an integer value stored in variable “num1” is added to a float value stored in variable “num2”, and the result was automatically converted to a float value stored in variable sum1 without explicitly telling the interpreter. This is an example of implicit data conversion. You may wonder that why the float value was not converted to an integer instead? This is due to type promotion that allows performing operations, whenever possible by converting data into a wider-sized data type without any loss of information.

1.17 DEBUGGING

A programmer can make mistakes while coding a program. These mistakes are called bugs or errors in programming language. The program contains bugs may not execute or generate wrong output. The process of identifying and removing such bugs, errors or mistakes is known as *debugging*. The Errors occurring in programs can be categorised as – *Syntax errors, Logical errors and Runtime errors*.

1.17.1 Syntax Errors

Like other programming languages, Python has its own rules that determine its syntax. The interpreter interprets the statements only if it is syntactically (as per the rules of Python) correct. If any syntax error is present, the interpreter shows error message(s) and stops the execution there. For example, parentheses must be in pairs, so the expression $(10 + 12)$ is syntactically correct, whereas $(7 + 11$ is not due to absence of right parenthesis. Such errors need to be removed before the execution of the program

1.17.2 Logical Errors

A logical error is a bug in the program that causes it to behave incorrectly. A logical error produces an undesired output but without abrupt termination of the execution of the program. Since the program interprets successfully even when logical errors are present in it, it is sometimes difficult to identify these errors. The only evidence to the existence of logical errors is the wrong output. While working backwards from the output of the program, one can identify what went wrong.

For example, if we wish to find the average of two numbers 10 and 12 and we write the code as $10 + 12/2$, it would run successfully and produce the result 16. Surely, 16 is not the average of 10 and 12. The correct code to find the average should have been $(10 + 12)/2$ to give the correct output as 11. Logical errors are also called semantic errors as they occur when the meaning of the program (its semantics) is not correct.

1.17.3 Runtime Error

A runtime error causes abnormal termination of program while it is executing. Runtime error is when the statement is correct syntactically, but the interpreter cannot execute it. Runtime errors do not appear until after the program starts running or executing.

For example, we have a statement having division operation in the program. By mistake, if the denominator entered is zero then it will give a runtime error.

Program 1.10. Program which generates runtime error *Division by zero*.

```

p10.11.py - C:/Users/Psscive/Desktop/JSD Class 11 Chapter 10...
File Edit Format Run Options Window Help
# Program to illustrate runtime error
num1 = 10.0
num2 = int(input("num2 = "))
# Division by zero encounter runtime error
print(num1/num2)
Ln: 9 Col: 0

IDLE Shell 3.10.4
File Edit Shell Debug Options Window Help
===== RESTART: C:/Users/Psscive/Desktop/JSD Class 11 Chapter 10/p10.11.py =====
num2 = 0
Traceback (most recent call last):
  File "C:/Users/Psscive/Desktop/JSD Class 11 Chapter 10/p10.11.py", line 5, in <module>
    print(num1/num2)
ZeroDivisionError: float division by zero
>>>
===== RESTART: C:/Users/Psscive/Desktop/JSD Class 11 Chapter 10/p10.11.py =====
num2 = hello
Traceback (most recent call last):
  File "C:/Users/Psscive/Desktop/JSD Class 11 Chapter 10/p10.11.py", line 3, in <module>
    num2 = int(input("num2 = "))
ValueError: invalid literal for int() with base 10: 'hello'
>>>
===== RESTART: C:/Users/Psscive/Desktop/JSD Class 11 Chapter 10/p10.11.py =====
num2 = 4
2.5
>>>
Ln: 26 Col: 0

```

Let us look at the output of Program 1.10 showing runtime errors while running the code. In first attempt when a user enters value '0' for num2, it gives error **ZeroDivisionError: float division by zero**. In 2nd attempt user enters a text value 'hello', it gives error **ValueError: invalid literal for int() with base 1**. In 3rd attempt, the program generates correct output when the user inputs an integer value 4 for num2 as shown in Output.

CHECK YOUR PROGRESS

A. Multiple Choice Questions

1. The with special meaning in Python, fixed for specific functionality are called (a) identifiers (b) functions (c) keywords (d) literals
2. Names given to different parts of Python program are (a) Identifiers (b) functions (c) Keywords (d) literals
3. The lines beginning with a certain character, and which are ignored by a compiler and not executed, are called _____ (a) operators (b) operands (c) functions (d) comments
4. Which of the following functions print the output to the console? (a) Output() (b) Print() (c) Echo() (d) print()
5. The input() returns the value as _____ type. (a) integer (b) string (c) floating point (d) none of these
6. The default separator character of print () is _____ (a) Tab (b) space (c) newline (d) dot
7. Which of the following cannot be used for naming an identifier in Python (a) Digits (0-9) (b) Letters(A-Z) (c) Underscore (_) (d) Special symbols(!, @, #, \$, %)
8. Which of the following is not a sequence data type (a) String (b) List (c) Tuple (d) Set
9. Which of the following is mapping data type (a) String (b) List (c) Tuple (d) Dictionary
10. Variables whose values can be changed in the same memory location, after they are created and assigned are called (a) mutable (b) immutable (c) static (d) dynamic
11. A mobile phone book is a good application of (a) Set (b) List (c) Tuple (d) Dictionary
12. The output of the following code is (a) 2 (b) 2.5 (c) 3 (d) 3.5

```
a=10
b=4
print(a/b)
```

13. The output of the following code is (a) 4 (b) 6 (c) 10 (d) 14

```
a=10
b=4
a+=b
print(a)
```

14. The output of the following code is (a) 0 (b) 1 (c) 01 (d) 10

```
a=0
b=1
print (a and b)
```

15. The output of the code $15.0 / 3 + (8 + 4.0)$ is (a) 12.0 (b) 13.0 (c) 15.0 (d) 17.0
16. Which of the following Python statement has syntax error (a) $10+20$ (b) $(10+20)$ (c) $(10+20)$ (d) $((10+20))$
17. Wrong output of a program is an indication of (a) Syntax errors (b) Logical errors (c) Runtime error (d) Compile time error
18. Division by zero is a (a) Syntax error (b) Logical error (c) Runtime error (d) Compile error

B. State whether True or False

1. Keywords can be used as identifier names.
2. The identifiers in Python can begin with an underscore.
3. Variables once assigned a value can be given any other value.
4. Python variables support dynamic typing.
5. A keyword can be renamed.
6. String values in Python can be single line strings, and multi-line strings.
7. A variable can contain values of different types at different times.
8. Expressions contain values/variables along with operators.

9. Comments are not executed by interpreter.
10. Set can have duplicate entries.

C. Fill in the Blanks

1. A _____ is a reserved word carrying special meaning and purpose.
2. Literals are the _____ values.
3. A legal combination of symbols that represents a value is _____.
4. In Python, the comments begin with _____ character.
5. The input() function returns the read value as of _____ type.
6. To convert an input()'s value in integer type, _____ function is used.
7. To convert an input()'s value in floating-point type _____ function is used.
8. Strings can be created with single quotes, double quotes and _____ quotes.
9. Python interpreter can be used in two modes _____ mode and _____ mode.
10. _____ None is a special data type with a _____ value.

D. Programming Questions

1. Which of the following identifier names are invalid and why?
 - Serial no.**
 - 1st_Room**
 - Hundred\$**
 - Total Marks**
 - total Marks**
 - total-Marks**
 - _Percentage**
2. Write the Python assignment statements to:
 - a) Assign 10 to variable length and 20 to variable breadth.
 - b) Assign the average of values of variables length and breadth to a variable sum.
 - c) Assign a list containing strings 'Paper', 'Gel Pen', and 'Eraser' to a variable stationery.
 - d) Assign strings 'Mohandas', 'Karamchand', and 'Gandhi' to variables first, middle and last
 - e) Assign the concatenated value of string variables first, middle and last to variable full name. Incorporate blank spaces appropriately between different parts of names.
3. Write logical expressions corresponding to the following statements in Python and evaluate the expressions, assuming variables num1, num2, num3, first, middle, last are already having meaningful values:
 - a) The sum of 20 and -10 is less than 12.
 - b) num3 is not more than 24.
 - c) 6.75 is between the values of integers num1 and num2.
 - d) The string 'middle' is larger than the string 'first' and smaller than the string 'last'.
 - e) List Stationery is empty.
4. Add a pair of parentheses to each expression so that it evaluates to True.
 - a) $0 == 1 == 2$
 - b) $2 + 3 == 4 + 5 == 7$
 - c) $1 < -1 == 3 > 4$
5. Write the output of the following:
 - a) **num1 = 4**
num2 = num1 + 1
num1 = 2 print (num1, num2)

- b) `num1, num2 = 2, 6`
`num1, num2 = num2, num1 + 2`
`print (num1, num2)`
- c) `num1, num2 = 2, 3`
`num3, num2 = num1, num3 + 1`
`print (num1, num2, num3)`

6. Which data type will be used to represent the following data values and why?

Data values	Data type	Reason
Number of months in a year		
Resident of Delhi or not		
Mobile number		
Pocket money		
Volume of a sphere		
Perimeter of a square		
Name of the student		
Address of the student		

7. What will be the output of statement `print(num1)` in the following example when `num1 = 4`, `num2 = 3`, `num3 = 2`

```
num1 += num2 + num
num1 = num1 ** (num2 + num3)
num1 **= num2 + num3
num1 = '5' + '5'
num1 = 2+9*((3*12)-8)/10
num1 = 24 // 4 // 2
num1 = float(10)
num1 = int('3.14')
```

8. What will be the output of following statements

- `print('Bye' == 'BYE')`
- `print(10 != 9 and 20 >= 20)`
- `print(10 + 6 * 2 ** 2 != 9//4 -3 and 29 >= 29/9)`
- `print(5 % 10 + 10 < 50 and 29 <= 29)`
- `print((0 < 6) or (not(10 == 6) and 10<0))`

9. Write a Python program to convert temperature in degree Celsius to degree Fahrenheit. If water boils at 100°C and freezes as 0°C, use the program to find out what is the boiling point and freezing point of water on the Fahrenheit scale. (Hint: $T(^{\circ}F) = T(^{\circ}C) \times 9/5 + 32$)

10. Write a Python program to calculate the amount payable if money has been lent on simple interest. Principal or money lent = P, Rate of interest = R% per annum and Time = T years. Then Simple Interest (SI) = $(P \times R \times T) / 100$.

Amount payable = Principal + SI. P, R and T are given as input to the program.

11. Write a program to calculate in how many days a work will be completed by three persons A, B and C together. A, B, C take x days, y days and z days respectively to do the job alone. The formula to calculate the number of days if they work together is $xyz/(xy + yz + xz)$ days where x, y, and z are given as input to the program.

12. Write a program to enter two integers and perform all arithmetic operations on them.

13. Write a program to enter five subject marks and print the average marks.

14. Write a program to swap two numbers using a third variable.

15. Write a program to swap two numbers without using a third variable.

16. Write a program to repeat the string “GOOD MORNING” n times, n is integer entered by user.
17. Write a program to find average of three numbers.
18. Write a Python program to find the volume of spheres with radius 7cm, 12cm, 16cm, respectively. The volume of a sphere with radius r is $\frac{4}{3}\pi r^3$.
19. Write a program that asks the user to enter their name and age. Print a message addressed to the user that tells the user the year in which they will turn 100 years old.
20. The formula $E = mc^2$ states that the equivalent energy (E) can be calculated as the mass (m) multiplied by the speed of light ($c = \text{about } 3 \times 10^8 \text{ m/s}$) squared. Write a program that accepts the mass of an object and determines its energy.

Session 2: Control Structures

In our daily life there are some situations where we have to follow a fixed sequence of steps to complete a task and in other situations we have some choices to complete a task. For example, in air travel every passenger has to follow the following steps to board on an airplane.

1. Show ticket and identity proof at the main entrance of the Airport.
2. Scanning of luggage.
3. Take your boarding pass.
4. Pass security check.
5. Finally, board on plane.

Passenger don't have choice to change the sequence of above written steps to board on the airplane. On the other hand, Passengers can take their boarding pass from airline's check-in counter or they can also get boarding pass at an electronic kiosk nearby as shown in Figure 2.1. Similarly, in programming generally statements are executed from beginning to end as in the sequence they are written. But we may encounter some situations in programming also where statements are required to change the normal sequence of execution.



Fig. 2.1: Steps to board on airplane

In this chapter, you will understand the control structures, their syntax and use in Python programming. The break and continue statements are also discussed.

2.2 Control structures

The order of execution of the statements in a program is known as flow of control. The flow of control can be implemented using control structures. Python supports two types of control structures—selection and repetition. Let us discuss both types of control structures in detail one by one.

2.3 Selection

Now suppose we have Rs.10 to buy a pen. On visiting the stationery shop, there are a variety of pens priced at Rs.10 each. Here, we have to decide which pen to buy. Similarly, when we use the direction services of a digital map, to reach from one place to another, we notice that sometimes it shows more than one path like the least crowded path, shortest distance path. We decide the path as per our priority. A decision involves selecting from one of the two or more possible options.

In programming, the concept of decision making or selection is implemented with the help of *if...else* statement. Suppose, we have to display the positive difference of the two numbers *num1* and *num2* as given in program 2.2. For that, we need to modify our approach. Look at the flowchart shown in Figure 2.2, first check the condition that *num1* is greater than *num2*. If *num1* is greater than *num2*, difference calculated by $diff = num1 - num2$ otherwise $diff = num2 - num1$. In other words, subtract smaller number from the bigger number so that the result is always a positive difference. This selection is based upon the values that are input for the two numbers *num1* and *num2*.

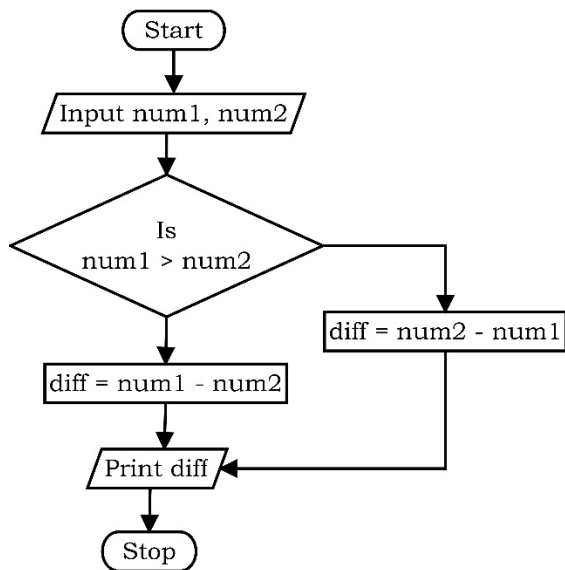


Fig. 2.2: Flow chart depicting decision making

The syntax of if statement is:

if condition:

statement(s)

Let us take an example to understand if control structure

Example 2.1

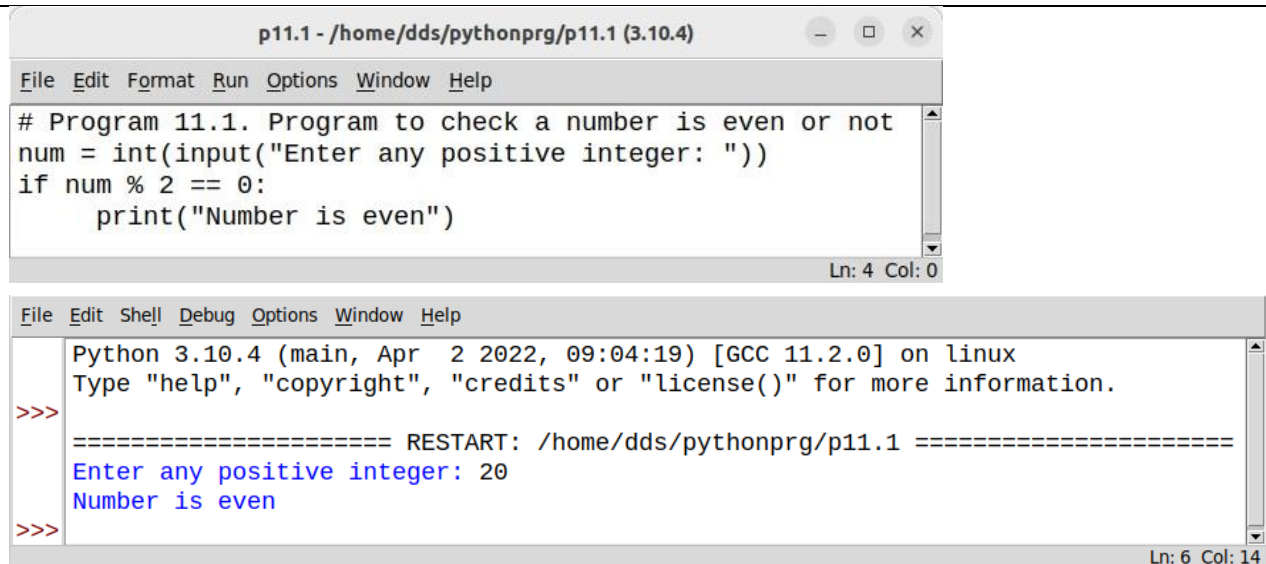
```
age = int(input("Enter your age "))
```

```
if age >= 18:
```

```
    print("Eligible to vote")
```

In the above example, if the age entered by the user is greater than 18, then print that the user is eligible to vote. If the condition is true, then the indented statement(s) are executed otherwise they are not.

The indentation implies that its execution is dependent on the condition. There is no limit on the number of statements that can appear as a block under the if statement.



```

p11.1 - /home/dds/pythonprg/p11.1 (3.10.4)
File Edit Format Run Options Window Help
# Program 11.1. Program to check a number is even or not
num = int(input("Enter any positive integer: "))
if num % 2 == 0:
    print("Number is even")
Ln: 4 Col: 0

Python 3.10.4 (main, Apr 2 2022, 09:04:19) [GCC 11.2.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /home/dds/pythonprg/p11.1 =====
Enter any positive integer: 20
Number is even
>>>
Ln: 6 Col: 14

```

A variant of if statement called *if...else* statement that allows to write two alternative paths and the control condition determines which path gets executed. The syntax for *if...else* statement is as follows.

```

if condition:
    statement(s)
else:
    statement(s)

```

Let us now modify the *Example 2.1* on voting to illustrate the use of *if else* structure.

```
age = int(input("Enter your age: "))
```

```

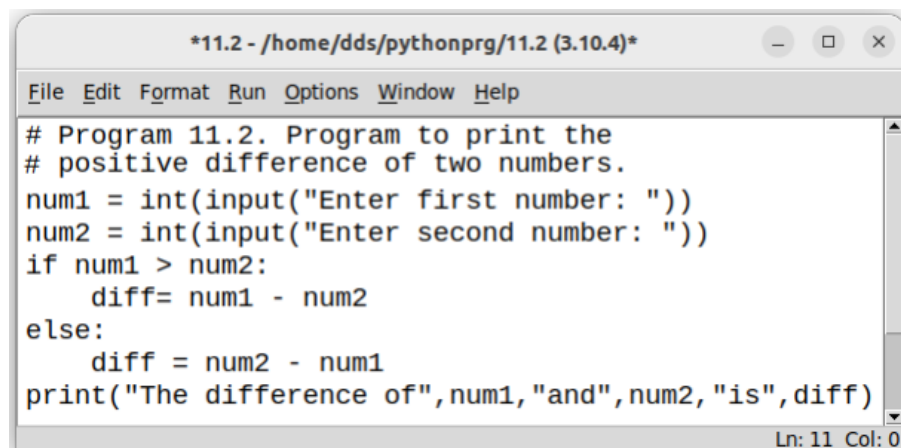
if age >= 18:
    print("Eligible to vote")
else:
    print("Not eligible to vote")

```

In above example, user will be asked to enter age. If the age entered by the user is greater than 18, then to display that the user is eligible to vote, otherwise display that the user is not eligible to vote.

Now let us use the same concept to modify the Program 2.1, so that it always gives a positive difference as the output. From the flowchart in Figure 2.2, it is required to decide whether *num1* > *num2* or not and take action accordingly.

We have to specify two blocks of statements since *num1* can be greater than *num2* or vice-versa as shown in Program 2.2.



```

*11.2 - /home/dds/pythonprg/11.2 (3.10.4)*
File Edit Format Run Options Window Help
# Program 11.2. Program to print the
# positive difference of two numbers.
num1 = int(input("Enter first number: "))
num2 = int(input("Enter second number: "))
if num1 > num2:
    diff= num1 - num2
else:
    diff = num2 - num1
print("The difference of", num1, "and", num2, "is", diff)
Ln: 11 Col: 0

```



```

File Edit Shell Debug Options Window Help
Python 3.10.4 (main, Apr 2 2022, 09:04:19) [GCC 11.2.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /home/dds/pythonprg/11.2 =====
Enter first number: 5
Enter second number: 6
The difference of 5 and 6 is 1
>>>
Ln: 9 Col: 0

```

In above example, condition $num1 > num2$ holds False for the values of $num1$ and $num2$ entered by user. Therefore, the statement after else will be executed.

There may be a situation, when you have multiple conditions to check and these all conditions are independent to each other. You can use *elif* statement to include multiple conditional expressions after the if condition or between the if and else control structure.

The syntax for a selection structure using *elif* is as shown below.

```

if condition1:
    statement1
elif condition2:
    statement2
elif condition3:
    statement3
else:
    statement4

```

In *elif* structure, if *condition1* is true then *statement1* will be executed. If *condition1* is false then control will go to *condition2*. If *condition2* is True then *statement2* will be executed. Similarly, if *condition2* is false then *condition3* will be checked and found true, then *statement3* will be executed. Now, if none of the conditions is true then only *statement4* will be executed. Here, *statement1*, *statement2*, *statement3*, *statement4* can be a single statement or block of statements.

Example 2.2 Check whether a number is positive, negative, or zero.

```

e11.2.py - C:/Users/Psscive/Desktop/Fig.Unit 3.Python Pr...
File Edit Format Run Options Window Help
number = int(input("Enter a number: "))
if number > 0:
    print("Number is positive")
elif number < 0:
    print("Number is negative")
else:
    print("Number is zero")
Ln: 12 Col: 0

```

```

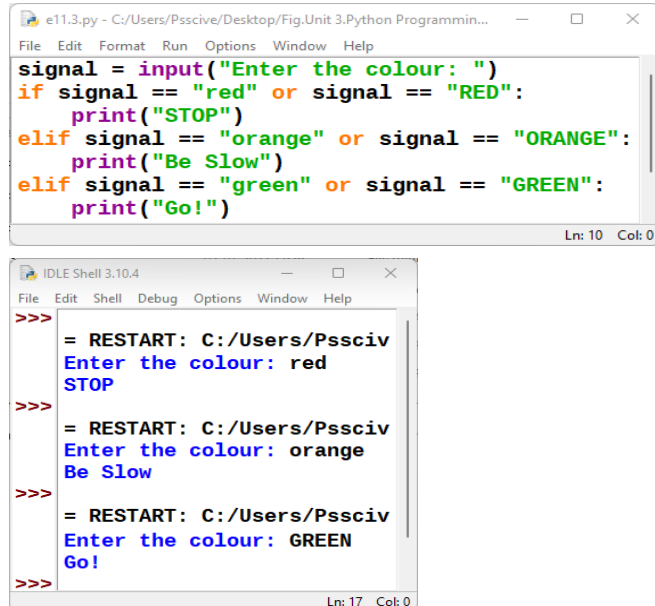
IDLE Shell 3.10.4
File Edit Shell Debug Options Window
>>>
= RESTART: C:/Users/Fig.Uni
Enter a number: 5
Number is positive
>>>
= RESTART: C:/Users/Fig.Uni
Enter a number: -2
Number is negative
>>>
= RESTART: C:/Users/Fig.Uni
Enter a number: 0
Number is zero
>>>
Ln: 15 Col: 0

```

Three different numbers are checked in the above program. In Output1, the entered number 5 is greater than 0. Here, the condition associated with if structure is true, so the statement to

print "Number is positive" is executed. In Output 2, the entered number -2 is less than 0. Here, the condition associated with *elif* structure is true, so the statement to print "Number is negative" is executed. In Output 3, the entered number 0 is neither greater than nor less than 0. Here, the condition associated with both if and *elif* is false, so the statement just after else keyword to print "Number is zero" is executed.

Example 2.3. Display appropriate message as per the colour of signal at the road crossing.



The screenshot shows two windows from the Python IDE. The top window is a Python script named 'e11.3.py' with the following code:

```
signal = input("Enter the colour: ")
if signal == "red" or signal == "RED":
    print("STOP")
elif signal == "orange" or signal == "ORANGE":
    print("Be Slow")
elif signal == "green" or signal == "GREEN":
    print("Go!")
```

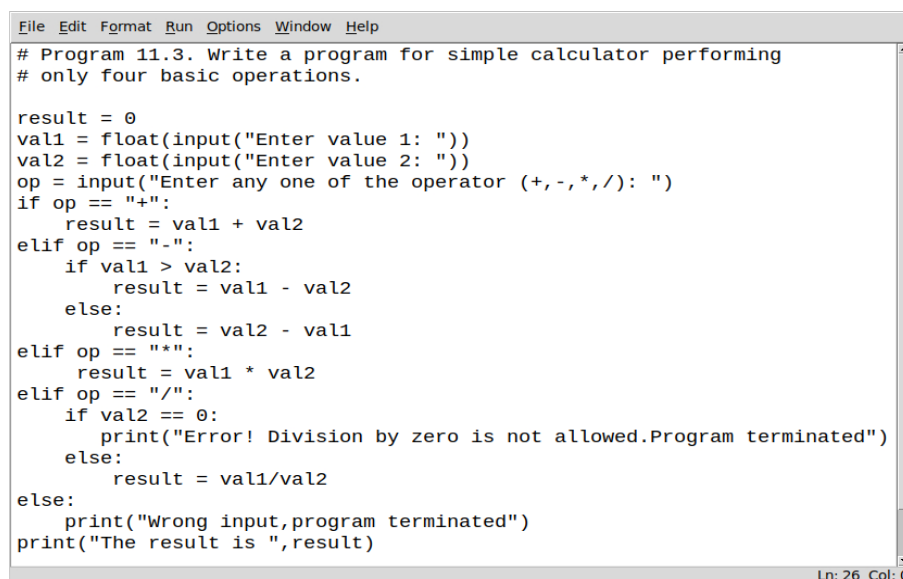
The bottom window is the IDLE Shell, showing the execution of the program three times:

```
>>> = RESTART: C:/Users/Pssciv
Enter the colour: red
STOP
>>> = RESTART: C:/Users/Pssciv
Enter the colour: orange
Be Slow
>>> = RESTART: C:/Users/Pssciv
Enter the colour: GREEN
Go!
```

Number of *elif* is dependent on the number of conditions to be checked. If the first condition is false, then the next condition is checked, and so on. If one of the conditions is true, then the corresponding indented block executes, and the if statement terminates.

Let us write a program to create a simple calculator to perform basic arithmetic operations on two numbers. The program should do the following:

1. Accept two numbers from the user.
2. Ask user to input any of the operator (+, -, *, /).
3. An error message is displayed if the user enters anything else.
4. Display only positive difference in case of the operator "-".
5. Display a message "Please enter a value other than 0" if the user enters the second number as 0 and operator '/' is entered.



The screenshot shows a Python script in the IDE with the following code:

```
# Program 11.3. Write a program for simple calculator performing
# only four basic operations.

result = 0
val1 = float(input("Enter value 1: "))
val2 = float(input("Enter value 2: "))
op = input("Enter any one of the operator (+,-,*,/): ")
if op == "+":
    result = val1 + val2
elif op == "-":
    if val1 > val2:
        result = val1 - val2
    else:
        result = val2 - val1
elif op == "*":
    result = val1 * val2
elif op == "/":
    if val2 == 0:
        print("Error! Division by zero is not allowed.Program terminated")
    else:
        result = val1/val2
else:
    print("Wrong input,program terminated")
print("The result is ",result)
```

```
File Edit Shell Debug Options Window Help
Enter value 1: 84
Enter value 2: 4
Enter any one of the operator (+,-,*,/): /
The result is 21.0
>>>
Ln: 20 Col: 0
```

In above program, for the operators "-" and "/", there exists an if...else condition within the elif block. This is called nested if. There can be many levels of nesting inside if...else statements. It is possible to use *if elif* structure as illustrated in Program 2.3. In Python 3.10 a much more powerful and flexible construct called Structural Pattern Matching is available. It can be used as a simple switch statement but is capable of much more.

Structural pattern matching

Structural pattern matching introduces the *match...case* statement and the pattern syntax to Python. The *match...case* statement follows the same basic outline as *switch...case* in other programming language. It takes an object, tests it against one or more match patterns, and takes an action if it finds a match.

Python performs matches by going through the list of cases from top to bottom. On the first match, Python executes the statements in the corresponding case block, then skips to the end of the match block and continues with the rest of the program. There is no “*fall-through*” between cases, but it’s possible to design your logic to handle multiple possible cases in a single case block. Let us take an example to use structural pattern matching.

Example 2.4. Example to use structural pattern matching

```
File Edit Format Run Options Window Help
digit=int(input("Enter any digit from 0 to 9: "))
match digit:
    case 0:
        print("Zero")
    case 1:
        print("One")
    case 2:
        print("Two")
    case 3:
        print("Three")
    case 4:
        print("Four")
    case 5:
        print("Five")
    case 6:
        print("Six")
    case 7:
        print("Seven")
    case 8:
        print("Eight")
    case 9:
        print("Nine")
    case _:
        print("Invalid input")
Ln: 26 Col: 12
```

In above program, value entered by user is stored in variable digit. The values written after case will be matched with value of digit one by one. If digit = 0 as specified in case 0 then statement to print zero will be executed. If digit is not equal to 0, control will go to next case statement specified as case 1. If digit =1, statement to print one will be executed. Similarly, it will go on for case 2 to case 9. For last case, if digit is anything except digits 0 to 9, It will print “*Invalid input*”.

```
>>> Enter any digit from 0 to 9: 3
Three
>>>
===== RESTART: .
Enter any digit from 0 to 9: 11
Invalid input
```

Assignment

1. Write a program to check whether a number is divisible by 7 or not.
2. Write a program to check whether an alphabet is a vowel or consonant.
3. Write a program to input month number and print the month name.
4. Write a program to check a triangle is equilateral, isosceles or scalene on the basis of the length of the sides provided by user.

2.3 Indentation

In most programming languages, the statements within a block are put inside curly brackets. However, Python uses indentation for block as well as for nested block structures. Leading whitespace (spaces and tabs) at the beginning of a statement is called indentation. In Python, the same level of indentation associates statements into a single block of code. The interpreter checks indentation levels very strictly and throws up syntax errors if indentation is not correct. It is a common practice to use a single tab for each level of indentation.

In the program 2.4, the if-else statement has two blocks of statements and the statements in each block are indented with the same amount of spaces or tabs.

```
File Edit Format Run Options Window Help
# Program 11.4 Program to find the larger of the two numbers.

num1 = int(input("Enter first number "))
num2 = int(input("Enter second number "))

if num1 > num2:                #Block1
    print("First number is larger")
    print("Bye")
else:                          #Block2
    print("Second number is larger")
    print("Bye Bye")

Ln: 17 Col: 0
```

```
File Edit Shell Debug Options Window Help
Enter first number 12
Enter second number 25
Second number is larger
Bye Bye
>>>
>>>
>>>

Ln: 13 Col: 0
```

In above program, as the condition $num\ 1 > num\ 2$ is false for the values of num1 and num2 taken in program. Therefore, the block of statements associated with else will be executed as shown in output.

2.4 Repetition

Sometimes we need to repeat the tasks such as payment of electricity bill is to be paid every month. Let us take an example which illustrates iterative process in nature also. Figure 2.3 shows the phases of the day like morning, midday and evening. These phases are repeated every day in the same order.

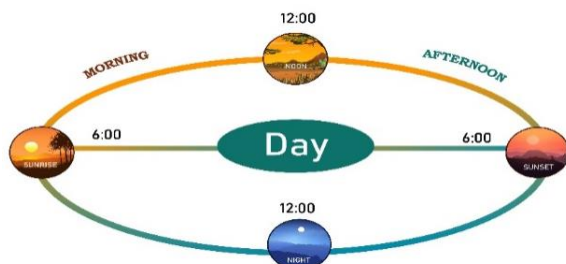
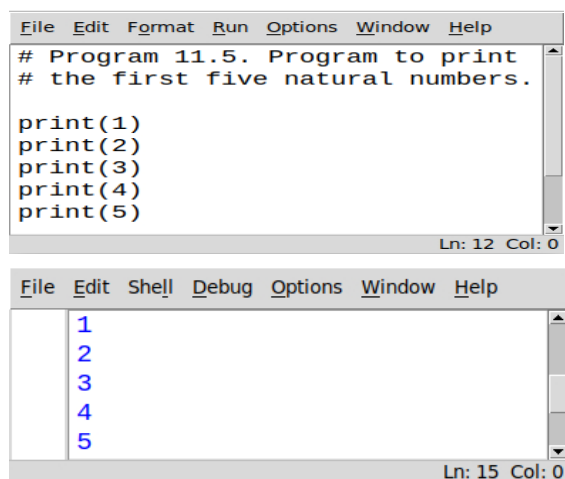


Fig. 2.3: Phases of the day

Another example where we repeat step, is calculation of mean marks for each student of the class separately. First, we list the steps to calculate mean of the marks scored in 5 subjects by a student

1. Read the marks scored by the student in five subjects.
2. Calculate sum of the marks of all 5 subjects.
2. Divide the sum by 5.
3. Result will be stored as mean.

Above steps will be repeated for each student of the class to get mean marks. This kind of repetition is also called iteration. Repetition of a set of statements in a program is made possible using looping constructs. To understand further, let us look at the program 2.5.



```
File Edit Format Run Options Window Help
# Program 11.5. Program to print
# the first five natural numbers.

print(1)
print(2)
print(3)
print(4)
print(5)
Ln: 12 Col: 0
```

```
File Edit Shell Debug Options Window Help
1
2
3
4
5
Ln: 15 Col: 0
```

In the above program, print () function is used 5 times to print 5 different natural numbers. But in the situation to print the first 100,000 natural numbers, it will not be efficient to write 100,000 print statements. In such case it is better to use loop or repetition in the program.

Looping constructs provide the facility to execute a set of statements in a program repetitively, based on a condition. The statements in a loop are executed again and again as long as particular logical condition remains **True**. This condition is checked based on the value of a variable called the loop control variable. When the condition becomes false, the loop terminates. It is the responsibility of the programmer to ensure that this condition eventually does become false so that there is an exit condition and it does not become an infinite loop. For example, if we did not set the condition count <= 100000, the program would have never stopped. There are two looping constructs in Python - for and while. Let us learn these looping constructs in detail.

2.4.1 For Loop

The for statement is used to iterate over a range of values or a fixed number of sequences or for each of the items in the range. These values can be numeric, string, list, or tuple. The flowchart depicting the execution of a for loop is given in Figure 2.4.

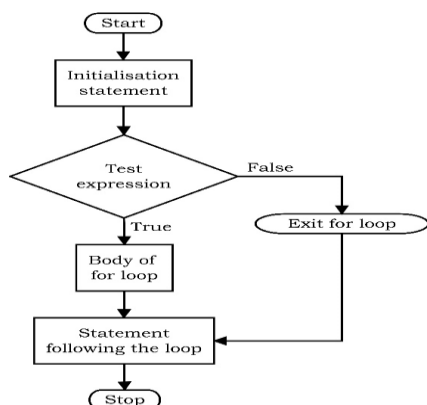


Fig. 2.4: Flow chart of for loop

Syntax of the For Loop

for <control-variable> in <sequence/items in range>:

<statements inside body of the loop>

With every iteration of the loop, the control variable checks whether each of the values in the range have been traversed or not. When all the items in the range are exhausted, the control is then transferred to the statement immediately following the for loop. While using for loop, it is known in advance the number of times the loop will execute.

```
File Edit Format Run Options Window Help
# Program 11.6. Program to print the
# characters in string using for loop.

for letter in "PYTHON":
    print(letter)
Ln: 11 Col: 0
```

```
File Edit Shell Debug Options Window Help
P
Y
T
H
O
N
Ln: 17 Col: 0
```

In the above program, letter variable represents a character of string “PYTHON”. For each iteration of for loop, it prints a character of string “PYTHON”.

```
File Edit Format Run Options Window Help
# Program 11.7. Program to print the numbers
# in a given sequence using for loop.

count = [10,20,30,40,50]
for num in count:
    print(num)
Ln: 3 Col: 0
```

```
File Edit Shell Debug Options Window Help
10
20
30
40
50
Ln: 26 Col: 0
```

In above program, count is a sequence of 5 integers. Variable *num* represents different integers in different iterations of for loop. So, it prints all the elements of count.

```
File Edit Format Run Options Window Help
# Program 11.8. Program to print even numbers
# in a given sequence using for loop.
print(" ")
numbers = [1,2,3,4,5,6,7,8,9,10]
for num in numbers:
    if (num % 2) == 0:
        print(num,'is an even Number')
Ln: 15 Col: 0
```

```
File Edit Shell Debug Options Window Help
2 is an even Number
4 is an even Number
6 is an even Number
8 is an even Number
10 is an even Number
Ln: 20 Col: 0
```

We discussed the programs that are example of using a for loop in Python. Let us also take a look at how range function can be used with for loop. If we enclose the range() inside a list(), then the values are returned in the form of a list.

The range() Function

The range () is a built-in function in Python. Syntax of range () function is:

range (start, stop, step)

It returns a sequence of integers from the given start value upto stop value (excluding stop value), with a difference of the given step value.

#Creating a list of first 10 natural numbers using range function.

```
>>>list(range(1,11,1))
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

#Creating a list of first 10 even numbers using range function.

```
>>> list(range(2,21,2))
```

```
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

#step value is 5

```
>>> list(range(0, 30, 5))
```

```
[0, 5, 10, 15, 20, 25]
```

In function range (), start, stop and step are parameters. The start and step parameters are optional. If start value is not specified, by default the list starts from 0. If step is also not specified, by default the value increases by 1 in each iteration. Let us take an example to understand this.

#start and step not specified

```
>>> list(range(11))
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

In the above example, value of only step parameter of range() function has been provided. It will take start=0 and step=1 by default.

All parameters of range() function must be integers. The step parameter can be a positive or a negative integer excluding zero.

Negative value of step parameter

```
>>> list(range(10,0,-1))
```

```
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
```

Negative value of step parameter of range () function will generate a decreasing sequence as illustrated in the above example. The function range () is often used in for loops for generating a sequence of numbers as illustrated below.

```
>>> for x in range (11):
```

```
    print(x)
```

Output:

```
0
```

```
1
```

```
2
```

```
3
```

```
4
```

5
6
7
8
9
10

In above python code, function *range (11)* will generate a sequence of numbers [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]. Variable *x* will hold the elements of this sequence one by one for different iterations of for loop. In each iteration of for loop *print(x)* statement will be executed. Therefore, all the elements of the sequence will be printed one by one as shown in output.

```
File Edit Format Run Options Window Help
# Program 11.9. Program to print the multiples
# of 10 for numbers in a given range.

for num in range(5):
    if num > 0:
        print(num * 10)
```

Ln: 11 Col: 0

```
File Edit Shell Debug Options Window Help
10
20
30
40
>>>
```

Ln: 11 Col: 0

In above program, *range (5)* function will generate a sequence of numbers [0,1,2,3,4]. Variable *num* will hold the elements of this sequence one by one for different iterations of for loop. In each iteration of for loop condition *num>0* will be evaluated. Condition *num>0* is false for first element of the sequence where *num=0*. Therefore, *print(num*10)* will not be executed only for first element that is 0. For rest of the elements, condition **num > 0** holds true. So, *print(num*10)* will be executed to get output as shown above.

Assignment

1. Write a program to print first 10 natural number in reverse order.
2. Write a program to print only odd numbers from the given list using for loop.
3. Write a program to display a list that stores squares of the elements of a given list.
4. Write a program using *for loop* that prompts the user for a hobby 3 times, then appends each one to a list named hobbies. Display the elements of list hobbies.

2.4.2 while Loop

The while statement executes a block of code repeatedly as long as the control condition of the loop is true. The control condition of the while loop is executed before any statement inside the loop is executed. After each iteration, the control condition is tested again and the loop continues as long as the condition remains true. When this condition becomes false, the statements in the body of loop are not executed and the control is transferred to the statement immediately following the body of while loop. If the condition of the while loop is initially false, the body is not executed even once. The flowchart of while loop is shown in Figure 2.5.

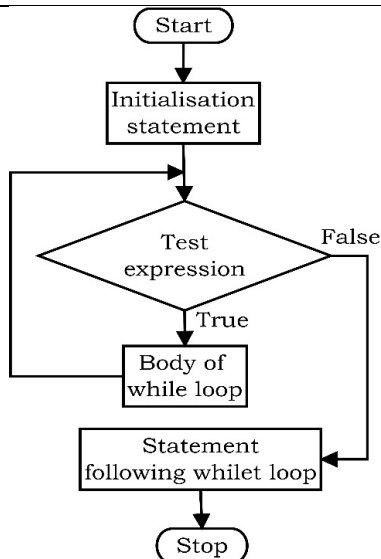


Fig. 2.5: Flow chart of while Loop

Syntax of while Loop

while test condition:

body of while

The statements within the body of the while loop must ensure that the condition eventually becomes false, otherwise the loop will become an infinite loop, leading to a logical error in the program.

```

File Edit Format Run Options Window Help
# Program 11.10. Program to print first
# 5 natural numbers using while loop.

count = 1
while count <= 5:
    print(count)
    count += 1
Ln: 12 Col: 0
  
```

```

File Edit Shell Debug Options Window Help
1
2
3
4
5
Ln: 16 Col: 0
  
```

```

File Edit Format Run Options Window Help
# Program 11.11. Program to find the factors
# of a whole number using while loop.
print(" ")
num = int(input("Enter a number to find its factor: "))
print (1, end=' ') #1 is a factor of every number
factor = 2
while factor <= num/2 :
    if num % factor == 0:
        print(factor, end=' ')
    factor += 1
print (num, end=' ') #every number is a factor of itself
Ln: 21 Col: 0
  
```

```
File Edit Shell Debug Options Window Help
Enter a number to find its factor: 6
1 2 3 6
>>>
Ln: 17 Col: 0
```

In the above program, body of the loop is indented with respect to while statement. Similarly, the statements within if are indented with respect to positioning of if statement.

Assignment

1. Write a program to print first 10 even numbers using while loop.
2. Write a program to find the sum of the digits of a number accepted from the user.
3. Write a program to reverse the number accepted from user using while loop.
4. Write a program to check the input number is Palindrome or not.
5. Write a program to check the input number is Armstrong or not.

2.5 break and continue Statement

Looping constructs allow programmers to repeat tasks efficiently. In certain situations, when some particular condition occurs, we may want to exit from a loop (come out of the loop forever) or skip some statements of the loop before continuing further in the loop. These requirements can be achieved by using break and continue statements, respectively. Python provides these statements as a tool to give more flexibility for the programmer to control the flow of execution of a program.

2.5.1 break Statement

The break statement alters as it terminates the current loop and resumes execution of the statement following that loop.

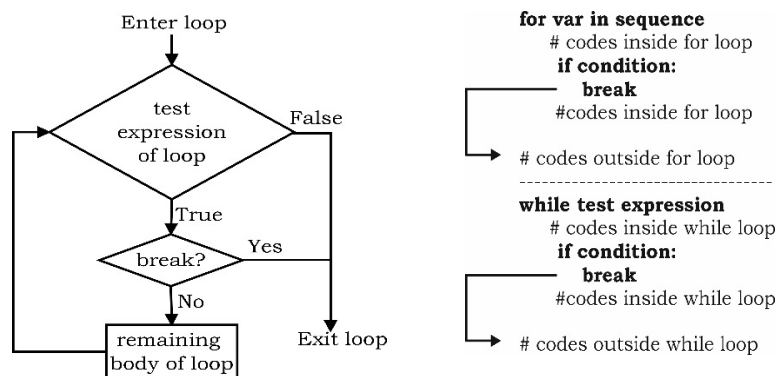


Fig. 2.6: Flowchart for using break statement in loop

Let us take a program to demonstrate use of break in for loop.

```
File Edit Format Run Options Window Help
# Program 11.12. Program to demonstrate
# use of break statement.
num = 0
for num in range(10):
    num = num + 1
    if num == 8:
        break
print('Num has value ' + str(num))
print('Encountered break!! Out of loop')
Ln: 15 Col: 0
```

```
File Edit Shell Debug Options Window Help
Num has value 8
Encountered break!! Out of loop
>>>
Ln: 19 Col: 0
```

In the above program, when value of **num** becomes 8, the break statement is executed and the for loop terminates.

```
File Edit Format Run Options Window Help
# Program 11.13. Find the sum of all the positive numbers using while loop
# Stop taking further input when entered negative number and display the sum
entry = 0
sum1 = 0
print("Enter numbers to find their sum, negative number ends the loop:")
while True:
#int() typecasts string to integer
    entry = int(input())
    if (entry < 0):
        break
    sum1 += entry
print("Sum =", sum1)
Ln: 28 Col: 0
```

```
File Edit Shell Debug Options Window Help
Enter numbers to find their sum, negative number ends the loop:
5
6
9
4
8
5
-7
Sum = 37
>>>
Ln: 24 Col: 0
```

```
File Edit Format Run Options Window Help
# Program 11.14. Program to check if the input number is prime or not.
num = int(input("Enter the number to be checked: "))
flag = 0 #presume num is a prime number
if num > 1 :
    for i in range(2, int(num / 2)):
        if (num % i == 0):
            flag = 1 #num is a not prime number break
            #no need to check any further
    if flag == 1:
        print(num , "is not a prime number")
    else:
        print(num , "is a prime number")
else :
    print("Entered number is <= 1, execute again!")
Ln: 18 Col: 0
```

```
File Edit Shell Debug Options Window Help
Enter the number to be checked: 25
25 is not a prime number
>>>
Ln: 12 Col: 0
```

Assignment

1. Write a program to find location of a particular item in a list.
2. Find output of the following Python code:

```
for val in "string":
    if val == "i":
```

```

break
print(val)
print("The end")

```

2.5.2 *continue* Statement

When a `continue` statement is encountered, the control skips the execution of remaining statements inside the body of the loop for the current iteration and jumps to the beginning of the loop for the next iteration. If the loop's condition is still true, the loop is entered again, else the control is transferred to the statement immediately following the loop. Figure 2.7 shows the flowchart of `continue` statement.

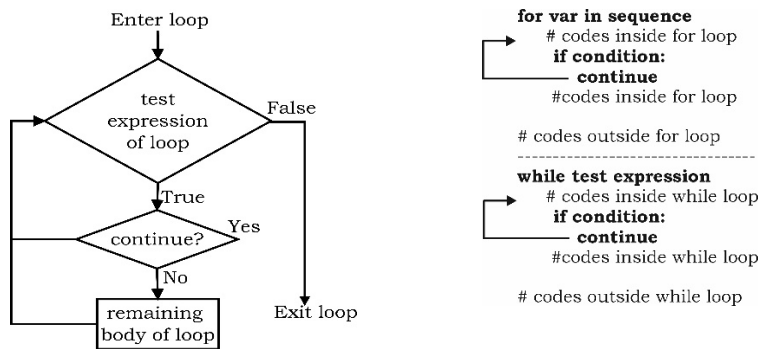


Fig. 2.7: Flow chart of `continue` statement

```

File Edit Format Run Options Window Help
# Program 11.15. Program to demonstrate
# the use of continue statement.
# Prints values from 0 to 6 except 3
num = 0
for num in range(6):
    num = num + 1
    if num == 3:
        continue
    print('Num has value ' + str(num))
print('End of loop')
Ln: 17 Col: 0

File Edit Shell Debug Options Window Help
Num has value 1
Num has value 2
Num has value 4
Num has value 5
Num has value 6
End of loop
>>>
Ln: 15 Col: 0

```

Observe that the value 3 is not printed in the output, but the loop continues after the `continue` statement to print other values till the `for` loop terminates.

2.5.3 `pass` statement

`Pass` statement is an empty statement or a null statement. Sometimes, when the user is not sure about a particular piece of code, `pass` statement can be used. It can be effectively used in `if` constructs, loops and function definitions.

For example:

```

for i in range(5):
    pass

```

The above code will not generate any output.

Assignment

1. Write a program to print values from 1 to 20 except multiples of 3.
2. Write a program to print all the prime numbers from 1 to 50.

2.6 Nested Loops

A loop may contain another loop inside it. A loop inside another loop is called a nested loop. The inner or outer loop can be any type, such as a while loop or for loop. For example, the outer for loop can contain a while loop and vice versa. The outer loop can contain more than one inner loop. There is no limitation on the chaining of loops.

Nested loops are typically used for working with number and star pattern programs. These are also useful to work with multidimensional data structures, such as printing two-dimension arrays, iterating a list that contains a nested list. You will learn about these array, multi dimension array and list in next chapters.

```
File Edit Format Run Options Window Help
#Program 11.16. Program to demonstrate nested for loop.
for var1 in range(3):
    print("Iteration " + str(var1 + 1) + " of outer loop")
    for var2 in range(2): #nested loop
        print(var2 + 1)
    print("Out of inner loop")
print("Out of outer loop")
Ln: 2 Col: 0
```

```
File Edit Shell Debug Options Window Help
Iteration 1 of outer loop
1
2
Out of inner loop
Iteration 2 of outer loop
1
2
Out of inner loop
Iteration 3 of outer loop
1
2
Out of inner loop
Out of outer loop
>>>
Ln: 23 Col: 0
```

```
File Edit Format Run Options Window Help
# Program 11.17. Print each adjective for every
# fruit in given lists using nested for loop
adj = ["Big", "Tasty"]
fruits = ["Apple", "Guava", "Mango"]
# Program to demonstrate nested for loops
adj = ["Big", "Tasty"]
fruits = ["Apple", "Guava", "Mango"]
for x in adj:
    for y in fruits:
        print(x, y)
Ln: 21 Col: 11
```

```
File Edit Shell Debug Options Window Help
Big Apple
Big Guava
Big Mango
Tasty Apple
Tasty Guava
Tasty Mango
>>>
Ln: 17 Col: 0
```

```

File Edit Format Run Options Window Help
# Program 11.18. Program to print the pattern input by the user.
# 1
# 1 2
# 1 2 3
# 1 2 3 4
# 1 2 3 4 5
num = int(input("Enter a number to generate its pattern = "))
for i in range(1,num + 1):
    for j in range(1,i + 1):
        print(j, end = " ")
    print()
Ln: 19 Col: 0

```

```

File Edit Shell Debug Options Window Help
Enter a number to generate its pattern = 5
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
>>>
Ln: 16 Col: 0

```

```

File Edit Format Run Options Window Help
# Program 11.19. Program to find prime numbers
# between 2 to 50 using nested loop.
num = 2
for i in range(2, 50):
    j= 2
    while ( j <= (i/2)):
        if (i % j == 0):    # factor found
            break          #break out of while loop
        j += 1
    if ( j > i/j) :        # no factor found
        print ( i, "is a prime number")
print ("Bye Bye!!")
Ln: 21 Col: 0

```

```

File Edit Shell Debug Options Window
Help
2 is a prime number
3 is a prime number
5 is a prime number
7 is a prime number
11 is a prime number
13 is a prime number
17 is a prime number
19 is a prime number
23 is a prime number
29 is a prime number
31 is a prime number
37 is a prime number
41 is a prime number
43 is a prime number
47 is a prime number
Bye Bye!!
>>>
Ln: 28 Col: 0

```

```
File Edit Format Run Options Window Help
# Program 11.20. Program to calculate the factorial of a given number.
num = int(input("Enter a number: "))
fact = 1
# check if the number is negative, positive or zero
if num < 0:
    print("Sorry, factorial does not exist for negative numbers")
elif num == 0:
    print("The factorial of 0 is 1")
else:
    for i in range(1, num + 1):
        fact = fact * i
print("Factorial of ", num, " is ", fact)
Ln: 19 Col: 0
```

```
File Edit Shell Debug Options Window Help
Enter a number: 5
Factorial of 5 is 120
>>>
Ln: 12 Col: 0
```

Assignment

- Write a program using nested loops to produce a rectangle of * with 6 rows and 20 * per row.
- Write a program to print a pattern like:

a.	b.	c.	d.	e.
1	1	A	1	5 4 3 2 1
2 2	2 1	A B	1 2	4 3 2 1
3 3 3	3 2 1	A B C	1 2 3	3 2 1
4 4 4 4	4 3 2 1	A B C D	1 2 3 4	2 1
5 5 5 5 5	5 4 3 2 1	A B C D E	1 2 3 4 5	1

CHECK YOUR PROGRESS**A. Multiple Choice Questions**

- Python supports two types of control structures (a) permutation and combination (b) module and library (c) built-in and user-defined (d) selection and repetition
- In programming, the concept of decision making or selection is implemented with the help of (a) if else statement (b) for loop (c) while loop (d) Functions
- In Python, the same level of indentation associate statements into (a) a single block of code (b) a single loop (c) a single function (d) a single control structure
- Each level of indentation is distinguished by _____ (a) single space (b) new line (c) tab space (d) #
- Repetition of a set of statements in a program is made possible using (a) if-else (b) loops (c) data types (d) functions
- When the condition associated with a loop becomes false, the loop (a) terminates (b) continues (c) fails (d) produce error
- Which function is used to create a list containing a sequence of integers from the given start value up to stop value (excluding stop value), with a difference of the given step value (a) range() (b) random() (c) maths() (d) int()
- The start and step parameters of range() function are (a) optional (b) mandatory (c) default (d) invalid

9. All parameters of range() function must be (a) integer (b) float (c) list (d) tuple
10. The step parameter of range() function can be a positive or a negative integer excluding (a) 0 (b) 1 (c) -1 (d) -2
11. Function range(x) will generate a sequence of numbers [0,1,2,3,4,5,6,7,8,9,10]. Then x=? (a) 10 (b) 11 (c) -10 (d) -11
12. Which of the following is not used as loop in Python? (a) for loop (b) while loop (c) do-while loop (d) None of the above
13. In a Python program, a flow control of execution: (a) defines program-specific data structures (b) directs the order of execution of the statements in the program (c) dictates what happens before the program starts and after it terminates (d) None of the above
14. How many times will the loop run? (a) 2 (b) 3 (c) 1 (d) 0

```
i=2
while(i>0):
i=i-1
```

15. What will be the output of the following code? (a) 12 (b) 1, 2 (c) 0 (d) Error

```
x = 12
for i in x:
print(i)
```

B. State whether True or False

1. There is no limit on the number of statements that can appear as a block under the if statement.
2. The statements in a loop are executed again and again as long as particular logical condition remains false.
3. The range () is a built-in function in Python.
4. The while statement executes a block of code repeatedly as long as the control condition of the loop is true.
5. There is no limitation on the chaining of loops.
6. Keyword "break" can be used to bring control out of the current loop.
7. A loop becomes infinite loop if a condition never becomes FALSE.
8. If the condition is TRUE the statements of if block will be executed otherwise the statements in the else block will be executed.
9. *do-while* is a valid loop a in Python.
10. *break* and *continue* are jump statements.

C. Fill in the Blanks

1. The order of execution of the statements in a program is known as _____.
2. Python uses _____ for block as well as for nested block structures.
3. The interpreter checks indentation levels very strictly and throws up _____ errors if indentation is not correct.
4. The _____ clause can occur with an if as well as with loops.
5. The break statement _____ the current loop and resumes execution of the statement following that loop.
6. When a continue statement is encountered, the control _____ the execution of remaining statements inside the body of the loop for the current iteration and jumps to the beginning of the loop for the next iteration.
7. A loop inside another loop is called a _____ loop.
8. Common use of nested loop is to print various _____ and _____ pattern.
9. For working with _____ data structures, such as printing two-dimensional arrays, nested loops are typically used.

10. The statement to check if a is equal to b is if _____.

D. Programming Questions

1. Write a program that takes the input name and age and displays a message whether the user is eligible to apply for a driving license or not. The eligible age is 18 years.
2. Write a program to print the table of a given number entered by the user.
3. Write a program that prints minimum and maximum of five numbers entered by the user.
4. Write a program to check if the year entered by the user is a leap year or not.
5. Write a program to generate the sequence: $-5, 10, -15, 20, -25\dots$ up to n , where n is an integer input by the user.
6. Write a program to find the sum of $1 + \frac{1}{8} + \frac{1}{27} \dots \frac{1}{n}$, where n is entered by the user.
7. Write a program to find the sum of digits of an integer number, input by the user.
8. Write a function that checks whether an input number is a palindrome or not.
9. Write a program to find largest number of a list of numbers entered through keyboard.
10. Write a program to input N number and then print the second largest number.

Session 3: Functions

In a marriage ceremony, a lot of work has to be done related to catering, decoration and other things. In general, we assign these different tasks to particular group of people who is service provider for a particular task. For example, catering work is assigned to caterers and decoration work is assigned to decorators. This makes the process easy to manage. Similarly, in programming also we use functions to do a specific task to make our program modular and easy to read. (Figure 3.1)



Fig. 3.1: Different tasks in a wedding ceremony

Suppose a factorial of a number can be calculated by simply executing the logic to calculate a factorial. But if that has to be done ten times in a day, writing the same logic again and again is going to be a long task. Instead of that it is better to write a function of that logic and that function can be used whenever we want, just by calling that function. This will reduce the complexity of code and save the time.

Till now we have written some programs and might have realized that as the problem gets complex, the number of lines in a program increase, which makes the program look bulky and difficult to manage.

Consider a problem of finding maximum of two numbers a and b . Let us take a simple Python program for this problem.

```

File Edit Format Run Options Window Help
# Program 12.1. Program to find maximum of
# two numbers without using functions

a=10
b=20
if a>b:
    max=a
else:
    max=b
print("Maximum =", max)
Ln: 14 Col: 0

File Edit Shell Debug Options Window Help
Maximum = 20
>>>
Ln: 9 Col: 0

```

In above program, a built-in function `print ()` is used to print the value. Another approach to solve the above problem is to divide the program into different blocks of code and keep the block of code in a function which is used to do some specific task. The process of dividing a computer program into separate independent blocks of code or separate sub-problems with different names and specific functionalities is known as modular programming.

In this chapter, you will understand the concept of functions and the benefits of using functions. We will discuss about user defined functions, flow of execution, scope of a variable and standard libraries in Python programming.

3.2 Functions

In programming, the use of function is one of the means to achieve modularity and re-usability. A function can be defined as a named group of instructions that accomplish a specific task when it is invoked. Once defined, a function can be called repeatedly from different places of the program without writing all the codes of that function every time, or it can be called from inside another function, by simply writing the name of the function and passing the required parameters, if any. The programmer can define as many functions as desired while writing the code.

The program 3.1 is rewritten using user defined functions as shown in program 3.2.

```

File Edit Format Run Options Window Help
# Program 12.2. Program to find maximum of
# two numbers using function

def max(x,y):
    if x>y:
        return x
    else:
        return y
print(" ")
a=10
b=20
print("Maximum in list1 =", max(a,b))
Ln: 9 Col: 10

File Edit Shell Debug Options Window Help
Maximum in list1 = 20
>>>
Ln: 10 Col: 0

```

If we compare program 3.1 and 3.2, it is evident that program 3.2 looks more organised and easier to read. In this program, the function `max ()` is defined and hence it works as user defined

function. The function `max()` is also a built in function in Python. When the function is defined by user with **def** keyword that works as a user defined function, even if it is built in function. If the function is used without defining it with **def** keyword then it will work as *built-in* function. In case of other programming language, it will give the error as the built in function cannot be used by user. This flexibility is offered in Python. Another function used in this program is `print()` which is a *built-in* function. In general, we have two types of functions in Python – *User defined functions* and *Built-in functions*.

The Advantages of Function

Following are the advantages of using functions in a program:

1. Increases readability, particularly for longer code as by using functions, the program is better organized and easy to understand.
2. Reduces code length as same code is not required to be written at multiple places in a program. This also makes debugging easier.
3. Increases re-usability, as function can be called from another function or another program. Thus, we can reuse or build upon already defined functions and avoid repetitions of writing the same piece of code.
4. Work can be easily divided among team members and completed in parallel.

3.3 User defined Functions

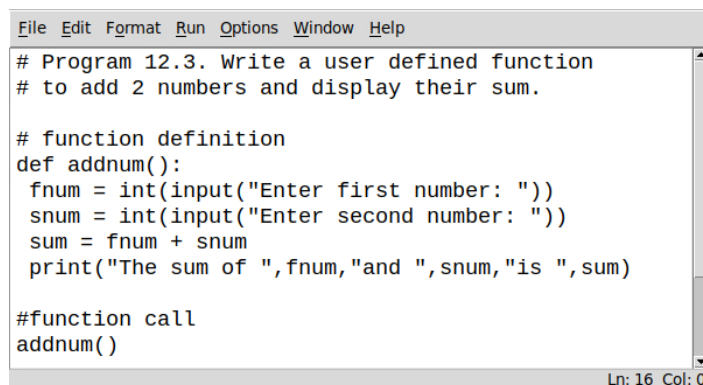
Taking advantage of re-usability feature of functions, there are large number of functions already available in Python under standard library. We can directly call these functions in our program without defining them. However, in addition to the standard library functions, we can define our own functions while writing the program. Such functions are called user defined functions. Thus, a function defined to achieve some tasks as per the programmer's requirement is called a user defined function.

3.3.1 Creating User Defined Function

A function definition begins with `def` (short for define). The syntax for creating a user defined function is given below.

```
def <Function name ( [parameter1, parameter2, ..... ] )  Function Header
    set of instruction to be executed                Function Body
    [return <value>]                                (should be intended with
                                                       the function header)
```

The items enclosed in "[]" are called parameters and they are optional. Hence, a function may or may not have parameters. Also, a function may or may not return a value. Function header always ends with a colon (:). Function name should be unique. Rules for naming identifiers also apply for function naming. The statements outside the function indentation are not considered as part of the function.



```
File Edit Format Run Options Window Help
# Program 12.3. Write a user defined function
# to add 2 numbers and display their sum.

# function definition
def addnum():
    fnum = int(input("Enter first number: "))
    snum = int(input("Enter second number: "))
    sum = fnum + snum
    print("The sum of ", fnum, "and ", snum, "is ", sum)

#function call
addnum()

Ln: 16 Col: 0
```

```
File Edit Shell Debug Options Window Help
Enter first number: 15
Enter second number: 63
The sum of 15 and 63 is 78
>>>
Ln: 14 Col: 0
```

In above program, **addnum()** is a user-defined function that takes two integer numbers as input, calculate sum of two numbers and display it. In order to execute the function, we need to call it. The function can be called in the program by writing function name followed by () as shown for **addnum()** function in the last line of Program 3.3.

The *def* is the keyword used to define a function and function name is written following *def* keyword. When it runs, it creates a new function object and assigns it a new name. As it is a statement, so there is no issue in using it inside any control structures like if else. Let us take an example to understand this:

Example 3.1:

```
File Edit Format Run Options Window Help
# Example 12.1:
def test():
    a=int(input("Enter a value: "))
    if a>0:
        print("Positive")
    else:
        print("Negative")

test() # Function call
Ln: 12 Col: 0
```

```
File Edit Shell Debug Options Window Help
Enter a value: 12
Positive
>>>
===== RESTART:
/home/dds/pythonprg/e12.1 =====
Enter a value: -63
Negative
>>>
Ln: 15 Col: 0
```

Assignment 3.1

1. Write a program to find the maximum of three numbers using a user-defined function.
2. Write a program to Find Factorial of Number using a user-defined function.
3. Write a program to print the sum of digit of a user entered number using user-defined function.
4. Write a program to print the day name by reading day number from user using a user-defined function.

3.3.2 Arguments and Parameters

In the above example, the numbers were accepted from the user within the function itself, but it is also possible for a user defined function to receive values at the time of being called. An argument is a value passed to the function during the function call which is received in corresponding parameter defined in function header.

```
File Edit Format Run Options Window Help
# Program 12.4. Program to find the sum of first n natural numbers
# using user defined function, # where n is passed as an argument
def sumSquares(n):      #n is the parameter
    sum = 0
    for i in range(1,n+1):
        sum = sum + i
    print("The sum of first",n,"natural numbers is: ",sum)

num = int(input("Enter the value for n: ")) #num is user defined argument
sumSquares(num) #function call
Ln: 8 Col: 0
```

```
File Edit Shell Debug Options Window Help
Enter the value for n: 6
The sum of first 6 natural numbers is: 21
>>>
Ln: 12 Col: 0
```

Let us assume that the user has input 5 during the execution of the Program 3.4. So, **num** refers to the value 5. It is then used as an argument in the function.

sumSquares(num)

Since the function is called, the control is transferred to execute the function

def sumSquares(n):

where parameter **n** also refers to the value 5 which **num** is referring to as shown in Figure 3.3. Since both **num** and **n** are referring to the same value, they are bound to have the same identity. We can use the `id()` function to find the identity of the object that the argument and parameter are referring to. Let us understand this with the help of the following example.

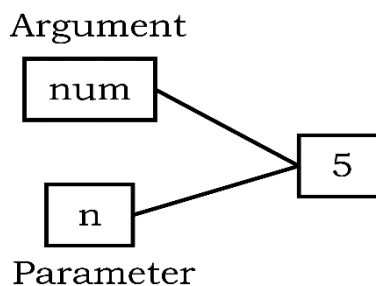


Fig. 3.2: Argument and parameter refer to the same value

```
File Edit Format Run Options Window Help
# Program 12.5. Program using user defined function that accepts an integer
# and increments the value by 5. Also display the id of argument (before
# function call), id of parameter before and after increment.

def incrValue(num):
    #id of Num before increment
    print("Parameter num has value:",num,"\nid =",id(num))
    num = num + 5
    #id of Num after increment
    print("num incremented by 5 is",num,"\nNow id is ",id(num))
    number = int(input("Enter a number: "))
    print("id of argument number is:",id(number)) #id of Number
    incrValue(number)
Ln: 18 Col: 0
```

```
File Edit Shell Debug Options Window Help
Enter a number: 5
id of argument number is: 140681514746224
Parameter num has value: 5
id = 140681514746224
num incremented by 5 is 10
Now id is 140681514746384
>>>
```

In above program output, *number* and *num* have the same id before value of *num* is incremented. After increment, the id of *num* has changed. After running this program code, you will get the different memory location as per your computer configuration.

Let us understand the above output through illustration as depicted in Figure 3.4.

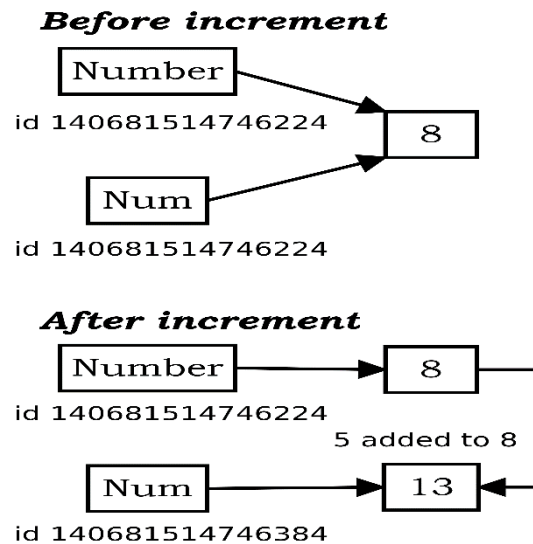


Fig. 3.3: Id of argument and parameter before and after increment

```
File Edit Format Run Options Window Help
# Program 12.6. Program using a user defined function
# myMean() to calculate the mean of floating values
# stored in a list.

def myMean(myList):
    total = 0
    count = 0
    for i in myList:
        total = total + i
        count = count + 1
    mean = total/count
    print("The calculated mean is:",mean)
#function ends here

myList = [1.2,2.3,3.4,4.5] #Function call
myMean(myList)
Ln: 21 Col: 0

File Edit Shell Debug Options Window Help
>>> The calculated mean is: 2.85
Ln: 10 Col: 0
```

```
File Edit Format Run Options Window Help
# Program 12.7. Write a program using a user
# defined function calcFact() to calculate and
# display the factorial of a number num passed
# as an argument.

def calcFact(num):
    fact = 1
    for i in range(num, 0, -1):
        fact = fact * i
    print("Factorial of", num, "is", fact)

num = int(input("Enter the number: "))

calcFact(num)
```

Ln: 19 Col: 0

```
File Edit Shell Debug Options Window Help
Enter the number: 5
Factorial of 5 is 120
>>>
```

Ln: 7 Col: 0

In above program, when function `calcFact()` is called; Control is transferred to the line where function is defined. Initially, value 1 is assigned to the variable `fact`. Then in for loop, variable `i` holds the value from the list [5, 4, 3, 2, 1] one by one for above output in which user entered 5. In each iteration, product of variable `fact` and `i` is assigned to `fact` as shown below in Table 3.1.

Table 3.1: Iterations of factorial() function

<i>iteration</i>	<i>fact</i>	<i>i</i>	<i>fact = fact*i</i>
1	1	5	5
2	5	4	20
3	20	3	60
4	60	2	120
5	120	1	120

Sometimes, we need to pass strings also as parameter to a function. In some cases, we want default values also for parameters. Next, we will learn to handle these situations in which we use following approaches – *String as Parameters and Default Parameter*.

String as Parameters – In programs 3.4 to 3.7, the arguments passed are of numeric type. However, in some programs, user may need to pass string values as an argument, as shown in Program 3.8.

```
File Edit Format Run Options Window Help
# Program 12.8. Program to concatenate two strings
# using a by user defined function.

def fullname(first, last):
    # + operator is used to concatenate strings
    fullname = first + " " + last
    print("Hello", fullname)
    #function ends here

first = input("Enter first name: ")
last = input("Enter last name: ")

fullname(first, last)    #function call
```

Ln: 17 Col: 0

```
File Edit Shell Debug Options Window Help
Enter first name: Aditi
Enter last name: Gaur
Hello Aditi Gaur
>>>
Ln: 18 Col: 0
```

Default Parameter

Python allows assigning a default value to the parameter. A default value is a value that is pre-decided and assigned to the parameter when the function call does not have its corresponding argument.

```
File Edit Format Run Options Window Help
# Program 12.9. Program to convert the denominator of the division into
# proper fraction using user deined function.

def mixedFraction(num,deno = 1):
    remainder = num % deno
#check if the fraction does not evaluate to a whole number
    if remainder!= 0:
        quotient = int(num/deno)
        print("The mixed fraction=", quotient,"(",remainder, "/", deno,")")
    else:
        print("The given fraction evaluates to a whole number")
#function ends here

num = int(input("Enter the numerator: "))
deno = int(input("Enter the denominator: "))
print("You entered:",num,"/",deno)
if num > deno: #condition to check whether the fraction is improper
    mixedFraction(num,deno) #function call
else:
    print("It is a proper fraction")
Ln: 23 Col: 0
```

```
File Edit Shell Debug Options Window Help
Enter the numerator: 25
Enter the denominator: 3
You entered: 25 / 3
The mixed fraction= 8 ( 1 / 3 )
>>>
Ln: 33 Col: 0
```

In the above program, the denominator entered is 3, which is passed to the parameter "*deno*" so the default value of the argument *deno* is overwritten. Let us consider the following function call: **mixedFraction(9)**. Here, num will be assigned 25 and *deno* will use the default value 1.

A function argument can also be an expression, such as

mixedFraction(num+5, deno+5)

In such a case, the argument is evaluated before calling the function so that a valid value can be assigned to the parameter. The parameters should be in the same order as that of the arguments.

The default parameters must be the trailing parameters in the function header that means if any parameter is having default value then all the other parameters to its right must also have default values. For example,

def mixedFraction(num,deno = 1)

def mixedFraction(num = 2,deno = 1)

Let us consider few more function definition headers:

def calcInterest(principal = 1000, rate, time = 5):

Above header is incorrect as default must be the last #parameter. So, the correct function header should be as follows.

```
def calcInterest(rate, principal = 1000, time = 5):
```

One more point is important to note that a function header cannot have expressions. Therefore, following function headers will give error:

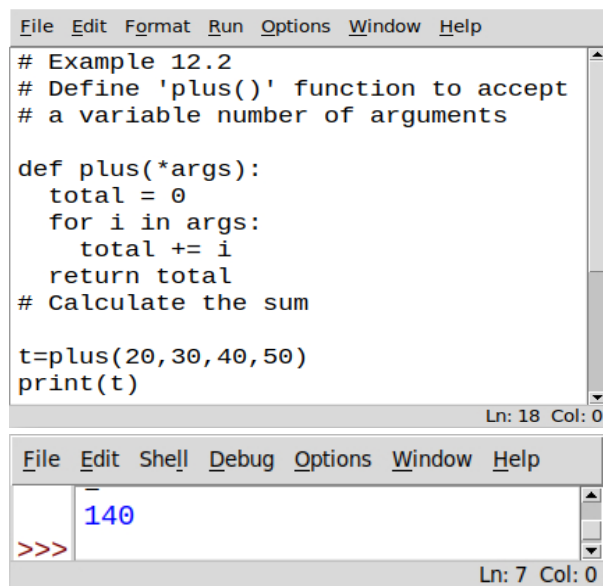
```
def mixedFraction(num+5,deno):
```

```
def mixedFraction(num+5,deno+5):
```

Variable Number of Arguments

In cases where you don't know the exact number of arguments that you want to pass to a function, you can use the following syntax with ***args**:

Example 3.2



```
File Edit Format Run Options Window Help
# Example 12.2
# Define 'plus()' function to accept
# a variable number of arguments

def plus(*args):
    total = 0
    for i in args:
        total += i
    return total
# Calculate the sum

t=plus(20, 30, 40, 50)
print(t)
Ln: 18 Col: 0

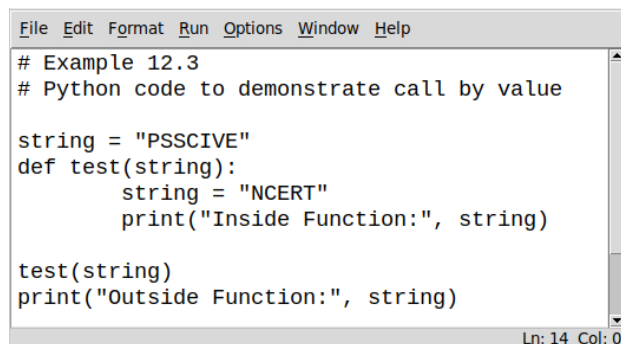
File Edit Shell Debug Options Window Help
140
>>>
Ln: 7 Col: 0
```

Pass-by-object-reference:

Python utilizes a system, which is known as “*Call by Object Reference*” or “*Call by assignment*” to pass arguments to a function. In the event that you pass arguments like whole numbers, strings or tuples to a function, the passing is like call-by-value because you can't change the value of the immutable objects being passed to the function as illustrated in Example 3.2. Whereas passing mutable objects can be considered as call by reference because when their values are changed inside the function, then it will also be reflected outside the function as illustrated in Example 3.3.

Example 3.3

Python code to demonstrate



```
File Edit Format Run Options Window Help
# Example 12.3
# Python code to demonstrate call by value

string = "PSSCIVE"
def test(string):
    string = "NCERT"
    print("Inside Function:", string)

test(string)
print("Outside Function:", string)
Ln: 14 Col: 0
```

```
File Edit Shell Debug Options Window Help
Inside Function: NCERT
Outside Function: PSSCIVE
>>>
Ln: 10 Col: 0
```

Example 3.4

Python code to demonstrate call by reference

```
File Edit Format Run Options Window Help
# Example 12.4
# Python code to demonstrate call by reference

def add_more(list):
    list.append(50)
    print("Inside Function: ", list)
print()
mylist = [10,20,30,40]
add_more(mylist)
print("Outside Function: ", mylist)
Ln: 10 Col: 25
```

```
File Edit Shell Debug Options Window Help
Inside Function: [10, 20, 30, 40, 50]
Outside Function: [10, 20, 30, 40, 50]
>>>
Ln: 16 Col: 0
```

3.3.3 Functions Returning Value

A function may or may not return a value when called. The return statement returns the values from the function. In the examples given so far, the function performs calculations and display result(s). They do not return any value. Such functions are called void functions. But a situation may arise, wherein we need to send value(s) from the function to its calling function. This is done using return statement. The return statement, returns the control to the calling function and return value(s) or None.

```
File Edit Format Run Options Window Help
# Program 12.10. Program to compute exponent using user
# defined function.

def calcpow(number,power):      #function definition
    result = 1
    for i in range(1,power+1):
        result = result * number
    return result
print()
base = int(input("Enter the value for the Base: "))
expo = int(input("Enter the value for the Exponent: "))
answer = calcpow(base,expo)    #function call
print(base,"raised to the power",expo,"is ",answer)
Ln: 20 Col: 0
```

```
File Edit Shell Debug Options Window Help
Enter the value for the Base: 5
Enter the value for the Exponent: 4
5 raised to the power 4 is 625
>>>
Ln: 22 Col: 0
```

The return statement takes zero or more values, separated by commas. Using commas actually returns a single tuple. To return multiple values, use a tuple or list. Returning multiple items separated by commas is equivalent to returning a tuple. Next, we will take examples where a function returns more than one values.

Example 3.5

```
File Edit Format Run Options Window Help
# Example 12.5
def test(x,y):
    return x*2, y*3
a,b=test(5,10)
print(a)
print(b)
Ln: 11 Col: 0
```

```
File Edit Shell Debug Options Window Help
10
30
>>> |
Ln: 27 Col: 0
```

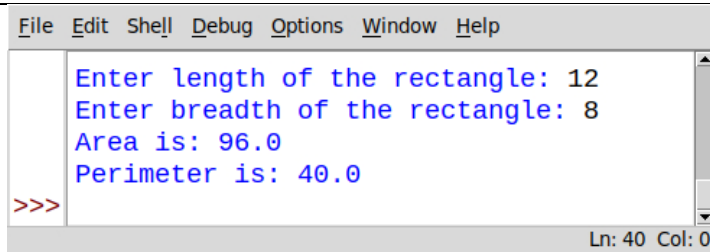
Example 3.6

```
File Edit Format Run Options Window Help
# Example 12.6
def test(x,y):
    return x+y, x-y, x*y
print()
(sum,sub,mult)=test(10,4)
print(sum)
print(sub)
print(mult)
Ln: 13 Col: 0
```

```
File Edit Shell Debug Options Window Help
14
6
40
>>> |
Ln: 33 Col: 0
```

Let us take a Program using function which returns two values area and perimeter of rectangle using tuple.

```
File Edit Format Run Options Window Help
# Program 12.11. Program to calculate area and perimeter
# of rectangle using user defined function
def calcAreaPeri(length,breadth):
    area = length * breadth
    perimeter = 2 * (length + breadth)
    return (area,perimeter)
l = float(input("Enter length of the rectangle: "))
b = float(input("Enter breadth of the rectangle: "))
#value of tuples assigned in order they are returned
area,perimeter = calcAreaPeri(l,b)
print("Area is:",area,"\nPerimeter is:",perimeter)
Ln: 19 Col: 0
```



```
File Edit Shell Debug Options Window Help
Enter length of the rectangle: 12
Enter breadth of the rectangle: 8
Area is: 96.0
Perimeter is: 40.0
>>>
Ln: 40 Col: 0
```

So far, we have learned that a function may or may not have parameter(s) and a function may or may not return any value(s). In Python, as per our requirements, we can have the function in either of the following ways:

- a) Function with no argument and no return value
- b) Function with no argument and with return value(s)
- c) Function with argument(s) and no return value
- d) Function with argument(s) and return value(s)

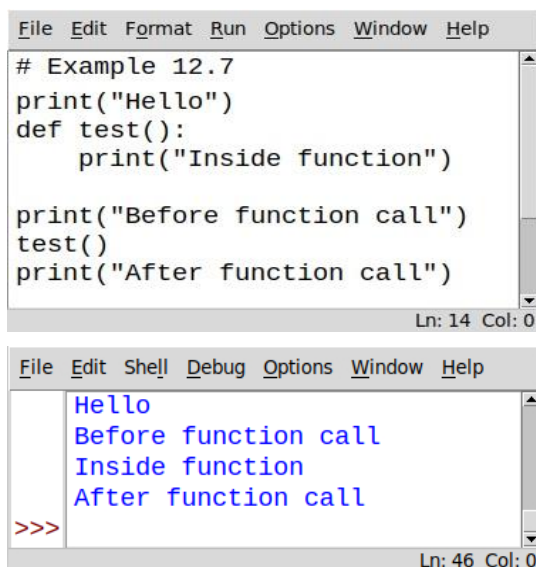
3.3.4 Flow of Execution

Flow of execution can be defined as the order in which the statements in a program are executed. The Python interpreter starts executing the instructions in a program from the first statement. The statements are executed one by one, in the order of appearance from top to bottom.

When the interpreter encounters a function definition, the statements inside the function are not executed until the function is called. Later, when the interpreter encounters a function call, there is a little deviation in the flow of execution. In that case, instead of going to the next statement, the control jumps to the called function and executes the statement of that function. After that, the control comes back to the point of function call so that the remaining statements in the program can be executed. Therefore, when we read a program, we should not simply read from top to bottom. Instead, we should follow the flow of control or execution. It is also important to note that a function must be defined before its call within a program.

Let us take an example to understand the flow of execution in a program where a function is used.

Example 3.7



```
File Edit Format Run Options Window Help
# Example 12.7
print("Hello")
def test():
    print("Inside function")

print("Before function call")
test()
print("After function call")
Ln: 14 Col: 0
```

```
File Edit Shell Debug Options Window Help
Hello
Before function call
Inside function
After function call
>>>
Ln: 46 Col: 0
```

```
File Edit Format Run Options Window Help
# Program 12.12. Program to understand the
# flow of execution using functions.
HelloPython()          #Function Call
def helloPython():     #Function definition
    print("I love Programming")
Ln: 3 Col: 0
```

On executing the above code the following error is produced:

```
File Edit Shell Debug Options Window Help
Traceback (most recent call last):
  File "/usr/lib/python3.10/idlelib/run.py",
  line 578, in runcode
    exec(code, self.locals)
  File "/home/dds/pythonprg/p12.12", line 4,
  in <module>
    HelloPython()          #Function Call
NameError: name 'HelloPython' is not defined
>>>
Ln: 24 Col: 0
```

The error *'function not defined'* is produced even though the function has been defined. When a function call is encountered, the control has to jump to the function definition and execute it. In the above program, since the function call precedes the function definition, the interpreter does not find the function definition and hence an error is raised.

That is why, the function definition should be made before the function call as shown below:

```
def helloPython(): #Function definition
    print("I love Programming")
helloPython()     #Function Call
```

```
[2] def Greetings(Name):#Function Header
[3] print("Hello "+Name)

[1] Greetings("John")    #Function Call
[4] print("Thanks")
```

Fig. 3.4: Order of execution of statements

Figure 3.5 explains the flow of execution for the above program. The number in square brackets shows the order of execution of the statements.

3.4 Scope of a variable

A variable defined inside a function cannot be accessed outside it. Every variable has a well-defined accessibility. The part of the program where a variable is accessible can be defined as the *scope* of that variable. A global variable can be accessed throughout a program and a local variable can only be accessed within a function or block in which it is defined.

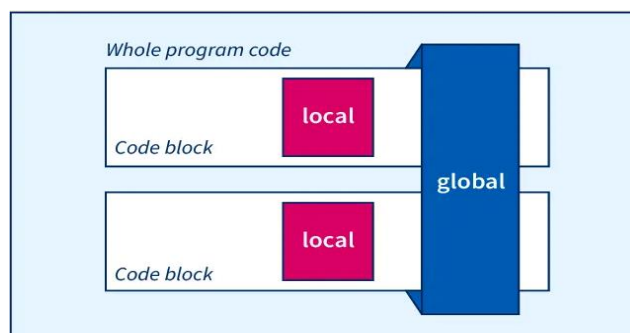
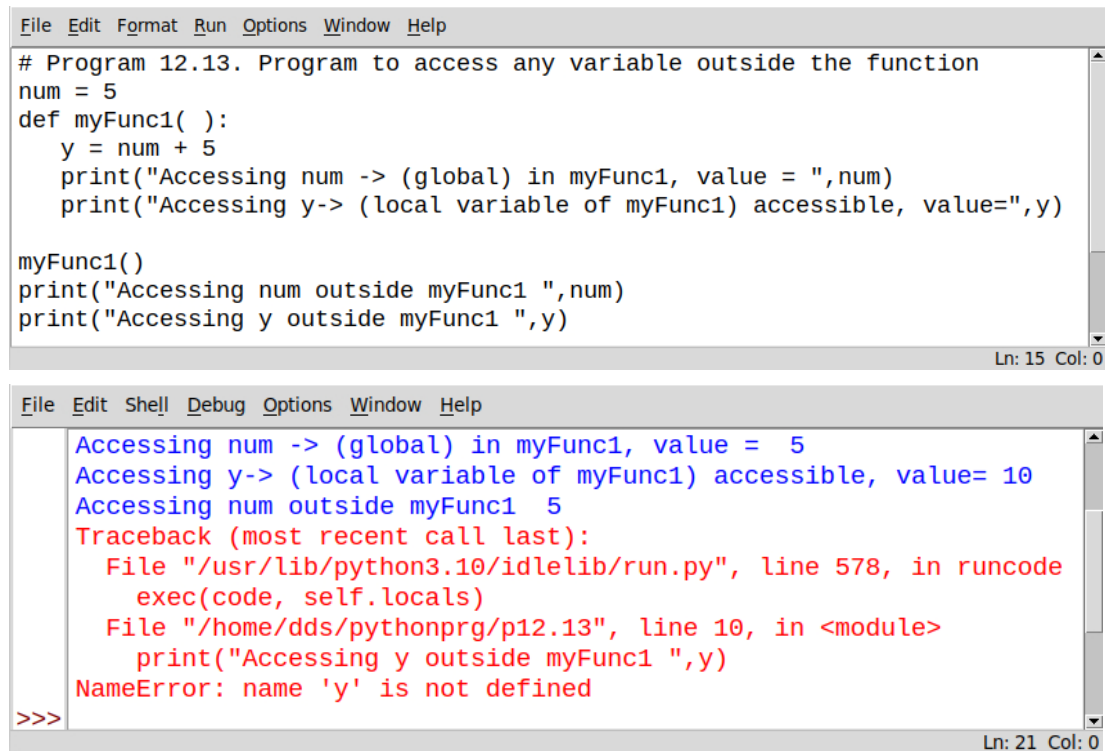


Fig. 3.5: Scope of a variable

Global Variable – In Python, a variable that is defined outside any function or any block is known as a global variable. It can be accessed in any functions defined onward. Any change made to the global variable will impact all the functions in the program where that variable can be accessed.

Local Variable – A variable that is defined inside any function or a block is known as a local variable. It can be accessed only in the function or a block where it is defined. It exists only till the function executes.



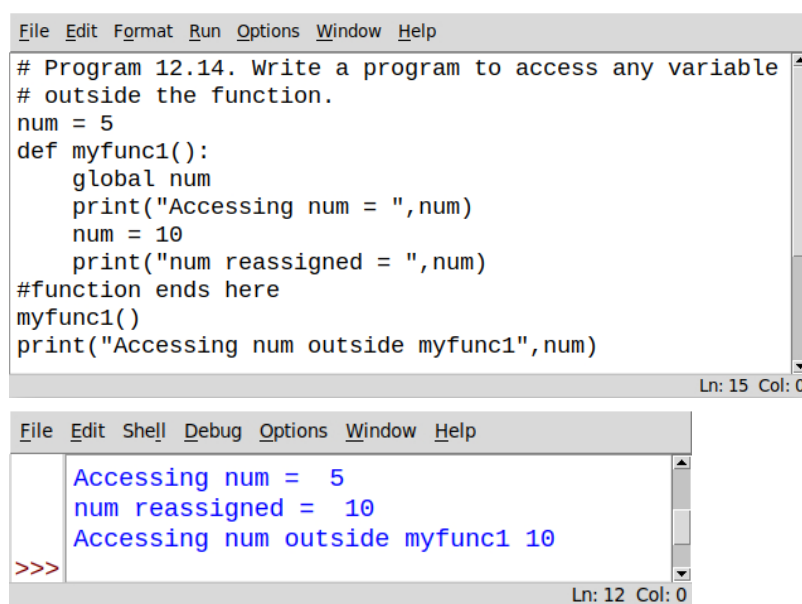
The screenshot shows two windows from a Python IDE. The top window displays the source code for Program 12.13, which defines a global variable 'num' and a function 'myFunc1' that prints the value of 'num' and 'y'. The bottom window shows the execution output, which includes a NameError because the function attempts to print 'y' without defining it locally.

```
File Edit Format Run Options Window Help
# Program 12.13. Program to access any variable outside the function
num = 5
def myFunc1( ):
    y = num + 5
    print("Accessing num -> (global) in myFunc1, value = ",num)
    print("Accessing y-> (local variable of myFunc1) accessible, value=",y)

myFunc1()
print("Accessing num outside myFunc1 ",num)
print("Accessing y outside myFunc1 ",y)
Ln: 15 Col: 0
```

```
File Edit Shell Debug Options Window Help
Accessing num -> (global) in myFunc1, value = 5
Accessing y-> (local variable of myFunc1) accessible, value= 10
Accessing num outside myFunc1 5
Traceback (most recent call last):
  File "/usr/lib/python3.10/idlelib/run.py", line 578, in runcode
    exec(code, self.locals)
  File "/home/dds/pythonprg/p12.13", line 10, in <module>
    print("Accessing y outside myFunc1 ",y)
NameError: name 'y' is not defined
>>>
Ln: 21 Col: 0
```

Any modification to global variable is permanent and affects all the functions where it is used. If a variable with the same name as the global variable is defined inside a function, then it is considered local to that function and hides the global variable. If the modified value of a global variable is to be used outside the function, then the keyword global should be prefixed to the variable name in the function.



The screenshot shows two windows from a Python IDE. The top window displays the source code for Program 12.14, which uses the 'global' keyword to access and modify the global variable 'num' inside a function. The bottom window shows the execution output, which successfully prints the updated value of 'num'.

```
File Edit Format Run Options Window Help
# Program 12.14. Write a program to access any variable
# outside the function.
num = 5
def myfunc1():
    global num
    print("Accessing num = ",num)
    num = 10
    print("num reassigned = ",num)
#function ends here
myfunc1()
print("Accessing num outside myfunc1",num)
Ln: 15 Col: 0
```

```
File Edit Shell Debug Options Window Help
Accessing num = 5
num reassigned = 10
Accessing num outside myfunc1 10
>>>
Ln: 12 Col: 0
```

In above program's output, Global variable **num** is accessed as the ambiguity is resolved by prefixing keyword **global** to it.

Anonymous Functions in Python

Anonymous functions are also called lambda functions in Python because instead of declaring them with the standard **def** keyword, you use the **lambda** keyword.

Example 3.8

```
File Edit Format Run Options Window Help
# Example 12.8
double = lambda x: x*2
y= double(5)
print(y)
Ln: 11 Col: 0

File Edit Shell Debug Options Window Help
10
>>>
Ln: 12 Col: 0
```

The pass Statement in Function definition

Function definitions cannot be empty, but if you for some reason have a function definition with no content, put in the **pass** statement to avoid getting an error.

Def myfunction():

pass

Assignment 3.2

1. Write a program using function that takes a positive integer and returns the one's position digit of the integer.
2. Write a program using function that takes two numbers **n** and returns the number that has minimum one's digit.

3.5 Standard Library

Python has a very extensive standard library. It is a collection of many built-in functions that can be called in the program as and when required, thus saving programmer's time of creating those commonly used functions every time. A function can belong to a standard library if it is a built-in function or in a module otherwise it may be user-defined function as shown in Figure 3.6.

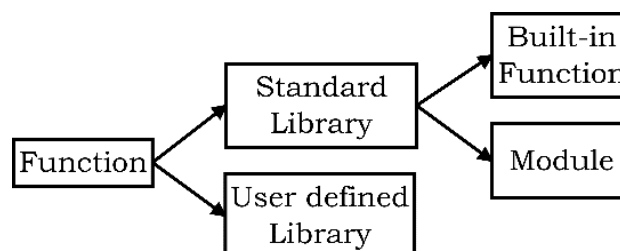


Fig. 3.6: Types of functions

3.5.1 Built-in functions

Built-in functions are the ready-made functions in Python that are frequently used in programs. Let us inspect the following Python program:

```
#Program to calculate square of a number
a = int(input("Enter a number: "))
b = a * a
print(" The square of ", a, "is", b)
```

In the above program `input()`, `int()` and `print()` are the built-in functions. The set of instructions to be executed for these built-in functions are already defined in the Python interpreter.

Let us consider the following Python program consisting of function call to a various built-in function.

```
File Edit Format Run Options Window Help
# Program 12.15. Program to display
# the input using built in function

name = input("Enter your name: ")
print("Hello " + name)
Ln: 11 Col: 0

File Edit Shell Debug Options Window Help
Enter your name: PSSCIVE
Hello PSSCIVE
>>>
Ln: 16 Col: 0
```

In the above program, `input ()` and `print ()` are built-in functions. Here, `print ()` function accepts a string "Enter your name" as argument. The function also returns a value here. Since there is an assignment (=) operator preceding the function name, it means that the function returns a value which is stored in the variable name. Hence, the function `input ()` accepts a value and returns a value. Another built-in function `print ()` is used in the above program to display "Hello" as it is as and value in string name.

Similarly, there are so many built-in functions in Python. For more examples of built-in functions refer to Appendix 1 (Table 3.2) in the end of this chapter.

Assignment 3.3

1. Write a program that reads a number, than converts it into octal and hexadecimal equivalent numbers using built-in functions of Python.
2. Write a program that inputs a real number and converts it to nearest integer using built-in functions. It also displays the given number rounded off to 3 places after decimal.

3.5.2 Module

Other than the built-in functions, the Python standard library also consists of a number of modules. While a function is a grouping of instructions, a module is a grouping of functions. As we know that when a program grows, functions are used to simplify the code and to avoid repetition. For a complex problem, it may not be feasible to manage the code in one single file. Then, the program is divided into different parts under different levels, called modules. Also, suppose we have created some functions in a program and we want to reuse them in another program. In that case, we can save those functions under a module and reuse them. A module is created as a Python (.py) file containing a collection of function definitions.

To use a module, we need to import the module. Once we import a module, we can directly use all the functions of that module. The syntax of import statement is as follows:

```
import modulename1 [, modulename2, ...]
```

This gives us access to all the functions in the module(s). To call a function of a module, the function name should be preceded with the name of the module with a dot(.) as a separator. The syntax is as shown below:

```
modulename.functionname()
```

Built-in Modules

Python library has many built-in modules that are really handy to programmers. Let us explore some commonly used modules and the frequently used functions that are available in those modules – *math*, *random*, *statistics*.

math Module – It contains different types of mathematical functions. Most of the functions in this module return a *float* value. To use a module, it must be imported only once anywhere in the program. So import *math* module it using the following statement.

```
import math
```

Let us take an example of function `gcd()` that is used to find greatest common divisor of two numbers.

```
>>> import math
>>> math.gcd(10,6)
2
```

In function `gcd(x,y)`, both the input parameters *x* and *y* are positive integers. Some of the commonly used functions in *math* module are given in Appendix 1 (Table 3.2) in the end of this chapter.

random Module – This module contains functions that are used for generating random numbers. For using this module, we can import it using the following statement:

```
import random
```

Let us take `random()` function that is used to Random Real Number (float) in the range 0.0 to 1.0.

```
>>> import random
>>> random.random()
0.011035221592589295
```

Some of the commonly used functions in *random* module are given in Appendix 1 (Table 3.3) in the end of this chapter.

statistics Module – This module provides functions for calculating statistics of numeric (Real-value) data. It can be included in the program by using the following statements.

Let us take `mean ()` function that is used to calculate arithmetic mean of given numbers.

```
>>> import statistics
>>> statistics.mean([1,2,3,4,5])
3
```

Some of the commonly used functions in *statistics* module are given in Appendix 1 (Table 3.4) in the end of this chapter.

from Statement

Instead of loading all the functions into memory by importing a module, *from* statement can be used to access only the required functions from a module. It loads only the specified function(s) instead of all the functions in a module. Its syntax is

```
>>> from module_name import function_name [, function_name,...]
```

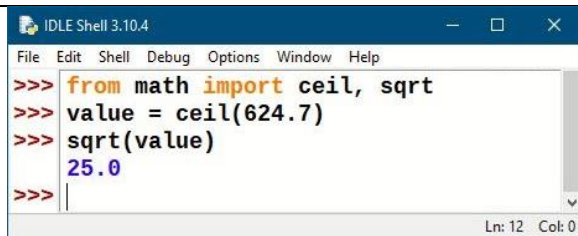
To use the function when imported using "*from* statement" we do not need to precede it with the module name. Rather we can directly call the function as shown in the following examples:

Example 3.9



```
IDLE Shell 3.10.4
File Edit Shell Debug Options Window Help
>>> from random import random
>>> random()
0.729111010781677
Ln: 6 Col: 0
```

Example 3.10



```

IDLE Shell 3.10.4
File Edit Shell Debug Options Window Help
>>> from math import ceil, sqrt
>>> value = ceil(624.7)
>>> sqrt(value)
25.0
>>>
Ln: 12 Col: 0

```

In Example 3.10, the ceil value of 624.7 is stored in the variable "value" and then sqrt function is applied on the variable "value". The above example can be rewritten as:

```
>>> sqrt (ceil (624.7))
```

The execution of the function sqrt () is dependent on the output of ceil () function.

To extract the integer part of 624.7, use trunc () function from math module.

#ceil and sqrt already been imported above

```
>>> from math import trunc
```

```
>>> sqrt ( trunc (625.7))
```

The above code will give result as 25.0.

A programming statement wherein the functions or expressions are dependent on each other's execution for achieving an output is termed as composition, here are some other examples of composition:

```
a = int (input ("First number: "))
```

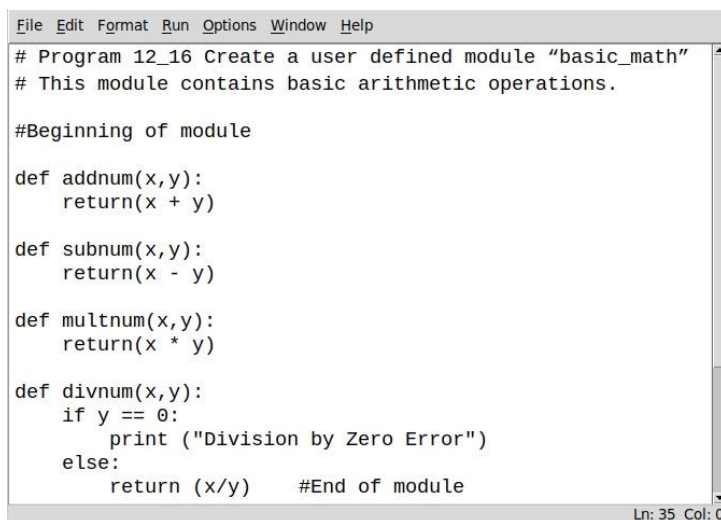
```
print ("Square root of", a, "=", math.sqrt(a))
```

```
print(floor(a+(b/c)))
```

```
math.sin (float(h)/float(c))
```

Importing user defined module

Besides the available modules in Python standard library, we can also create our own module consisting of our own functions. All you need to do is create a file that contains legitimate Python code and then give the file a name with .py extension. To use a module, we need to import the module using import statement. Once we import a module, we can directly use all the functions of that module. Let us learn how to create a module and importing the module to use its function.



```

File Edit Format Run Options Window Help
# Program 12_16 Create a user defined module "basic_math"
# This module contains basic arithmetic operations.

#Beginning of module

def addnum(x,y):
    return(x + y)

def subnum(x,y):
    return(x - y)

def multnum(x,y):
    return(x * y)

def divnum(x,y):
    if y == 0:
        print ("Division by Zero Error")
    else:
        return (x/y) #End of module
Ln: 35 Col: 0

```

Note: Save this module with file name as p12_16.py instead of p3.16 as practiced in naming the file name in this book. Because .(dot) has the special meaning in Python which cannot be used for naming the module which has to be referred in another Python program.

Save the above code in a separate file name as p12_16.py. Now write the following below code in a new file. Save it as p3.17 and then Run to get output.

```

File Edit Format Run Options Window Help
# Program 12.17 Statements for using module p12_16

import p12_16
print()
a = p12_16.addnum(2,5) #Call addnum() function of module p12_16
print(a)

b = p12_16.subnum(2,5) #Call subnum() function of module p12_16
print(b)

c = p12_16.multnum(2,5) #Call multnum() function of module p12_16
print(c)

d = p12_16.divnum(2,5) #Call divnum() function of module p12_16
print(d)
Ln: 22 Col: 0

File Edit Shell Debug Options Window Help
7
-3
10
0.4
>>>
Ln: 12 Col: 0

```

Assignment 3.4

Create a module `length_conversion.py` that stores functions for various lengths conversion like `mile_to_km ()`, `km_to_mile ()`, `feet_to_inches ()` and `inches_to_feet ()`. Write a program to use the above module.

CHECK YOUR PROGRESS**A. Multiple Choice Questions**

1. A named group of instructions that accomplish a specific task when it is invoked is called a (a) string (b) control (c) tuple (d) Function
2. Which keyword is used for function? (a) fun (b) define (c) def (d) function
3. What are the two main types of functions? (a) Custom function (b) Built-in function & User defined function (c) User function (d) System function
4. Which of the following is the use of `id ()` function in Python? (a) Id returns the identity of the object (b) Every object doesn't have a unique id (c) All of the mentioned (d) None of the mentioned
5. Which of the following refers to mathematical function? (a) sqrt (b) rhombus (c) add (d) minus
6. Python supports the creation of anonymous functions at runtime, using a construct called _____ (a) lambda (b) pi (c) anonymous (d) none of the mentioned
7. What will be the output of the following Python code? (a) 12 (b) 14 (c) 27 (d) 64


```

y = 4
z = lambda x: x * y
print (z(3))

```
8. What will be the output of the following Python code? (a) 27, 81, 343 (b) 6, 9, 12 (c) 9, 27, 81 (d) 64, 27,343


```

L = [lambda x: x ** 2, lambda x: x ** 3, lambda x: x ** 4]
for f in L:
    print(f(3))

```

9. Which one of the following is not a function of statistics module? (a) fabs() (b) mean() (c) median() (d) mode()
10. A function by user in one program can be reused in another program. Such function can be stored under (a) string (b) control (c) tuple (d) module

B. State whether True or False

1. Use of function is one of the means to achieve modularity and reusability.
2. Function header always ends with a semi-colon (;).
3. Lambda function contains return statements.
4. Lambda is an anonymous function in Python.
5. Module is a grouping of functions.
6. Once we import a module, we can directly use all the functions of that module.
7. To use the function when imported using "from statement" we do not need to precede it with the module name.
8. It is not possible to create your own function, besides the available modules in Python standard library.
9. randrange () is a function of random module.
10. fmod () is a function of statistics module.

C. Fill in the Blanks

1. The process of dividing a computer program into separate independent blocks of code or separate sub-problems with different names and specific functionalities is known as _____ programming.
2. A function defined to achieve some tasks as per the programmer's requirement is called a _____ function.
3. An _____ is a value passed to the function during the function call which is received in corresponding parameter defined in function header.
4. The _____ statement returns the values from the function.
5. A collection of many built- in functions that can be called in the program as and when required is known as _____.
6. A module is created as a Python file with extension _____ containing a collection of function definitions.
7. To use a module, we need to import the module using _____ statement.
8. To call a function of a module, the function name should be preceded with the name of the module with a _____ as a separator.
9. Functions that are used for generating random numbers are containing in _____ module.
10. Instead of loading all the functions into memory by importing a module, _____ statement can be used to access only the required functions from a module.

D. Programming Questions

1. Identify the errors if any in the following code.
 - (a)

```
def create (text, freq):
    for i in range (1, freq):
        print text
    create (5)           #function call
```
 - (b)

```
from math import sqrt,ceil
def calc ():
    print cos (0)
calc ()                 #function call
```

```
(c) mynum = 9 def add9():
    mynum = mynum + 9
    print mynum
    add9()          #function call
(d) def findValue( vall = 1.1, val2, val3):
    final = (val2 + val3)/ vall
    print(final)
    findvalue ()   #function call
(e) def greet ():
    return ("Good morning")
    greet () = message#function call
```

- Write a program to check the divisibility of a number by 7 that is passed as a parameter to the user defined function.
- Write a program that uses a user defined function that accepts name and gender (as M for Male, F for Female) and prefixes Mr./Ms. on the basis of the gender.
- Write a Program that uses two user defined function to convert temperatures to and from Celsius, Fahrenheit and Fahrenheit to Celsius.
- Write a program that has a user defined function to accept the coefficients of a quadratic equation in variables and calculates its determinant. For example: if the coefficients are stored in the variables a, b, c then calculate determinant as b^2-4ac . Write the appropriate condition to check determinants on positive, zero and negative and output appropriate result.
- ABC School has allotted unique token IDs from (1 to 600) to all the parents for facilitating a lucky draw on the day of their annual day function. The winner would receive a special prize. Write a program using Python that helps to automate the task. (Hint: use random module)
- Write a program that implements a user defined function that accepts Principal Amount, Rate, Time, Number of Times the interest is compounded to calculate and displays compound interest. (Hint: $CI = P(1 + \frac{R}{N})^{NT}$)
- Write a program that has a user defined function to accept 2 numbers as parameters, if number 1 is less than number 2 then numbers are swapped and returned, i.e., number 2 is returned in place of number 1 and number 1 is reformed in place of number 2, otherwise the same order is returned.
- Write a program that contains user defined functions to calculate area, perimeter or surface area whichever is applicable for various shapes like square, rectangle, triangle, circle and cylinder. The user defined functions should accept the values for calculation as parameters and the calculated value should be returned. Import the module and use the appropriate functions.
- Write a program that creates a GK quiz consisting of any five questions of your choice. The questions should be displayed randomly. Create a user defined function score () to calculate the score of the quiz and another user defined function remark (score value) that accepts the final score to display remarks as follows:

Marks	Remarks
5	Outstanding
4	Excellent
3	Good
2	Read more to score more
1	Needs to take interest
0	General knowledge will always help you. Take it seriously.

Session 4. Strings

Venkateshwari was filling her details like Name, Mobile Number and Address in an online application form for registering in a music event. But by mistake, she typed her Mobile number in the field of Name. So, she got a message of invalid input. On asking her father, he explained that Name always consists of alphabets only. Similarly, Mobile number of the applicant will consist of only digits. Further he explained that in computerized system, every field is being validated for proper data input. He elaborated that in computer based online application forms, a data-type validation is used to prevent user to enter invalid inputs. In Python programming, string data type is used data in the form of characters only. If any digit exists in a string, it will also be treated as a character. (Figure 4.1)



Fig. 4.1: Use of strings in our daily life

We have studied that a sequence is an orderly collection of items and each item is indexed by an integer. String, List and Tuple are sequence data types in Python. In this chapter, you will understand the basic concepts of string, string operations, traversing a string, string methods & built-in functions and string as argument to a function.

4.2 Strings

String is a sequence which is made up of one or more UNICODE characters. UNICODE was introduced to include every character in all languages and bring uniformity in encoding. Here the character can be a letter, digit, whitespace or any other symbol. A string can be created by enclosing one or more characters in single, double or triple quote. Let us take examples of creating strings using different type of quotes.

```
>>> str1 = 'Hello World!'
>>> str2 = "Hello World!"
>>> str3 = """Hello World!"""
>>> str4 = '''Hello World!'''
```

str1, str2, str3, str4 are all string variables having the same value 'Hello World!'. Values stored in str3 and str4 can be extended to multiple lines using triple quotes as can be seen in the following example:

```
>>> str3 = """Hello World!
welcome to the world of Python"""
>>> str4 = '''Hello World!
welcome to the world of Python'''
```

4.2.1 Accessing Characters in a String

Each individual character in a string can be accessed using a technique called indexing. The index specifies the character to be accessed in the string and is written in square brackets. The index of the first character (from left) in the string is 0 and the last character is $n-1$ where n is the length of the string. If we give index value out of this range then we get an `IndexError` message. The index must be an integer (positive, zero or negative).

```
#initializes a string str1
>>> str1 = 'Hello World!'
#gives the first character of str1
>>> str1[0]
'H'
#gives seventh character of str1
>>> str1[6]
'W'
#gives last character of str1
>>> str1[11]
'!'
#gives error as index is out of range
>>> str1[15]
IndexError: string index out of range
```

The index can also be an expression including variables and operators but the expression must evaluate to an integer.

```
#an expression resulting in an integer index
#so gives 6 character of str1
>>> str1[2+4]
'W'
#gives error as index must be an integer
>>> str1[1.5]
```

TypeError: string indices must be integers

Python allows an index value to be negative also. Negative indices are used to access the characters of the string from right to left. Starting from right hand side, the first character has the index as -1 and the last character has the index $-n$ where n is the length of the string. Table 4.1 shows the indexing of characters in the string *'Hello World!'* in both the cases, i.e., positive and negative indices.

```
>>> str1[-1] #gives first character from right
'!'
>>> str1[-12] #gives last character from right
'H'
```

Table 4.1 Indexing of characters in string 'Hello World!'

Positive Indices	0	1	2	3	4	5	6	7	8	9	10	11
String	H	e	l	l	o		W	o	r	l	d	!
Negative Indices	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

An in-built function `len()` in Python returns the length of the string that is passed as parameter. For example, the length of string `str1 = 'Hello World!'` is 12.

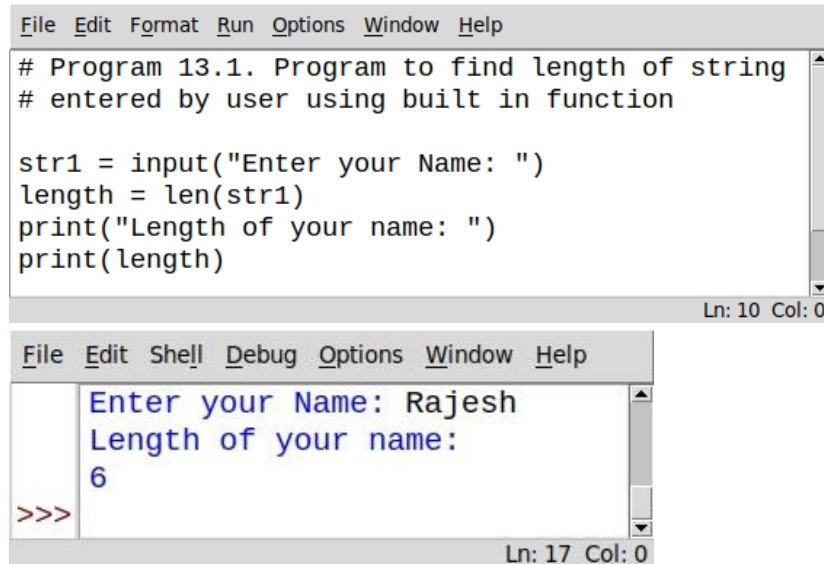
```
#gives the length of the string str1
>>> len(str1)
12
```

```
#length of the string is assigned to n
>>> n = len(str1)
>>> print(n)
12
#gives the last character of the string
>>> str1[n-1]
'!'
#gives the first character of the string
>>> str1[-n]
'H'
```

4.2.2 String is Immutable

A string is an immutable data type. It means that the contents of the string cannot be at the same memory location where it was created. An attempt to do this would lead to an error.

```
>>> str1 = "Hello World!"
#if we try to replace character 'e' with 'a'
>>> str1[1] = 'a'
TypeError: 'str' object does not support item Assignment
```



```
File Edit Format Run Options Window Help
# Program 13.1. Program to find length of string
# entered by user using built in function

str1 = input("Enter your Name: ")
length = len(str1)
print("Length of your name: ")
print(length)
Ln: 10 Col: 0

File Edit Shell Debug Options Window Help
Enter your Name: Rajesh
Length of your name:
6
>>>
Ln: 17 Col: 0
```

In above program, in-built function `len()` is used to find the length of the string `str1`.

Assignment

1. Write a program to find length of the name of the city entered by user. Display its first and last character also.
2. Write a program to display character at index -1 of a given string.
3. Write a program to display character at index (2+3) of string "Python Learners".

4.3 String Operations

As we know that string is a sequence of characters. Python allows certain operations on string data type, such as concatenation, repetition, membership and slicing. These operations are explained in the following subsections with suitable examples.

4.3.1 Concatenation

To concatenate means to join. Python allows to join two strings using concatenation operator plus which is denoted by symbol `+`.

```
>>> str1 = 'Hello'           #First string
>>> str2 = 'World!'        #Second string
>>> str1 + str2            #Concatenated strings
'HelloWorld!'
```


#str1 and str2 remain same after this operation.

```
>>> str1
'Hello'
>>> str2
'World!'
```

4.3.2 Repetition

Python allows to repeat the given string using repetition operator which is denoted by symbol *.

```
#assign string 'Hello' to str1
>>> str1 = 'Hello'
#repeat the value of str1 2 times
>>> str1 * 2
'HelloHello'
#repeat the value of str1 5 times
>>> str1 * 5
'HelloHelloHelloHelloHello'
```

An important point to note here is that str1 still remains the same after the use of repetition operator.

4.3.3 Membership

Python has two membership operators '*in*' and '*not in*'. The '*in*' operator takes two strings and returns True if the first string appears as a sub-string in the second string, otherwise it returns False.

```
>>> str1 = 'Hello World!'
>>> 'W' in str1
True
>>> 'Wor' in str1
True
>>> 'My' in str1
False
```

The '*not in*' operator also takes two strings and returns True if the first string does not appear as a sub-string in the second string, otherwise returns False.

```
>>> str1 = 'Hello World!'
>>> 'My' not in str1
True
>>> 'Hello' not in str1
False
```

4.3.4 Slicing

In Python, to access some part of a string or sub-string, we use a method called slicing. This can be done by specifying an index range. Given a string str1, the slice operation str1[n:m] returns the part of the string str1 starting from index n (inclusive) and ending at m (exclusive). In other words, we can say that str1[n:m] returns all the characters starting from str1[n] till str1[m-1]. The numbers of characters in the sub-string will always be equal to difference of two indices m and n, i.e., (m-n).

```
>>> str1 = 'Hello World!'
#gives substring starting from index 1 to 4
>>> str1[1:5]
'ello'
#gives substring starting from 7 to 9
>>> str1[7:10]
'orl'
#index that is too big is truncated down to
```

#the end of the string

```
>>> str1[3:20]
```

```
'lo World!'
```

#first index > second index results in an

#empty " string

```
>>> str1[7:2]
```

If the first index is not mentioned, the slice starts from index.

#gives substring from index 0 to 4

```
>>> str1[:5]
```

```
'Hello'
```

If the second index is not mentioned, the slicing is done till the length of the string.

#gives substring from index 6 to end

```
>>> str1[6:]
```

```
'World!'
```

The slice operation can also take a third index that specifies the 'step size'. For example, `str1[n:m:k]`, means every `k` character has to be extracted from the string `str1` starting from `n` and ending at `m-1`. By default, the step size is one.

```
>>> str1[0:10:2]
```

```
'HloWr'
```

```
>>> str1[0:10:3]
```

```
'HIWI'
```

Negative indexes can also be used for slicing.

#characters at index -6,-5,-4,-3 and -2 are

#sliced

```
>>> str1[-6:-1]
```

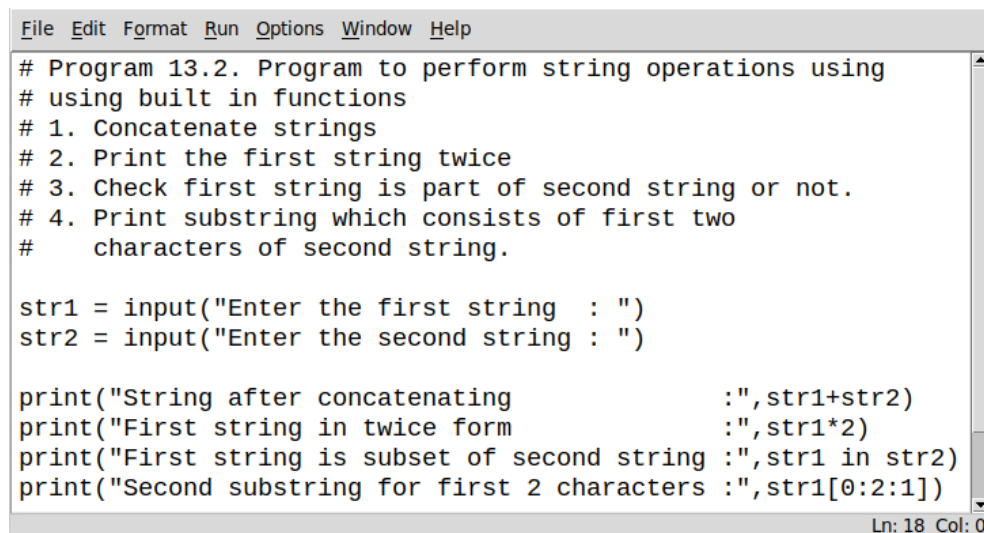
```
'World'
```

If we ignore both the indexes and give step size as -1

#str1 string is obtained in the reverse order

```
>>> str1[::-1]
```

```
'!dlroW olleH'
```



```
File Edit Format Run Options Window Help
# Program 13.2. Program to perform string operations using
# using built in functions
# 1. Concatenate strings
# 2. Print the first string twice
# 3. Check first string is part of second string or not.
# 4. Print substring which consists of first two
# characters of second string.

str1 = input("Enter the first string : ")
str2 = input("Enter the second string : ")

print("String after concatenating          :",str1+str2)
print("First string in twice form         :",str1*2)
print("First string is subset of second string :",str1 in str2)
print("Second substring for first 2 characters :",str1[0:2:1])

Ln: 18 Col: 0
```

```

File Edit Shell Debug Options Window Help
Enter the first string : Him
Enter the second string : Sagar

String after concatenating           : HimSagar
First string in twice form           : HimHim
First string is subset of second string : False
Second substring for first 2 characters : Hi
>>>
Ln: 107 Col: 0

```

In above program, operator + is used to concatenate the strings. Symbol * is used with value 2 to print str1 twice. The membership operator in is used to check str1 is part of str2 or not. The concept of slicing is used to print first two characters of string1.

Assignment

- Write a program to input two strings and do following operations on them
 - Concatenate strings and print them.
 - Print the second string twice.
 - Check second string is part of first string or not.
 - Print sub-string which consists of first two characters of second string.
 - Print first string in reverse order.
- Write a Python program to get a string made of the first 2 and the last 2 characters from a given string. If the string length is less than 2, return instead of the empty string.

4.4 Traversing a String

We can access each character of a string or traverse a string using for loop and while loop.

(a) String Traversal Using for Loop:

```

>>> str1 = 'Hello World!'
>>> for ch in str1:
    print(ch, end = ")

```

Hello World! #output of for loop

In the above code, the loop starts from the first character of the string str1 and automatically ends when the last character is accessed.

(b) String Traversal Using while Loop:

```

>>> str1 = 'Hello World!'
>>> index = 0
#len(): a function to get length of string
>>> while index < len(str1):
    print(str1[index], end = ")
    index += 1

```

Hello World! #output of while loop

Here while loop runs till the condition **index < len(str)** is True, where index varies from 0 to **len(str) - 1**.

4.5 String Methods and Built-In Functions

Python has several built-in functions that allow us to work with strings. Table 4.2 describes some of the commonly used built-in functions for string manipulation.

Table 4.2 Built-in functions for string manipulations

Method	Description	Example
--------	-------------	---------

title()	Returns the string with first letter of every word in the string in uppercase and rest in lowercase.	>>> str1 = 'hello WORLD!' >>> str1.title() 'Hello World!'
lower()	Returns the string with all uppercase letters converted to lowercase	>>> str1 = 'hello WORLD!' >>> str1.lower() 'hello world!'
upper()	Returns the string with all lowercase letters converted to uppercase	>>> str1 = 'hello WORLD!' >>> str1.upper() 'HELLO WORLD!'
count (str,start, end)	Returns number of times substring str occurs in the given string. If we do not give start index and end index then searching starts from index 0 and ends at length of the string	>>> str1 = 'Hello World! Hello Hello' >>> str1.count('Hello',12,25) 2 >>> str1.count('Hello') 3
find(str,start, end)	Returns the first occurrence of index of substring str occurring in the given string. If we do not give start and end then searching starts from index 0 and ends at length of the string. If the substring is not present in the given string, then the function returns -1.	>>> str1 = 'Hello World! Hello Hello' >>> str1.find('Hello',10,20) 13 >>> str1.find('Hello',15,25) 19 >>> str1.find('Hello') 0 >>> str1.find('Hee') -1
index(str, start, end)	Same as find () but raises an exception if the substring is not present in the given string	>>> str1 = 'Hello World! Hello Hello' >>> str1.index('Hello') 0 >>> str1.index('Hee') ValueError: substring not found
endswith()	Returns True if the given string ends with the supplied substring otherwise returns False	>>> str1 = 'Hello World!' >>> str1.endswith('World!') True >>> str1.endswith('!') True >>> str1.endswith('lde') False
startswith()	Returns True if the given string starts with the supplied substring otherwise returns False	>>> str1 = 'Hello World!' >>> str1.startswith('He') True >>> str1.startswith('Hee') False
isalnum()	Returns True if characters of the given string are either alphabets or numeric. If whitespace or special symbols are part of the given string or the string is empty it returns False	>>> str1 = 'HelloWorld' >>> str1.isalnum() True >>> str1 = 'HelloWorld2' >>> str1.isalnum() True >>> str1 = 'HelloWorld!!!'

		<pre>>>> str1.isalnum() False</pre>
islower()	Returns True if the string is non-empty and has all lowercase alphabets, or has at least one character as lowercase alphabet and rest are non-alphabet characters	<pre>>>> str1 = 'hello world!' >>> str1.islower() True >>> str1 = 'hello 1234' >>> str1.islower() True >>> str1 = 'hello ??' >>> str1.islower() True >>> str1 = '1234' >>> str1.islower() False >>> str1 = 'Hello World!' >>> str1.islower() False</pre>
isupper()	Returns True if the string is non-empty and has all uppercase alphabets, or has at least one character as uppercase character and rest are non-alphabet characters	<pre>>>> str1 = 'HELLO WORLD!' >>> str1.isupper() True >>> str1 = 'HELLO 1234' >>> str1.isupper() True >>> str1 = 'HELLO ??' >>> str1.isupper() True >>> str1 = '1234' >>> str1.isupper() False >>> str1 = 'Hello World!' >>> str1.isupper() False</pre>
isspace()	Returns True if the string is non-empty and all characters are white spaces (blank, tab, newline, carriage return)	<pre>>>> str1 = ' \n \t \r' >>> str1.isspace() True >>> str1 = 'Hello\n' >>> str1.isspace() False</pre>
istitle()	Returns True if the string is non-empty and title case, i.e., the first letter of every word in the string in uppercase and rest in lowercase	<pre>>>> str1 = 'Hello World!' >>> str1.istitle() True >>> str1 = 'hello World!' >>> str1.istitle() False</pre>
lstrip()	Returns the string after removing the spaces only on the left of the string	<pre>>>> str1 = 'Hello World! ' >>> str1.lstrip() 'Hello World!'</pre>
rstrip()	Returns the string after removing the spaces only on the right of the string	<pre>>>> str1 = 'Hello World!' >>> str1.rstrip() 'Hello World!'</pre>
strip()	Returns the string after removing the spaces both on the left and the right of the string	<pre>>>> str1 = 'Hello World!' >>> str1.strip() 'Hello World!'</pre>
Replace	Replaces all occurrences of old	<pre>>>> str1 = 'Hello World!'</pre>

(oldstr, newstr)	string with the new string	<pre>>>> str1.replace('o','*') 'Hell* W*rd!' >>> str1 = 'Hello World!' >>> str1.replace('World','Country') 'Hello Country!' >>> str1 = 'Hello World! Hello' >>> str1.replace('Hello','Bye') 'Bye World! Bye'</pre>
join()	Returns a string in which the characters in the string have been joined by a separator	<pre>>>> str1 = ('HelloWorld!') >>> str2 = '-' #separator >>> str2.join(str1) 'H-e-l-l-o-W-o-r-l-d-!'</pre>
partition()	Partitions the given string at the first occurrence of the substring (separator) and returns the string partitioned into three parts. 1. Substring before the separator 2. Separator 3. Substring after the separator If the separator is not found in the string, it returns the whole string itself and two empty strings	<pre>>>> str1 = 'India is a Great Country' >>> str1.partition('is') ('India ', 'is', ' a Great Country') >>> str1.partition('are') ('India is a Great Country', '', '')</pre>
split()	Returns a list of words delimited by the specified substring. If no delimiter is given then words are separated by space.	<pre>>>> str1 = 'India is a Great Country' >>> str1.split() ['India','is','a','Great', 'Country'] >>> str1 = 'India is a Great Country' >>> str1.split('a') ['Indi', ' is ', ' Gre', 't Country']</pre>
capitalize()	Return a copy of the string with its first character capitalized and the rest lowercased.	<pre>>>>str3="Bhopal is My favorite city" >>> str3.capitalize() 'Bhopal is my favorite city'</pre>
swapcase()	Return a copy of the string with uppercase characters converted to lowercase and vice versa.	<pre>>>> str1="Bhopal" >>> str1.swapcase() 'bHOPAL'</pre>

Let us take a program to demonstrate the use of built-in functions for string manipulations.

```
File Edit Format Run Options Window Help
# Program 13.3. Program to perform following
# String manipulation using built in functions
# 1. Print the string in uppercase
# 2. Print the string in lowercase
# 3. Replace all occurrences of a character "a" '*'
# 4. Print the string in reverse case
str1 = input("Enter your First Name: ")
str2 = input("Enter your Last Name : ")
print("String in upper case      : ",str1.upper())
print("String in lower case     : ",str2.lower())
print("String after replacement : ",str1.replace('a','*'))
print("String in reverse case   : ",str2.swapcase())
Ln: 18 Col: 0
```

```
File Edit Shell Debug Options Window Help
Enter your First Name: Vibha
Enter your Last Name : Anshu

String in upper case      : VIBHA
String in lower case     : anshu
String after replacement : Vibh*
String in reverse case   : aNSHU
>>>
Ln: 131 Col: 0
```

In above program, upper () function is used to print the string str1 in uppercase letters. The function lower () is used to print the string str2 in lowercase letters. The function replace () is used to replace all occurrences of 'a' with '*' in str1. The function swapcase() is used to print the string str2 with uppercase characters converted to lowercase and vice versa.

Assignment

Write a program to input two strings (str1 and str2) and do following manipulations on them

- (i) Print the string str1 with all uppercase letters converted to lowercase.
- (ii) Print the string str2 with all lowercase letters converted to uppercase.
- (iii) Replace all occurrences of 'a' with '*' in str1.

4.6 String as argument to function

A string can be used as argument to a function. Let us take some programs consisting user defined functions in Python to perform different operations on strings.

In below program, an user-defined function charCount() takes a character ch that has to be searched and a string as an argument. Now, each character of the string is compared with the given character ch using if structure. If character is matched, a counter variable is increased by 1. Final value of counter gives the number of occurrences of character ch in the given string.

```
File Edit Format Run Options Window Help
# Program 13.4. Program to find the number of occurrences of a
# character in a string using user defined function

def charCount(ch,st):
    count = 0
    for character in st:
        if character == ch:
            count += 1
    return count
#end of function
print()
st = input("Enter the input string      : ")
ch = input("Enter the search character : ")
count = charCount(ch,st)
print()
print("Number of occurances of",ch,"in the string is:",count)
Ln: 19 Col: 0
```

```
File Edit Shell Debug Options Window Help
Enter the input string      : Python Programming
Enter the search character : P

Number of occurances of P in the string is: 2
>>>
Ln: 175 Col: 0
```

```

File Edit Format Run Options Window Help
# Program 13.5. Program to replace all vowels in
# the string with '*' using user defined function

def replaceVowel(st):
    newstr = ''      # create an empty string
    for character in st:
        # check if next character is a vowel
        if character in 'aeiouAEIOU':
            newstr += '*'  # Replace vowel with *
        else:
            newstr += character
    return newstr
# end of function
print()
st = input("Enter a String: ")
st1 = replaceVowel(st)
print("The original String is:",st)
print("The modified String is:",st1)
Ln: 21 Col: 0

```

```

File Edit Shell Debug Options Window Help
Enter a String: AUTHENTICATION
The original String is: AUTHENTICATION
The modified String is: **TH*NT*C*T**N
>>>
Ln: 207 Col: 0

```

In above program, an user-defined function **replaceVowel()** takes a string as an argument and check each character of that string is vowel or not. If the character is vowel, this character of the string is replaced with '*'. This is a simple and easy example of passing a string as argument to a function.

```

File Edit Format Run Options Window Help
# Program 13.6. Program to print the string
# in reverse order without creating a new string

st = input("Enter a string: ")
print("String in reverse order is as under:")
for i in range(-1,-len(st)-1,-1):
    print(st[i],end='')
Ln: 9 Col: 0

```

```

File Edit Shell Debug Options Window Help
Enter a string: Python Programming
String in reverse order is as under:
gnimmargorP nohtyP
>>>
Ln: 229 Col: 0

```

In the above program, the string is printed in reverse order without using a user defined function.


```

File Edit Format Run Options Window Help
# Program 13.7. Program to reverse the string
# passed as argument and store into new
# string using user defined function

#Function to reverse a string
def reverseString(st):
    newstr = '' #create a new string
    length = len(st)
    for i in range(-1,-length-1,-1):
        newstr += st[i]
    return newstr
#end of function

st = input("Enter a String: ")
st1 = reverseString(st)
print("The original String is:",st)
print("The reversed String is:",st1)
Ln: 21 Col: 0

```

```

File Edit Shell Debug Options Window Help
Enter a String: PSSCIVE BHOPAL
The original String is: PSSCIVE BHOPAL
The reversed String is: LAPOHB EVICSSP
>>>
Ln: 234 Col: 0

```

In the above program, the string is printed in reverse order with the help of a using a user defined function.

```

File Edit Format Run Options Window Help
# program 13.8. Program using to check the string
# is a palindrome or not using user defined function

def checkPalin(st):
    i = 0
    j = len(st) - 1
    while(i <= j):
        if(st[i] != st[j]):
            return False
        i += 1
        j -= 1
    return True
#end of function
print()
st = input("Enter a String: ")
result = checkPalin(st)
if result == True:
    print("The given string",st,"is a palindrome")
else:
    print("The given string",st,"is not a palindrome")
Ln: 24 Col: 0

```

```

File Edit Shell Debug Options Window Help
Enter a String: madam
The given string madam is a palindrome
>>>
Ln: 248 Col: 0

```

In above program, an user-defined function checkPalin() takes a string as an argument. This function compares first character with last character of the string, second character is compared with second last character and so on for the whole string. If at the end all the characters from first is same as characters from last it means entered string is a palindrome otherwise not.

CHECK YOUR PROGRESS

A. Multiple Choice Questions

- The index of the first character (from left) in the string of (a) 0 (b) 1 (c) n-1 (d) n
- Which type of quotes are used for multi-line string? (a) Single (b) Double (c) Triple (d) None of the above
- Which data type is suitable to store a sequence of characters (a) String (b) List (c) Tuple (d) Dictionary
- What will be the output of Python code `str1="10/4"`? (a) 2 (b) '10/4' (c) 2.5 (d) str1
- What will be the output of below Python code? (a) oca (b) ocat (c) cat (d) cati

```
str1="Vocational"
print(str1[2:5])
```
- Which of the following is not a legal string operator? (a) in (b)+ (c) * (d)/
- Which of the following functions will return the total number of characters in a string? (a) count () (b) index () (c) len() (d) all of these
- Which of the following code will return the last two characters of a string str? (a) str [2:] (b) str [:2] (c) str [-2:] (d) str [: -2]
- Which of the following functions will return the string in all caps? (a) upper () (b) toupper() (c) isupper() (d) to-upper()
- Which of the following functions will return a list containing all words of the string? (a) find () (b) index () (c) partition () (d) split ()
- Which of the following functions will always return a tuple of 3 elements? (a) find() (b) index() (c) partition() (d) split()
- What is the output of the following code? (a) False True (b) False False (c) True False (d) True True

```
str1 = "Bhopal 462002"
str2 = "462002"
print (str1.isdigit(), str2.isdigit())
```
- Which method should be used to convert String *"Python programming is fun"* to *"Python Programming Is Fun"*? (a) capitalize () (b) title () (c) istitle() (d) upper()
- What will be the output of the following String operations (a) BhopalBhopal (b) TypeError: unsupported operand type(s) for *: 'str' and 'int' (c) BBhhooppaall (d) Bhopal2

```
str1 = 'Bhopal'
print(str1*2)
```
- Select the correct output of the following String operations. (a) 'I am from india.' (b) 'i Am From India.' (c) 'I Am From India.' (d) TypeError: unsupported operand type(s) for * : 'str' and 'int'

```
str = "i am from India."
print(str.capitalize())
```
- Which of the following functions will raise an error if the given substring is not found in the string? (a) find() (b) index() (c) replace() (d) all of these
- Which of the following code will return the string s in reverse order? (a) s[: : 1] (b) s[: : -1] (c) s[: : 0] (d) str[: -2]
- Which of the following arithmetic operators cannot be used with strings in Python? (a) + (b) * (c) - (d) not in

B. State whether True or False

1. Strings have both positive and negative indexes.
2. In Python a single character is treated as strings of length one.
3. Strings are immutable in Python, which means a string cannot be modified.
4. Like '+', all other arithmetic operators are also supported by strings.
5. Functions capitalize() and title() return the same result.
6. Functions partition() and split() work identically.
7. The find() and index() are similar functions.
8. The find() does not raise an exception if the substring is not found.
9. The split() returns always a 3-element list.
10. A string can be used as argument to a function.

C. Fill in the Blanks

1. The string indexes begin ____ onwards.
2. For strings, _____ operator performs concatenation.
3. For strings, _____ operator performs replication.
4. The in and _____ in are membership operators for strings.
5. If a string contains letters and digits, function _____ will return true.
6. 'ab'.isalpha() will return value as _____.
7. To get each word's first letter capitalized, _____ function is used.
8. Function _____ divides a line of text into individual words.
9. Function _____ returns a copy of the string with uppercase characters converted to lowercase and vice versa.
10. s[: :-1] will result the string s in _____ order.

D. Programming Questions

1. The string mySubject = "Computer Science". What will be the output of the following string operations.
 - a) print(mySubject[0:len(mySubject)])
 - b) print(mySubject[-7:-1])
 - c) print(mySubject[::2])
 - d) print(mySubject[len(mySubject)-1])
 - e) print(2*mySubject)
 - f) print(mySubject[:::-2])
 - g) print(mySubject[:3] + mySubject[3:])
 - h) print(mySubject.swapcase())
 - i) print(mySubject.startswith('Comp'))
 - j) print(mySubject.isalpha())
2. Consider the string myAddress = "WZ-1, New Ganga Nagar, New Delhi". What will be the output of following string operations?
 - a) print(myAddress.lower())
 - b) print(myAddress.upper())
 - c) print(myAddress.count('New'))
 - d) print(myAddress.find('New'))
 - e) print(myAddress.rfind('New'))
 - f) print(myAddress.split(','))
 - g) print(myAddress.split(' '))
 - h) print(myAddress.replace('New','Old'))
 - i) print(myAddress.partition(','))

```
j) print(myAddress.index('Agra'))
```

3. Write a program to input line(s) of text from the user until enter is pressed. Count the total number of characters in the text including white spaces, total number of alphabets, total number of digits, total number of special symbols and total number of words in the given text.
4. Write a user defined function to convert a string with more than one word into title case string where string is passed as parameter.
5. Write a function deleteChar() which takes two parameters, one is a string and other is a character. The function should create a new string after deleting all occurrences of the character from the string and return the new string.
6. Input a string having some digits. Write a function to return the sum of digits present in this string.
7. Write a function that takes a sentence as an input parameter where each word in the sentence is separated by a space. The function should replace each blank with a hyphen and then return the modified sentence.

Session 5. Lists

Taleem has to fetch few things of daily use from market. He made a list of items to be purchased and went to nearest local market. When he reached supermarket, he got a call from his mother and told him to buy few more items. Taleem added new items in the list and strikes down the items to be skipped. Generally, we use these type lists to record the details about different items and we can also change in items at any time in lists. Similar type of facility is available in Python in the form of **list data type**. (Figure 5.1)



Fig. 5.1 Shopping List

Such shopping list is prepared on a piece of paper, which makes it easier to use. In the same way, when we make a program that works with many separate but similar items, it is possible to make a bunch of variables. However, it's often far easier to make a list.

In this chapter you will understand the basic concepts of list data type including list operations, list methods and built-in functions, nested lists, copying lists, list as argument to a function and list manipulation.

5.1 List

The data type list is an ordered sequence which is mutable and made up of one or more elements. Unlike a string which consists of only characters, a list can have elements of different data types, such as integer, float, string, tuple or even another list. A list is very useful to group together elements of mixed data types. Elements of a list are enclosed in square brackets and are separated by comma. Let us take some examples of lists in Python programming.

#list1 is the list of five numbers

```
>>> list1 = [10,20,30,40,50]
```

```
>>> print(list1)
[10,20,30,40,50]
#list2 is the list of vowels
>>> list2 = ['a', 'e', 'i', 'o', 'u']
>>> print(list2)
['a', 'e', 'i', 'o', 'u']
#list3 is the list of mixed data types
>>> list3 = [100, 23.5, 'Hello']
>>> print(list3)
[100, 23.5, 'Hello']
#list4 is the list of lists called nested #list
>>> list4 =[['Physics',101], ['Chemistry',202], ['Maths',303]]
>>> print(list4)
[['Physics', 101], ['Chemistry', 202], ['Maths', 303]]
```

5.1.1 Accessing Elements in a List

The elements of list are accessed in the same way as characters are accessed in string. Like string indices, list indices also start from 0. Let us take some examples of accessing element of a list using its index.

```
#initializes a list list1
>>> list1 = [2,4,6,8,10,12]
>>> list1[0] #return first element of list1
2
>>> list1[3] #return fourth element of list1
8
#return error as index is out of range
>>> list1[15]
IndexError: list index out of range
#an expression resulting in an integer index
>>> list1[1+4]
12
>>> list1[-1] #return first element from right
12
#length of the list list1 is assigned to n
>>> n = len(list1)
>>> print(n)
6
#return the last element of the list1
>>> list1[n-1]
12
#return the first element of list1
>>> list1[-n]
2
```

5.1.2 Lists are Mutable

In Python, lists are mutable. It means that the contents of the list can be changed after it has been created. Let us take an example of changing a particular element of a list in Python programming.

```
#List list1 of colors
>>> list1 = ['Red', 'Green', 'Blue', 'Orange']
#change/override the fourth element of list1
>>> list1[3] = 'Black'
>>> list1 #print the modified list list1
```

```
['Red', 'Green', 'Blue', 'Black']
```

5.2 List Operations

The data type list allows manipulation of its contents through various operations as shown below.

5.2.1 Concatenation

Python allows us to join two or more lists using concatenation operator depicted by the symbol +.

```
#list1 is list of first five odd integers
>>> list1 = [1,3,5,7,9]
#list2 is list of first five even integers
>>> list2 = [2,4,6,8,10]
#elements of list1 followed by list2
>>> list1 + list2
[1, 3, 5, 7, 9, 2, 4, 6, 8, 10]
>>> list3 = ['Red', 'Green', 'Blue']
>>> list4 = ['Cyan', 'Magenta', 'Yellow', 'Black']
>>> list3 + list4
['Red', 'Green', 'Blue', 'Cyan', 'Magenta', 'Yellow', 'Black']
```

Note that, there is no change in ongoing lists, i.e., **list1**, **list2**, **list3**, **list4** remain the same after concatenation operation. To merge these two lists, use an assignment statement to assign the merged list to another new list. The concatenation operator '+' requires that the operands should be of list type only. Concatenating a list with elements of some other data type will give TypeError.

```
>>> list1 = [1,2,3]
>>> str1 = "abc"
>>> list1 + str1
```

TypeError: can only concatenate list (not "str") to list

5.2.2 Repetition

Python allows to replicate a list using repetition operator depicted by symbol *.

```
>>> list1 = ['Hello']
#elements of list1 repeated 4 times
>>> list1 * 4
['Hello', 'Hello', 'Hello', 'Hello']
```

5.2.3 Membership

Like strings, the membership operators "in" checks if the element is present in the list and returns True, else returns False.

```
>>> list1 = ['Red', 'Green', 'Blue']
>>> 'Green' in list1
True
>>> 'Cyan' in list1
False
```

The not in operator returns True if the element is not present in the list, else it returns False.

```
>>> list1 = ['Red', 'Green', 'Blue']
>>> 'Cyan' not in list1
True
>>> 'Green' not in list1
False
```

5.2.4 Slicing

Like strings, the slicing operation can also be applied to lists.

```
>>> list1 = ['Red', 'Green', 'Blue', 'Cyan', 'Magenta', 'Yellow', 'Black']
```

```

>>> list1[2:6]
['Blue', 'Cyan', 'Magenta', 'Yellow']
#list1 is truncated to the end of the list
>>> list1[2:20] #second index is out of range
['Blue', 'Cyan', 'Magenta', 'Yellow', 'Black']
#First index > second index so it results in an empty list.
>>> list1[7:2]
[]
#return sublist from index 0 to 4
>>> list1[:5] #first index missing
['Red', 'Green', 'Blue', 'Cyan', 'Magenta']
#slicing with a given step size
>>> list1[0:6:2]
['Red', 'Blue', 'Magenta']
#negative indexes
#elements at index -6,-5,-4,-3 are sliced
>>> list1[-6:-2]
['Green', 'Blue', 'Cyan', 'Magenta']
#both first and last index missing
>>> list1[::2] #step size 2 on entire list ['Red', 'Blue', 'Magenta', 'Black']
#negative step size to print the list in reverse order
>>> list1[::-1]
['Black', 'Yellow', 'Magenta', 'Cyan', 'Blue', 'Green', 'Red']

```

5.3 Traversing a List

We can access each element of the list or traverse a list using for loop or while loop.

(A) List Traversal Using for Loop:

```

>>> list1 = ['Red', 'Green', 'Blue', 'Yellow', 'Black']
>>> for item in list1:
    print(item)

```

Output:

```

Red
Green
Blue
Yellow
Black

```

Another way of accessing the elements of the list is using range() and len() functions:

```

>>> for i in range(len(list1)):
    print(list1[i])

```

Output:

```

Red
Green
Blue
Yellow
Black

```

(B) List Traversal Using while Loop:

```

>>> list1 = ['Red', 'Green', 'Blue', 'Yellow', 'Black']
>>> i = 0
>>> while i < len(list1):
    print(list1[i])
    i += 1

```

Output:

Red
Green
Blue
Yellow
Black

```

File Edit Format Run Options Window Help
# Program 14.1. Program to do the following operations on list
# 1. Print first element of the list1.
# 2. Concatenate list1 and list2
# 3. Print the list2 thrice
# 4. Print last 2 elements of list1
.
list1 = ["A", "B", "C"]
list2 = [1,2,3]
print("First element of list 1 is      :",list1[0])
print("Concatenation of both list      :", list1+list2)
print("List 2 in trice form            :", list2 * 3)
n=len(list1)
print("Last 2 elements of lists 2 are :",list1[n:n-3:-1])
Ln: 16 Col: 0

File Edit Shell Debug Options Window Help
First element of list 1 is      : A
Concatenation of both list      : ['A', 'B', 'C', 1, 2, 3]
List 2 in trice form           : [1, 2, 3, 1, 2, 3, 1, 2, 3]
Last 2 elements of lists 2 are : ['C', 'B']
>>>
Ln: 21 Col: 0

```

In above program, index 0 is used to access the first element of list1. Operator + is used to concatenate the lists. Symbol * is used with value 3 to print list2 thrice and the concept of slicing is used to print last two elements of list1.

Assignment

1. Write a Python program to sum all the items in a list.
2. Write a Python program to print the average of list item.
3. Write a Python program to get the largest and smallest number from a list.
4. Write a Python program to print the difference between maximum and minimum number of a list.

5.4 List Methods and Built-In Functions

The data type list has several built-in methods that are useful in programming. Some of them are listed in Table 5.1.

Table 5.1 Built-in functions for list manipulations

Method	Description	Example
len()	Returns the length of the list passed as	>>> list1 = [10,20,30,40,50] >>> len(list1) 5
list()	Creates an empty list if no argument is passed. Creates a list if a sequence is passed as an argument	>>> list1 = list() >>> list1 [] >>> str1 = 'aeiou' >>> list1 = list(str1) >>> list1 ['a', 'e', 'i', 'o', 'u']

append()	Appends a single element passed as an argument at the end of the list. The single element can also be a list.	<pre>>>> list1 = [10,20,30,40] >>> list1.append(50) >>> list1 [10, 20, 30, 40, 50] >>> list1 = [10,20,30,40] >>> list1.append([50,60]) >>> list1 [10, 20, 30, 40, [50, 60]]</pre>
extend()	Appends each element of the list passed as argument to the end of the given list	<pre>>>> list1 = [10,20,30] >>> list2 = [40,50] >>> list1.extend(list2) >>> list1 [10, 20, 30, 40, 50]</pre>
insert()	Inserts an element at a particular index	<pre>>>> list1 = [10,20,30,40,50] >>> list1.insert(2,25) >>> list1 [10, 20, 25, 30, 40, 50] >>> list1.insert(0,5) >>> list1 [5, 10, 20, 25, 30, 40, 50]</pre>
count()	Returns the number of times a given element occurs in list	<pre>>>> list1 = [10,20,30,10,40,10] >>> list1.count(10) 3 >>> list1.count(90) 0</pre>
index()	Returns index of the first occurrence of	<pre>>>> list1 = [10,20,30,20,40] >>> list1.index(20) 1 >>> list1.index(90) ValueError: 90 is not in list</pre>
remove()	Removes the given element from the list. If the element is present multiple times, only the first occurrence is removed. If the element is not present, then ValueError is generated.	<pre>>>> list1 = [10,20,30,40,50,30] >>> list1.remove(30) >>> list1 [10, 20, 40, 50, 30] >>> list1.remove(90) ValueError: list.remove(x): x not in list</pre>
pop()	Returns the element whose index is passed as parameter to this function and also removes it from the list. If no parameter is given, then it returns and removes the last element of the list.	<pre>>>> list1 = [10,20,30,40,50,60] >>> list1.pop(3) 40 >>> list1 [10, 20, 30, 50, 60] >>> list1 = [10,20,30,40,50,60] >>> list1.pop() 60</pre>

reverse()	Reverses the order of elements in the given list	<pre>>>> list1 = [34,66,12,89,28,99] >>> list1.reverse() >>> list1 [99, 28, 89, 12, 66, 34] >>> list1 = ['Tiger', 'Zebra', 'Lion', 'Cat' , 'Elephant', 'Dog'] >>> list1.reverse() >>> list1 ['Dog', 'Elephant', 'Cat', 'Lion', 'Zebra', 'Tiger']</pre>
sort()	Sorts the elements of the given list in-place	<pre>>>> list1 = ['Tiger', 'Zebra', 'Lion', 'Cat' , 'Elephant', 'Dog'] >>> list1.sort() >>> list1 ['Cat', 'Dog', 'Elephant', 'Lion', 'Tiger', 'Zebra'] >>> list1 = [34, 66, 12, 89, 28, 99] >>> list1.sort(reverse = True) >>> list1 99, 89, 66, 34, 28, 12]</pre>
sorted()	It takes a list as parameter and creates a new list consisting of the same elements arranged in sorted order.	<pre>>>> list1=[23, 45, 11, 67, 85, 56] >>> list2 = sorted(list1) >>> list1 [23, 45, 11, 67, 85, 56] >>> list2 [11, 23, 45, 56, 67, 85]</pre>
min()	Returns minimum or smallest element of the list	<pre>>>> list1= [34, 12, 63, 39, 92, 44] >>> min(list1) 12</pre>
max()	Returns maximum or largest element of the list	<pre>>>> list1 = [34, 12, 63, 39, 92, 44] >>> max(list1) 92</pre>
sum()	Returns sum of the elements of the list	<pre>>>> list1 = [34, 12, 63, 39, 92, 44] >>> sum(list1) 284</pre>

File Edit Format Run Options Window Help

```
#Program 14.2. Program to do following manipulations on list.
# 1. Insert an element at second index in the list1
# 2. Append each element of the list2 passed as argument to the end of the list1.
# 3. Sort the elements of the list1.
# 4. Print minimum value among elements of list2.
|
list1 = ["A", "B", "C"]
list2 = [1,3,2]
list1.insert(2,"D")
print("List1 after inserting items at index no. 2:",list1)

list1.extend(list2)
print("List 1 after exetending list 2          :",list1)

list2.sort()
print("List 2 aftre sorting                    :",list2)

print("Minimum of list 2 is                    :",min(list2))
```

Ln: 6 Col: 0

```
File Edit Shell Debug Options Window Help
List1 after inserting items at index no. 2: ['A', 'B', 'D', 'C']
List 1 after extending list 2           : ['A', 'B', 'D', 'C', 1, 3, 2]
List 2 after sorting                    : [1, 2, 3]
Minimum of list 2 is                   : 1
>>>
```

In above program, insert() function is used to insert an element 'D' at second index in the list list1. The function extend() is used to append each element of the list2 passed as argument to the end of the list1. The function sort () is used to sort the elements of the list1. The function min() is used to find the minimum value among elements of list2.

Assignment

Write a program to input two lists list1 and list2 and do following manipulations on them using built-in functions

1. Append a single element passed as an argument at the end of the list1.
2. Appends each element of the list2 passed as argument to the end of the list1.
3. Find the number of times a given element occurs in list2.
4. Removes the last element from list2.

5.5 Nested Lists

When a list appears as an element of another list, it is called a nested list.

Example 5.2

```
>>> list1 = [1, 2, 'a', 'c', [6, 7, 8], 4, 9]
#fifth element of list is also a list
>>> list1[4]
[6, 7, 8]
```

To access the element of the nested list of list1, you have to specify two indices list1[i][j]. The first index i will take us to the desired nested list and second index j will take us to the desired element in that nested list.

```
>>> list1[4][1] 7
#index i gives the fifth element of list1 #which is a list
#index j gives the second element in the #nested list
```

5.6 Copying Lists

Given a list, the simplest way to make a copy of the list is to assign it to another list.

```
>>> list1 = [1, 2, 3]
>>> list2 = list1
>>> list1
[1, 2, 3]
>>> list2
[1, 2, 3]
```

The statement list2 = list1 does not create a new list. Rather, it just makes list1 and list2 refer to the same list object. Here list2 actually becomes an alias of list1. Therefore, any changes made to either of them will be reflected in the other list.

```
>>> list1.append(10)
>>> list1
[1, 2, 3, 10]
>>> list2
[1, 2, 3, 10]
```

We can also create a copy or clone of the list as a distinct object by three methods. The first method uses slicing, the second method uses built-in function list() and the third method uses copy() function of Python library copy.

Method 1 : We can slice our original list and store it into a new variable as follows:

```
newList = oldList[:]
```

Let us take an example to use slicing to copy a list

```
>>> list1 = [1, 2, 3, 4, 5]
>>> list2 = list1[:]
>>> list2
[1, 2, 3, 4, 5]
```

Method 2: We can use the built-in function list () as follows:

```
newList = list(oldList)
```

Let us take an example to use list() function to copy a list

```
>>> list1 = [10,20,30,40]
>>> list2 = list(list1)
>>> list2
[10, 20, 30, 40]
```

Method 3

First of all, we will import the library copy as follows:

```
import copy
```

Now, we can use the copy () function of copy library as follows:

```
newList = copy.copy(oldList)
```

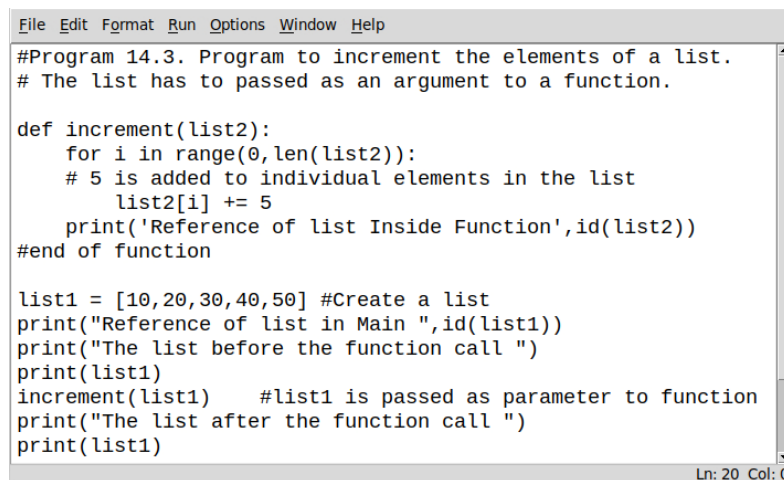
Let us take an example to use copy() function to copy a list

```
>>> import copy
>>> list1 = [1, 2, 3, 4, 5]
>>> list2 = copy.copy(list1)
>>> list2
[1, 2, 3, 4, 5]
```

5.7 List as Argument to a Function

Whenever a list is passed as an argument to a function, we have to consider two scenarios:

Elements of the original list may be changed, i.e. changes made to the list in the function are reflected back in the calling function.



```
File Edit Format Run Options Window Help
#Program 14.3. Program to increment the elements of a list.
# The list has to passed as an argument to a function.

def increment(list2):
    for i in range(0,len(list2)):
        # 5 is added to individual elements in the list
        list2[i] += 5
        print('Reference of list Inside Function',id(list2))
#end of function

list1 = [10,20,30,40,50] #Create a list
print("Reference of list in Main ",id(list1))
print("The list before the function call ")
print(list1)
increment(list1) #list1 is passed as parameter to function
print("The list after the function call ")
print(list1)
Ln: 20 Col: 0
```

```
File Edit Shell Debug Options Window Help
Reference of list in Main 140391557454656
The list before the function call
[10, 20, 30, 40, 50]
Reference of list Inside Function 140391557454656
The list after the function call
[15, 25, 35, 45, 55]
>>>
Ln: 20 Col: 0
```

In above program, list list1 of numbers is passed as an argument to function increment(). This function increases every element of the list by 5.

Assignment

1. Write a program to decrement the elements of a list. The list is passed as an argument to a function.
2. Write a program to get square of the elements of a list containing first ten natural numbers. The list is passed as an argument to a function.

Observe that, when we pass a list as an argument, we actually pass a reference to the list. Hence any change made to list2 inside the function is reflected in the actual list list1.

(2). If the list is assigned a new value inside the function, then a new list object is created and it becomes the local copy of the function. Any changes made inside the local copy of the function are not reflected back to the calling function.

```
File Edit Format Run Options Window Help
# Program 14.4. Program to increment the elements of the list passed as parameter.
def increment(list2):
    print("\nID of list inside function before assignment:", id(list2))
    list2 = [15,25,35,45,55] #List2 assigned a new list
    print("ID of list changes inside function after assignment:", id(list2))
    print("The list inside the function after assignment is:")
    print(list2)
    #end of function

list1 = [10,20,30,40,50] #Create a list
print("ID of list before function call :",id(list1))
print("The list before function call :")
print(list1)
increment(list1)          #list1 passed as parameter to function
print('\nID of list after function call :',id(list1))
print("The list after the function call: ")
print(list1)
Ln: 21 Col: 0
```

```
File Edit Shell Debug Options Window Help
ID of list before function call : 139894898955968
The list before function call :
[10, 20, 30, 40, 50]

ID of list inside function before assignment: 139894898955968
ID of list changes inside function after assignment: 139894877081088
The list inside the function after assignment is:
[15, 25, 35, 45, 55]

ID of list after function call : 139894898955968
The list after the function call:
[10, 20, 30, 40, 50]
>>>
Ln: 58 Col: 0
```

5.8 List Manipulation

In this chapter, we have learn to create a list and the different ways to manipulate lists. In the following programs, we will apply the various list manipulation methods.

```

# Program 14.5. Program to perform various list manipulation operations

myList = [22,4,16,38,13] #myList already has 5 elements
choice = 0

while True:
    print("The list 'myList' has the following elements", myList)
    print("\nL I S T   O P E R A T I O N S")
    print(" 1. Append an element")
    print(" 2. Insert an element at the desired position")
    print(" 3. Append a list to the given list")
    print(" 4. Modify an existing element")
    print(" 5. Delete an existing element by its position")
    print(" 6. Delete an existing element by its value")
    print(" 7. Sort the list in ascending order")
    print(" 8. Sort the list in descending order")
    print(" 9. Display the list")
    print("10. Exit")

    choice = int(input("ENTER YOUR CHOICE (1-10): "))
    #append element
    if choice == 1:
        element = int(input("Enter the element to be inserted: "))
        myList.append(element)
        print("The element has been appended\n")

    #insert an element at desired position
    elif choice == 2:
        element = int(input("Enter the element to be inserted: "))
        pos = int(input("Enter the position:"))
        myList.insert(pos,element)
        print("The element has been inserted\n")

    #append a list to the given list
    elif choice == 3:
        newList=[]
        size=int(input('Enter no of elements in new list'))
        for i in range(size):
            item = int(input("Enter element of new list: "))
            newList.append(item)
        myList.extend(newList)
        print("The list has been appended\n")

    #modify an existing element
    elif choice == 4:
        i = int(input("Enter the position of the element to be modified: "))
        if i < len(myList):
            newElement = int(input("Enter the new element: "))
            oldElement = myList[i]
            myList[i] = newElement
            print("The element",oldElement,"has been modified\n")
        else:
            print("Position of the element is more than the length of list")

    #delete an existing element by position
    elif choice == 5:
        i = int(input("Enter the position of the element to be deleted: "))
        if i < len(myList):
            element = myList.pop(i)
            print("The element",element,"has been deleted\n")
        else:
            print("\nPosition of the element is more than the length of list")

    #delete an existing element by value
    elif choice == 6:
        element = int(input("\nEnter the element to be deleted: "))
        if element in myList:
            myList.remove(element)
            print("\nThe element",element,"has been deleted\n")
        else:
            print("\nElement",element,"is not present in the list")

```

```

#list in sorted order
elif choice == 7:
    myList.sort()
    print("\nThe list has been sorted")

#list in reverse sorted order
elif choice == 8:
    myList.sort(reverse = True)
    print("\nThe list has been sorted in reverse order")

#display the list
elif choice == 9:
    print("\nThe list is:", myList)

#exit from the menu
elif choice == 10:
    print("\nExited.Thank you.")
    break
else:
    print("Choice is not valid")
    print("\n\nPress any key to continue.")
    ch = input()

```

Output:

The list 'myList' has the following elements [22, 4, 16, 38, 13]

L I S T O P E R A T I O N S

1. Append an element
2. Insert an element at the desired position
3. Append a list to the given list
4. Modify an existing element
5. Delete an existing element by its position
6. Delete an existing element by its value
7. Sort the list in ascending order
8. Sort the list in descending order
9. Display the list
10. Exit

ENTER YOUR CHOICE (1-10): 1

Enter the element to be inserted: 25

The element has been appended

The list 'myList' has the following elements [22, 4, 16, 38, 13, 25]

L I S T O P E R A T I O N S

1. Append an element
2. Insert an element at the desired position
3. Append a list to the given list
4. Modify an existing element
5. Delete an existing element by its position
6. Delete an existing element by its value
7. Sort the list in ascending order
8. Sort the list in descending order
9. Display the list
10. Exit

ENTER YOUR CHOICE (1-10): 2

Enter the element to be inserted: 50

Enter the position:3

The element has been inserted

The list 'myList' has the following elements [22, 4, 16, 50, 38, 13, 25]

L I S T O P E R A T I O N S

1. Append an element
2. Insert an element at the desired position
3. Append a list to the given list
4. Modify an existing element
5. Delete an existing element by its position
6. Delete an existing element by its value
7. Sort the list in ascending order
8. Sort the list in descending order
9. Display the list
10. Exit

ENTER YOUR CHOICE (1-10): 5

Enter the position of the element to be deleted: 2

The element 16 has been deleted

The list 'myList' has the following elements [22, 4, 50, 38, 13, 25]

L I S T O P E R A T I O N S

1. Append an element
2. Insert an element at the desired position
3. Append a list to the given list
4. Modify an existing element
5. Delete an existing element by its position
6. Delete an existing element by its value
7. Sort the list in ascending order
8. Sort the list in descending order
9. Display the list
10. Exit

ENTER YOUR CHOICE (1-10): 7

The list has been sorted

The list 'myList' has the following elements [4, 13, 22, 25, 38, 50]

```
File Edit Format Run Options Window Help
# Program 14.6. Program to calculate average marks of n
# students using a function where n is entered by user.

def computeAverage(list1,n):
    total = 0          # initialize total
    for marks in list1: # add marks to total
        total = total + marks
    average = total / n
    return average
# End of function

list1 = []            # create an empty list
print("How many students marks you want to enter: ")
n = int(input())
for i in range(0,n):
    print("Enter marks of student ",(i+1),":")
    marks = int(input())
    list1.append(marks) # append marks in the list

average = computeAverage(list1,n)
print("Average marks of",n,"students is:",average)
```

Ln: 26 Col: 0

The image displays three screenshots of a Python IDE, likely PyCharm, showing the execution of a program and its source code.

Top Screenshot: Shows the execution output of a program. The prompt asks for the number of students, followed by three marks (250, 275, 300), and finally displays the average mark as 275.0. The status bar indicates line 29, column 0.

```

File Edit Shell Debug Options Window Help
How many students marks you want to enter:
3
Enter marks of student 1 :
250
Enter marks of student 2 :
275
Enter marks of student 3 :
300
Average marks of 3 students is: 275.0
>>>
Ln: 29 Col: 0

```

Middle Screenshot: Shows the source code of the program. It defines a function `linearSearch` that checks if a number is present in a list and returns its position. The main code creates an empty list, prompts for the number of elements, collects the elements, prompts for a search number, and uses `linearSearch` to find its position. The status bar indicates line 28, column 0.

```

File Edit Format Run Options Window Help
# Program 14.7. Program to create a user defined function
# to check if a number is present in the list or not

def linearSearch(num, list1):
    for i in range(0, len(list1)):
        if list1[i] == num: # num is present
            return i      # return the position

    return None          # num is not present in the list
# end of function

list1 = []             # Create an empty list
print("How many numbers do you want to enter in the list: ")
maximum = int(input())
print("Enter a list of numbers: ")
for i in range(0, maximum):
    n = int(input())
    list1.append(n)    #append numbers to the list

num = int(input("Enter the number to be searched: "))
result = linearSearch(num, list1)
if result is None:
    print("Number", num, "is not present in the list")
else:
    print("Number", num, "is present at", result + 1, "position")
Ln: 28 Col: 0

```

Bottom Screenshot: Shows the execution output of the program. The prompt asks for the number of numbers, followed by three numbers (12, 25, 36), then a search number (25), and finally displays "Number 25 is present at 2 position". The status bar indicates line 131, column 0.

```

File Edit Shell Debug Options Window Help
How many numbers do you want to enter in the list:
3
Enter a list of numbers:
12
25
36
Enter the number to be searched: 25
Number 25 is present at 2 position
>>>
Ln: 131 Col: 0

```

In above program, a number and a list are the two arguments of the function `linearSearch ()`. The number is compared with each element of the list one by one. If any element of the list is equal to the number, the position of that element is returned.

CHECK YOUR PROGRESS

A. Multiple Choice Questions

1. Python code to get the first element of list list1 is (a) list1[n] (b) list1[n-1] (c) list1[-n] (d) list1[1]
2. Output of the following code is (a) ['India'], ['India'] (b) ['India', 'India'] (c) ['India*2'] (d) ['IndiaIndia']

```
list1 = ['India']
print(list1*2)
```
3. Which function is used to append each element of the list passed as argument to the end of the given list (a) append () (b) insert () (c) extend () (d) add ()
4. Code to insert x at index y in a list is (a) append(x,y) (b) append(y,x) (c) insert(x,y) (d) insert(y,x)
5. Code to get sum of the elements of the list list1 (a) sum(list1) (b) sum (). list1 (c) list1.sum () (d) list1(). sum
6. When a list appears as an element of another list, it is called a (a) nested list (b) singly list (c) empty list (d) doubly list
7. Which of the following will create an empty list L? (a) L = list (b) L = list (0) (c) L = list () (d) L = List(empty)
8. If L1 = [1, 3, 5] and L2 = [2, 4, 6] then L1 + L2 will yield (a) [1, 2, 3, 4, 5, 6] (b) [1, 3, 5, 2, 4, 6] (c) [3, 7, 11] (d) [1, 3, 5, [2, 4, 6]]
9. Given a list L= [1, 2, 3, 4, 5, 6, 7], what would L [1: 5] return? (a) [1, 2, 3, 4] (b) [2, 3, 4, 5] (c) [2, 3, 4] (d) [1,2, 3, 4, 5]
10. Given a list L= [1, 2, 3, 4, 5, 6, 7], what would L [2: -2] return? (a) [1, 2, 3, 4] (b) [2, 3, 4, 5] (c) [2, 3, 4] (d) [3, 4, 5]
11. Given a list L= [1, 2, 3, 4, 5, 6, 7], what would L [-3: 99] return? (a) [2, 3, 4] (b) [3, 4, 5] (c) [5, 6, 7] (d) Error
12. What is printed by the Python code? print (list (range (3))) (a) [0, 1, 2, 3] (b) [1, 2, 3] (c) [0, 1, 2] (d) 0, 1, 2
13. What is the output when we execute list("hello")? (a) ['h', 'e', 'l', 'l', 'o'] (b) ['hello'] (c) ['llo'] (d) ['olleh']
14. What is the output of following code? (a) H (b) a (c) Hasan (d) Dia

```
names = ['Hasan', 'Balwant', 'Sean', 'Dia']
print (names [-1] [-1])
```
15. Which of the following will always return a list? (a) max () (b) min () (c) sort () (d) sorted ()

B. State whether True or False

1. List data type in Python is mutable.
2. A list can have elements of different data types, such as integer, float, string, tuple or even another list.
3. A = [] and A = list () will produce the same result.
4. Lists once created cannot be changed.
5. To sort a list, sort () and sorted (), both can be used.
6. The extend () adds a single element to a list.
7. The append () can add an element in the middle of a list.
8. The insert () can add an element in the middle of a list.
9. The del statement can only delete list slices and not single elements from a list.
10. The del statement can work similar to the pop () function.

C. Fill-in the blanks

1. List is an _____ sequence.
2. Elements of a list are enclosed in _____ brackets.
3. List indices start from _____.

4. If no parameter is given, then pop () function returns and removes the _____ element of the list.
5. To create an empty list, function _____ can used.
6. The ____ operator adds one list to the end another list.
7. The ____ operator replicates a list.
8. To check if an element is in list, _____ operator is used.
9. To delete a list slice from a list, _____ statement is used
10. A _____ list contains another list as its member.
11. The _____ function is used to insert element at a designated position in a list.
12. The _____ function is used to delete element to remove an element from designated index in a list.
13. The _____ function can append a list element to a list.
14. The _____ function sorts a list and makes changes in the list.
15. The _____ function sorts a list and returns another list.

D. Programming Questions

1. What will be the output of the following statements?
 - (a)

```
list1 = [12,32,65,26,80,10]
list1.sort()
print(list1)
```
 - (b)

```
list1 = [12,32,65,26,80,10]
sorted(list1)
print(list1)
```
 - (c)

```
list1 = [1,2,3,4,5,6,7,8,9,10]
list1[: -2]
list1[:3] + list1[3:]
```
 - (d)

```
list1 = [1,2,3,4,5]
list1[len(list1)-1]
```
2. Consider the following list myList. What will be the elements of myList after the following two operations:


```
myList = [10,20,30,40]
myList.append([50,60])
myList.extend([80,90])
```
3. What will be the output of the following code segment:


```
myList = [1,2,3,4,5,6,7,8,9,10]
for i in range (0, len(myList)):
    if i%2 == 0:
        print(myList[i])
```
4. What will be the output of the following code segment:
 - a.

```
myList = [1,2,3,4,5,6,7,8,9,10]
del myList[3:]
print(myList)
```
 - b.

```
myList = [1,2,3,4,5,6,7,8,9,10]
del myList[:5]
print(myList)
```
 - c.

```
myList = [1,2,3,4,5,6,7,8,9,10]
del myList[:2]
print(myList)
```
5. The record of a student (Name, Roll No., Marks in five subjects and percentage of marks) is stored in the following list:

```
stRecord = ['Raman','A-36', [56,98,99,72,69], 78.8]
```

Write Python statements to retrieve the following information from the list stRecord.

- a) Percentage of the student
 - b) Marks in the fifth subject
 - c) Maximum marks of the student
 - d) Roll no. of the student
 - e) Change the name of the student from 'Raman' to 'Raghav'
6. Write a program to find the number of times an element occurs in the list.
 7. Write a program to read a list of n integers (positive as well as negative). Create two new lists, one having all positive numbers and the other having all negative numbers from the given list. Print all three lists.
 8. Write a function that returns the largest element of the list passed as parameter.
 9. Write a function to return the second largest number from a list of numbers.
 10. Write a program to read a list of n integers and find their median.
 11. Write a program to read a list of elements. Modify this list so that it does not contain any duplicate elements, i.e., all elements occurring multiple times in the list should appear only once.
 12. Write a program to read a list of elements. Input an element from the user that has to be inserted in the list. Also input the position at which it is to be inserted. Write a user defined function to insert the element at the desired position in the list.
 13. Write a program to read elements of a list.
 - (a) The program should ask for the position of the element to be deleted from the list. Write a function to delete the element at the desired position in the list.
 - (b) The program should ask for the value of the element to be deleted from the list. Write a function to delete the element of this value from the list.
 14. Read a list of n elements. Pass this list to a function which reverses this list in-place without creating a new list.

Session 6: Tuples and Dictionaries

Sharmishtha was helping her younger sister Moushamiin preparing her school assignment. This assignment was to fill names of days of the week and names of months in a year. When Moushami wrote the assignment, she made some mistakes in the sequence of days and months. Then, Sharmishtha explained her that these are fixed values which cannot be changed. Similarly, in programming also sometimes we require a sequence where elements can't be changed. For such type of data, tuples are used. (Figure 6.1)



Fig. 6.1: Real life example of Tuple

If our data is being constantly modified or we need a fast lookup based on a custom key or we need a logical association between the key: value pair, it is advised to use dictionaries. A mobile phone book is a good application of dictionary.



Fig. 6.2: Example of Dictionary

In this chapter, you will understand the basic concepts of data types tuples and dictionaries.

6.2 Tuples

A tuple is an ordered sequence of elements of different data types, such as integer, float, string, list or even a tuple. Elements of a tuple are enclosed in parenthesis (round brackets) and are separated by commas. Like list and string, elements of a tuple can be accessed using index values, starting from 0. Let us take few examples of tuple

#tuple1 is the tuple of integers

```
>>> tuple1 = (1,2,3,4,5)
```

```
>>> tuple1
```

```
(1, 2, 3, 4, 5)
```

#tuple2 is the tuple of strings

```
>>> tuple2 = ("Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday")
```

```
>>> tuple2
```

```
('Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday')
```

#tuple3 is the tuple of mixed data types

```
>>> tuple3 = ('Economics', 87, 'Accountancy', 89.6)
```

```
>>> tuple3
```

```
('Economics', 87, 'Accountancy', 89.6)
```

#tuple4 is the tuple with list as an element

```
>>> tuple4 = (10, 20, 30, [40, 50])
```

```
>>> tuple4
```

```
(10, 20, 30, [40, 50])
```

#tuple5 is the tuple with tuple as an element

```
>>> tuple5 = (1, 2, 3, 4, 5, (10,20))
```

```
>>> tuple5
```

```
(1, 2, 3, 4, 5, (10, 20))
```

If there is only a single element in a tuple then the element should be followed by a comma. If we assign the value without comma it is treated as integer. It should be noted that a sequence without parenthesis is treated as tuple by default.

#incorrect way of assigning single element to #tuple

#tuple5 is assigned a single element

```
>>> tuple5 = (20)
```

```
>>> tuple5
```

```
20
```

```
>>>type(tuple5)
```

```
#tuple5 is not of type tuple
```

```
<class 'int'>
```

```
#it is treated as integer
```

#Correct Way of assigning single element to

#tuple

```
#tuple5 is assigned a single element
>>> tuple5 = (20,)          #element followed by comma
>>> tuple5
(20,)
>>> type(tuple5)           #tuple5 is of type tuple
<class 'tuple'>
#a sequence without parentheses is treated as
#tuple by default
>>> seq = 1,2,3            #comma separated elements
>>> type(seq)              #treated as tuple
<class 'tuple'>
>>> print(seq)            #seq is a tuple
(1, 2, 3)
```

6.1.1 Accessing Elements in a Tuple

Elements of a tuple can be accessed in the same way as a list or string using indexing and slicing. Let us take few examples to illustrate this.

```
>>> tuple1 = (2, 4, 6, 8, 10, 12)  #initializes a tuple tuple1
#returns the first element of tuple1
>>> tuple1[0]
2
#returns fourth element of tuple1
>>> tuple1[3]
8
#returns error as index is out of range
>>> tuple1[15]
IndexError: tuple index out of range
#an expression resulting in an integer index
>>> tuple1[1+4]
12
#returns first element from right
>>> tuple1[-1]
12
```

6.1.2 Tuple is Immutable

Tuple is an immutable data type. It means that the elements of a tuple cannot be changed after it has been created. An attempt to do this would lead to an error as illustrated in the example given below:

```
>>> tuple1 = (1, 2, 3, 4, 5)
>>> tuple1[4] = 10
TypeError: 'tuple' object does not support item assignment
```

However, an element of a tuple may be of mutable type, e.g., a list.

```
#4th element of the tuple2 is a list
>>> tuple2 = (1, 2, 3, [8, 9])
#modify the list element of the tuple tuple2
>>> tuple2[3][1] = 10
#modification is reflected in tuple2
>>> tuple2
(1, 2, 3, [8, 10])
```

```

File Edit Format Run Options Window Help
# Program 15.1. Program to create tuples and implement
# the following operations on tuples.

# 1. The tuple1 is an empty tuple
# 2. Initialize tuple2 with first five natural numbers,
#    print it's last element.
# 3. Assign tuple2 to a sequence of 5 elements, print
#    first and 3rd element of sequence.

tuple1=()
print("Tuple1 is empty:",tuple1)
tuple2=(1,2,3,4,5)
print("Tuple2 = ",tuple2)
(s1,s2,s3,s4,s5) = tuple2
print("s1 =",s1)
print("s3 =",s3)
Ln: 19 Col: 0

```

```

File Edit Shell Debug Options Window Help
Tuple1 is empty: ()
Tuple2 = (1, 2, 3, 4, 5)
s1 = 1
s3 = 3
>>>
Ln: 18 Col: 0

```

In above example, parentheses are used to create an empty tuple tuple1. Assignment operator in the statement tuple2 = (1, 2, 3, 4, 5) is used to initialize tuple2 with first five natural numbers. Next statement, (s1, s2, s3, s4, s5) = tuple2 is used to assign first element of tuple2 to s1, second element of tuple2 to s2 and so on.

6.2 Tuple Operations

A number of operations like concatenation, repetition, membership and slicing can be performed on tuples. Let us learn these tuple operations one by one.

6.2.1 Concatenation

Python allows us to join tuples using concatenation operator depicted by symbol +. We can also create a new tuple which contains the result of this concatenation operation. Let us take few examples to illustrate this.

```

>>> tuple1 = (1, 3, 5, 7, 9)
>>> tuple2 = (2, 4, 6, 8, 10)
>>> tuple1 + tuple2 #concatenates two tuples
(1, 3, 5, 7, 9, 2, 4, 6, 8, 10)
>>> tuple3 = ('Red', 'Green', 'Blue')
>>> tuple4 = ('Cyan', 'Magenta', 'Yellow', 'Black')
#tuple5 stores elements of tuple3 and tuple4
>>> tuple5 = tuple3 + tuple4
>>> tuple5
('Red', 'Green', 'Blue', 'Cyan', 'Magenta', 'Yellow', 'Black')

```

Concatenation operator can also be used for extending an existing tuple. When we extend a tuple using concatenation a new tuple is created.

```

>>> tuple6 = (1, 2, 3, 4, 5)
#single element is appended to tuple6
>>> tuple6 = tuple6 + (6,)
>>> tuple6

```

```
(1, 2, 3, 4, 5, 6)
#more than one elements are appended
>>> tuple6 = tuple6 + (7, 8, 9)
>>> tuple6
(1, 2, 3, 4, 5, 6, 7, 8, 9)
```

6.2.2 Repetition

Repetition operation is depicted by the symbol *. It is used to repeat elements of a tuple. We can repeat the tuple elements. The repetition operator requires the first operand to be a tuple and the second operand to be an integer only.

```
>>> tuple1 = ('Hello', 'World')
>>> tuple1 * 3
('Hello', 'World', 'Hello', 'World', 'Hello', 'World')
#tuple with single element
>>> tuple2 = ("Hello",)
>>> tuple2 * 4
('Hello', 'Hello', 'Hello', 'Hello')
```

6.2.3 Membership

The in operator checks if the element is present in the tuple and returns True, else it returns False.

```
>>> tuple1 = ('Red', 'Green', 'Blue')
>>> 'Green' in tuple1
True
```

The not in operator returns True if the element is not present in the tuple, else it returns False.

```
>>> tuple1 = ('Red', 'Green', 'Blue')
>>> 'Green' not in tuple1
False
```

6.2.4 Slicing

Like string and list, slicing can be applied to tuples also. Let us take few examples to illustrate this.

```
#tuple1 is a tuple
>>> tuple1 = (10, 20, 30, 40, 50, 60, 70, 80)
#elements from index 2 to index 6
>>> tuple1[2:7]
(30, 40, 50, 60, 70)
#all elements of tuple are printed
>>> tuple1[0:len(tuple1)]
(10, 20, 30, 40, 50, 60, 70, 80)
#slice starts from zero index
>>> tuple1[:5]
(10, 20, 30, 40, 50)
#slice is till end of the tuple
>>> tuple1[2:]
(30, 40, 50, 60, 70, 80)
#step size 2
>>> tuple1[0:len(tuple1):2]
(10, 30, 50, 70)
#negative indexing
>>> tuple1[-6:-4]
(30, 40)
#tuple is traversed in reverse order
```



```
>>> tuple1[::-1]
(80, 70, 60, 50, 40, 30, 20, 10)
```

```
File Edit Format Run Options Window Help
# Program 15.2. Program to do following operations on tuples
# 1. Concatenate tuple1 and tuple2.
# 2. Print the elements of tuple1 twice.
# 3. Print elements of tuple1 in reverse order.
# 4. Check for any value exists in tuple2 or not.

tuple1 = (1,2,3,4,5)
tuple2 = ('a','b','c')
print("Concatenation of both tuple:", tuple1+tuple2)
print("Twice elemnts of tuple1 are      :", tuple1 * 2)
print("Elements in reverse order      :",tuple1[: :-1])
print("Status of presence for character b is", end=" ")
print('b' in tuple2)

Ln: 17 Col: 0

File Edit Shell Debug Options Window Help
Concatenation of both tuple: (1, 2, 3, 4, 5, 'a', 'b', 'c')
Twice elemnts of tuple1 are      : (1, 2, 3, 4, 5, 1, 2, 3, 4, 5)
Elements in reverse order      : (5, 4, 3, 2, 1)
Status of presence for character b is True

>>>

Ln: 70 Col: 0
```

In above program, operator + is used to concatenate the tuples. Symbol * is used with value 2 to print tuple1 twice. Membership operator is used check that 'b' exists in tuple2 or not. The concept of slicing is used to print elements of tuple1 in reverse order.

6.3 Tuple Methods and Built-In Functions

Python provides many functions to work on tuples. Table 6.1 list some of the commonly used tuple methods and built-in functions.

Table 6.1 Built-in functions and methods for tuples

Method	Description	Example
len()	Returns the length or the number of elements of the tuple passed as the argument	>>> tuple1 = (10,20,30,40,50) >>> len (tuple1) 5
tuple()	Creates an empty tuple if no argument is passed. Create a tuple if a sequence is passed as argument	>>> tuple1 = tuple () >>> tuple1 () >>> tuple1 = tuple ('aeiou') #string >>> tuple1 ('a', 'e', 'i', 'o', 'u') >>> tuple2 = tuple ([1,2,3]) #list >>> tuple2 (1, 2, 3) >>> tuple3 = tuple(range(5)) >>> tuple3 (0,1,2,3,4)
count()	Returns the number of times the given element appears in the tuple	>>> tuple1 =(10,20,30,10,40,10,50) >>> tuple1.count(10) 3 >>> tuple1.count (90) 0

index()	Returns the index of the first occurrence of the element in the given tuple	>>> tuple1 = (10,20,30,40,50) >>> tuple1.index(30) 2 >>> tuple1.index(90) ValueError: tuple.index(x) : x not in tuple
sorted()	Takes elements in the tuple and returns new sorted list. It should be noted that, sorted() does not make any change to the original tuple	>>> tuple1 = ("Rama", "Heena", "Raj", "Mohsin", "Aditya") >>> sorted (tuple1) ['Aditya', 'Heena', 'Mohsin', 'Raj', 'Rama']
min()	Returns minimum or smallest element of the tuple	>>> tuple1=(19,12,56,18,9,87,34) >>> min(tuple1) 9
max()	Returns maximum or largest element of the tuple	>>> tuple1=(19,12,56,18,9,87,34) >>> max (tuple1) 87
sum()	Returns sum of the elements of the tuple	>>> tuple1=(19,12,56,18,9,87,34) >>> sum (tuple1) 235

```
File Edit Format Run Options Window Help
# Program 15.3. Program for tuple manipulations
# using built-in functions

# 1. Find length of the tuple1.
# 2. Find minimum value in tuple2.
# 3. Find sum of the elements of tuple1.

tuple1 = (1,2,3,4,5)
print("Length of tuple 1 is      :",len(tuple1))
print("Minimum vaue in tuple 1 is :",min(tuple1))
print("Sum of items in tuple 1 is :",sum(tuple1))
Ln: 14 Col: 0
```

```
File Edit Shell Debug Options Window Help
Length of tuple 1 is      : 5
Minimum vaue in tuple 1 is : 1
Sum of items in tuple 1 is : 15
>>>
Ln: 77 Col: 0
```

In above program, len() function is used to find length of the tuple1. The function min() is used to find minimum of the tuple2. The function sum() is used to get sum of the elements of tuple1.

6.4 Tuple Assignment

Assignment of tuple is a useful feature in Python. It allows a tuple of variables on the left side of the assignment operator to be assigned respective values from a tuple on the right side. The number of variables on the left should be same as the number of elements in the tuple. Let us take an example to illustrate this.

#The first element 10 is assigned to num1 and

#the second element 20 is assigned to num2.

```
>>> (num1,num2) = (10,20)
```

```
>>> print(num1)
```

```
10
```

```
>>> print(num2)
```

```
20
```

```
>>> record = ("Pooja",40,"CS")
```

```
>>> (name, rollNo, subject) = record
>>> name
'Pooja'
>>> rollNo
40
>>> subject
'CS'
>>> (a, b, c, d) = (5,6,8)
ValueError: not enough values to unpack
(expected 4, got 3)
```

If there is an expression on the right side then first that expression is evaluated and finally the result is assigned to the tuple. Let us take an example to illustrate this.

```
#15 is assigned to num3 and
#25 is assigned to num4
>>> (num3, num4) = (10+5,20+5)
>>> print(num3)
15
>>> print(num4)
25
```

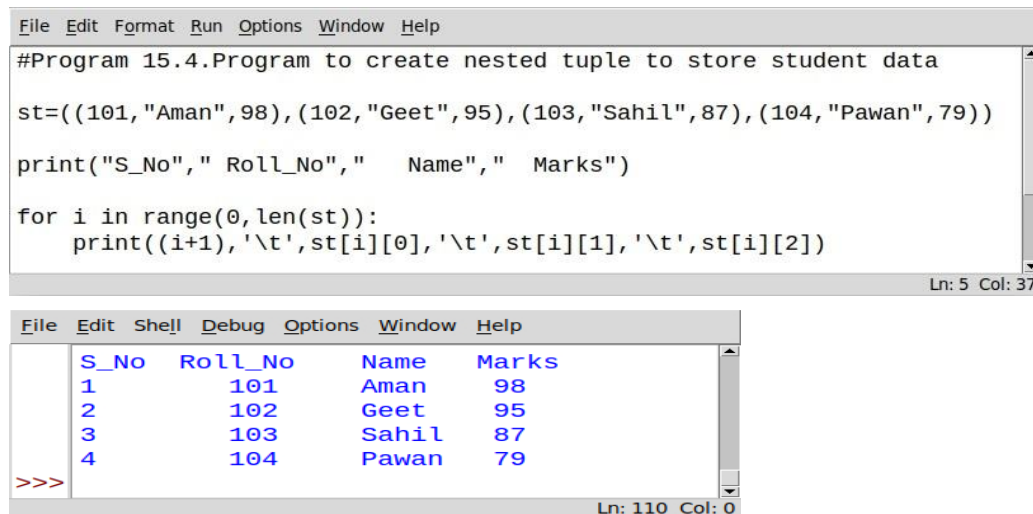
6.5 Nested Tuples

A tuple inside another tuple is called a nested tuple. In a nested tuple, each tuple is considered as an element. Loop control structures can be used to access the elements in a nested tuple. Let us take few examples of nested tuple.

```
>>> tuple1 = (1,2,3, (4,5))
>>> tuple1
(1, 2, 3, (4, 5))
>>> tuple2 = ((1,2), (3,4), (5,6))
>>> tuple2
((1, 2), (3, 4), (5, 6))
```

Here, (4,5) is a nested tuple as it an element of another tuple tuple1. Similarly, (1, 2), (3, 4) and (5, 6) are nested tuples as these tuples are elements of another tuple tuple2.

Nested tuple can be used to represent a specific data record. For example, records of many students consisting RollNo, Name and Marks can be stored in a nested tuple. Let us do an activity to demonstrate the use of nested tuples to store records of students and print them.



```
File Edit Format Run Options Window Help
#Program 15.4.Program to create nested tuple to store student data
st=((101, "Aman", 98), (102, "Geet", 95), (103, "Sahil", 87), (104, "Pawan", 79))
print("S_No", " Roll_No", " Name", " Marks")
for i in range(0, len(st)):
    print((i+1), '\t', st[i][0], '\t', st[i][1], '\t', st[i][2])
Ln: 5 Col: 37
```

```
File Edit Shell Debug Options Window Help
S_No Roll_No Name Marks
1 101 Aman 98
2 102 Geet 95
3 103 Sahil 87
4 104 Pawan 79
>>>
Ln: 110 Col: 0
```

In the above program, details like roll number, name and marks of students are saved in a tuple. To store details of many such students we created nested tuples using for loop.

6.6 Tuple Handling

```
File Edit Format Run Options Window Help
# Program 15.5. Program to swap two numbers without
# using a temporary variable.

num1 = int(input('Enter the first number: '))
num2 = int(input('Enter the second number: '))
print("\nNumbers before swapping:")
print("First Number:", num1)
print("Second Number:", num2)

(num1, num2) = (num2, num1)
print("\nNumbers after swapping:")
print("First Number:", num1)
print("Second Number:", num2)
Ln: 16 Col: 0
```

```
File Edit Shell Debug Options Window Help
Enter the first number: 5
Enter the second number: 10

Numbers before swapping:
First Number: 5
Second Number: 10

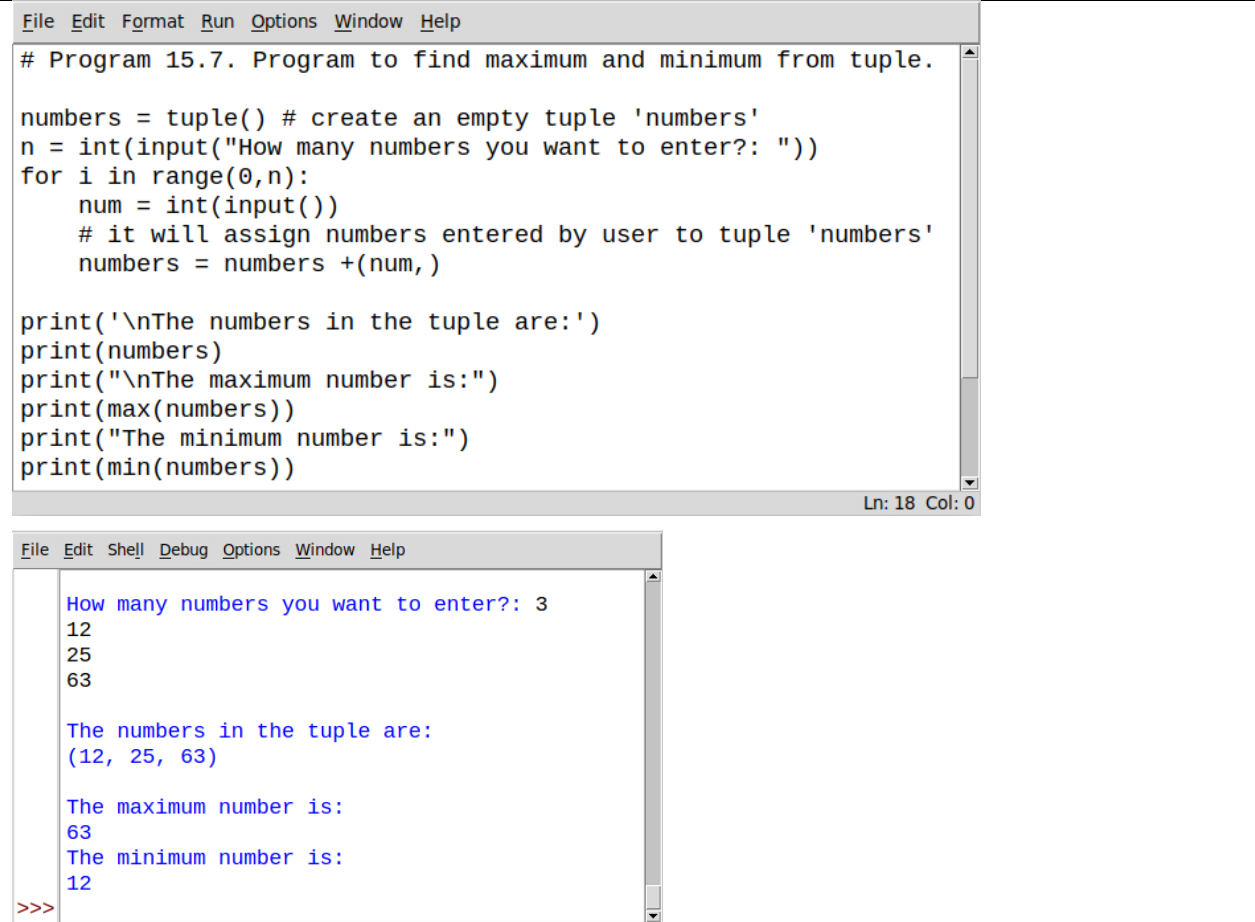
Numbers after swapping:
First Number: 10
Second Number: 5
>>>
Ln: 124 Col: 0
```

```
File Edit Format Run Options Window Help
# Program 15.6. Program to compute the area and circumference
# of a circle using user defined function

def circle(r):
    area = 3.14*r*r
    circumference = 2*3.14*r
    #returns a tuple having two elements area and circumference
    return (area, circumference)
#end of function

radius = float(input('Enter radius of circle: '))
area, circumference = circle(radius)
print('Area of circle is:', area)
print('Circumference of circle is:', circumference)
Ln: 18 Col: 0
```

```
File Edit Shell Debug Options Window Help
Enter radius of circle: 3.25
Area of circle is: 33.16625
Circumference of circle is: 20.41
>>>
Ln: 187 Col: 0
```



```

File Edit Format Run Options Window Help
# Program 15.7. Program to find maximum and minimum from tuple.

numbers = tuple() # create an empty tuple 'numbers'
n = int(input("How many numbers you want to enter?: "))
for i in range(0,n):
    num = int(input())
    # it will assign numbers entered by user to tuple 'numbers'
    numbers = numbers +(num,)

print('\nThe numbers in the tuple are:')
print(numbers)
print("\nThe maximum number is:")
print(max(numbers))
print("The minimum number is:")
print(min(numbers))
Ln: 18 Col: 0

File Edit Shell Debug Options Window Help
How many numbers you want to enter?: 3
12
25
63

The numbers in the tuple are:
(12, 25, 63)

The maximum number is:
63
The minimum number is:
12
>>>
Ln: 258 Col: 0

```

6.7 Introduction to Dictionaries

The data type dictionary falls under mapping. It is a mapping between a set of keys and a set of values. The key-value pair is called an item. A key is separated from its value by a colon (:) and consecutive items are separated by commas. Items in dictionaries are unordered, so we may not get back the data in the same order in which we had entered the data initially in the dictionary.

6.7.1 Creating a Dictionary

To create a dictionary, the items entered are separated by commas and enclosed in curly braces. Each item is a key value pair, separated through colon (:). The keys in the dictionary must be unique and should be of any immutable data type like number, string or tuple. The values can be repeated and can be of any data type. Let us take few examples of creating dictionaries.

#dict1 is an empty Dictionary created

#curly braces are used for dictionary

```
>>> dict1 = {}
```

```
>>> dict1
```

```
{}
```

#dict2 is an empty dictionary created using

#built-in function

```
>>> dict2 = dict()
```

```
>>> dict2
```

```
{}
```

#dict3 is the dictionary that maps names

#of the students to respective marks in #percentage

```
>>> dict3 = {'Mohan':95, 'Ram':89, 'Suhel':92, 'Sangeeta':85}
```

```
>>> dict3
```

```
{'Mohan': 95, 'Ram': 89, 'Suhel': 92, 'Sangeeta': 85}
```

6.7.2 Accessing Items in a Dictionary

We have already seen that the items of a sequence (string, list and tuple) are accessed using a technique called indexing. The items of a dictionary are accessed via the keys rather than via their relative positions or indices. Each key serves as the index and maps to a value.

The following example shows how a dictionary returns the value corresponding to the given key:

```
>>> dict3 = {'Mohan':95, 'Ram':89, 'Suhel':92, 'Sangeeta':85}
>>> dict3['Ram']
89
>>> dict3['Sangeeta']
85
#the key does not exist
>>> dict3['Shyam']
KeyError: 'Shyam'
```

In the above examples the key 'Ram' always maps to the value 89 and key 'Sangeeta' always maps to the value 85. So, the order of items does not matter. If the key is not present in the dictionary we get KeyError.

6.8 Dictionaries are mutable

Dictionaries are mutable which implies that the contents of the dictionary can be changed after it has been created.

6.8.1 Adding a new item

We can add a new item to the dictionary as shown in the following example:

```
>>> dict1 = {'Mohan':95, 'Ram':89, 'Suhel':92, 'Sangeeta':85}
>>> dict1['Meena'] = 78
>>> dict1
{'Mohan': 95, 'Ram': 89, 'Suhel': 92, 'Sangeeta': 85, 'Meena': 78}
```

6.8.2 Modifying an Existing Item

The existing dictionary can be modified by just overwriting the key-value pair. Example to modify a given item in the dictionary:

```
>>> dict1 = {'Mohan':95, 'Ram':89, 'Suhel':92, 'Sangeeta':85}
#Marks of Suhel changed to 93.5
>>> dict1['Suhel'] = 93.5
>>> dict1
{'Mohan': 95, 'Ram': 89, 'Suhel': 93.5, 'Sangeeta': 85}
```

6.9 Dictionary Operations

Except membership, other operations like concatenation, repetition and slicing are not supported by dictionaries. Let us learn membership operation on dictionary data type in Python programming.

Membership

The membership operator **'in'** checks if the key is present in the dictionary and returns True, else it returns False.

```
>>> dict1 = {'Mohan':95, 'Ram':89, 'Suhel':92, 'Sangeeta':85}
>>> 'Suhel' in dict1
True
```

The not in operator returns True if the key is not present in the dictionary, else it returns False.

```
>>> dict1 = {'Mohan':95, 'Ram':89, 'Suhel':92, 'Sangeeta':85}
>>> 'Suhel' not in dict1
```

False

6.10 Traversing A Dictionary

We can access each item of the dictionary or traverse a dictionary using for loop.

```
>>> dict1 = {'Mohan':95, 'Ram':89, 'Suhel':92, 'Sangeeta':85}
```

Method 1

In this method we use key of the dictionary to get corresponding value among all the elements of dictionary as illustrated below:

```
>>> for key in dict1:
    print(key, ':', dict1[key])
```

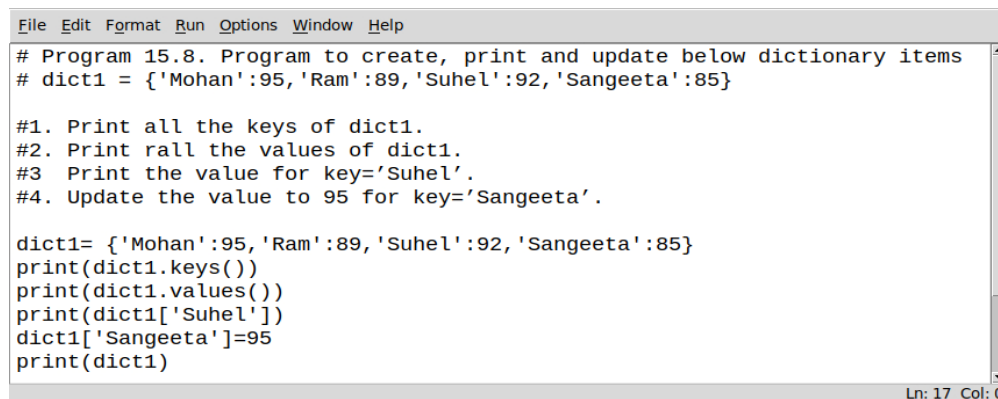
```
Mohan: 95
Ram: 89
Suhel: 92
Sangeeta: 85
```

Method 2

In this method we use both key and value of each element to access elements of the dictionary as illustrated below:

```
>>> for key, value in dict1.items():
    print(key, ':', value)
```

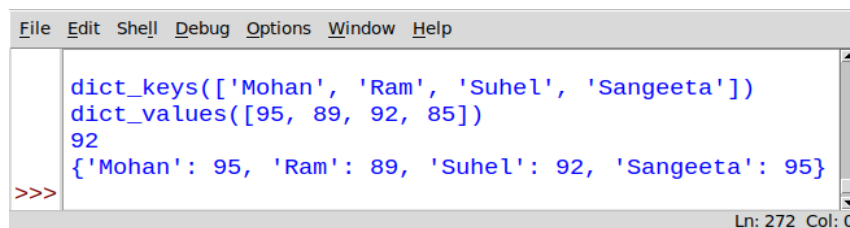
```
Mohan: 95
Ram: 89
Suhel: 92
Sangeeta: 85
```



```
File Edit Format Run Options Window Help
# Program 15.8. Program to create, print and update below dictionary items
# dict1 = {'Mohan':95, 'Ram':89, 'Suhel':92, 'Sangeeta':85}

#1. Print all the keys of dict1.
#2. Print rall the values of dict1.
#3 Print the value for key='Suhel'.
#4. Update the value to 95 for key='Sangeeta'.

dict1= {'Mohan':95, 'Ram':89, 'Suhel':92, 'Sangeeta':85}
print(dict1.keys())
print(dict1.values())
print(dict1['Suhel'])
dict1['Sangeeta']=95
print(dict1)
Ln: 17 Col: 0
```



```
File Edit Shell Debug Options Window Help
dict_keys(['Mohan', 'Ram', 'Suhel', 'Sangeeta'])
dict_values([95, 89, 92, 85])
92
{'Mohan': 95, 'Ram': 89, 'Suhel': 92, 'Sangeeta': 95}
>>>
Ln: 272 Col: 0
```

In above program, the function keys() is used to print all the keys of dict1.

The function values () is used to print all the values of dict1.

The statement print(dict1['Suhel']) is used to print the value for key = 'Suhel'.

The statement dict1['Sangeeta'] =95 is used to update the value to 95 for key = 'Sangeeta'.

6.11 Dictionary Methods and Built-In Functions

Python provides many functions to work on dictionaries. Table 6.2 lists some of the commonly used dictionary methods.

Table 6.2 Built-in functions and methods for dictionary

Method	Description	Example
len()	Returns the length or the number of key: value pairs of the dictionary passed as the argument.	<pre>>>> dict1 = {'Mohan' :95, 'Ram' :89, 'Suhel':92,'Sangeeta' :85} >>> len (dict1) 4</pre>
dict()	Create a dictionary from a sequence of key-value pairs.	<pre>Pair1 = [('Mohan',95), ('Ram',89), ('Suhel',92),('Sangeeta',85)] >>> pair1 [('Mohan',95), ('Ram',89), ('Suhel',92), ('Sangeeta',85)] >>> dict1 = dict (pair1) >>> dict1 {'Mohan': 95, 'Ram': 89, 'Suhel': 92, 'Sangeeta': 85}</pre>
key()	Returns a list of keys in the dictionary.	<pre>>>> dict1 = {'Mohan': 95, 'Ram': 89, 'Suhel': 92, 'Sangeeta': 85} >>> dict1.keys() dict_keys(['Mohan', 'Ram', 'Suhel', 'Sangeeta'])</pre>
values()	Returns a list of values in the dictionary.	<pre>>>> dict1 = {'Mohan': 95, 'Ram': 89, 'Suhel': 92, 'Sangeeta': 85} >>> dict1.values() dict_values([95,89,92,85])</pre>
items()	Returns a list of tuples(key-value) pair.	<pre>>>> dict1 = {'Mohan': 95, 'Ram': 89, 'Suhel': 92, 'Sangeeta': 85} >>> dict1.items() dict_items([('Mohan',95), ('Ram',89), ('Suhel',92), ('Sangeeta',85)])</pre>
get()	Returns the value corresponding to the key passed as the argument. If the key is not present in the dictionary it will return None	<pre>>>> dict1 = {'Mohan': 95, 'Ram': 89, 'Suhel': 92, 'Sangeeta': 85} >>> dict1.get('Sangeeta') >>> 85</pre>
update()	Appends the key-value pair of the dictionary passed as the argument to key-value pair of the given dictionary	<pre>>>> dict1 = {'Mohan': 95, 'Ram': 89, 'Suhel': 92, 'Sangeeta': 85} >>> dict2 = {'Sohan' :79, 'Geeta' :89} >>> dict1.update(dict2) >>> dict1 {'Mohan': 95, 'Ram': 89, 'Suhel': 92, 'Sangeeta': 85, 'Sohan' :79, 'Geeta': 89} >>> dict2 {'Sohan' :79, 'Geeta' :89}</pre>

del()	Deletes the item with the given key. To delete the dictionary from the memory we write: del Dict_name	>>> dict1 = {'Mohan': 95, 'Ram': 89, 'Suhel': 92, 'Sangeeta': 85} >>> del dict1 ['Ram'] >>> dict1 {'Mohan': 95, 'Suhel': 92, 'Sangeeta': 85} >>> del(dict1 ['Mohan']) >>> dict1 {'Suhel': 92, 'Sangeeta': 85} >>> del dict1 >>> dict1 NameError: name 'dict1' is not defined
clear()	Deletes or clear all the items of the dictionary.	>>> dict1 = {'Mohan': 95, 'Ram': 89, 'Suhel': 92, 'Sangeeta': 85} >>> dict1.clear() >>> dict1 {}

```
File Edit Format Run Options Window Help
# Program 15.9. Program to create, print and delete items
# from below dictionary items
# dict1= {'Mohan':95, 'Ram':89, 'Suhel':92, 'Sangeeta':85}

#1. Print all the elements of dict1.
#2. Delete the item with key Suhel.
#3. Delete all the elements of dict1.

dict1= {'Mohan':95, 'Ram':89, 'Suhel':92, 'Sangeeta':85}
print(dict1.items())
del(dict1['Suhel'])
print(dict1)
dict1.clear()
print(dict1)
Ln: 18 Col: 0

File Edit Shell Debug Options Window Help
dict_items([('Mohan', 95), ('Ram', 89), ('Suhel', 92), ('Sangeeta', 85)])
{'Mohan': 95, 'Ram': 89, 'Sangeeta': 85}
{}
>>>
Ln: 297 Col: 0
```

In above program, the function items() is used to print all the elements of dict1. The function del() is used to delete the item with key Suhel. The function clear() is used to delete all the elements of dict1.

6.12 Manipulating Dictionaries

In this chapter, we have learn how to create a dictionary and apply various methods to manipulate it. The Program 6.10 shows the application of those manipulation methods on dictionaries.

```

File Edit Shell Debug Options Window Help
>>> # Program 15.10. Create a dictionary of odd numbers and perform the following
... # operations
... # (a) Display the keys
... # (b) Display the values
... # (c) Display the items
... # (d) Find the length of the dictionary
... # (e) Check if 7 is present or not
>>> # (f) Check if 2 is present or not
>>> # (g) Retrieve the value corresponding to the key 9
>>> # (h) Delete the item from the dictionary corresponding to the key 9
>>> # Solution
>>> # (a) Display the keys
>>> ODD = {1:'One',3:'Three',5:'Five',7:'Seven',9:'Nine'}
>>> ODD
{1: 'One', 3: 'Three', 5: 'Five', 7: 'Seven', 9: 'Nine'}
>>> # (b) Display the values
>>> ODD.values()
dict_values(['One', 'Three', 'Five', 'Seven', 'Nine'])
>>> # (c) Display the items
>>> ODD.items()
dict_items([(1, 'One'), (3, 'Three'), (5, 'Five'), (7, 'Seven'), (9, 'Nine')])
>>> # (d) Find the length of the dictionary
>>> len(ODD)
5
>>> # (e) Check if 7 is present or not
>>> 7 in ODD
True
>>> # (f) Check if 2 is present or not
>>> 2 in ODD
False
>>> # (g) Retrieve the value corresponding to the key 9
>>> ODD.get(9)
'Nine'
>>> # (h) Delete the item from the dictionary corresponding to the key 9
>>> del ODD[9]
>>> ODD
{1: 'One', 3: 'Three', 5: 'Five', 7: 'Seven'}
>>>
Ln: 40 Col: 0

```

```

File Edit Format Run Options Window Help
# Program 15.11. Program to store employees details in a dictionary.

num = int(input("Enter the number of employees whose data to be stored:"))
count = 1
employee = dict() #create an empty dictionary
while count <= num:
    name = input("Enter the name of the Employee: ")
    salary = int(input("Enter the salary: "))
    employee[name] = salary
    count += 1

print("\n\nEMPLOYEE_NAME\tSALARY")
for k in employee:
    print(k, '\t\t', employee[k])

Ln: 16 Col: 0

```

```

File Edit Shell Debug Options Window Help

Enter the number of employees whose data to be stored:3
Enter the name of the Employee: RAJU
Enter the salary: 15000
Enter the name of the Employee: HARI
Enter the salary: 18000
Enter the name of the Employee: SONU
Enter the salary: 16000

EMPLOYEE_NAME    SALARY
RAJU              15000
HARI              18000
SONU              16000
>>>
Ln: 53 Col: 0

```

In above Program, dictionary emp is created which stores names of the employee as key and their salary as values.

```
File Edit Format Run Options Window Help
# Program 15.12. Program to count the occurrences
# of a character i n a given string using dictionary.
print()
st = input("Enter a string: ")
dic = {} #creates an empty dictionary
for ch in st:
    if ch in dic:    # if next character is already in the dictionary
        dic[ch] += 1
    else:
        dic[ch] = 1 # if ch appears for the first time

for key in dic:
    print(key, ':', dic[key])
Ln: 22 Col: 0
```

```
File Edit Shell Debug Options Window Help
Enter a string: reena
r : 1
e : 2
n : 1
a : 1
Ln: 77 Col: 0
```

```
File Edit Format Run Options Window Help
# Program 15.13. Function to convert a number to its name

def convert(num):
# numberNames is a dictionary of digits and corresponding number names
    numberNames = {0:'Zero',1:'One',2:'Two',3:'Three',4:'Four',\
                    5:'Five',6:'Six',7:'Seven',8:'Eight',9:'Nine'}

    result = ''
    for ch in num:
        key = int(ch) # converts character to integer
        value = numberNames[key]
        result = result + ' ' + value
    return result

num = input("Enter any number: ") #number is stored as string
result = convert(num)
print("The number is:",num)
print("The numberName is:",result)
Ln: 23 Col: 0
```

```
File Edit Shell Debug Options Window Help
Enter any number: 1562
The number is: 1562
The numberName is:  One Five Six Two
>>>
Ln: 90 Col: 0
```

In above program, numberNames is a dictionary holding digits (0 to 9) as key and their Name as corresponding values in a user-defined function convert(). Now name of each digit of the number entered by user is stored in the variable result by accessing the value from the dictionary using that digit as key of the dictionary.

CHECK YOUR PROGRESS

A. Multiple Choice Questions

1. Elements of a tuple are enclosed in (a) round brackets () (b) square brackets [] (c) flower brackets { } (d) bar brackets | |
2. What will be the output of the following code (a) 10 (b) 20 (c) 30 (d) 10,20,30


```
t=(10,20,30,40)
print(t[-3])
```
3. Which of the following statement creates an empty tuple? (a) t = Empty () (b) t = () (c) t = tuple () (d) t = empty_tuple()
4. What will be the output of the following code (a) <class 'int'> <class 'tuple'> (b) <class 'tuple'> <class 'tuple'> (c) <class 'tuple'> <class 'int'> (d) <class 'int'> <class 'int'>


```
t1 = (10)
t2 = (10,)
type(t1)
type(t2)
```
5. What will be the output of the following code (a) <class 'int'> (b) <class 'string'> (c) <class 'tuple'> (d) <class 'float'>


```
s = '10','20','30'
type(s)
```
6. What will be the output of the following code (a) 3 (b) 7 (c) 9 (d) 13


```
tuple1 = (3,5,7,9,11,13,15)
tuple1[5-2]
```
7. What will be the output of the following code (a) (1,2,3,4,5) (b) (1,2,8,4,5) (c) (1,8,3,4,5) (d) TypeError


```
tuple1 = (1,2,3,4,5)
tuple1[2] = 8
print(tuple1)
```
8. What will be the output of the following code (a) True (b) False (c) Error (d) Index=1


```
City = ('Bhopal','Patna','Lucknow')
'Patna' in City
```
9. What will be the output of the following code (a) ('Bhopal', 'Patna', 'Lucknow') (b) ('Lucknow', 'Patna', 'Bhopal') (c) True (d) False


```
City = ('Bhopal','Patna','Lucknow')
City[::-1]
```
10. What will be the output of the following code (a) 0 (b) 1 (c) 3 (d) 4


```
tuple1 = (1,2,3,1,4,1,5,1,7)
tuple1.count(1)
```
11. What will be the output of the following code (a) 0 (b) 3 (c) 4 (d) 0,3,5,7


```
tuple1 = (1,2,3,1,4,1,5,1,7)
tuple1.index(1)
```
12. What will be the output of the following code (a) 0 (b) 6 (c) '1','2','3' (d) TypeError


```
tuple1 = ('1','2','3')
sum(tuple1)
```

13. What will be the output of the following code (a) 1 (b) (1,2,3) (c) ValueError (d) TypeError
- ```
(a,b,c,d) = (1,2,3)
```
14. What will be the output of the following code (a) 5 (b) 10 (c) ValueError (d) TypeError
- ```
(a,b) = (10-5,10+5)
print(b-a)
```
15. A tuple inside another tuple is called a (a) nested tuple (b) singly tuple (c) doubly tuple (d) complex tuple
16. Which of the following statement creates an empty dictionary? (a) d = empty () (b) d = {} (c) d = dict[] (d) d = empty_dict{ }
17. What will be the output of the following code (a) 0 (b) 1 (c) 89 (d) 85
- ```
Dict1 = {'M':95,'R':89,'S':92,'T':85}
Dict1['R']
```
18. What will be the output of the following code (a) 0 (b) 89 (c) 100 (d) ValueError
- ```
Dict1 = {'M':95,'R':89,'S':92,'T':85}
Dict1['R'] = 100
print(Dict1['R'])
```
19. What will be the output of the following code (a) True (b) False (c) 'R' (d) ValueError
- ```
Dict1 = {'M':95,'R':89,'S':92,'T':85}
'R' in dict3
```
20. What will be the output of the following code (a) True (b) False (c) 'R' (d) ValueError
- ```
Dict3 = {'M':95,'R':89,'S':92,'T':85}
100 in dict3
```

B. State whether True or False

1. A tuple is an ordered sequence.
2. Repetition operation for Tuples is depicted by the symbol +.
3. The not in operator returns True if the element is not present in the tuple.
4. Slicing cannot be applied to tuples.
5. Loop control structures can be used to access the elements in a nested tuple.
6. Tuples can't be made keys of a dictionary.
7. In Python, a dictionary can have two same values with different keys.
8. The value of a dictionary can be accessed with the help of indices.
9. Dictionaries aren't ordered.
10. Keys of a dictionary may be any data type that is immutable.

C. Fill in the blanks:

1. Tuple is an _____ data type.
2. Elements of a tuple can be accessed using index values, starting from ____.
3. The _____ operator is used to check whether particular element is a part of tuple or not.
4. Python allows us to join tuples using _____ operator depicted by symbol +.
5. In dictionary, each key serves as the _____ and maps to a value.
6. Function _____ returns a list of tuples(key-value) pair of a dictionary.
7. Function clear () returns an _____ dictionary.
8. Dictionaries are _____.
9. The method _____ removes a random key-value pair from a dictionary.
10. Except _____, other operations like concatenation, repetition and slicing are not

supported by dictionaries.

D. Programming Questions

1. Consider the following tuples, tuple1 and tuple2:

```
tuple1 = (23,1,45,67,45,9,55,45)
```

```
tuple2 = (100,200)
```

Find the output of the following statements:

- `print(tuple1.index(45))`
- `print(tuple1.count(45))`
- `print(tuple1 + tuple2)`
- `print(len(tuple2))`
- `print(max(tuple1))`
- `print(min(tuple1))`
- `print(sum(tuple2))`
- `print(sorted (tuple1)) print(tuple1)`

2. Consider the following dictionary stateCapital:

```
stateCapital = {"MadhyaPradesh":"Bhopal", "Bihar":"Patna", "Maharashtra":"Mumbai",  
"Rajasthan":"Jaipur"}
```

Find the output of the following statements:

```
print(stateCapital.get("Bihar"))  
print(stateCapital.keys())  
print(stateCapital.values())  
print(stateCapital.items())  
print(len(stateCapital))  
print("Maharashtra" in stateCapital)  
print(stateCapital.get("Assam"))  
del stateCapital["MadhyaPradesh"]  
print(stateCapital)
```

3. Write a program to read email IDs of n number of students and store them in a tuple. Create two new tuples, one to store only the usernames from the email IDs and second to store domain names from the email IDs. Print all three tuples at the end of the program. **[Hint: You may use the function split ()]**
4. Write a program to input names of n students and store them in a tuple. Also, input a name from the user and find if this student is present in the tuple or not. We can accomplish these by:
- writing a user defined function
 - using the built-in function
5. Write a Python program to find the highest 2 values in a dictionary.
6. Write a Python program to create a dictionary from a string.
7. Write a program to input your friends' names and their Phone Numbers and store them in the dictionary as the key-value pair. Perform the following operations on the dictionary:
- Display the name and phone number of all your friends
 - Add a new key-value pair in this dictionary and display the modified dictionary
 - Delete a particular friend from the dictionary
 - Modify the phone number of an existing friend
 - Check if a friend is present in the dictionary or not
 - Display the dictionary in sorted order of names

Appendix I**Table 12.2 Commonly used functions in math module**

Function Syntax	Arguments	Returns	Example Output
math.ceil(x)	x may be an integer or floating-point number	ceiling value of x	>>> math.ceil(-9.7) -9 >>> math.ceil(9.7) 10 >>> math.ceil(9) 9
math.floor(x)	x may be an integer or floating-point number	floor value of x	>>> math.floor(-4.5) -5 >>> math.floor(4.5) 4 >>> math.floor(4) 4
math.fabs(x)	x may be an integer or floating-point number	absolute value of x	>>> math.fabs(6.7) 6.7 >>> math.fabs(-6.7) 6.7 >>> math.fabs(-4) 4.0
math.factorial(x)	x is a positive integer	factorial of x	>>> math.factorial(5) 120
math.fmod(x,y)	x and y may be an integer or floating-point number	x % y with sign of x	>>> math.fmod(4,4.9) 4.0 >>> math.fmod(4.9,4.9) 0.0 >>> math.fmod(-4.9,2.5) -2.4 >>> math.fmod(4.9,-4.9) 0.0
math.gcd(x,y)	x, y are positive integers	gcd (greatest common divisor) of x and y	>>> math.gcd(10,2) 2
math.pow(x,y)	x, y may be an integer or floating-point number	x^y (x raised to the power y)	>>> math.pow(3,2) 9.0 >>> math.pow(4,2.5) 32.0 >>> math.pow(6.5,2) 42.25 >>> math.pow(5.5,3.2) 233.97
math.sqrt(x)	x may be a positive integer or floating-point number	square root of x	>>> math.sqrt(144) 12.0 >>> math.sqrt(.64) 0.8
math.sin(x)	x may be an integer or floating-point number in radians	sine of x in radians	>>> math.sin(0) 0 >>> math.sin(6) -0.279

Table 12.3 Commonly used functions in random module

Function Syntax	Argument	Return	Example Output
random.random()	No argument (void)	Random Real Number (float) in	>>> random.random() 0.65333522

		the range 0.0 to 1.0	
random. randrange(x,y)	x and y are positive integers signifying the start and stop value	Random integer between x and y	>>> random.randrange(2,7) 2
random. randrange(y)	y is a positive integer signifying the stop value	Random integer between 0 and y	>>> random.randrange(5) 4

Table 12.4 Some of the function available through statistics module

Function Syntax	Argument	Return	Example Output
statistics.mean(x)	x is a numeric sequence	arithmetic mean	>>> statistics. mean([11,24,32,45,51]) 32.6
statistics.median(x)	x is a numeric sequence	median (middle value) of x	>>>statistics. median([11,24,32,45,51]) 32
statistics.mode(x)	x is a sequence	mode (the most repeated value)	>>> statistics. mode([11,24,11,45,11]) 11 >>> statistics. mode(("red","blue","red")) 'red'

Module 4:

Data Structure

Module Overview

In the modern world, data and its information have significance, and there are different implementations taking place to store it in different ways. Data is simply a collection of facts and figures, or set of values in a particular format that refers to a single set of item values. The data items are then classified into sub-items, which is the group of items that are not called the simple primary form of the item.

In the context of computers, the data structure is a specific way of storing and organizing data in the computer's memory so that these data can be easily retrieved and efficiently used when needed later. The data can be managed in many different ways, such as a logical or mathematical model for a particular organization of data is called a data structure.

There are two types of data structure – *Linear* and *Non-linear*. In linear data structure the elements are arranged in the linear order or in a sequence. The examples of the linear data structure are: *Arrays, Queues, Stacks, Linked lists*. In non-linear data structure the data is arranged with hierarchical relationship between different elements. The examples of the non-linear data structure are: *Tree and Graph*.

In this unit you will learn all these data structures with their memory representation and algorithms for traversal, insertion and deletion of elements.

Learning Outcomes

After completing this module, you will be able to:

- Describe the concepts of Data Structure.

- Describe the Linear Data Structures and its Algorithms.
- Describe the Non-Linear Data Structures and its Algorithms.

Module Structure

Session 1: Introduction to Data Structure

Session 2: Linear Data Structures

Session 3: Non-Linear Data Structures

Session 1: Introduction to Data Structure

Everybody of us are having one or other kind of identity card. Let us take an example of a unique ID Aadhaar card issued by the Government of India. In this card you can easily observe that there are lot of data items such as Name, Date of Birth (DOB), Gender as Male or Female, your photograph and Aadhar number which is unique. (Figure 1.1)



Fig. 1.1 Data structures used in Aadhaar card

In real life we always want to deal with different sets of values such as the name of a student or person. Every person living in India has an Aadhaar number, every city or village in India has a pin code and also every student will score some marks in examination. Whenever we are dealing with different situations then we are dealing with the different sets of values.

In this Chapter you will understand the concept of data structure, different types of data structures and operations performed on data structures.

1.1 Data Structure

Data Structure is the specialized means of organizing and storing data in computers to access the data easily. If the data is stored in organised manner then it is possible to perform operations on the data more efficiently. There are different ways to organize data and accordingly there are different types of data structures. Some examples of Data Structures are arrays, Linked List, Stack, Queue, etc. Data Structures are widely used in almost every aspect of Computer Science i.e. Operating System, Compiler Design, Artificial intelligence, Graphics and many more. It is the fundamental and key topic to begin with the software development. The software developers should have the sound knowledge of data structures. Computer programming languages are used to implement algorithms on computers.

Data Structures are the main part of many computer science algorithms as they enable the programmers to handle the data in an efficient way. It plays a vital role in enhancing the performance of software or a program as the main function of the software is to store and retrieve the user's data as fast as possible.

1.2 Basic Terminology

It is important to understand the basic terms before knowing to all about the data structures. The data items are classified into primitive and group data items.

Primitive data item – Data are the elementary value or the collection of values. The data elements or items that cannot be further divided are called as the primitive data elements. For

example, “Roll Number” of a student is a unique number and it cannot be further divided into different items.

Group data item – The data elements that can be further divided into data items are called as the group items. For example, the “Student Name” which can be further divided into three parts such as, First Name, Middle Name and Last Name, such data items can be called as a group data item.

Attribute and Entity – The data items with some property or attributes is called as “Entity”. This entity can be assigned numeric or non-numeric value. For example, for the property or attribute “Age”, you can assign any numeric values such as 17.

The entities with similar attributes are called as entity set. For example, students in a class, or employees in the organisation.

Every attribute of entity set has a range of values. For example, range of age can be from 0 to 100. The data with given attribute is called as information or processed data. The data of the student such as marks obtained in the subject can be processed to obtain the information in the form of result as “pass” or “fail”.

1.3 Elementary data organization

Data can be organized in linear data structure the elements are into hierarchy of fields, records and files. The Figure 1.2, shows the file containing three records with each record is having three fields. Every field has certain data value associated with it. Let us discuss these terms in detail.

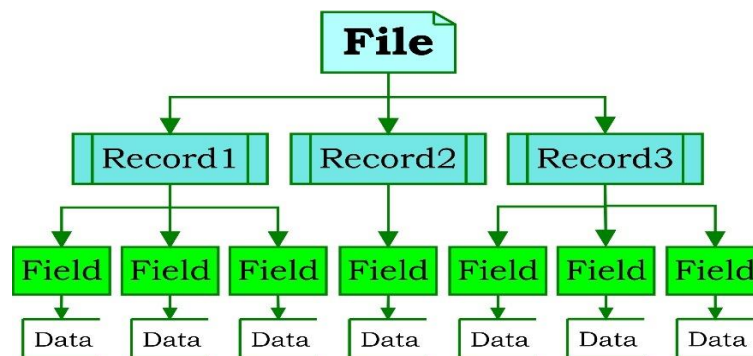


Fig. 1.2 Elementary data organization

Field – Field is a single elementary unit of information representing the attribute of an entity. There can be any number of fields for any single entity. For example, Roll number, Name of student, and Marks scored by the student are different fields.

Record – Record is a collection of field values. For example, for **student entity**, Roll Number, Name and Marks scored are the three different fields. Then the data value for these fields such as “4301, “Anil”, “71” form one record.

A record may contain many fields but any one field has a unique value that will not be repeated for other record is called as “Primary Key”. For example, Roll Number of a student cannot be repeated for other student and hence it is a unique and forms a primary key for this record.

File – A File is a collection of various records of one type of entity. For example, the data of several students for these three fields form the several records and collection of all the records will form the file.

A file can have a fixed length records or variable length records. Many times, the length of the records is fixed but it is also possible to have a variable length record. It is necessary to specify the minimum and maximum length for each field of the record. For example, we can have 4 digits for Roll Number, 20 characters for Name and 2 digits for Marks.

1.4 Need of Data Structures

Fields, Records and Files are the elementary data organisation techniques. But these techniques are not sufficient to process all type of data in many applications in real life. Hence, we require

more complex data structure such as arrays, stacks, queues, linked lists, trees, and graphs. As applications are getting complex and amount of data is increasing day by day, following problems may arise.

Processor speed – To handle very large amount of data, high speed processing is required, but as the data is growing day by day to the billions of files per entity, processor may fail to deal with that much amount of data.

Data Search – Consider an inventory size of 106 items in a store, if our application needs to search for a particular item, it needs to traverse 106 items every time, results in slowing down the search process.

Multiple requests – If thousands of users are searching the data simultaneously on a web server, then there are the chances that a very large server can be failed during that process

in order to solve the above problems, data structures are used. Data is organized to form a data structure in such a way that all items are not required to be searched and required data can be searched instantly.

1.5 Advantages of Data Structures

Efficiency – Efficiency of a program depends upon the choice of data structures. For example: suppose, we have some data and we need to perform the search for a particular record. In that case, if we organize our data in an array, we will have to search sequentially element by element. hence, using array may not be very efficient here. There are better data structures which can make the search process efficient like ordered array, binary search tree or hash tables.

Re-usability – Data structures are reusable, i.e. once we have implemented a particular data structure, we can use it at any other place. Implementation of data structures can be compiled into libraries which can be used by different clients.

Abstraction – Data structure is specified by the ADT which provides a level of abstraction. The client program uses the data structure through interface only, without getting into the implementation details.

1.6 Classification of Data Structure

Data Structures can be classified basically into two categories – *primitive* and *non-primitive* data structure as shown in Figure 1.3. The data structures that cannot be further divided into other sub-items are called as the primitive data structures or simple data structures or simply data types. For example, it is not possible to further divide the integer or real number. The primitive data structures have integer, real, character and Boolean data types. The non-primitive data structures are categorized in linear and nonlinear data structures.

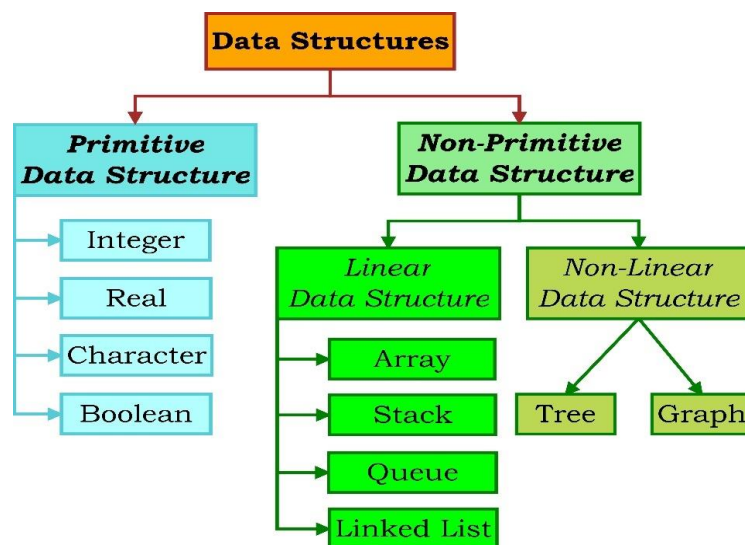


Fig. 1.3: Classification of data structure

1.6.1 Linear Data Structures

A data structure is called linear if all of its elements are arranged in the linear order or in a sequence. In linear data structures, the elements are stored in non-hierarchical way where each element has the successors and predecessors except the first and last element. For example, the stacks, queues and linked list are the linear data structure.

Types of Linear Data Structures are given below:

Arrays – An array is a collection of similar type of data items and each data item is called an element of the array. The data type of the element may be any valid data type like char, int, float or double.

The elements of array share the same variable name but each one carries a different index number known as subscript. The array can be one dimensional, two dimensional or multidimensional.

The individual elements of the array are:

age [0], age [1], age [2], age [3], age [98], age [99].

Stack – Stack is a linear list in which insertion and deletions are allowed only at one end, called **top**. A stack is an abstract data type (ADT), can be implemented in most of the programming languages. It is named as stack because it behaves like a real-world stack, for example: – piles of plates or deck of cards etc.

Queue – Queue is a linear list in which elements can be inserted only at one end called **rear** and deleted only at the other end called **front**.

It is an abstract data structure, similar to stack. Queue is opened at both end therefore it follows First-In-First-Out (FIFO) methodology for storing the data items.

Linked List – Linked list is a linear data structure which is used to maintain a list in the memory. It can be seen as the collection of nodes stored at non-contiguous memory locations. Each node of the list contains a pointer to its adjacent node.

1.6.2 Non-Linear Data Structures

This data structure does not form a sequence i.e. each item or element is connected with two or more other items in a non-linear arrangement. The data elements are not arranged in sequential structure. Trees and graphs are the nonlinear data structures.

Trees – Trees are multilevel data structures with a hierarchical relationship among its elements known as nodes. The bottommost nodes in the hierarchy are called **leaf node** while the topmost node is called **root node**. Each node contains pointers to point adjacent nodes.

Tree data structure is based on the parent-child relationship among the nodes. Each node in the tree can have more than one child except the leaf nodes. Each node must have parent node except the root node. Trees can be classified into many categories which will be discussed later in this unit.

Graphs – Graphs can be defined as the pictorial representation of the set of elements represented by vertices connected by the links known as edge. A graph is different from tree in the sense that a graph can have cycle while the tree cannot have the one.

1.7 Operations on data structure

It is possible to write computer programs to perform any operation on the data structure. For this it is required to write an algorithm. Algorithm is a set of steps written to perform the operation. Every algorithm is associated with complexity. The complexity of algorithm refers to the time and space requirement to execute that algorithm. The most common operations that can be performed on data structures are as follows.

Traversing – Every data structure contains the set of data elements. Traversing the data structure means visiting each element of the data structure in order to perform some specific operation like searching or sorting. For example, if we need to calculate the average of the marks obtained by a student in 6 different subjects, we need to traverse the complete array of marks

and calculate the total sum, and then we will divide that sum by the number of subjects i.e. 6, in order to find the average.

Insertion – Insertion can be defined as the process of adding the elements to the data structure at any location. If the size of data structure is n then we can only insert $n-1$ data elements into it.

Deletion – The process of removing an element from the data structure is called Deletion. We can delete an element from the data structure at any random location. If we try to delete an element from an empty data structure then **underflow** occurs.

Searching – The process of finding the location of an element within the data structure is called Searching. There are two algorithms to perform searching, Linear Search and Binary Search. These will be discussed later in this unit.

Sorting – The process of arranging the data structure in a specific order is known as Sorting. There are many algorithms that can be used to perform sorting, for example, insertion sort, selection sort, bubble sort, etc.

Merging – When two lists List A and List B of size M and N respectively, of similar type of elements, clubbed or joined to produce the third list, List C of size $(M+N)$, then this process is called merging.

Summary

- Data structure is a way to organize data in computers so that it can be accessed efficiently.
- Data can be organized into hierarchy of fields, records and files.
- Field represents an attribute of entity.
- Record is a collection of field values.
- File is a collection of records.
- A file can have a fixed length records or variable length records. The minimum and maximum length of records can be specified.
- Data structure is classified into primitive and non-primitive data structures.
- Primitive data structure are simple data types such as Integer, Real, Character and Boolean.
- Arrays, linked list, stacks and queues are the linear data structure as its elements form a sequence.
- Trees and Graphs are the non-linear data structure as its elements cannot be arranged sequentially.
- An array is a linear list of elements. It can have one or more dimensions.
- A Linked list is a list in which each element has a link to Next element. Link lists are more efficient than arrays.
- Stack is a liner list in which insertion and deletion takes place at one end only.
- Queue is a liner list in which insertion takes place at rear end and deletion takes place at front end.
- In Tree data structure has root and leaves that represents hierarchical relationship between elements.
- Graph is a pictorial representation of elements along with links. There may not be hierarchical relationship.
- Operations performed on data structures are: Insertion, Deletion, Traversing, Searching, Sorting, and Merging.
- Algorithms i.e. set of steps can be written to perform each operation. Complexity of algorithm refers to time and space required in terms of input size.

CHECK YOUR PROGRESS

A. Multiple choice questions

1. The specialized means of organizing and storing data in computers to access the data easily are (a) algorithm (b) flowchart (c) data structures (d) data flow diagram
2. The data items with some property or attributes is called as (a) Entity (b) File (c) Field (d) Record
3. Record is a collection of (a) Entity values (b) Files (c) Field values (d) Bytes
4. Which of the following is not a linear data structure (a) array (b) stack (c) queue (d) tree
5. Which is odd data structure (a) array (b) stack (c) queue (d) tree

B. Fill in the Blanks

1. The data structures that cannot be further divided into other sub-items are called as the _____ data structures.
2. The entities with similar attributes are called as _____
3. Data structure is called as _____ if its elements can be arranged in a sequence.
4. In _____ data structure, elements cannot be arranged in sequence.
5. File is a collection of _____.

C. State whether True or False

1. A file can have a fixed length records or variable length records.
2. In linked list access to data is random.
3. A tree is a hierarchical structure.
4. An array is a non-linear list of elements.
5. A file can have a fixed or variable length records.

D. Short answer questions

1. What is a data structure?
2. What are linear and non-linear data structures?
3. What are the common operations performed on data structures?
4. What do you mean by elementary data organization?
5. Write down various real-world applications where linear data structure is used?

Session 2: Linear Data Structures

Suppose you have prepared the list of items to purchase from the shop. If the number of items is not much more then you just remember it. When you visit the shop you just recall it from your memory or list prepared on the piece of paper to purchase these items. When you store such items in computer and retrieve it one by one, the similar process is carried out in the data structure. In linear data structure, the data elements are arranged sequentially or linearly where the elements are connected to the previous and next elements. As the elements are stored sequentially, they can be accessed or traversed in a single run. The data structure such as array, stack, queue, linked list are the linear data structures.

In this chapter, you will understand the linear data structure and different operations performed on them with algorithms.

2.1 Array

An array is a linear list of elements where all elements are of similar type. Array can be of one dimension or more than one dimension. It uses a script variable or index variable to refer to elements of array. The index of an array always starts with 0 and goes up to N-1 where N is size of the array. The starting index 0 is called as a *lower bound* and the highest index N-1 is called as an *upper bound*.

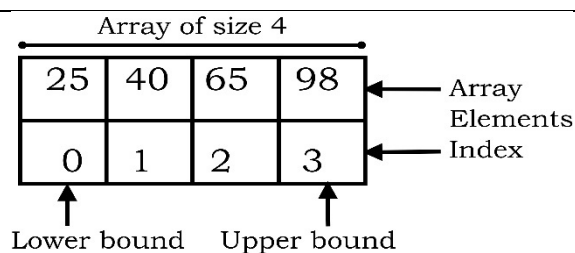


Fig. 2.1 Representation of array elements and index

Figure 2.1 shows the array of one dimension. There can be the different elements of the array and these elements of the array can be stored in successive memory locations. The index variable or subscript is used to refer the element of array. For example, A [3] refers to the 4th element of array A i.e. 98.

The two-dimensional array has two index variable for rows and columns to form a table. Such table values can be stored in memory by using two-dimensional array.

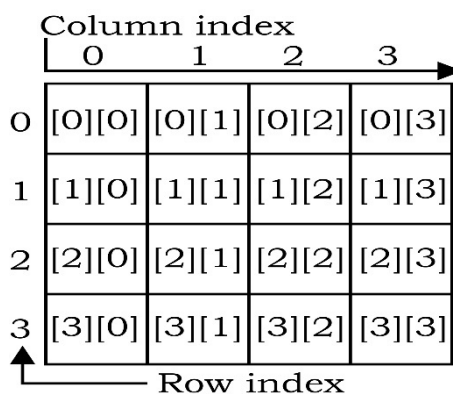


Fig. 2.2 Row and column index in two-dimensional array

Observe the Figure 2.2, the first element is in first row and first column, hence it has indices [0,0], second element is in first row and second column with indices [0,1]. The first index digit is row index and second index digit are the column index. For example, A [2][3] refers to the element of third row and fourth column.

2.1 Representing array in memory

The elements of array are stored in successive memory locations as shown in Figure 2.3. For example, the first element is stored in the memory location starting with address 1001. Suppose each element requires two memory locations, then the element will be stored in memory location 1001 and 1002 and the Next memory location 1003 will be available for second element. The value of first element is 20, the value of the second element is 50 and in this way many elements can be stored in the array.

Memory Address	Value	Index
1001	20	0
1002		
1003	50	1
1004		
1005	102	2
1006		
1007	600	3
1008		
1009	2	4
1010		
1011		

1012	34	5
1013	500	6
1014		
1015	100	7
1016		

Fig. 2.3 Array representation in memory

The first element memory address is called as a base address. In the above example the base address of the array is 1001. Suppose LA is the name of the Linear array, then the its base address can be written as Base (LA). It is possible for computer to calculate the location address of any element of array and for such calculation it is not necessary to visit each and every element of array. It can be calculated by using the following formula.

$$LOC(LA[K]) = \text{Base}(LA) + W(K - \text{Lower Bound})$$

Where W is number of words per memory cell, LA is linear array, and Base (LA) is base address of Linear Array LA.

For example, to find out the address of 4th element,

$$\begin{aligned} LOC(LA[4]) &= 1001(LA) + 2(4 - 1) \\ &= 1001 + 6 \\ &= 1007 \end{aligned}$$

So, the memory address of the fourth elements starts at memory location 1007.

2.2 Operations on Array

Arrays are used in many programming situations. The various operations performed on arrays include, traversing, insertion, deletion, searching, sorting and merging. Algorithm of these operations are explained here.

2.2.1 Traversing Linear Array

Traversing means accessing each element of array exactly once. The algorithm for traversing a linear array is given below.

Algorithm for traversing of linear array

1. [Initialize counter] Set K: = LB.
2. Repeat Step 3 and 4 while K <= UB.
3. [Visit element] Apply PROCESS to LA[K]
4. [Increment counter] Set K: = K + 1
5. [End of step 2 loop]
6. Exit

In the above algorithm of traversing of array, initially the counter K is set to lower bound means the lowest value of the index variable. Then the elements of the array are accessed one by one by incrementing the counter K till the counter has reached to the upper bound. This process will be repeated till the value of K is less than or equal to upper bound. Thus, it is possible to traverse or access array elements.

This process can be used for reading or printing the element or making some computation on the array elements.

The computation of complexity of this algorithm will be of the order n . If there are n elements in the array then it is necessary to visit each element once. This will increase the complexity of the order of n .

2.2.2 Insertion

Insertion means adding element into already created array. An element can be easily added at the end if the memory space is available. To add an element in between the two elements then it

is necessary to create a space between these two elements first. To create a space at particular position, it is required to shift the rest of elements down. The following algorithm inserts the element “*ITEM*” at the K^{th} position.

Algorithm for insertion of element in a linear array

(Inserting into a Linear Array) INSERT (LA, N, K, ITEM)

Here LA is a linear array with N elements and K is a positive integer such that $K \leq N$

This algorithm inserts an elements ITEM into the K^{th} position in LA.

1. [Initialize counter.] Set J: = N.
2. Repeat Steps 3 and 4 while $J \geq K$.
3. [Move J^{th} element downward] Set $LA[J + 1] := LA[J]$.
4. [Decrease counter.] Set J: = J - 1.
[End of step 2 loop.]
5. [Insert element.] Set $LA[K] := \text{ITEM}$.
6. [Reset N] Set $N := N + 1$.
7. Exit.

In the above algorithm LA is linear array and N is the number of elements in the array, ITEM is the value of the element of array to be added at the K^{th} position. So, K should be always less than or equal to N.

Initially set the counter variable J to N by equation $J = N$. Then move the J^{th} element to downward by one position to create the space in that position. This process will be repeated until the counter J is greater than or equal to the position K. This can be done by making the use of statement $LA(J + 1) = LA(J)$ and then decrease the counter J by 1. Then assign the new element ITEM to LA (K). This will insert the new value ITEM in the K^{th} position. Since a new element is inserted in the array, set the number of elements from N to N+1.

2.2.3 Deletion

Deletion means removing an element from the existing elements of the array. It is easy to delete the element at the end of array. To delete an element located in between the two elements then it is required to move the subsequent element in the side direction so that the vacancy that is being created can be filled. Algorithm to perform the deletion operation in array is as follows.

Algorithm for deleting an element in array

(Deleting from a Liner Array) DELETE (LA, N, K, ITEM)

Here LA is a liner array with N elements and K is a positive integer such that $K \leq N$

This algorithm deletes the K^{th} element from LA.

1. Set $\text{ITEM} := LA[K]$.
2. Repeat for J = K to N - 1:
[Move J + 1st element upward.] Set $J := LA [J + 1]$.
[End of loop]
3. [Reset the number N of element in LA] Set $N := N - 1$.
4. Exit.

In the above algorithm DELETE the arguments are LA the linear array, N is the number of number of elements in the array, K is the position of the element to be deleted and ITEM is the value of the element to be deleted.

Initially set the ITEM to K^{th} position. Now move the element of J+1 position. This process is repeated for J = K to N - 1. This will move the J+1st element upward. This is done by using the statement $LA(J) = LA (J+1)$. Then reset the number of elements of linear array from N to N - 1 as one element is deleted from the array and therefore it is necessary to reduce the number of elements from the array by one.

2.2.4 Searching in array

Searching is a process of finding an element in array. In many applications, it is required to find whether the element is present in the list or not. The searching algorithm is required in such cases. Complexity of search algorithm is measured in terms of number of comparisons be to made to find the element. Searching operation need to be performed several times in computer programming.

There are two types of searching – Linear search and Binary search.

a. Linear search

It is the simplest type of search operation. In this technique the search element is compared with each element of array until that element found in the list. If all the elements are compared and the search element is not found in the list then it is declared that the element is not present in the array. Linear search is also called as sequential search, because in this technique the search element is sequentially compared with the first, second up to last element in the list. For example, consider an array of characters A, B, S, O, L, U, T, E as shown in Figure 2.4 and we want to find whether the element “U” is stored in the array or not.

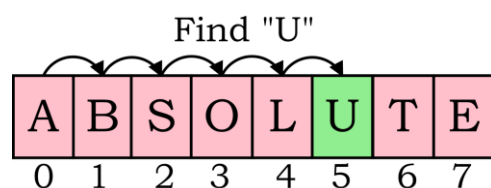


Fig. 2.4 Graphic view of Linear search

It is possible to find out the search element using linear searching. For this first the search element is compared with the first element. If it is not matched with the search element then it will be compared with the second element, then third element. This will be continued until the search element is matched with the element in the array. When the element is found in the list the search process will stop with the result that the element found in the particular location. But if the element is not found after comparing all the elements then the result is declared that the search element is not present in the list. Algorithm for this type of searching using linear search techniques is given below.

Algorithm for searching using linear search

Linear Search (Array A, Value x)

Step 1: Set i to 1

Step 2: if $I > n$ then go to step 7

Step 3: if $A[i] = x$ then go to step 6

Step 4: Set i to $i + 1$

Step 5: Go to Step 2

Step 6: Print Element x Found at index i and go to step 8

Step 7: Print element not found

Step 8: Exit

The algorithm “Linear Search” with arguments as Array A and Value x. In this algorithm initially, the control variable “i” is set to 1 and then check the value of I with the number of elements N in the array. If value of “i” is greater than the number of elements N in the array, then the search ends with the message that element not found in the list. In the third step the given value of search element x is compared with the first element in the array. If the value matches, then it will print the message that the search element is found in the particular location “i” of array. But if the value does not match then the control variable “i” will be incremented by 1 and again the process to find the search element will be repeated with the second element and so on until it will find the match with the search element or the control reaches to the last element of array. This algorithm is a simple algorithm. The search element will probably be found in the array by

comparing all the elements or in worst case the element may not found in the array and thus the complexity of the algorithm can be of the order of n , i.e. $O(n)$ as there are n elements in the array then it requires to check all the n elements if the search element is not found in the list.

b. Binary Search

It is the most popular type of search operation. In binary search method, the given array should be sorted array. In Binary Search, a sorted array is repeatedly dividing the search interval in half and if the value of the search key is less than the item in the middle of the interval, the interval narrow down the interval to the lower half. Otherwise, the upper half interval will be considered for searching the element. This process is performed repeatedly until the value is found or the interval reached to empty. For example, consider a sorted array with the elements as 1,2,3,4,5,6,7 as shown in Figure 2.5.

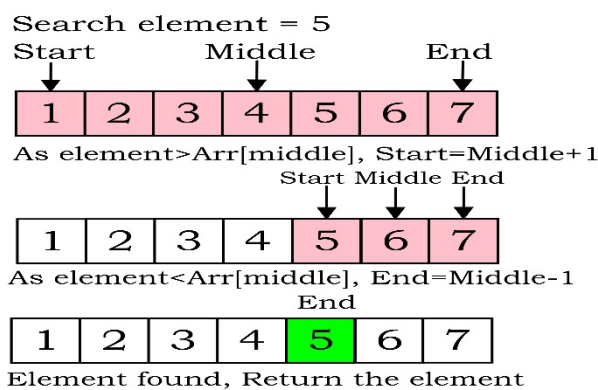


Fig. 2.5 Graphic view of Binary search

Suppose the search key is 5. Now in this array the start element is 1 and end element is 7. Hence the middle element will be 4th element. Then the value of the middle element is compared with the search key. It is found that the search key is greater than the middle element. Hence it is clear that the search element will be in the upper half part of the array. The upper part of the array consists of three elements, where the start element is 5, end element is 7 and middle element is 6. Then again, the search key “5” is compared with the middle element which is “6”. Now the search key is less than the middle element, hence the lower part of the search interval will be considered only to search the search element. The lower part of the array contains only one element i.e. 5. Hence the search key will be compared with the element of the lower part i.e. “5”. Now at this stage the match is found with the search element. Hence the result is declared that search element found in the array. This completes the successful searching of the search element. In this method, it is observed that the numbers of comparison are quite less than the earlier method and hence it is much faster than the “Linear Search”. Therefore the complexity of the algorithm is very low of the order of $\log n$ i.e. $O(\log n)$. If the array is very short i.e. the numbers of elements are very less in the array then only it is not possible to get the advantage of binary search method. But if the list is very large then this method is much faster than Linear Search method. Algorithm for binary search can be written in number of ways. One of the popular ways to write the algorithm for binary search is iterative method as follows.

Algorithm for Binary Search

First = 1

Last = N

Middle = $\lfloor \text{First} + \text{Last} \rfloor / 2$

While $\langle \text{First} \leq \text{Last} \rangle$

<

If $\langle \text{Array}[\text{Middle}] < \text{Key} \rangle$

First = Middle + 1

Else If $\langle \text{Array}[\text{Middle}] =$

Key \rangle

```

    <
        Print "Key Found at
Location"
        Break
    >
    Else
        Last = Middle - 1
        Middle = < First + Last > / 2
    >
    If < First > Last >
        Print "Not Found"
    Return

```

In this algorithm the first element is assigned the value 1 and last element is assigned the value N. The middle element is computed by using the formula $(\text{first} + \text{last})/2$. So initially all these three variables first, last and middle are computed. Then in loop will be executed till the first element is less than or equal to last. In the loop the array of middle is checked with the key. If it is less than the key then assigns the First to Middle + 1. Else if Array of Middle is equal to key then it displays that *"Element found in that location"*. Else the Last is assigned to Middle - 1.

If it is equal to key value then it is found that search of key element is successful and element is found at the particular location.

2.2 Stack

Many times, it is observed that books are kept on table one after another. In the same way, the plates are also placed on one another. This type of arrangement of placing of items on one another is known as stack. (Figure 2.6) It seems quite interesting that most of the concepts of computer are actually brought from the real-life situations. The concept of stack data structure is also taken from the real-life example of stack of books or plates.

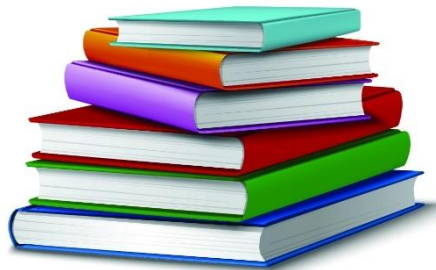


Fig. 2.6 Stack of books

A linear list that allows to insert and delete the element from one end only is called as stack. In the example of books arranged one after another in stack, a book can be removed from topmost book or any new book can be placed on the top of the stack of book. A stack is a container where first inserted element goes to the bottom as shown in Figure 2.7. Stack has a pointer known as stack pointer or *top* pointing to the topmost element of the stack.

Stack is called as a Last In First Out (LIFO) as the elements that are inserted in the end will be removed first from such list. In stack, the topmost element which is inserted last is processed first and first inserted element is processed in the last. Inserting the first book will go to the bottom and the second book will come on the top. Inserting the third book will bring that book on the top. Inserting an element into the stack called as a PUSH and removing an element from the stack is called as POP. A pointer keeps track of top element of stack.

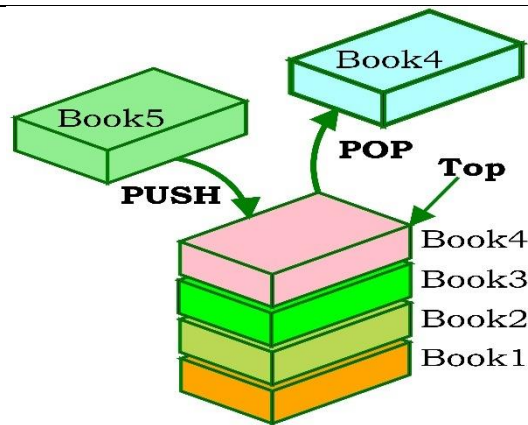


Fig. 2.7 Stack data structure

Applications of stack

Stack is very useful data structure in implementation of programs for many real-life applications using recursive functions, evaluations of arithmetic expressions and to solve the problems that make use of backtracking.

2.2.1 Representation of stack in memory

The stack data structure is mostly not inherently supported by the programming languages. It can be implemented using array or linked list. It is possible to access stack and elements through index variable of array. The value of the index variable can be assigned to the stack pointer or TOP. In Figure 2.8, an array represents the stack because insertion and deletion can be done at one end only TOP.

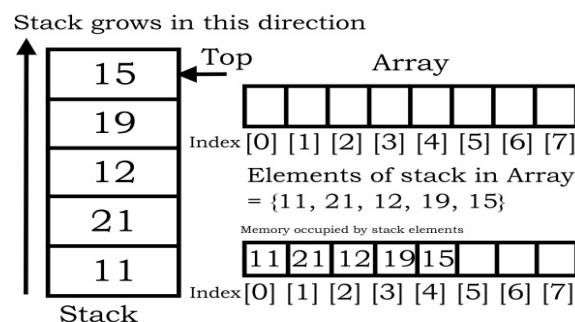


Fig. 2.8 Implementation of Stack using Array

To insert element in the stack, first it is required to increase the TOP pointer by 1 and to delete an element from the stack, decrease the TOP pointer by 1. Array restricts to create a stack of fixed size only. On the other hand, if the size of the stack is not known then the linked list is used to create the stack instead of array.

2.2.2 Creation of stack using array

A stack can be created using one dimensional array of specific size. The data items are inserted or deleted by using LIFO principle. A variable called TOP is used to insert or delete the element in stack. Initially the value of TOP is -1. It is incremented by 1 while inserting the element in the stack and decremented by 1 while deleting the element from the stack.

Operations on stack

Now let's understand some important operations that can be performed on the stack.

2.2.3 Push Operation

To insert or add a new data element into stack is called as push operation. Now in order to achieve this push operation, perform the following steps.

Algorithm: Push operation on stack

1. Check if stack is full. If yes then output an error

- and exit
2. If not, increment top to point to Next empty space
 3. Add data element to the stack location pointed by top
 4. Return success

First check whether there is space available for the new element. Check if the stack is full or not. If the stack is full, it is not possible to insert the element. It will output an error. If the Stack is not full then it is possible to insert the new element into the stack. In such a case, increment the TOP by one to point to the next empty space. Then add the data element to the stack location that is being pointed by the top pointer or stack pointer and ultimately will return the success.

Let us take an example of a empty stack as shown in Figure 2.9. The stack is initially empty with TOP equal to -1 . To add the new element in the stack, first increase the TOP by 1. Now value of TOP becomes 0. Then insert (PUSH) the first element "A". To add the second element, again increase the TOP by 1. Now value of TOP becomes 1. Then insert (PUSH) the first element "B". To add the third element, increase the TOP by 1. Now value of TOP becomes 2. Then insert (PUSH) the first element "C". In this way by using PUSH operation, new elements can be added in the stack until it becomes full.

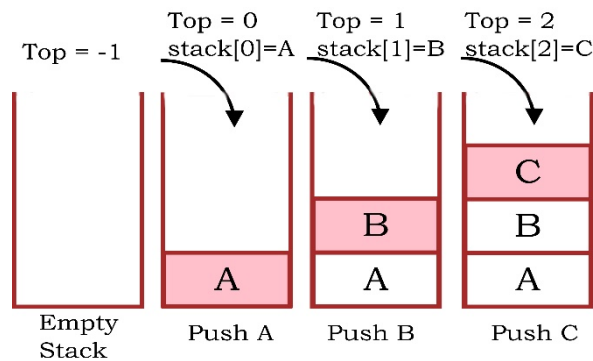


Fig. 2.9 Insertion (Push) operation on stack

2.2.4 Pop Operation

Deletion of an element from the stack is called as POP operation. The steps to perform for POP operation are given in the following algorithm.

Algorithm : Pop operation on stack

1. Check if stack is empty. If yes, then output an error and exit
2. If not, access data element at which top is pointing.
3. Decrease the value of Top by 1.
4. Return success

First, check the stack is empty or not. In empty stack it is not possible to perform the POP operation and hence it will give "*Underflow Error*". If the stack is not empty then, decrease the value of the TOP by 1, then POP is successful.

Let us take an example of a stack consisting of the elements A, B, C, where C is the TOP as shown in Figure 2.10.

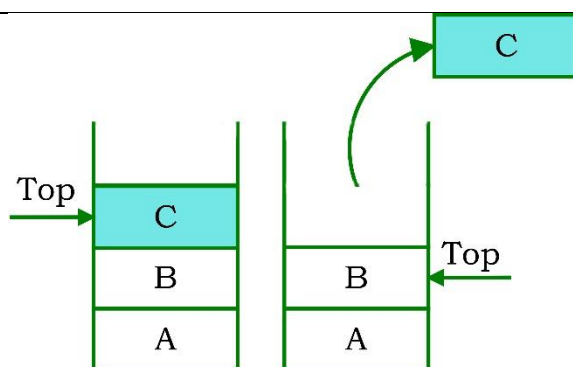


Fig. 2.10 Deletion (Pop) operation on stack

To remove element “C” from the stack, decrease the TOP pointer by 1. Now, it will point to B. In this way, C will be removed from stack. Space used by C can be made free. POP is successful.

2.2.5 Traversal Operation

Traversal operation visits the each element of stack at least once. The following are the steps to traverse/display the elements of a stack.

Algorithm : Traversal operation on Stack

1. Check whether stack is Empty or not.
2. If it is empty, then display "Stack is Empty!" and exit.
3. If it is Not Empty, then assign value of Top in var x.
4. Now Access data element at which x is pointing
5. Decrease the value of x by 1 until it becomes 0 (zero).
6. Return success

First, check the stack is empty or not. In empty stack it is not possible to display the items of hence it will give “Underflow Error”. If the stack is not empty then, copy the value of Top is one another X variable and display the value of stack at X. Decrease the value of X by 1 until it becomes -1 and display contents.

2.3 Queue

Whenever there are number of persons approaching for the same task such as purchasing a ticket at the ticket counter of bus, train or theater, they form a queue to process the task one by one as shown in Figure 2.11. A person who enters into the queue first will get ticket first and the person who enters in the queue at the last will get the ticket at the last. In many places a queue is observed in our daily lives.



Fig. 2.11 : Example of queue

Queue is linear data structure in which insertion of the elements takes place at one end and deletion takes place at another end. The new element is inserted in the rear end or back end and deletion of the element takes place from the front end. So this data structure is called as a First

In First Out list (FIFO). In this data structure the element that is inserted first will be deleted first, just as the person who enters in the queue first can exit first.

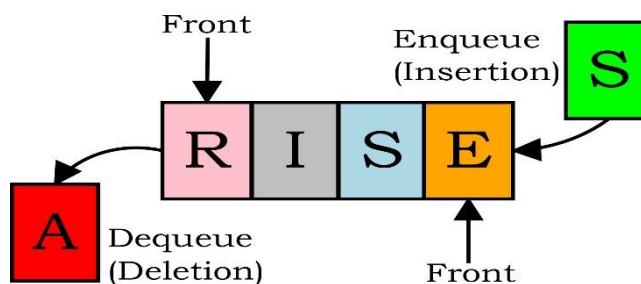


Fig. 2.12 Operations on Queue

There are two common operations performed on queue that are insertion and deletion, also called as *Enqueue* and *Dequeue*.

Enqueue – Whenever an insertion of element is performed at rear end, then this operation is called as *Enqueue*.

Dequeue – Whenever a deletion of element is performed from front end, then this operation is called as *Dequeue*.

Queue is a very useful data structure used to store data values temporary in the memory. Queue is used to write algorithms or programs to achieve sharing of the resources. For example, in the laboratory of school there is one printer shared among the number of students. Then such printer sharing is achieved by using the queue data structure.

2.3.1 Representation of queue using array

A queue can be represented by using linear array. There are two variables – Front and Rear that are used for insertions and deletions operations. Initially the value of front is -1 which indicates the queue is empty. Consider an array of five elements with values E, M, E, R, G, E. The Front has initial value 0, which will point to the element E. The Rear has value 5 that will point to the last character E. The representation of queue in memory using array is shown in Figure 2.13.

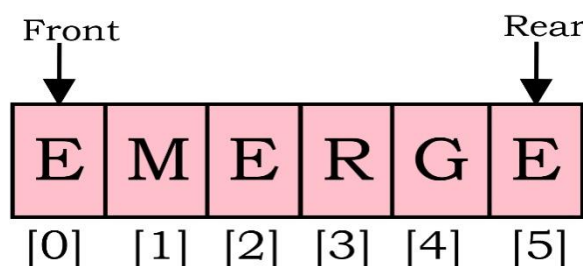


Fig. 2.13 Representation of queue using array

2.3.2 Insertion operation

Initially if the queue is empty and the first element is to be inserted in the queue then set the value of front and rear to 0 and insert the element at the rear end. To insert the new elements, keep on increasing the value of Rear by one and insert each element one by one with Rear as index. The insertion in the queue is possible until the queue becomes full. To check whether the queue is already full or not, the Rear is compared with $\text{MAX} - 1$ where MAX being the size of the queue. If queue is full, then return an error “*Overflow*”.

The algorithm for performing the insertion operation in the queue is as follows.

Algorithm: To insert an element in the queue

Step 1: If $\text{Rear} = \text{Max} - 1$
 Print “Overflow”
 Go to Step 4

[End of If]

Step 2: If Front = -1 And Rear = -1

Set Front = Rear = 0

Else

Set Rear = Rear + 1

[End of If]

Step 3: Set Queue [Rear] = Num

Step 4: Exit

Consider the above queue as shown in Figure 2.14. Now to insert the new element “D” in the queue, increase the Rear variable by 1 so that it becomes 6. Now insert the new element “D” in the queue, as shown in Figure 2.14.

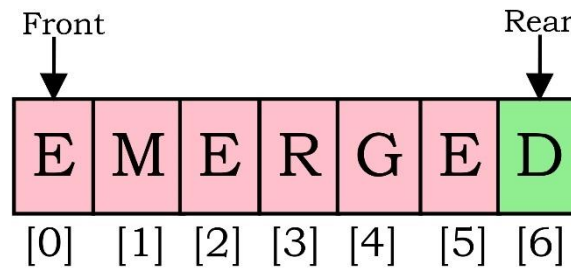


Fig. 2.14 Queue after inserting new element

2.3.3 Deletion operation on queue

It is also possible to perform deletion operation on queue. To perform deletion operation first it is required to check whether the queue is empty or not.

If, the value of front is -1 or value of front is greater than rear, then the queue is empty. Print the message “Underflow” and exit. Otherwise, keep increasing the value of front by 1 and return the element stored at the front end of the queue at each time.

The algorithm for performing the deletion operation in the queue is as follows.

Algorithm: To delete an element from the queue

Step 1: If Front = -1 or Front > Rear

Print “Underflow”

Else

Set Val = Queue [Front]

Set Front = Front + 1

[End of If]

Step 2: Exit

Consider the above queue as shown in Figure 2.15. To delete the element from the queue remove the first element and increase the Front variable by 1 so that it becomes 1. Observe that first element E gets deleted as shown in Figure 2.15.

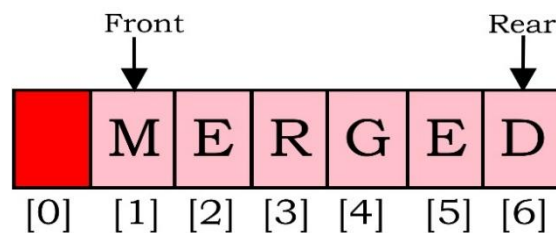


Fig. 2.15 Deletion operation on queue

2.3.4 D Queue

D Queue in short called as Deck is a double ended queue in which it is possible to insert the element from both the ends either from the frontend or from the backend of the queue as shown

in Figure 2.16. It is also possible to remove the element either from the frontend or from the backend of the queue. So D Queue is a linear list in which elements can be added or removed at both the ends, but it is not possible to add or remove the elements from the middle of the queue. This is a hybrid data structure that provides all capabilities of stack and Queue. Deck can be implemented by the doubly linked list or array. In D Queue the traversing of the element is much faster than that of the Queue. The time required in searching of the elements is much less in Deck than a normal queue. It is also possible to have the different versions of the D Queue such as input restricted deck or output restricted deck. In the input restricted deck insertion is allowed at one end and deletion is allowed at both ends. In the output restricted deck deletion is allowed at one end and insertion is allowed at both ends. These are the two forms of double ended queue.

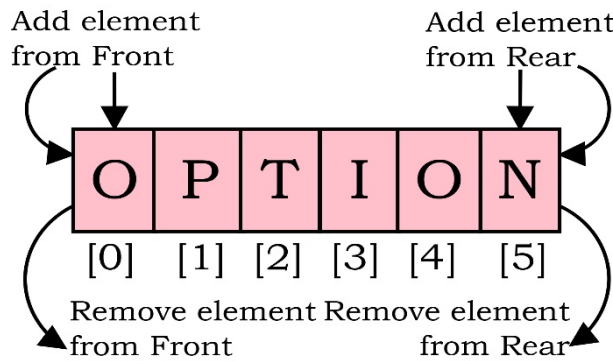


Fig. 2.16 D Queue

2.3.5 Priority Queue

The advantages of the double ended queue are that it can be used for the creation of priority queue. As you know that queue is FIFO list and whenever we have a queue and where there are some important jobs in queue and such job requires to wait until their turn comes in. To solve this problem, it is possible to assign the priority to each element. The element with the highest priority will be processed with the elements with the lowest priority. In this type of queue, the priority of the element of the queue will be taken into the consideration. The elements having the higher priority will be processed first and the element with the lower priority will be processed later. If there are two elements of the same priority than any suitable method can be used to process the element. You can make use of FIFO or process the element depending upon the value of the element. In time sharing system we can make use of this queue where programs with high priority will be processed first. It is possible to perform insertion and deletion operation on the priority queue.

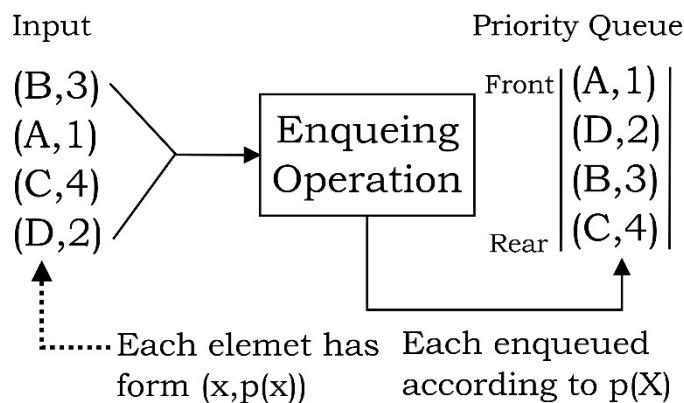


Fig. 2.17 Priority queue – insertion

As shown in the Figure 2.17, every element in the priority queue will be assigned with the priority level. The elements of queue are arranged in such as way that the elements with higher priority will go at the Front end and the element with the lower priority will go to the Rear end. The double ended queue can be used to implement the priority queue. Items in the priority queue are ordered by some key called as *priority index*. Items with the highest priority will be kept in

the Front end and the items of the lowest priority are kept in the Rear end for easy access to these elements.

2.4 Linked list

Suppose you entered in your class and you want to find a seat for sitting. In the class few seats are vacant in different locations. You cannot find a consecutive location to sit adjacent to your friend. To pass on your copy to your friend you need to know the chain of your few friends through which you can pass your copy to your concerned friend shown in Figure 2.18. This could be achieved by passing your copy to the next friend starting from you.



Fig. 2.18 Example of linked list

See how the chain of friends pointing to next friend becomes useful to transfer your copy to your concerned friend. This situation can be assumed as a linked list data structure in computer terminology. You can think of classroom as a memory, the virtual address space as the seats, and any particular memory address as a seat number.

You see, arrays are suitable when we need fast access. Like who's seating at seat number 13 can be answered in constant time. But arrays require you to declare a fixed size at the compile-time, due to which memory can be either wasted or fall short.

Linked list is a linear collection of the data elements, which are not stored in the successive memory location. So, it is possible to store these elements in different parts of the memory even though they are the part of the same list. Figure 2.19 shows the linked list.

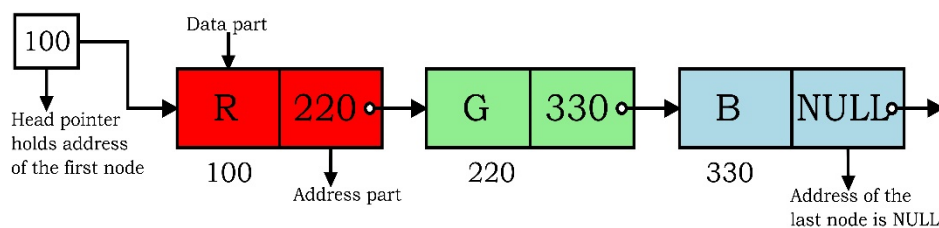


Fig. 2.19 Linked list

The linked list is suitable when we need to modify existing data by insertion and deletion because it doesn't have a fixed size. So, our memory consumption is determined at run time as the linked list shrinks and grows dynamically in constant time.

Every data element in the linked list will consist of a node and the node will have two parts. The first part is called as a data and other part is called as a pointer that points to the next element. This list is called as linked list because there is a linking between the different elements of the list. So linked list is a collection of nodes where every node will consist of a data and a pointer to the Next element.

The first element is called as a *Head* and the last element is called as a *Tail*. *Head* point to the first element of the linked list and *Tail* point to the last element. Since *Tail* is the last element whose pointer will not point to the Next element. Hence the pointer to *Tail* is always NULL.

Accessing the elements of linked list is much easier than that the arrays. In arrays, the computer needs to compute the location of each and every element but in case of linked list the node itself contains the address of the Next element. Therefore, making access to the elements of the linked list is quite easy and faster.

The memory requirement for the linked list is comparatively larger than that of array. In case of array only one form of memory location is required where the data values are stored. But in case of linked list two arrays (columns) are required. One column to represent the value of the element and the other column will represent the linked values or memory address values. Thus, the memory requirement of linked list is larger than that of array.

Linked list is the dynamic data structure. It is possible to increase the number of elements of the link list even during the program execution, which is not possible in case of array.

2.4.1 Types of linked list

There are different types of linked list as follows:

a. Singly Linked List – This is a basic linked list in which each node contains data and pointer to Next element. Last element pointer has NULL value as shown in Figure 2.20. In this type of linked list, traversing of the elements is possible only in forward direction only.

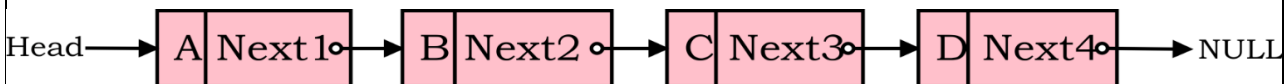


Fig. 2.20 Singly linked list

In this list the first element has the value “A” and “Next1” is the pointer to the Next element B. The second element is B and “Next2” is the pointer to C. In this way the final element will have a value D and its pointer will have a “NULL” value since there are no further elements in the list. It indicates that D is a final element of the linked list. This is the simplest type of list.

b. Doubly Linked list – In this type of linked list each node contains two pointers instead of one pointer. The first pointer is left pointer “Prev” and second pointer is right pointer “Next”. Figure 2.21 shows the two pointers to the first node i.e. left pointer is “NULL” and right pointer in “Next”. It is possible to traverse this list in both forward and backward direction.

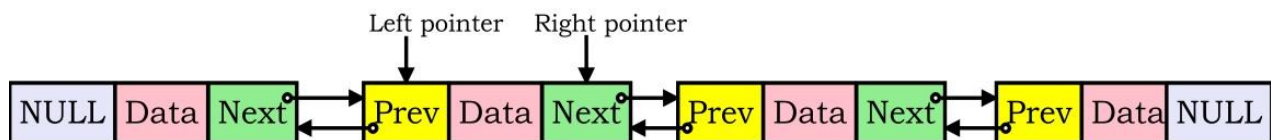


Fig. 2.21 Doubly linked list

Similarly, the next element D will have the left pointer “Prev” and right pointer “Next”. The traversing of this list is possible in forward as well reverse direction, i.e. we can visit each element of the list from both sides. In such list it is possible that leftmost element pointer “Prev” will have a “NULL” value or the rightmost element pointer “Next” will have a “NULL” value.

c. Circular Linked List – This is a singly linked list where last node points to first node. Figure 2.22 shows that the last node T is having the pointer “Next” that will point to the first element “L”. This forms a circular approach hence it is called as circular linked list.

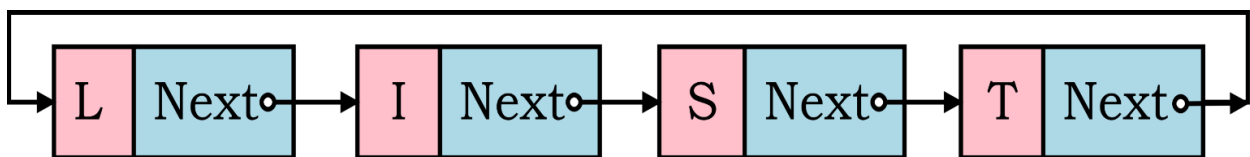


Fig. 2.22 Circular linked list

2.4.2 Representation of linked list in the memory

The advantage with the linked list is that it is not necessary to represent all the elements in the memory in successive memory location. They can be stored in any available free memory locations.

Each element of the link list requires two locations as data value and pointer value. So, to represent the link list in memory, two array (columns) are required. The first array “Data” is used to hold the data values and the second array “Next” is used to hold the pointer or memory address of the Next element.

For example, consider a linked list as shown in Figure 2.23. The linked list starts with the head node pointing to the first node. The first node with element “13” has a pointer pointing to the next node with element “15”. In the same way last node with element “127” has the NULL value as terminating node.

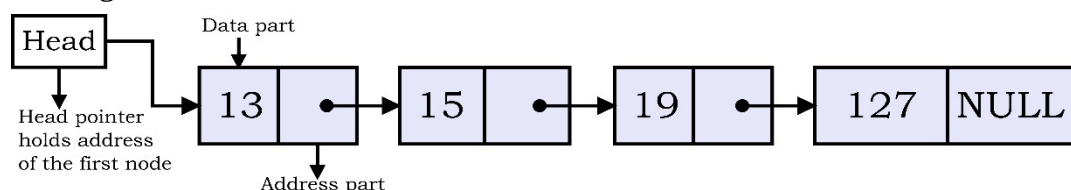


Fig. 2.23 Representation of linked list

The second node has value 15 and pointer value is 3045 which points to the third node. The third node has value 19 and pointer value is 4025, which points to the last element. The last element has value 27 and pointer value is NULL and hence it will not point to any node further. This linked list is represented in the memory as shown in the Figure 2.24.

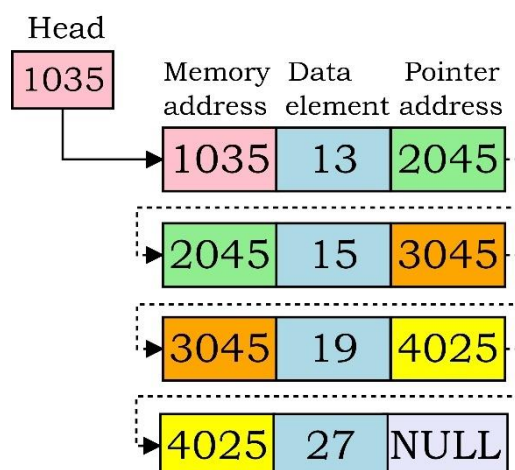


Fig. 2.24 Memory representation of link list

There are various operations that can be performed on the linked list are *Traversing, Insertion, Deletion, and Searching*.

2.4.3 Traversing a linked list

Traversing means accessing or visiting of each and every element of linked list at least once. Now suppose we have a start pointer or the head pointer which will point to the first element. Suppose we have an array called *Info* which contains the values and the array *Next* which will point to the Next element. Then it is possible to write the algorithm as given below to perform traversing operation for the linked list.

Algorithm : Traversing operation on linked list

1. if start = NULL
 Print “List is empty”
 Exit
2. Set Next = start

3. Repeat steps 4 and 5 until Next != NULL
4. Access and appl Next → info
5. set Next = Next → Next
 End repeat
6. Exit

In the above algorithm, if the start pointer is equal to NULL means the list is empty. Therefore, exit the algorithm. Otherwise set the pointer to “start” and repeat the Next two steps until the pointer value is not equal to NULL. Thus, each and every element of “info” can be accessed by using Next pointer. Then set the pointer to the *Next* and end the loop. In this algorithm it is observed that each and every element of the linked list is accessed or visited at least once.

2.4.4 Insertion in the linked list

Adding a new element in the linked list is called as insertion. Insertion of the element in the linked list can be done either at the beginning or at the end or at specific location of the linked list. To insert an element in linked list at any location, it is essential to create a new node. This new node can be inserted at the beginning or at the end or at the specific location of linked list. The algorithm is discussed below.

a. Insertion at the beginning of the linked list

Consider a linked list as shown in Figure 20.8 with three nodes having the values 5, 10 and 15. The pointer of the first node 5 will point to the second element, the pointer of the second element 10 will point to the third element and pointer of the third element 15 have a NULL value as it is the last element of linked list.

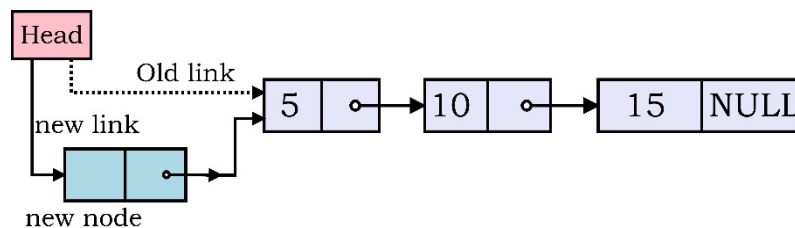


Fig. 2.25 Insertion at the beginning of the linked list

The head or start pointer points to the first element of the linked list. To insert a new element at the beginning of the linked list, the pointer of the new node should point to the element of the first node, so that the start pointer will point to the new node. The pointer of newly created node will point to the element of the first node i.e. 5, as shown in Figure above 20.8.

The algorithm to perform the insertion at the beginning of the linked list is given below.

Algorithm: Insert the node at the beginning of the linked list

- Step 1:** If Ptr = NULL
 Print “Overflow”
 Go To Step 7
 [End of If]
- Step 2:** Set New_Node = Ptr
- Step 3:** Set Ptr = Ptr → Next
- Step 4:** Set New_Node → Data = Val
- Step 5:** Set New_Node → Next = Head
- Step 6:** Set Head = New_Node
- Step 7:** Exit

As per the above algorithm, first check the memory space is available for a *New* node or not. If memory space is not available then print message “Overflow”. Once the memory space is

allocated to the *New* node, assign the head pointer to the new *New* node. Then assign the new node pointer to the first node of the linked list.

b. Insert the node at the end of the linked list

It is also possible to insert the node at the end of the linked list. To insert a node at the end of the linked list, consider a linked list consisting of three elements as shown in Figure 2.26. To insert node at the end of the list, it is required to break the link so that the last node should not have a pointer value equal to NULL. Instead this pointer value should be set to the *New* node. Allocate memory for the *New* node and assign the pointer value of the last node. For this perform the traversing operation through the linked list. To end up the list assign the pointer value of the *New* node to a NULL.

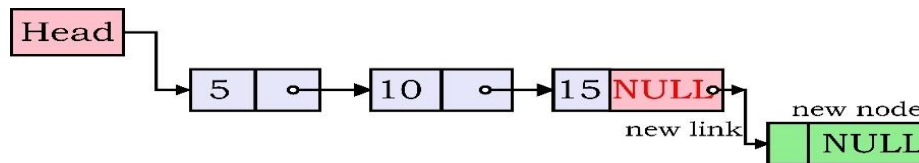


Fig. 2.26 Insertion at the end of the linked list

The steps of algorithm to insert an element at the end of the linked list is given below.

Algorithm : Insert node at the end of the linked list

- Step 1:** If Ptr = NULL
 Print "Overflow"
 Go To Step 1
 [End of If]
- Step 2:** Set New_Node = Ptr
- Step 3:** Set Ptr = Ptr -> Next
- Step 4:** Set New_Node -> Data = Val
- Step 5:** Set New_Node -> Next = NULL
- Step 6:** Set Ptr = Head
- Step 7:** Repeat Step 8 While Ptr -> Next != NULL
- Step 8:** Set Ptr = Ptr -> Next
 [End of Loop]
- Step 9:** Set Ptr -> Next = New_Node
- Step 10:** Exit

In the above algorithm, first step is to check whether there is a free space to create a *New* node. If no free space is available then print the message, "Overflow" and exit the program. If it is possible to create a *New* node then allocate the memory space to the *New* node. Then set the *Data* value for *New* node by assigning *Info* to *Data* and *Next* to NULL. The new node is the last element of the linked list and therefore it is necessary to set the pointer value of the *New* node to NULL. Then traverse the list to check if the lists consisting of an element or not. If the list is empty then this will be the first element. Otherwise traverse through the list by making the use of *Repeat - While* loop. Then at the end of the loop set the *Next* pointer to *New*. This pointer which is the last element of the linked list has to be set to the *New* node.

c. Insert an element into the linked list at a specific location

It is possible to insert an element at the specific location of the linked list. Consider a linked list whose *Start* or *Head* point to the first element as shown in Figure 2.27. It will consist of *Data* and the *Next* pointer. To insert a new element after the second element, it is required to break this link and create some *New* node. Then assign the pointer value of the second element to the *Data* value of the *New* node and then the set the pointer value of the *New* node to the *Data* value of the *Next* node. Thus, the *New* element get inserted after the second element in the linked list as third element and Previous third element becomes the fourth element of the list.

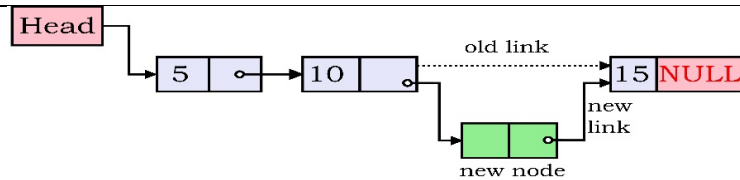


Fig. 2.27 Inserting node at a specific location of linked list

The steps of algorithm to insert an element at a specific location of linked list is given below.

Algorithm: Insert a Node after specified location

Step 1: If Ptr = NULL
 Print "Overflow"
 Goto Step 12
 End of If

Step 2: Set New_Node = Ptr

Step 3: New_Node → Data = Val

Step 4: Set Temp = Head

Step 5: Set I = 0

Step 6: Repeat Step 5 And 6 Until I

Step 7: Temp = Temp → Next

Step 8: If Temp = NULL
 Print "Desired Node Not Present"
 Goto Step 12
 End of If

End of Loop

Step 9: Ptr → Next = Temp → Next

Step 10: Temp → Next = Ptr

Step 11: Set Ptr = New_Node

Step 12: Exit

The above steps specify that you need to traverse the linked list up to the position – 1 node. You should know the position where you want to insert the new node. Once all position – 1 node are traversed then it is possible to allocate the memory and then data for the given nodes. Then point to the Next pointer of the new node to the Next of the current node. Next point the Next of the current node to the New Node. In this way, the pointer values of the position – 1 can be changed and the pointer value of the new node. Once these settings are done then it is possible to insert an element anywhere in between the linked list.

Summary

- Array is a linear data structure where all elements are of similar type can be accommodate only.
- Array uses a script variable or index variable to access elements which starts with 0 and goes up to N-1 where N is size of the array.
- A Two-dimensional array could have rows and columns.
- Any array elements are stored in successive memory locations.
- It is possible for computer to calculate the address of any element by using the following formula. $LOC(LA[K]) = Base(LA) + W(K - Lower\ Bound)$
- Various operation that could be performed of array data structure are Insertion, Deletion, Traversing, Searching, Sorting and Merging
- Traversing operation means accessing each element of array exactly once.
- Insertion operation means adding new element into already created array.
- Deletion operation means removing an element from the existing elements of the array.
- Searching operation is a process of finding an element in array.

- Searching operation can be done in two ways i.e. Linear search or Binary search.
- Stack is Last in First out (LIFO) list. Element inserted in end will be taken out first.
- In stack, insertion and deletion takes place at one end only.
- In stack insertion of element is called as Push and deletion of element is called as Pop operation.
- To insert element in stack, Top is incremented by one.
- To delete element from stack, Top is decremented by one.
- A pointer Top keep track of top element of stack.
- Stack is used in many applications such as Recursion, keeping track of function calls, evaluation of expression, servicing hardware interrupts and solving problems using backtracking.
- Stack can be implemented by using Array or Linked Lists.
- By using array, a stack of fixed size can be created.
- By using linked list, a stack of any size can be created.
- Queue is linear data structure in which insertion takes place at one end and deletion takes place at another end.
- A New element is inserted at Rear End (Back End) and existing element is deleted from Front End.
- Queue is also called as first in first out (FIFO) list.
- Two most important operations performed on queue are insertion and deletion.
- Queue can be represented by a linked list or an array.
- The start pointer of linked list is used as Front and Rear will store the address of last element in queue.
- If Front and Rear are equal to NULL then the queue is empty.
- For deletion front is increased by one and for insertion, rear is increased by one.
- Linked list is a linear collection of the data elements, which are not stored in the successive memory location.
- Every data element in the linked list will consist of a node which has two parts i.e data part and pointer part.
- The first part is called as a data part to store data values and second part is called as a pointer that points to the next element.
- *Head* is a variable that points to the first element of the linked list and *Tail* point to the last element of the linked list.
- Linked list is the dynamic data structure. It is possible to increase the number of elements of the link list even during the program execution, which is not possible in case of array.
- There are three variants of linked list like Singly, Double and Circular linked list.
- Singly Linked List is the basic linked list in which each node contains data and pointer to Next element.
- Doubly Linked list node contains one data part and two pointers instead of one pointer. The first pointer is left pointer is left pointer to indicate previous node and second pointer is right pointer to indicate "Next" node.
- Circular Linked List is the list where last node points to first node.
- The various operations that can be performed on linked list are traversing, insertion, deletion and searching.
- Insertion operation of new node could be in beginning, middle or in the last of the linked list.

CHECK YOUR PROGRESS

A. Multiple choice questions

1. An Array is (a) linear (b) non-linear (c) circular (d) volatile
2. The element of third row and fourth column is referred as (a) A [3][2] (b) A [3][3] (c) A [2][2] (d) A [2][3]
3. Accessing each element of data structure is called (a) insertion (b) deletion (c) traversing (d) merging
4. Operation to combine two or more elements to form a new array is known as (a) insertion (b) deletion (c) traversing (d) merging
5. The first element memory address is called as a (a) first address (b) base address (c) high address (d) low address
6. Stack pointer point to the _____ element of the stack (a) first (b) last (c) topmost (d) bottom
7. In programming, stack can be implemented using (a) array (b) queue (c) list (d) array or linked list.
8. In stack, insertion and deletion take place only at (a) front end (b) rear end (c) one end (d) both end
9. Stack follows (a) LIFO (b) FIFO (c) LILO (d) LOLI
10. Removing an element from the stack is called as (a) Push (b) Pop (c) Delete (d) Remove
11. A linear list that allows insertion of the elements at one end and deletion at another end is called (a) Stack (b) Queue (c) Tree (d) Graph
12. Queue follows (a) LIFO (b) FIFO (c) FILO (d) LOLI
13. Insertion of an item in a Queue is called (a) Push (b) Pop (c) Enqueue (d) Dequeue
14. Insertion of element in a Queue is performed at (a) rear (b) front (c) top (d) bottom
15. Removing an element from the queue is called (a) Push (b) Pop (c) Enqueue (d) Dequeue
16. Linked list is _____ data structure (a) Linear (b) non-linear (c) primary (d) volatile
17. Each element of a linked list is represented as (a) node (b) graph (c) stack (d) link
18. The first element of linked list is called as (a) head (b) tail (c) front (d) leg
19. Finding an element in the list (a) insertion (b) deletion (c) traversing (d) searching
20. In singly linked list, it is possible to traverse the elements only in (a) forward direction (b) backward direction (c) both A and B (d) None

B. Fill in the blanks

1. Array uses _____ variable to refer its elements.
2. Arranging data in ascending/descending order is called _____.
3. The complexity of traversing algorithm for an array will be of the order ____
4. In binary search method, the given array should be _____ array.
5. The complexity of the binary search algorithm is _____.
6. The pointer pointing to the topmost element of the stack is called _____.
7. Stack of variable size can be created using _____.
8. In a stack, element that is inserted in end will be taken out _____.
9. When we delete element from a stack, Top is _____ by one.
10. By using array, we can create a stack of _____ size.
11. If Front and Rear are equal to NULL then the queue is _____.
12. For insertion in a Queue, rear is increased by _____
13. When we delete element from a Queue, Front is _____ by one.
14. By using array, we can create a Queue of _____ size.

15. Queue can be implemented by the _____ list.
16. The last element of a linked list is called as _____.
17. Linked list is the _____ data structure.
18. Circular Linked List is a singly linked list where last node points to _____ node.
19. To represent the linked list in memory, _____ arrays are required.
20. Start pointer will point to the _____ element of the linked list.

C. State whether True or False

1. An element can be easily added at the end of array if the memory space is available.
2. Searching is a process of finding an element in array.
3. Binary search is also called as sequential search.
4. If the list is very large, then binary search is much faster than Linear Search.
5. Complexity of Linear search is $O(n^2)$.
6. In stack, first inserted element is processed in the last (T)
7. Array is used to create a stack of fixed size only. (T)
8. Push operation cannot be performed on empty stack. (F)
9. Pop operation on an empty stack will lead to overflow error. (F)
10. Stack is used in execution of recursive functions. (T)
11. In D Queue, the traversing of the element is much faster than that of the Queue.
12. The elements having the higher priority will be processed last.
13. In Priority queue, items of lowest priority are kept in the Rear end for easy access.
14. The double ended queue can be used to implement the priority queue.
15. Queue is used in process scheduling by CPU.
16. Accessing the elements of linked list is much easier than that of the arrays
17. In a singly linked list, Traversal of items can be done in the backward direction only.
18. The memory requirement for the linked list is less than that of array.
19. It is not possible to insert an element in the middle of the linked list.
20. If the start pointer is equal to NULL then the list is full.

D. Short answer questions

1. What is array?
2. Write algorithm for insertion in an array.
3. How to delete an element located in between two elements of an array.
4. What is Linear search?
5. Write algorithm for Binary search.
6. What do you mean by stack?
7. Write algorithm to perform Push operation in stack.
8. State the pop operation in stack with an example.
9. How stack is represented in memory?
10. Why stack is called LIFO based structure?
11. What do you mean by Queue?
12. Write algorithm to perform insertion operation in Queue.
13. State the deletion operation in Queue with an example.
14. What is D Queue?
15. Discuss the significance of Priority Queue?

16. What do you mean by Linked list?
17. Write algorithm to perform traversing operation for the linked list.
18. How to insert the element into the linked list at the end?
19. How linked list is represented in memory?
20. Differentiate between singly linked and circular linked list.

Session 3: Non-Linear Data Structure

Till now we have learned the linear data structures such as arrays, linked list, stack, and queue. Many times, in real life we need to deal with the data which are not arranged in sequential manner and also there is a need to access or retrieve the data in non-sequential manner. In such cases the data structure which are used are nonlinear data structures. In these data structures data items are not arranged in a sequence. As the arrangement is non-sequential, the data elements cannot be traversed or accessed in a single run. Non-linear data structures are not easy to implement in comparison to linear data structures. Tree and graph are the examples of Non-linear Data structures.

3.1 Tree

Tree is a nonlinear data structure. It is mainly used for representing the data which contain the hierarchical relationship among its elements. A tree is a popular non-linear data structure used in a wide range of applications. It is a collection of data (Node) which is organized in hierarchical structure recursively.

For example, in any organization there are many people working and there is some hierarchy maintained in the post of these people. The topmost post can be a chief executive officer (CEO) then middle managers working under CEO, then managers working under each director and then many more workers working under the managers as shown in Figure 3.1. Such hierarchical data can be easily handled by using a data structure called as tree.



Fig. 3.1 Hierarchical data structure tree

In tree data structure, every individual element is called as *Node*. Node in a tree data structure stores the actual data of that particular element and link to next elements in hierarchical structure. A tree with N number of nodes has the maximum N-1 number of links.

Records or table of contents can also be represented in the form of trees. A tree is a collection of zero or more nodes. Out of these nodes one node is called as a root and this tree can have either zero or one or more sub-trees.

Important key terms for tree data structure (Figure 3.2)

Path – Path refers to the sequence of nodes along the edges of a tree.

Root – The node at the top of the tree is called root. There is only one root per tree and one path from the root node to any node.

Parent – Any node except the root node has one edge upward to a node called parent.

Child – The node below a given node connected by its edge downward is called its child node.

Leaf – The node which does not have any child node is called the leaf node.

Sub-tree – Sub-tree represents the descendants of a node.

Traversing – Traversing means passing through nodes in a specific order.

Levels – Level of a node represents the generation of a node. If the root node is at level 0, then its next child node is at level 1, its grandchild is at level 2, and so on.

Keys – Key represents a value of a node based on which a search operation is to be carried out for a node.

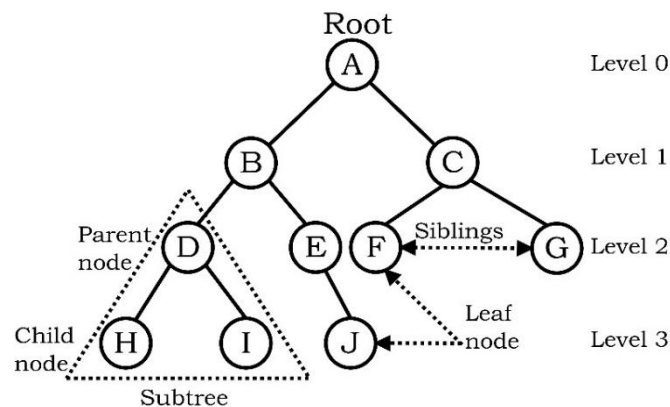


Fig. 3.2 Tree structure

In Figure 3.2 the node A is the ROOT of the tree having the child B and C. Node B further can have the child D and E. In the sub-tree, D is a parent node, H and I are the child node. When the node does not have any child then that node is called as a LEAF. In this Figure 3.2, H, I, J, F, G are the leaf nodes. Nodes which belong to same parent are called as SIBLINGS like F and G are siblings. The node which has at-least one child is called as INTERNAL Node. Here B, D, E, C are the internal nodes. All the nodes are interconnected with each other by edges.

3.1.1 Binary Tree

Binary tree is a tree with each node having at the most two children. It is not possible to have three children for any node in binary tree.

Figure 3.3 shows a binary tree with root as 1 and two child nodes as 2 and 3. Again the node 2 has two child nodes 5 and 6. The children nodes in the binary tree are named as Left child and Right child. In this example 5 is the left child and 6 are the right child of node 2.

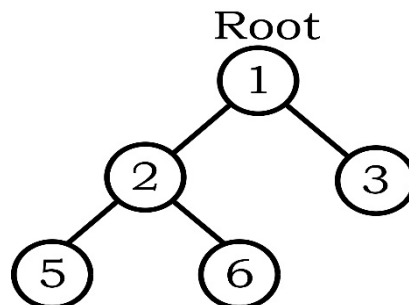


Fig. 3.3 Binary Tree

Each node of binary tree consists of three parts – *data*, *pointer to the left child* and *pointer to the right child*. Figure 3.4 shows the first node as the root which will have a pointer to the left child and pointer to the right child. So, if binary tree is having n nodes then it is confirmed that it will

have in $n-1$ edge. For example, consider a binary tree with five nodes 1, 2, 3, 5 and 6 as shown in Figure 3.4. Therefore, it will have 4 edges.

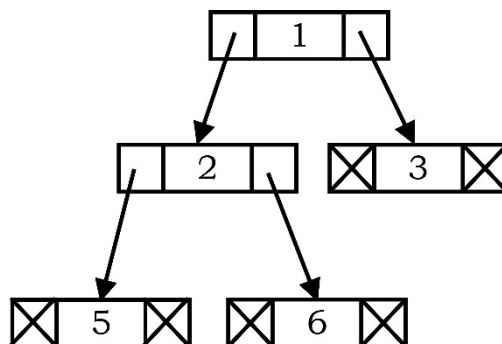


Fig. 3.4 Representation of binary tree using linked list

Every node has one parent and two nodes are connected by single path. For example, the parent of node 5 and 6 is 2. There is a single path for both the nodes 5 and 6.

If the height of tree is K then the number of nodes in the binary tree can be given by $2^{k+1} - 1$ for $K \geq 0$. If you consider the height of this binary tree as 3 then the number of nodes will be $2^3 - 1 = 7$. It means at the most this binary tree has seven nodes.

It is also possible to represent algebraic or mathematical expression by a binary tree. A linked list or array can also be used to represent the binary tree in the memory. Algebraic expression can be represented by binary tree. Binary tree is represented in memory by linked list or array.

3.1.2 Representation of Binary Tree in Memory

Let us understand how the binary tree is represented in memory. For example, consider a binary tree as shown in Figure 3.5.

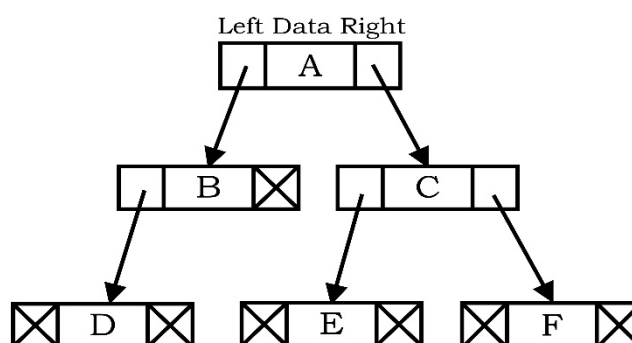


Fig. 3.5 Representation of binary tree using linked list

Suppose first node is the root of the tree with data value A. It will have the two pointers which will point to the left child and right child. By using the data value, left pointer and right pointer, it is possible to represent the binary tree using the linked list. The linked list is basically a collection of nodes and every node consists of the data, the left pointer and right pointer. The left pointer holds the address of the left child and the right pointer holds the address of the right child. If any sub-tree of a node is empty then the corresponding pointer of the node will hold a NULL value indicated by the cross sign as shown in Figure 3.5.

A binary tree can also be represented by an array. For example, consider a binary tree as shown in Figure 3.6. The value of root node is A in this binary tree. The root node with value A has the left and right child nodes with value B and C respectively. The node with value B has the left child with value D and no right child. The node with value C has the left child with value F and right child with value G. Such binary tree can be represented using array as shown in Figure 21.6.

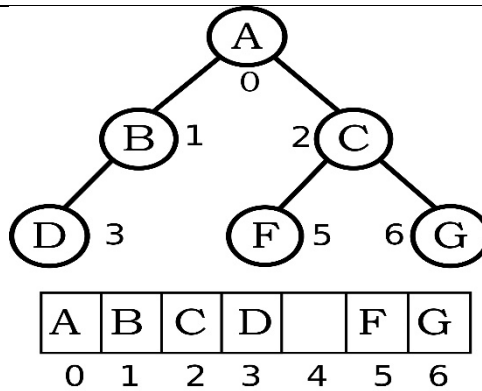


Fig. 3.6 Representation of binary tree using array

The root with value A is represented with the index 0, the Next value B is represented with index 1, C is represented with index 2 and D is represented with index 3. There is no right child for the node having value B which is at fifth location with index 4. Therefore, there will be vacancy over here. The nodes with value F and G are represented with the index 5 and 6 respectively.

When the tree is represented as array, the root is stored at the index value 0. Then the left and right child of the root are stored in the next subsequent locations. Thus, for a given height K , the size of the array will be is $2^{k+1} - 1$. Accordingly, the size of array should be declared.

3.1.3 Traversing of Binary Tree (Pre Order, Post Order and In Order)

Traversing operation of the binary tree is just visiting each and every element of the binary tree. It is possible to perform traversing operation of the binary tree by three ways – Pre-Order, In-Order and Post-Order.

a. Pre-order (NLR)

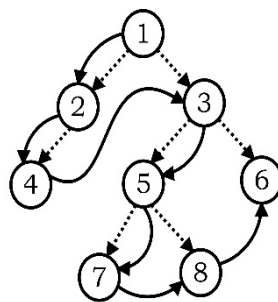
In pre-order, the sequence of visiting the element is first, node (N), then left sub-tree (L) and then right sub-tree (R). Algorithm for pre-order traversing is as follows.

Initially process the root R and then traverse the left sub-tree of R in the pre-order and then traverse the right sub-tree in pre-order. The function pre-order is called to process the sub-tree in pre-order. So, the algorithm for pre-order processing is as follows.

Algorithm for Pre-order (NLR) traversing

1. Process root N.
2. Traverse left sub-tree of R in pre-order. Pre-order (LST).
3. Traverse right sub-tree of R in pre-order. Pre-order (RST).

For example, consider a binary tree for pre-order traversing, as shown in Figure 3.7. The sequence of visiting each and every element of the binary tree in pre-order will be 1 then 2 then 4 then 3 then 5 then 7 then 8 then 6.



Preorder : 1,2,4,3,5,7,8,6

Fig. 3.7 Traversing of binary tree in Pre-Order (NLR)

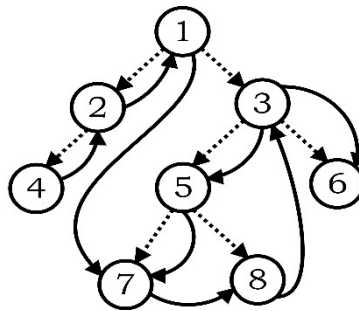
b. In order (LNR)

In in-order traversing, the sequence is the left sub-tree (LST) then the node (N) and then the right sub-tree (RST). Here the left sub-tree traverse first in in-order then process the root R and then traverse the right sub-tree in in-order. The function in-order is called to process the sub-tree in in-order. So, the algorithm for in-order processing is as follows.

Algorithm for Preorder (LNR) traversing

1. Traverse left sub-tree of R in in order. In-order (LST).
2. Process root N.
3. Traverse right sub-tree of R in in order. In-order (RST).

For example, consider a binary tree for in-order traversing, as shown in Figure 3.8. The sequence of visiting each and every element of the binary tree in In-order will be 4 then 2 then 1 and then move to 7 than 5 then 8 then 3 and then finally the 6.



Inorder : 4,2,1,7,5,8,3,6

Fig. 3.8 Traversing of binary tree in In-order (LNR)

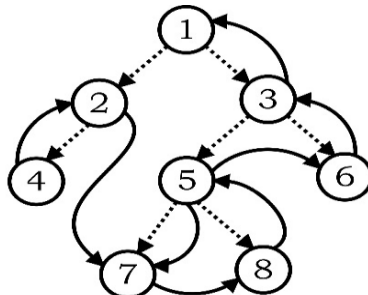
c. Post-order (LRN)

In post-order traversing, the sequence is the left sub-tree (LST) then the right sub-tree (RST) and then process the node (N). Here the left sub-tree (LST) traverse first in post-order then the right sub-tree (RST) traverse in post-order and then the root R is processed in post-order. The function post-order is called to process the sub-tree in post-order. So, the algorithm for in-order processing is as follows.

Algorithm for Post-order (LRN) traversing

1. Traverse left sub-tree of R in post-order. Post-order (LST).
2. Traverse right sub-tree of R in post-order. Post-order (RST).
3. Process root N.

For example, consider a binary tree for post-order traversing, as shown in Figure 3.9. The sequence of visiting each and every element of the binary tree in post-order will be 4 to 2 then 7 then 8 then 5 then 6 then 3 and then finally 1.



Postorder : 4,2,7,8,5,6,3,1

Fig. 3.9 Traversing of binary tree in Post-order (LRN)

It is easily observed that in all these three methods the recursive functions are used. In the algorithm the same function is called again. This method is called as recursion method. Also, whenever implementing traversing algorithm make use of stack.

3.2 Graph

Now a days it is quite common that all of us make use of social networks such as Facebook, Twitter and Linked-in to chat with others. Have you ever thought how the people on the social networks are connected to each other? You will see that all the people on the social network are connected to each other by some edges and this can be called as a graph as shown in Figure 3.10. So, we can say that the graph is a tool that is used for the connection of the people on the social network.



Fig. 3.10 Graph

Graph is a non-linear data structure consisting of *nodes* and *edges*. The *nodes* of the graph are called as the *vertices* and the *lines* or *arcs* of the graph are called as *edges*.

A graph is represented as $G = (V, E)$, where V is the set of vertices and E is the set of edges.

For example, a graph shown in Figure 3.11 has 5 vertices and all these vertices are connected to each other by the edges or lines. In this graph, the vertices are A, B, C, D and E and the edges are (A, B), (A, E), (B, C), (B, D), (B, E), (C, D) and (D, E).

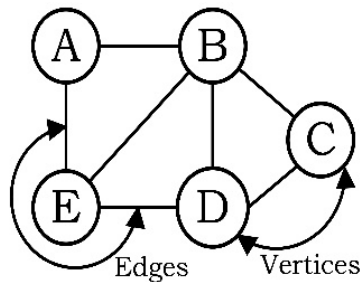


Fig. 3.11 Edges and vertices of graph

Graphs are very useful data structure to solve many real-world applications. They are used to represent network such as telecommunication network and circuit network. Graphs are used in social network where each person is represented by a vertex or node. The node will contain the information such as name, gender, and address of the person.

Google map also makes the use of graph. Every location on the Google map is represented by the vertices. Graph are used to find the shortest distance between any two locations on the Google map.

3.2.1 Terminologies related to Graph

Let us understand some terminologies associated with Graph in Figure 3.12.

Vertex – Each node in the graph is called as vertex.

Edge – Path between two vertices is called as edge and it is represented by adjacent vertices. For example, A is connected to the B so there is an edge between A and B similarly there is an edge between C and G as shown in Figure 22.3.

Weighted Edge – In many applications, each edge of a graph has an associated numerical value, called a weight. Usually, the edge weights are nonnegative integers. Weighted graphs may be either directed or undirected. Observe edge I to L has weight 42.

Parallel Edges – In a graph, if a pair of vertices is connected by more than one edge, then those edges are called parallel edges.

Adjacency – Two vertices are adjacent to each other when they are connected to each other by edge. For example, A and B are adjacent, A and D are adjacent and A and E are also adjacent.

Loop – Whenever an edge starts from some vertex and at the end reach to the same vertex then it forms a loop. For example, start from K to J, J to L, then L to H and then come back to K. This is called as a loop of the graph.

Self Loop – Whenever an edge of a graph which starts and ends at the same vertex, it forms a self loop to that vertex.

Path – A sequence of edges between two vertices is called as path. For example, K to J, J to L, then L to I is the path from K to I.

Connected Graph – when there is a path from any vertex to every other vertex then it is called as connected graph. When there is no any isolated vertex which is not connected to the other vertices then that graph can be called as a connected graph.

Degree of vertex – Number of incident edges is called as degree of vertex.

For example, A has 3 edges and therefore the degree of the vertex A is 3. Isolated vertex will not have any connection edge between the other vertices and hence it has degree 0. The pendent vertex has only one edge to the other vertex, hence it has degree 1.

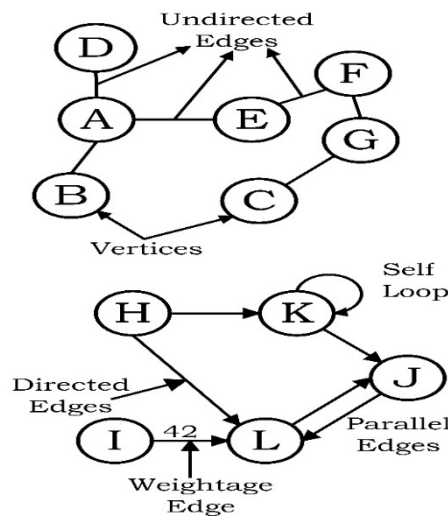


Fig. 3.12 Terminologies related to Graph

3.2.2 Types of graph

A graph can be undirected, directed and cyclic graph, as shown in Figure 3.13.

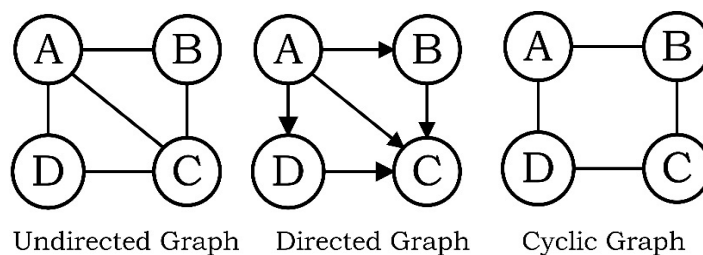


Fig. 3.13 Types of graph

In undirected graph the edges do not have a particular direction. In directed graph the edges will have a direction. In directed graph, it is possible to traverse the graph in particular direction. When it is possible to reach the same vertex after traveling or visiting the other vertex of the graph, then it is called as cyclic graph.

3.2.3 Representation of Graph in Memory

There are two methods to represent a graph in memory. The first method is called as the *sequential representation*. In this method, it is necessary to create the adjacency matrix for the given graph to represent it in memory.

Another way to represent the graph is the *linked representation*. In this method the graph is represented by making the use of the linked list.

The adjacency matrix A for graph G = (V, E) with n vertices. The adjacency matrix is n×n matrix, such that

$$A_{ij} = 1, \text{ if there is an edge from } V_i \text{ to } V_j \text{ and}$$

$$0, \text{ if there is no edge}$$

For example, consider a graph as shown in Figure 3.14. There is an edge from A to B, from A to E and therefore you will find that from A to B there is there is 1 represented in the adjacency matrix and from A to E the 1 is being represented while the others are represented by the number 0. Similarly, edge from B to A then B to E and B to C and therefore the corresponding element in the adjacency matrix is represented by 1 and other elements are represented by 0. In this way we can prepare the adjacent matrix for any given graph.

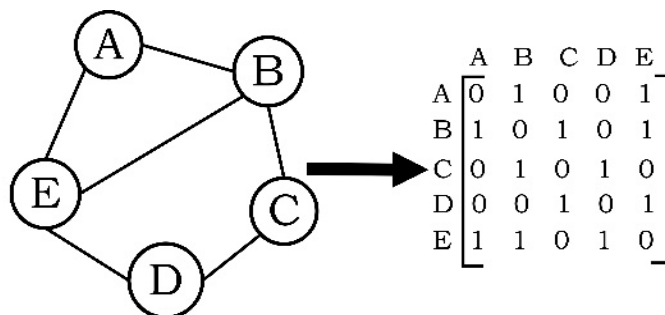


Fig. 3.14 Sequential representation of graph

In the *linked representation*, it is necessary to create the adjacency list and then represent the Graph by making the use of this list.

For example, consider a graph as shown in Figure 3.15, where A is connected to B and B is connected to D. Therefore, it can have a pointer from A to B and from B to D. Then the pointer of the D will hold a NULL value. Similarly, from B there are edges to A, D and C and therefore it can have a pointer from B to A, then to D and then to C. The pointer value for the node of C will have a NULL value. In this way the adjacency list for any given graph is prepared. This adjacency list can be represented by making the use of the linked list and this matrix can be represented by making the use of two-dimensional array.

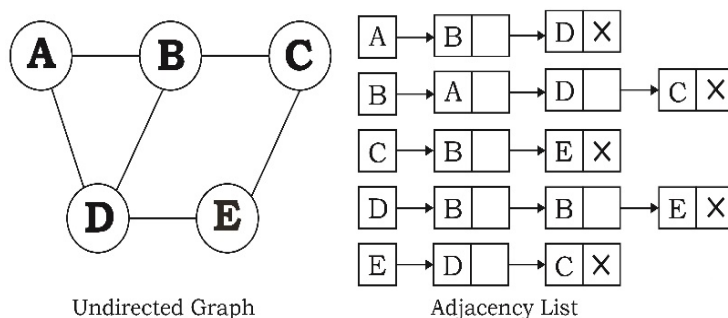


Fig. 3.15 Linked representation of graph

3.2.4 Operations of Graph

The operations that can be performed on graph includes, *Traversing, Searching, Insertion, Deletion, Merging of Vertices, Spitting of Vertices, Edge contraction.*

Traversing – This is the most common operation performed on the graph, where each and every node of the graph is visited. For example, consider a graph as shown in Figure 3.16.

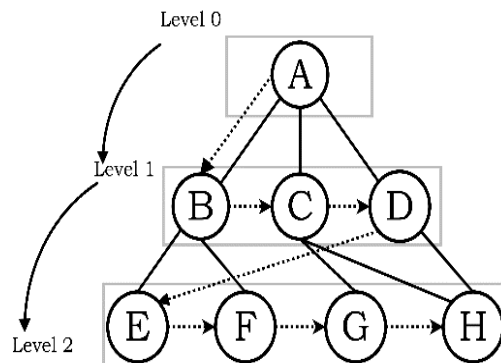


Fig. 3.16 Traversing a Graph

In this traversing operation, start from the level 0 with a root and then go on visiting each and every element. You can start from any node and then go on visiting each and every node of the graph. As shown in Figure 22.6, start from A then go to $B \rightarrow C \rightarrow D \rightarrow E \rightarrow F \rightarrow G$ and finally to H. In this way traversing of the graph is performed.

Summary

- Tree is a nonlinear data structure.
- A tree is a collection of zero or more nodes.
- All the nodes are interconnected with each other by edges.
- A sub-tree is just nothing but a part of the tree.
- Traversing operation of the binary tree is performed by three ways – Pre-Order, In-Order and Post-Order.
- Graph is a non-linear data structure consisting of *nodes* and *edges*.
- A graph is represented as $G = (V, E)$, where V is the set of vertices and E is the set of edges.
- Graphs are very useful data structure to solve many real-world applications.
- Graphs are used in social network where each person is represented by a vertex or node.
- Graph are used to find the shortest distance between any two locations on the Google map.
- Each node in the graph is called as vertex.
- Path between two vertices is called as edge and it is represented by adjacent vertices.
- Two vertices are adjacent to each other when they are connected to each other by edge.
- Whenever an edge starts from some vertex and at the end reach to the same vertex then it forms a loop in graph.
- Number of incident edges is called as degree of vertex.
- A graph can be *undirected, directed* or *cyclic graph*.
- There are two ways to represent Graph in memory, first is sequential representation and second is linked representation.
- Sequential representation uses adjacency matrix while linked representation uses linked list data structure.
- The Operations that can be performed on graph are Traversing, Searching, Insertion, Deletion, Merging of Vertices, Spitting of Vertices and Edge contraction.

CHECK YOUR PROGRESS

A. Multiple Choice Questions

1. Which of the following is non-linear data structure? (a) Stack (b) Queue (c) Array (d) Tree
2. Which of the following data structure is used to represent hierarchical relationship? (a) Stack (b) Queue (c) Array (d) Tree
3. Number of child nodes of a leaf node are (a) 0 (b) 1 (c) 2 (d) 3
4. A tree in which each node has at the most two children is called (a) Binary tree (b) Full tree (c) Complete tree (d) AVL tree
5. Height of a Balanced binary tree with n nodes is (a) $n+1$ (b) $2n$ (c) $n*n$ (d) $\log n$
6. Which of the following is non-linear data structure? (a) Stack (b) Queue (c) Array (d) Graph
7. Which of the following data structure is used to represent social network connections? (a) Stack (b) Queue (c) Tree (d) Graph
8. Each node in the graph is called (a) Edge (b) Vertex (c) Link (d) Connection
9. A sequence of edges between two vertices is called (a) Vertex (b) Path (c) Link (d) Connection
10. If there is a path from any vertex to every other vertex in a graph then it is called (a) connected graph (b) full graph (c) NULL graph (d) directed graph

B. Fill in the Blanks

1. Number of edges in a binary tree of n nodes is _____.
2. If the height is K then the number of nodes in the binary tree is _____.
3. In a balanced binary tree, both right and left sub-tree differ by _____ level.
4. In a degenerate binary tree, every internal node only has _____ child.
5. In pre-order, the sequence of visiting the element is first, root node, then left sub-tree and then _____.
6. Number of incident edges is called as _____ of vertex.
7. When it is possible to reach the same vertex after traveling or visiting the other vertex of the graph, then it is called as _____ graph.
8. Adjacency matrix represent the connection of _____ in a graph.
9. In adjacency matrix A of a Graph, $A_{ij} = \text{_____}$, if there is an edge from V_i to V_j .
10. It is possible to traverse the graph in particular direction in _____ graph.

C. State whether True or False

1. Root node of a binary tree may have three children.
2. Algebraic or mathematical expression cannot be represented by binary tree.
3. In a complete binary tree, all the levels are completely filled except the last level.
4. Node of a tree which has no parent node is root node.
5. A tree can have cycles.
6. Google map makes the use of the graph.
7. Computer network can be represented by a graph.
8. If there is no edge from a node to another node in a graph, corresponding entry in adjacency matrix will be 0.
9. Linked list can be used to represent a graph.
10. A graph cannot have cycles.

D. Short answer questions

1. What do you mean by Tree?
2. Write algorithm to perform Pre-order traversal in Binary Tree.
3. Discuss different types of Binary trees?
4. What is In-order traversal of a Binary tree?
5. State the representation of Binary tree in memory.
6. What do you mean by Graph?
7. State the representation of Graph in memory.
8. Discuss different operations on Graph.
9. Illustrate edge, path and loop in a graph with example.
10. Write the difference between directed and undirected graph.

Answer

Module 1: Software Construction Essentials

Session 1: Computer System Architecture

A. Multiple choice questions

1. (c) 2. (b) 3. (d) 4. (b) 5. (b) 6. (b) 7. (a) 8. (b) 9. (b) 10. (d) 11. (a) 12. (c) 13. (a) 14. (a) 15. (c) 16. (b) 17. (b) 18. (a) 19. (c) 20. (a)

B. Fill in the blanks

1. Tablet and iPad 2. Computere 3. logical 4. program 5. ALU 6. input 7. output 8. Read Only Memory 9. CPU 10. impact and non impact

C. True and False

1. (T) 2. (T) 3. (F) 4. (F) 5. (F) 6. (T) 7. (T) 8. (F) 9. (T) 10. (T) 11. (F) 12. (F)

Session 2: Data Representation in Computer

A. Multiple choice questions

1. (d) 2. (c) 3. (b) 4. (c) 5. (c) 6. (c) 7. (d) 8. (c) 9. (d) 10. (a)

B. Fill in the blanks

1. (machine code) 2. (bit) 3. (encoding) 4. (radix or base) 5. (24 bits) 6. (hexadecimal)

C. True and False

1. (T) 2. (F) 3. (F) 4. (T) 5. (F) 6. (T) 7. (F) 8. (T) 9. (F) 10. (F) 11. (T) 12. (T) 13. (F) 14. (F) 15. (T)

Session 3: Basic Concepts of Mathematics and Statistics

A. Multiple choice questions

1. (c) 2. (a) 3. (d) 4. (d) 5. (a) 6. (d) 7. (c) 8. (b) 9. (d) 10. (a)

B. Fill in the blanks

1. (mathematical, discrete) 2. (logical reasoning) 3. (both) 4. (Cartesian product) 5. (void) 6. (domain, range) 7. (relation diagram) 8. (set of premises, conclusion) 9. (Tautology) 10. (Contingency)

C. True and False

1. (T) 2. (F) 3. (F) 4. (F) 5. (F) 6. (T) 7. (T) 8. (T) 9. (F) 10. (T)

Session 4: Problem Solving Methods

A. Multiple choice questions

1. (d) 2. (d) 3. (c) 4. (b) 5. (c) 6. (c)

B. Fill in the blanks

1. 5. 2. algorithm 3. pieces 4. programming language 5. complexity 6. programmer documentation, user documentation 7. Selection 8. Syntax

C. True and False

1. (T) 2. (T) 3. (T) 4. (F) 5. (T) 6. (F) 7. (F) 8. (T)

Session 5: Programming Language Concepts

A. Multiple choice questions

1. (c) 2. (d) 3. (d) 4. (c) 5. (d) 6. (b) 7. (b) 8. (d) 9. (e) 10. (d)

B. Fill in the blanks

1. (computational) 2. (arithmetic, logical) 3. (expression) 4. (Integrated Development Environment)(LISP)

C. True and False

1. (T) 2. (F) 3. (T) 4. (T) 5. (T)

Module 2: Operating System and Computer Network**Session 1: Operating System**

A. Multiple Choice Questions

1. (b) 2. (a) 3. (d) 4. (d) 5. (c) 6. (b) 7. (c) 8. (c) 9. (d) 10. (c)

B. Fill in the blanks

1. operating system 2. kernel, shell 3. multiprogramming, multitasking
4. processors 5. desktop 6. Cut and Paste
7. Recycle Bin 8. files, sub folders 9. Executable 10. safeguard

C. State whether True or False

1. (T) 2. (T) 3. (T) 4. (F) 5. (F) 6. (T) 7. (F) 8. (T) 9. (T) 10. (F)

Session 2: Computer Networks

A. Multiple Choice Questions

1. (c) 2. (d) 3. (d) 4. (c) 5. (c) 6. (d) 7. (b) 8. (c) 9. (d) 10. (b)

B. Fill in the blanks

1. node 2. Ethernet, Gigabit Ethernet 3. 10 metres 4. Bridge 5. Modulator, DEModulator
6. analog signal, digital signal 7. MAC address 8. switches, routers 9. ethernet 10. multiple

C. State whether True or False

1. (T) 2. (F) 3. (T) 4. (F) 5. (F) 6. (F) 7. (T) 8. (T) 9. (F) 10. (F)

Session 3: Transmission Media and Network Protocols

A. Multiple Choice Questions

1. (d) 2. (c) 3. (c) 4. (a) 5. (d) 6. (d) 7. (d) 8. (d) 9. (a) 10. (d)

B. Fill in the blanks

1. 3 KHz to 900 THz 2. 2.402 GHz, 2.480 GHz 3. 255 4. Low
5. 1-2 Mbps 6. long-distance 7. Piconet 8. 255
9. IP address 10. lowest, highest 11. voice, television
12. short -distance 13. infrared light waves 14. low-cost 15. primary protocol

C. State whether True or False

1. (T) 2. (T) 3. (F) 4. (T) 5. (T) 6. (T) 7. (F) 8. (F) 9. (F) 10. (F)

Session 4: Network Security

A. Multiple Choice Questions

1. (d) 2. (b) 3. (a) 4. (d) 5. (d) 6. (b) 7. (a) 8. (c) 9. (d) 10. (a)

B. Fill in the blanks

1. unauthorized 2. Malicious 3. malicious, hamper 4. self-replicate

5. email, web, database 6. user data 7. external entity 8. external entity

9. detect, remove 10. browsing information

C. State whether True or False

1. (T) 2. (T) 3. (F) 4. (T) 5. (F) 6. (F) 7. (T) 8. (F) 9. (T) 10. (F)

Module 3: Python Programming

Session 1: Python Basics

A. Multiple Choice Questions

1. (c) 2. (a) 3. (d) 4. (d) 5. (b) 6. (b) 7. (d) 8. (d) 9. (d) 10. (a) 11. (d) 12. (b) 13. (d) 14. (a) 15. (d) 16. (c) 17. (b) 18. (c)

B. State whether True or False

1. (F) 2. (T) 3. (T) 4. (T) 5. (T) 6. (F) 7. (T) 8. (T) 9. (T) 10. (T) 11. (F) 12. (T)

C. Fill in the blanks

1. keyword 2. Identifiers 3. fixed 4. Expression 5. # 6. case 7. string
8. int () 9. float () 10. triple 11. Interactive, Script 12. Single

Session 2: Control Structures

A. Multiple Choice Questions

1. (d) 2. (a) 3. (a) 4. (c) 5. (b) 6. (a) 7. (a) 8. (a) 9. (a) 10. (a) 11. (b) 12. (c) 13. (b) 14. (a) 15. (d)

B. State whether True or False

1. True 2. False 3. True 4. True 5. True 6. True 7. True 8. True 9. False 10. True

C. Fill in the Blanks

1. flow of control 2. indentation 3. syntax 4. else 5. terminates 6. skips 7. nested 8. star, number 9. multidimensional 10. a==b:

Session 3: Functions

A. Multiple Choice Questions

1. (d) 2. (c) 3. (b) 4. (a) 5. (a) 6. (a) 7. (a) 8. (c) 9. (a) 10. (d)

B. State whether True or False

1. (T) 2. (F) 3. (F) 4. (T) 5. (T) 6. (T) 7. (T) 8. (F) 9. (T) 10. (F)

C. Fill in the Blanks

1. Modular 2. user-defined 3. argument 4. return 5. Standard Library
6. .py 7. import 8. dot(.) 9. random 10. from

Session 4: Strings

A. Multiple Choice Questions

1. (a) 2. (c) 3. (a) 4. (b) 5. (c) 6. (d) 7. (c) 8. (c) 9. (a) 10. (d)
11. (c) 12. (a) 13. (b) 14. (a) 15. (a) 16. (b) 17. (b) 18. (c)

B. State whether True or False

1. (T) 2. (T) 3. (T) 4. (F) 5. (F) 6. (F) 7. (T) 8. (T) 9. (T) 10. (F) 11. (T)

C. Fill in the Blanks

1. 0 2. + 3. * 4. not in 5. immutable 6. isalnum() 7. True

8. title() 9. index() 10. split() 11. swapcase() 12. reverse

Session 5: Lists

A. Multiple Choice Questions

1. (c) 2. (b) 3. (c) 4. (d) 5. (a) 6. (a) 7. (c) 8. (b) 9. (b) 10. (d) 11. (c) 12. (c) 13. (a) 14. (b) 15. (d)

B. State whether True or False

1. (T) 2. (T) 3. (T) 4. (F) 5. (T) 6. (F) 7. (F) 8. (T) 9. (F) 10. (T)

C. Fill in the Blanks

1. ordered 2. square 3. 0 4. last 5. list ()
 6. + 7. * 8. in 9. del 10. nested
 11. insert () 12. pop () 13 extend () 14. sort () 15. sorted ()

Session 6: Tuples and Dictionaries

A. Multiple Choice Questions

1. (a) 2. (b) 3. (c) 4. (a) 5. (c) 6. (c) 7. (d) 8. (a) 9. (b) 10. (d)
 11. (a) 12. (d) 13. (c) 14. (b) 15. (a) 16. (b) 17. (c) 18. (b) 19. (a) 20. (b)

B. State whether True or False

1. (T) 2. (F) 3. (T) 4. (F) 5. (T) 6. (F) 7. (T) 8. (F) 9. (T) 10. (T)

C. Fill in the Blanks

1. immutable 2. 0 3. in 4. concatenation 5. Index 6. items() 7.
 empty 8. mutable 9. popitem() 10. membership

Module 4. Data Structure

Session 1: Introduction to Data Structure

A. Multiple choice questions

1. (c) 2. (a) 3. (c) 4. (d) 5. (d)

B. Fill in the Blanks

1. primitive 2. entity set 3. Linear 4. hierarchical 5. records

C. State whether True or False

1. (T) 2. (F) 3. (T) 4. (F) 5. (T)

Session 2: Linear Data Structure

A. Multiple choice questions

1. (a) 2. (d) 3. (c) 4. (d) 5 (b) 6. (c) 7. (d) 8. (c) 9. (a) 10. (b) 11. (b) 12. (b) 13. (c) 14. (a) 15. (d) 16.
 (a) 17. (a) 18. (a) 19. (d) 20. (a)

B. Fill in the blanks

1. index 2. sorting 3. n 4. sorted 5. $O(\log n)$ 6. Top 7. linked list 8. first 9. decremented 10. fixed
 11. empty 12. one 13. incremented 14. fixed 15. linked 16. Tail 17. dynamic 18. first 19. two 20.
 first

C. State whether True or False

1. (T) 2. (T) 3. (F) 4. (T) 5. (F) 6. (T) 7. (T) 8. (F) 9. (F) 10. (T) 11. (T) 12. (F) 13. (T) 14. (T) 15. (T) 16.
 (T) 17. (F) 18. (T) 19. (F) 20. (F)

Session 3: Non-Linear Data Structure

A. Multiple choice questions

1. (d) 2. (d) 3. (a) 4. (a) 5. (d) 6. (d) 7. (d) 8. (b) 9. (b) 10. (a)

B. Fill in the blanks

1. $n-1$ 2. $2^{k+1} - 1$ 3. 1 4. one 5. right sub-tree 6. degree 7. cyclic 8. nodes 9. 1 10. directed

C. State whether True or False

1. (F) 2. (F) 3. (T) 4. (T) 5. (F) 6. (T) 7. (T) 8. (T) 9. (T) 10. (F)

PSSCIVE Draft Study Material © Not to be Published