



Higher Secondary Course

Computer Applications
Commerce

XI



Government of Kerala
Department of General Education

State Council of Educational Research and Training (SCERT) Kerala
2026



THE NATIONAL ANTHEM

Jana-gana-mana adhinayaka, jaya he
Bharatha-bhagya-vidhata
Punjab-Sindh-Gujarat-Maratha
Dravida-Utkala-Banga
Vindhya-Himachala-Yamuna-Ganga
Uchchala-Jaladhi-taranga
Tava subha name jage,
Tava subha asisa mage,
Gahe tava jaya gatha
Jana-gana-mangala-dayaka jaya he
Bharatha-bhagya-vidhata
Jaya he, jaya he, jaya he,
Jaya jaya jaya, jaya he.

PLEDGE

India is my country. All Indians are my brothers and sisters.
I love my country, and I am proud of its rich and varied heritage.
I shall always strive to be worthy of it.
I shall give my parents, teachers and all
elders, respect and treat everyone with courtesy.
To my country and my people, I pledge my devotion. In their well-
being and prosperity alone, lies my happiness.

Prepared by:

State Council of Educational Research and Training (SCERT)
Poojappura, Thiruvananthapuram - 695012 Kerala
website : scertkerala.gov.in. Email : scertkerala@gmail.com.
Phone : 0471-2341883. Fax : 0471-2341869
Designed by : SCERT, First Edition : 2026
Printed at : Centre for Advanced Printing and Training (C-apt),
Thiruvananthapuram-695013
© Department of General Education, Govt. of Kerala.
ISBN 978-93-47213-85-4



Foreword

The State Council of Educational Research and Training (SCERT), Kerala, is committed to providing learners with quality, relevant, and future-ready educational resources that align with the evolving demands of society and technology. In this context, the textbook Computer Applications – Commerce, Class XI has been thoughtfully prepared to introduce students to the foundational and applied aspects of computing in the modern world.

This book presents computing not merely as a subject of study, but as a practical tool for understanding data, information processing, software systems, and responsible technology usage. Beginning with the evolution and organization of computer systems, the text book carefully guides learners through core concepts such as data processing, software classifications, number systems, green computing, the social relevance of Free and Open-Source Software (FOSS). It also covers software development using python, computer networks, cyber security and artificial intelligence. The content has been structured in a learner-friendly manner with illustrations, activities, examples, and review questions that encourage conceptual clarity and analytical thinking.

Special attention has been given to ensure that students from a commerce background can appreciate the relevance of computing in business, data management, decision-making, and sustainable technological practices. Contemporary topics such as e-waste management, open-source movements, and modern data processing strategies are included to build awareness of the ethical and environmental responsibilities associated with technology.

The textbook also promotes experiential learning through activities, case studies, and practical tasks that help students connect theoretical knowledge with real-life applications. By integrating historical perspectives with present-day technological advancements, this book aims to nurture digitally literate, environmentally conscious, and socially responsible learners.

It is hoped that this textbook will serve as a valuable resource for students and teachers alike, fostering curiosity, creativity, and competence in the field of computer applications.

Dr. Jayaprakash R K
Director
SCERT, Kerala

TEXT BOOK DEVELOPMENT TEAM

Chairperson

Dr. Sabu M K

Professor, Department of Computer Applications, Cochin University of Science and Technology, Ernakulam

Expert

Dr. N S Sreekanth

Professor
Department of Information Technology, Kannur University

Participants

Biswas K C

Sree Narayana HSS , Okkal, Ernakulam

Sunilkumar Mathoden

KPRGS Govt.HSS, Kalliassery, Kannur

Mukesh V K

Govt.HSS Chalissery, Palakkad

S M Manoj Murali

SNDPY HSS Neeravil, Kollam

Gigeeshkumar M V

Govt.Girls HSS Mavelikara, Alappuzha

Binoj Chacko

J J Murphy Memorial HSS, Yendayar, Kottayam

Midhunlal B

VHSS Kadavathur, Kannur

Sudheer P S

AKNMMAM HSS Kattukulam, Thiruvazhiyode, Palakkad

Sreejith P

CJHSS Chenmanad, Kasargode

Language Experts

Dr. Joyson Jose

Department of English, Bharata Mata College, Thrikkakara, Ernakulam

Sony Mathew

Department of English, Government College, Thrippunithara, Ernakulam

Artist

Radhakrishnan P

Retired Principal, Sree Krishna HSS, Guruvayur, Thrissur

Academic Coordinator

Dr. Renjith Subhash

Research Officer, SCERT Kerala



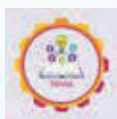
State Council of Educational Research and Training (SCERT)
VidyaBhavan, Poojapura, Thiruvananthapuram 695 012

CONTENTS

1	Computing System	7-38
2	Data Representation	39-58
3	Programming Concepts	59-80
4	Elements of Python	81-100
5	Python Control Structures and Data Handling	101-132
6	Computer Networks	133-162
7	Cyber Security	163-180
8	AI and its Applications	181-206



Icons used in this textbook for convenience



Trivia



Try Yourself

THE CONSTITUTION OF INDIA

PREAMBLE

WE, THE PEOPLE OF INDIA, having solemnly resolved to constitute India into a ¹**[SOVEREIGN SOCIALIST SECULAR DEMOCRATIC REPUBLIC]** and to secure to all its citizens :

JUSTICE, social, economic and political;

LIBERTY of thought, expression, belief, faith and worship;

EQUALITY of status and of opportunity; and to promote among them all

FRATERNITY assuring the dignity of the individual and the ²[unity and integrity of the Nation];

IN OUR CONSTITUENT ASSEMBLY this twenty-sixth day of November, 1949 do **HEREBY ADOPT, ENACT AND GIVE TO OURSELVES THIS CONSTITUTION.**

1. Subs. by the Constitution (Forty-second Amendment) Act, 1976, Sec.2, for "Sovereign Democratic Republic" (w.e.f. 3.1.1977)
2. Subs. by the Constitution (Forty-second Amendment) Act, 1976, Sec.2, for "Unity of the Nation" (w.e.f. 3.1.1977)



1

Computing System



“Computer is not a device any more. It is an extension of your mind and your gateway to other people.”

Mark Shuttleworth.

1.1 Introduction

The journey of computing devices spans thousands of years from simple counting tools to modern supercomputers and smart phones. Over a relatively short period, computing devices have progressed from performing basic calculations to handling complex data processing tasks with incredible speed. Computers today are fast, compact and powerful. But they were not always like this. This chapter covers the historical evolution, technical organisation, concepts of data processing, software classifications, importance of open source software, green computing and issues of e-waste. Studying this evolution and other aspects helps us to realise the development of technology and its ongoing influence on our lives. Now the development is like *‘We dream in the morning, a reality by noon and obsolete by evening’*.

1.1.1 Evolution of Computing

The word compute means to ‘calculate’. Computing devices have transformed from basic calculators to advanced data processors in just a few decades. The *Abacus* is considered to be the first known calculating tool. It uses beads on rods to perform basic arithmetic operations like addition and subtraction. It has been in use for thousands of years across different cultures. *Charles Babbage*, who is known as the *‘Father of the Computer’*, designed a mechanical general-

purpose computer in 1837. Although it was never fully built in his lifetime, the design laid the foundation for modern computing. In 1843, *Augusta Ada Lovelace*, wrote the first algorithm intended for a machine (Charles Babbage's Analytical Engine). She is therefore often regarded as *the first computer programmer*. Although the Analytical Engine was never built in her lifetime, her contributions laid the foundation for the development of computational theory. In 1890, *Herman Hollerith* invented a punch-card-based machine used in the U.S. census. It marked the shift towards data processing. The evolution of computing devices is shown in Figure 1.1

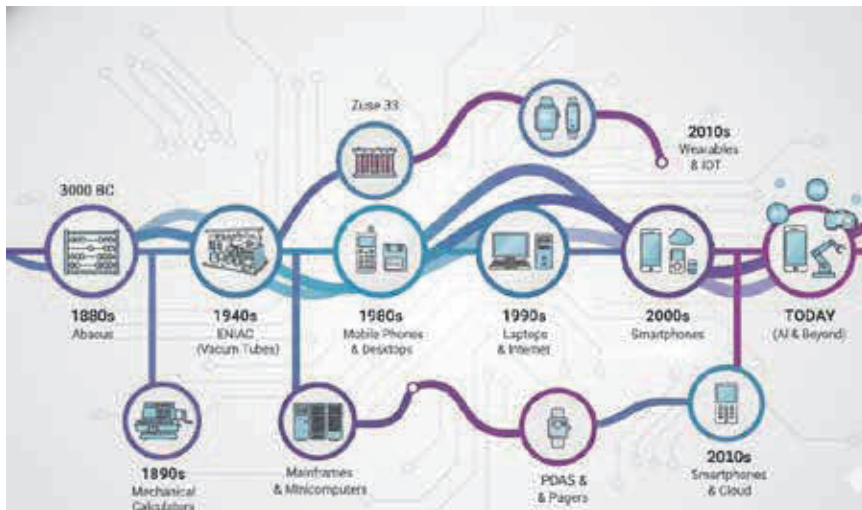


Figure 1.1 - Timeline diagram of generation of computing devices

1.1.2 Generations of Computing

The *Generations of Computing* refers to the classification of computers based on major technological developments over different periods. These developments increased the speed, reduced the size and cost, and enhanced the efficiency of computers. The evolution of computers is classified into *five generations*.

i) First Generation (1940–1956)

This era marks the beginning of modern electronic computing. The technology used was *vacuum tubes* for circuitry, and magnetic drums for memory. These tubes were easily broken, generated a lot of heat, and consumed a huge amount of power. Computers were bigger in size, often occupied the entire room. Programming was done with the help of machine language in which instructions are written using 0s and 1s. *ENIAC* (Electronic Numerical Integrator and Computer) is considered to be one of the first general-purpose electronic computers introduced in 1945. It was used by the U.S. Army for calculating ballistics tables. *UNIVAC I* (Universal

Automatic Computer) was the first commercial computer produced in the United States. It was used to predict the outcome of the 1952 presidential election on live television. Another milestone in this period was *EDSAC* (Electronic Delay Storage Automatic Calculator), one of the earliest stored-program computers, designed and built at the University of Cambridge in England. It became operational in 1949 and played a crucial role in the development of computing.

ii) Second Generation (1956–1963)

The invention of the transistor replaced vacuum tubes in computers. Transistors were much smaller, more reliable, and more energy-efficient than vacuum tubes. This generation saw the rise of the computer industry and the start of widespread use of commercial computers. High level programming languages like COBOL and FORTRAN were developed, making programming easier and more accessible. Computers became smaller in size, and new machines *IBM 1401* and *IBM 7090* were also introduced in this period.

iii) Third Generation (1964–1971)

This generation is defined by the *integrated circuit* (IC), which was invented by Jack Kilby and Robert Noyce. It is a small chip, also known as microchip, containing thousands of transistors and other components. Computers became even smaller, faster and more affordable, leading to their use in a wider range of industries. Operating systems were introduced, allowing computers to run multiple programs simultaneously and manage resources more efficiently. *IBM 360* series machines were the newly introduced computing device in this era.

iv) Fourth Generation (1971–Present)

The development of the *microprocessor*, a single IC that contains all the functions of a computer's central processing unit (CPU), marked a turning point. This generation brought computing to the general public, transforming society and giving rise to the digital age. The microprocessor became the brain of the computer. The invention of the *Intel 4004* microprocessor in 1971 is a landmark event. The development of more user-friendly software and the rise of the Graphical User Interface (GUI) made computers accessible to the general public. The invention of the CPU led to the creation of personal computers (PCs), laptops and eventually mobile devices. The internet was also developed during this era. The important machines introduced were *Apple II*, and *IBM PC*.

v) Fifth Generation (Present and Beyond)

While the fourth generation is still prominent, the concept of fifth generation is often used to describe future and present technologies that go beyond traditional computing. We are currently witnessing and utilising the parallel or mixed use of fourth and fifth-generation computers. This generation aims to create computers that can interact with the world in more natural ways and solve problems of incredible complexity, shaping the future of science, medicine and technology. Technologies like Artificial Intelligence (AI), Parallel Processing, Quantum Computing and DNA Computing have been incorporated in to the computing process of this era. Currently, IoT (Internet of Things), Modern AI systems and quantum computers are also in development stage.

- **Artificial Intelligence (AI):** It is the ability of a computer or machine to perform tasks that normally require human intelligence, such as learning, reasoning, linguistic intelligence, perception and problem-solving.
- **Parallel Processing:** It is the use of multiple processors working together to solve a single problem, increasing speed and efficiency.
- **Internet of Things (IoT):** The Internet of Things (IoT) refers to a vast network of physical objects (things) that are embedded with sensors, software, and other technologies for the purpose of connecting and exchanging data with other devices and systems over the internet.
- **Quantum Computing:** A new approach that uses the principles of quantum mechanics to perform calculations, with the potential to solve problems that are currently impossible for traditional computers.
- **DNA Computing :** This is a form of biological computation that uses DNA molecules, enzymes, and biological reactions instead of traditional silicon-based transistors to process information.



1. The first known calculating device is _____.
2. The second generation of computers used _____ instead of vacuum tubes.
3. _____ is known as the father of computers.
4. The second generation of computing introduced _____ programming language.



5. GUI was initially used in _____ generation of computers.
6. What are the contributions of Ada Lovelace in computing.
7. Distinguish between first and second generation computers.
8. Expand ENIAC, EDSAC and UNIVAC.
9. What is artificial intelligence?
10. Identify the main technology of each generation of computing

Match the following

1. First Generation	a) Artificial Intelligence
2. Second Generation	b) Vacuum Tubes
3. Third Generation	c) Microprocessors
4. Fourth Generation	d) Transistors
5. Fifth Generation	e) Integrated Circuits

1.2 Computer System

Computer is an electronic data processing machine that can be programmed to input, store and process data to produce output. It is made up of different hardware and software components.

Computer system usually consists of hardware components such as input/output (I/O) units, Central Processing Unit (CPU) and Storage Unit. All these components are mutually dependent and work together as a single unit to produce the desired output. It becomes a system when the different units perform their individual tasks to achieve a common goal. Nowadays, computers come in various forms such as servers, personal computers (PCs), laptops, tablets, and smart phones. Some of the devices are shown in Figure 1.2.



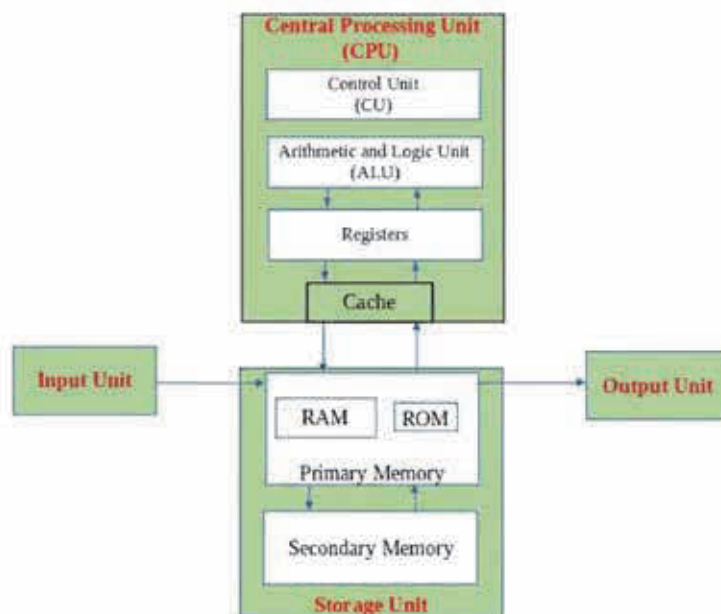
Figure 1.2 - Computing devices

1.2.1 Computer Organisation

A computer is an electronic device that processes raw data into meaningful information. This transformation is made possible through the organised arrangement of various hardware components, known as functional units. A computer operates using both hardware and software, working together to perform the tasks efficiently. Its internal organisation is designed to ensure the smooth flow of data from input devices to processing units, storage components, and finally to output devices. Computer organisation refers to the structure, interconnection, and operation of these hardware components that enable the execution of instructions. This session focuses on understanding the hardware components of a computer and how they are organised to function as an integrated system.

The computer system is organised into different functional units that perform specific tasks. These units work together to achieve the overall purpose of data processing. The main *functional units* are *Input Unit*, *Central Processing Unit (CPU)*, *Storage Unit (Memory)* and *Output Unit*. The basic structure of a computer system can be broken down into three main categories - Hardware, Software and User. Hardware refers to the physical, tangible components of the computer. Software is the set of instructions and programs that tell the hardware to perform a task. User is the one (Human, Device or AI tool) who interacts with the computer system.

1.2.2 Block Diagram of Computer Organisation



The Von Neumann architecture, published by John Von Neumann in 1945, is the foundational design for almost all modern computers. It is based on the stored-program concept, meaning that both instructions and data are stored in the same memory unit. The block diagram is depicted in Figure 1.3.

Figure 1.3 - Block Diagram of Computer Organisation

i) Input Unit

The input unit is used to feed data into a computer system. Input devices accept data and instructions, and convert them into digital form that a computer can understand. The data is encoded into machine-readable form (binary) and is transferred to the processing unit. Some of the input devices are keyboard, mouse, scanner, microphone, barcode reader, touch screen, etc. Specially designed input devices are commonly available today. Besides, we can input data through voice, visuals, biometric sensing, etc. Voice search and mobile biometric passwords are some examples.

Computing has evolved far beyond the traditional usage of keyboard and mouse. Nowadays, we have shifted to '*multimodal*' control. This simply means you can pick the easiest way to talk to your devices using your *voice, touch, eyes, or even your thoughts*. Modern gadgets are now '*ambient*'. That means, '*ambient gadgets*' are devices that work quietly in the background of your life you don't have to think about them or control them like a traditional computer. Instead of a computer sitting on a desk, technology is built into our clothes, our bodies, or the rooms around us. For example smart lights, smart watches and voice assistants. Because these devices are getting smarter, we have changed how we use them to make life faster and easier for everyone. The biggest trend right now is the '*Always-Available*' strategy. Think of it like the air around you, it is always there, even if you cannot see it. We use small sensors worn on our body so that we can control your digital world instantly from anywhere. Figure 1.4 depicts some input devices and modern strategies.

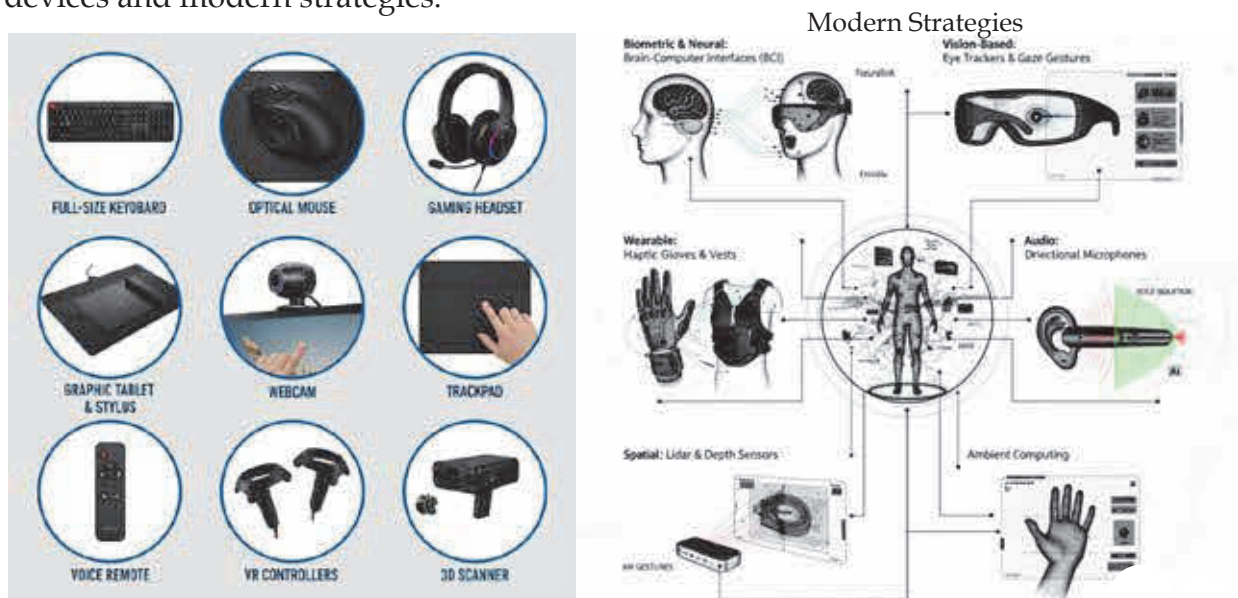


Figure 1.4 - Input devices and modern strategies

ii) Central Processing Unit (CPU)

It performs all the processing, and controls the functions of other units inside the computing device. CPU is commonly known as the processor. It consists of several microchips usually called Integrated Circuits (IC).

The CPU works based on the instructions given through programs. It fetches data and programs from memory and performs arithmetic and logical operations as per the given instructions to produce results, which are then stored back into memory. During processing, the CPU stores data and instructions in high-speed memory called

registers, which are a part of the CPU.



CPU

Apart from registers, the CPU has two more main components -Arithmetic Logic Unit (ALU) and Control Unit (CU). CU acts like a 'traffic police man' who controls the traffic system, inside the computer. It directs the flow of data between different functional units and interprets instructions in the programs.

ALU performs all arithmetic operations like addition, subtraction, multiplication, division and executes logical operations like comparisons such as greater than, equal to, or less than, etc. This component has the main role in decision-making processes.

Registers are the smallest and the fastest storage locations within the CPU. They temporarily hold data, instructions, and memory address necessary for the CPU to perform operations. CPU registers are categorized based on their specific function in the instruction execution cycle.

The two main categories are *Special-Purpose Registers (SPRs)* and *General-Purpose Registers (GPRs)*. *Special-Purpose Registers (SPRs)* are essential for managing the state and control of the CPU and the program's execution flow. *General-Purpose Registers (GPRs)* are flexible and can be used by the programmer (via assembly language) or the compiler to store temporary data and intermediate results during the execution of a program. The different SPRs and GPRs are listed in Table 1.1. We will discuss more about microprocessors in section 1.2.3.

Table 1.1 - SPRs and GPRs

Special-Purpose Registers (SPR)	General-Purpose Registers (GPR)
Program Counter (PC)	AX (Accumulator)
Instruction Register (IR)	BX (Base)
Memory Address Register (MAR)	CX (Counter)
Memory Buffer Register – MBR(also called Memory Data Register-MDR)	DX (Data)
Status Register	SI (Source Index)
Stack Pointer (SP)	DI (Destination Index)

iii) Output Unit

The *output unit* is responsible for converting the processed data from the computer's digital format into human-readable format. It is used to present the results after processing the data. The monitor, projector, headphone, speaker and printer are some examples. Visual display Unit (VDU) like monitor and projector are the most commonly used *softcopy* output devices, and printer is the most widely used *hardcopy* output device. Inkjet, laser jet and dot matrix are the widely used printers for office purpose. Recently, 3D printers are also available for 3D design printing to build exact physical replicas of 3D images. 3D printing has a wide range of uses in prototyping and manufacturing industries like automotive, aerospace, medical, architectural and construction. Some of the output devices are shown in Figure 1.5.



Figure 1.5 - Output devices

iv) Storage unit

The computer system needs to store data and instructions before processing and the results after processing. The storage unit or memory is where data and instructions are held for future use. It is essential to store data temporarily or permanently. At the physical

level data is stored in the form of 0s and 1s. The binary digits 0 and 1 are the basic units of memory called *bits* (**binary** digits). A 4 bit combination of bits is known as *nibble* and eight bit combination is called *byte*. In computing, memory size is measured using a hierarchy of units based on the number of bits. Table 1.2 contains the different memory units and their size.

Table 1. 2 - Memory units and their size

Unit	Abbreviation	Size
Bit	b	1 or 0 (1 bit)
Nibble	N (or nb)	half byte (4 bits)
Byte	B	1 Byte (8 Bits)
Kilobyte	KB	1,024 bytes
Megabyte	MB	1,024 KB
Gigabyte	GB	1,024 MB
Terabyte	TB	1,024 GB
Petabyte	PB	1,024 TB
Exabyte	EB	1,024 PB
Zettabyte	ZB	1,024 EB
Yottabyte	YB	1,024 ZB

Storage unit is generally classified into Primary Storage (Main Memory) and Secondary Storage (Auxiliary Memory). Primary Memory is also called internal memory which is directly accessible by the CPU, and used for short-term storage. RAM (Random Access Memory) is the most common type of primary storage. It is fast but volatile, in the sense that data is lost when the power is turned off. ROM (Read Only memory) and Cache memories are also considered as primary memory due to their direct accessibility by the CPU. RAM and cache are volatile, whereas ROM is non-volatile. RAM is used to hold the data and programs that are being used by the computer system while running. ROM stores essential programs such as the BIOS (Basic Input Output System) and other firmware, which are necessary to start a computer. For example, when we press the power button of a computer, the bootloader stored in to ROM initiates and loads the operating system from secondary storage into RAM. This process is called *booting*.

Boot Loader

A boot loader is a small program that runs every time you power on a computer, smart phone, or any other computing device. Its primary role is to load the operating system into RAM, a process that follows the Power-On Self-Test (POST).

The Cache memory is a small, high-speed memory located inside or very close to the CPU. Usually it is placed in between RAM and CPU. Even though RAM is faster than secondary memory, it is not as fast as CPU. Due to this slowness of RAM, CPU will be idle for an interval of time without getting data from RAM. This may lead to the lack of maximum utilisation of CPU. To avoid this delay, Cache memory is used with CPU. It stores copies of frequently accessed data from RAM. When the CPU needs data, it first searches the cache memory. If the data is in the cache, it is read from there; otherwise, it is fetched from RAM. Different types of Cache memories in the CPU are L1 (Level 1), L2 (Level 2) and L3 (Level 3).

Secondary Storage or External Memory is used for long-term storage of data and programs. It is non-volatile, slower and has a larger capacity than primary storage. Some secondary storage devices are Hard Disk Drives (HDDs), Solid-State Drives (SSDs), USB flash drives, and memory cards. Nowadays, comparing with HDD, SSD is more common in use because it is much faster, more durable, has less noise, lower power consumption, lightweight and compact. The emerging technology like quantum computing may change the existing memory devices. Different storage devices are shown in Figure 1.6.



Memory hierarchy



Figure 1.6 - Storage devices

1.2.3 Microprocessor

A microprocessor is a computer processor where the data processing logic and control are included on a single integrated circuit (IC), or a microchip. Figure 1.7 shows the image of a microprocessor. Moore's Law, as stated by Intel co-founder Gordon Moore in 1965 (revised in 1975), is the observation that the number

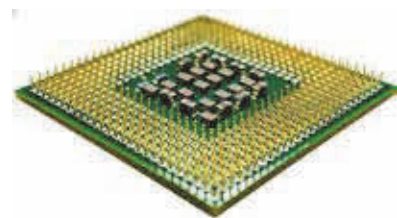


Figure 1.7 - Microprocessor

of transistors that can be economically placed on an integrated circuit (IC) doubles approximately every two years. In the early 1960s, SSI (Small Scale Integration) chips were manufactured with approximately 1 to 100 transistors. By the early 1970s, technology moved to MSI (Medium Scale Integration), where between 100 and 1,000 transistors were integrated onto a single chip. Later in 1970s, LSI (Large Scale Integration) chips that consist of 1,000 to 10,000 transistors, in 1980s, VLSI (Very Large Scale Integration) chips that use 10,000 to millions of transistors, and at present ULSI (Ultra Large Scale Integration) chip comprising millions to billions of transistors are introduced. These developments are shown in Figure 1.8.

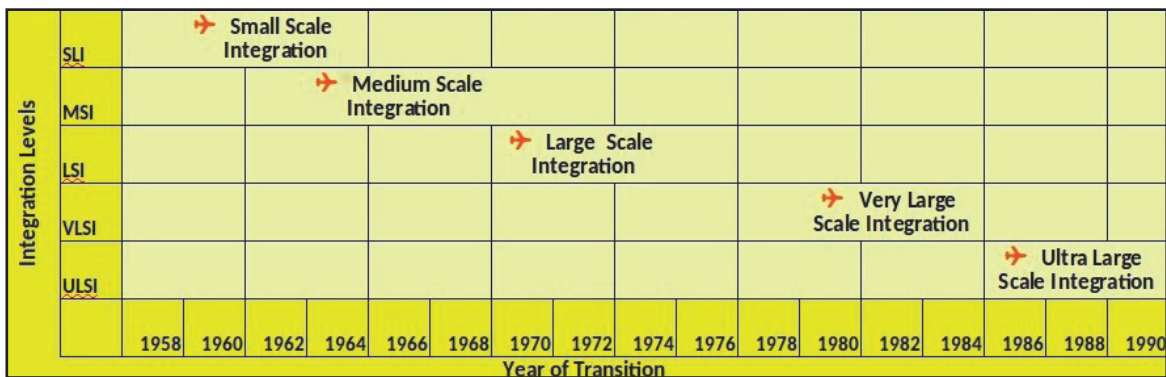


Figure 1.8 - Timeline diagram of micro processor evolution

1.2.4 Performance Characteristics of Microprocessors

The microprocessors are used based on several key criteria that reflect their architecture, performance, and intended applications. The followings are some performance characteristics of microprocessors.

i) Word Length

This refers to the number of bits the microprocessor can process at a time. Table 1.3 shows microprocessors with various word lengths.

Table 1.3 Microprocessors with different word lengths

Microprocessor	Word Length
Intel 4004	4-bit
Intel 8085, Zilog Z80	8-bit
Intel 8086, Motorola 68000	16-bit
Intel 80386, ARM7	32-bit
Intel Core i7, AMD Ryzen	64-bit

ii) Clock Speed

Clock speed simply means the number of pulses generated per second by the clock inside a computer. The clock speed indicates the speed at which the computer can execute instructions. Earlier, it was measured in Hertz (Hz) and Kilohertz (kHz). But with advancement in technology and chip density, it is now measured in Gigahertz (GHz), i.e., billions of pulses per second.

iii) Number of Cores

Imagine an office with only a single worker responsible for every task. If there are two or more workers, tasks can be performed in parallel, resulting in faster job completion. Think of a single-core CPU as an office with just one worker handling every job sequentially. Adding more workers (cores) allows tasks to be done in parallel, leading to much faster performance. Thus we can say core is the physical processing unit within a microprocessor (CPU) that executes instructions. Today we are using multi-core processors such as Dual-core, Quad-core, Octa-core, Hexa-core etc., in modern CPUs.

1.2.5 Motherboard

A motherboard is the main circuit board of a computer. The motherboard acts as the backbone of the computer system, providing power, connections, and communication paths for all hardware components to work together. It is the main printed circuit board (PCB) that connects and allows communication between all major components, such as

- Processor
- RAM ,ROM
- Storage devices like HDD,SSD, etc.
- Input/Output devices (keyboard, mouse, etc.)
- Expansion cards (graphics, sound, network cards)



Motherboard



1. _____ is called the brain of a computer.
2. _____ memory stores data permanently.
3. The main function of _____ is to perform arithmetic and logical operations.
4. ULSI stands for _____ .
5. 1KB = _____ bytes.
6. Explain the role of Control Unit.
7. What is the importance of cache memory?



8. What is word length?
9. List any six storage devices.
10. Name any six input and output devices.
11. Compare RAM and ROM.
12. Differentiate between Primary and Secondary memories.
13. Differentiate between softcopy and hardcopy outputs.
14. What are registers? List the registers inside CPU.
15. Explain core in a microprocessor.
16. Expand SSI, MSI, LSI.
17. Explain the working of a computer system with the help of a block diagram.
18. Explain the role of motherboard. List any two components connected with it.

Match the following

1. Input Unit	(a) Monitor
2. Output Unit	(b) Keyboard
3. ALU	(c) RAM
4. Storage	(d) Arithmetic operations

Activity

1. Collect the latest computing device configurations from advertisements in periodicals. (Paper cuttings or written documents)
2. Identify different input, output and storage devices in computer lab.

1.3 Software

Computer system is a combination of hardware and software. We are already familiar with the physical parts or hardware of computer system. Hardware of a computer system is nothing unless it is operated by instructions. Software is a set of instructions (programs) that tell the computer to perform an action. Without software, a computer is just an inert machine. Some examples of software are operating systems like Linux, Windows and Android, word processing tools like LibreOffice Writer and Microsoft Word,

Spreadsheets like LibreOffice Calc, Presentations like impress and Powerpoint, video players like VLC Player, photo editors like GIMP and database applications like Base, MS Access, MySql and MariaDB. Based on the mode of interaction with hardware and functionality, software is broadly classified into two types: System Software and Application Software.

1.3.1 System Software

System software is a set of programs designed to control and manage the computer hardware. System software manages the overall functioning of the computer system. It acts as a bridge between the hardware and the user's applications. Without system software, application software cannot run. Examples of system software include operating systems, system utilities, device drivers, firmware, language translators, etc.

i) Operating Systems (OS)

An *Operating System (OS)* is an interface between user and the hardware. It is the main control program that manages all other programs in a computer. The OS is the first software loaded into the computer's memory when the system boots up. Table 1.4 shows the functions of Operating System.

Table 1.4 - Functions of an OS

Functions	Description
Memory Management	Allocating and de-allocating memory for running programs
Process Management	Managing the execution of all tasks and processes
File Management	Organizing and storing files on storage devices
Device management	Controlling all I/O devices such as keyboard, mouse, printer
Security and Access Control	Protecting data and system resources from unauthorized access
User Interface	Providing a way for the user to interact with the computer

Some common examples of Operating Systems are

Desktop/Laptop : Microsoft Windows , MacOS, Linux.

Mobile Phones : Android, iOS.

ii) Utilities

Utility software (Utility program) is used to manage, maintain, and optimize the computer system's performance, security, and disk organization. They are the tools that provide support for the smooth functioning of a computer. Some examples are antivirus software, disk clean up tools, file compression software, backup tools, disk defragmenter, etc. The different utility programs and their uses are shown in Table 1.5.

Table 1.5 - Utility programs and their use

Utility Software	Purpose	Examples
Antivirus/ Anti-Malware	Protects the computer system from viruses, spyware, ransomware, and other malicious software	Windows Defender, Avast, Avira, McAfee, Malwarebytes, ClamAV/ ClamWin, Panda
File Compression/ Decompression Tools	Reduces the size of files to save storage space and make files easier to transfer (e.g., via email)	WinRAR, 7-Zip, WinZip, PeaZip, gzip, bzip2, LZ4
Disk Management Tools	Manages the computer's hard drive, including partitioning, formatting, and analyzing disk space	Windows Disk Management, MiniTool Partition Wizard, GParted
Disk Cleanup Tools	Scans the hard drive for unnecessary files (like temporary files, cache, and log files) and removes them to free up disk space	Windows Disk Cleanup, CCleaner
Disk Defragmenters	Reorganizes fragmented data on a hard disk drive to store pieces of each file close together, improving data access speed	Windows Disk Defragmenter (or Optimize Drives), MyDefrag
Backup and Recovery	Creates copies of important data (files, folders, or entire systems) and allows for restoration in case of data loss or system failure	Windows Backup and Restore, Time Machine (macOS), Acronis True Image

Uninstallers	Completely removes unwanted software and all related files and registry entries from the system	Revo Uninstaller, IObit Uninstaller
System Monitoring/ Profiler	Provides real-time information and detailed reports on the system's performance, resource usage (CPU, RAM, disk), and hardware/software inventory	Windows Task Manager, Activity Monitor (macOS), CPU-Z

iii) Device Drivers

Device driver or Driver software is a software component that allows the operating system to communicate with the hardware devices in a computer. It works as a translator and a bridge between the operating system and hardware devices by converting commands into instructions that the device can understand and send data back to the operating system. Manufacturers provide these drivers and keeping them updated is crucial for performance, stability, and accessing all device features. The device driver specifications are listed in Table 1.6.

Table 1.6 - Examples of device drivers

Device Type	Driver Function
Printer	Tells the OS how to send print jobs
Graphics Card	Enables screen display, resolution, 3D support
Sound Card	Handles audio input/output
Wi-Fi Adapter	Manages wireless network connections
Touchpad	Provides multi-touch gesture support
USB Ports	Manages connected devices like pen drives

1.3.2 Application Software

Application software, often called an '*app*' is a program designed to perform a specific task or solve a particular problem for the user. It operates between the user and the system software. The figure shows the hierarchy of software and hardware. A computer system cannot work without *system software*, but it can function without *application software*. For example, without an operating system, a computer or mobile phone cannot run. However, even without



Hierarchy of software and hardware

application software such as a calculator, graphics editor, word processor, or spreadsheet, a computer or smart phone can still function. Some application software are web browsers, word processors, computer games, media players, and presentation tools. Application software is classified as follows.

i) General-purpose Software

The application software developed for generic purposes is called General-purpose software. They are usually used for performing common tasks.

So many types of software like Word Processors, electronic spreadsheets, Presentations, Database Management System (DBMS), Web Browsers, Graphics Software are available in this category.

ii) Special-purpose Software

The application software exclusively designed to perform a specific task is called Special-purpose software. Some of the examples are Accounting Software, Reservation Systems, Inventory Control Systems, Payroll Systems and Media Players.

1.3.3 Programming Tools

A *program* is a set of instructions that a computer can execute to perform a specific task. *Programming* is the process of writing, testing, and maintaining these instructions. Programs are written in a specific *programming language*. Programming tools, also known as software development tools, are used to create, debug, test, and execute programs. They are essential for the entire software development process. As we know, if the computer wants to do a particular task, we need to give some instructions to process the input data for the desired output. We can communicate with each other using natural languages like Kannada, Malayalam, Tamil, etc. Humans can understand different languages but computers can understand only machine language (Binary language of 1s and 0s). Programmers write instructions in high level language and computers store and process them in machine language. So there must be a translator for the conversion. They are usually called language translators. To write instructions, code editor like Geany, IDLE or Eclipse is required. Here, we will briefly describe some of the program development components such as Programming languages, Language translators and Program development tools.

i) Programming Languages and Language Translators

Writing instructions in the form of 1s and 0s is a tedious task for the human being. To simplify the difficulties while writing instructions, different types of computer programming languages are introduced. These languages are basically classified into two types, Low-level and High-level languages. Low-level languages include machine language and assembly language, and they are machine dependent. 1s and 0s are used in machine language to write instructions which are directly understood and executed by the computer system. As we mentioned earlier, writing code in machine language by remembering operation codes, character codes and machine addresses is very difficult. Error detection and correction in the code is also difficult. Since the code written in machine language is directly understandable by the computer system, there is no need of a translator.

Assembly language was developed to simplify the code writing by using English like notations as symbolic opcodes and mnemonics instead of 1s and 0s. The main problem of writing code in this language is that, it is totally machine specific or machine dependent. The code written for one machine cannot be executed by another machine. Here the code written in assembly language cannot be directly understood by the computer and hence there should be a translator. The language translator called Assembler will translate a program written in assembly language into machine language.

Opcode(Operation Code)

An opcode is the portion of machine language instructions that specifies the operations to be performed. It is the only language represented in binary, a CPU's control unit understands.

Mnemonics

A mnemonic is an abbreviated name used to represent an opcode in assembly language. Instead of memorizing strings of binary digits, programmers use short words like ADD, MOV and JMP.

Writing and executing codes in low-level language is very difficult because they are totally machine dependent. To reduce and simplify the coding complexity, high-level languages were introduced. They are machine independent and use English-like sentences. The code written in High-level language is not directly understood by the computer system. Hence, translators are needed to convert high-level language code into machine language code.



The language translators, *compiler* and interpreter are used for this case. A compiler translates the entire high-level language program into machine code all at once before the program is executed (GNU Compiler Collection (GCC), Java compiler, etc.). An *interpreter* translates and executes the high-level language program line by line during runtime (Python interpreter, BASIC interpreter, etc).

ii) Program Development Tools

To write the program codes we need some text editors. An editor is a software that allows us to create a text file when we type instructions and store the file as the source code. Lightweight text editors are specifically designed for coding, offering features like syntax highlighting and code completion. In order to simplify the programming procedure we usually use a single screen environment. These are commonly known as Integrated Development Environments (IDEs). This environment integrates all necessary tools (editor, compiler, debugger, etc.) into a single, comprehensive interface. Python IDLE, Geany, Netbeans, Eclipse, Atom and Lazarus are examples for IDEs.

1.3.4 Free and Open-Source Software (FOSS)

Software that grants users the freedom to run, study, change, and share the software and its source code is called *Free and Open-Source Software (FOSS)*. We use FOSS everyday through different computing devices. Operating Systems such as Linux (e.g. Ubuntu, Fedora, Android are based on the Linux kernel), Web Browsers like Mozilla Firefox, Office Suites such as *LibreOffice* (an alternative to Microsoft Office), *Graphics & Design* software like GIMP (GNU Image Manipulation Program, an alternative to Photoshop), Blender (3D creation suite), Web Servers like Apache HTTP Server and Media Players such as VLC Media Player are some examples of FOSS.

The Free Software Foundation (FSF) is a global movement of individuals who support free software that promotes values and ethics. FSF was founded by Richard Stallman in 1985. *Richard Stallman* is co-founder of the GNU project and the free license movement, which has given rise to the biggest digital commons: GNU/Linux, Mozilla Firefox, LibreOffice, Wikipedia and Creative Commons. The word 'Free' refers to freedom, not price free. You can join the FSF community by becoming a Free Software Supporter. KITE (Kerala Infrastructure and Technology for Education) is a promoter of FSF activities.



Richard Stallman

i) Free Software

Free software refers to software that grants users the freedom to run, study, modify, and distribute it. It is important to note that *open-source* and *free software* are usually the same things, just with different goals. The goal of the free software movement is *social justice* and *user liberty*, whereas the open source movement focuses on *better software development*. The following are the four essential freedoms for free software.

1. **Freedom 0** :-Freedom to run the program for any purpose.
2. **Freedom 1** :-Freedom to study how it works and changes it.
3. **Freedom 2** :-Freedom to redistribute copies.
4. **Freedom 3** : Freedom to distribute modified versions.

ii) Open Source Software

Open Source Software is one in which, the *source code is made publicly available* to anyone. Users are typically granted the right to *view, modify, and distribute* the code under the terms of an open source license. Some examples are Ubuntu, Fedora, Android, Mozilla Firefox, Chromium, LibreOffice, Apache HTTP Server, WordPress, etc.

1.3.5 Closed Source Software

Closed Source Software, also known as *Proprietary Software* is the software whose *source code is kept confidential* and is only accessible to the original developers or the organization that owns it. Some examples are Microsoft Windows, Apple Mac-OS, Apple iOS, Word, Excel, PowerPoint, Adobe Creative Suite(Photoshop, Illustrator), Zoom, etc. Table 1.7 shows the features of Open -Source and Closed-source software.

Table 1.7 - Features of open and closed source software

Feature	Open-Source Software	Closed-Source (Proprietary) Software
Source Code	Open. You can view, study, and modify it	Closed. Kept secret by the company
Updates/Fixes	Collaboration by a global community; fixes can be rapid	Only the developing company can apply fixes
Security	Transparent. "Many eyes" principle, vulnerabilities are often found and fixed quickly	Obscure. Must trust the vendor that no backdoors or bugs exist
Customization	High. You can change the program to suit your needs	Low. You must wait for the company to add a feature



Richard Stallman - The Father of Free Software

Richard Matthew Stallman (born March 16, 1953), often known by his initials RMS, is an American computer programmer and the founder of the Free Software Movement. He is one of the most influential figures in the history of computing, advocating for the ethical right of users to control their own technology.

Early Life and Career

Stallman's journey began in the early 1970s at the MIT Artificial Intelligence Laboratory. During this time, programmers lived in a "hacker culture" where they freely shared and improved each other's code. However, by the early 1980s, software began to be 'privatised', meaning companies restricted access to the source code. Stallman found this ethically wrong, believing that software should be a community resource.

Major Contributions

Stallman is best known for three landmark contributions to the world of Information Technology.

The GNU Project (1983): He launched this project to create a complete, Unix-like operating system that was entirely free. The name GNU is a recursive acronym for 'GNU is Not Unix.'

The Free Software Foundation (FSF) (1985): A non-profit organization established to support the GNU Project and promote the legal and philosophical principles of software freedom.

GNU General Public License (GPL): Stallman introduced the concept of Copyleft. Unlike traditional copyright that restricts users, the GPL uses law to ensure that the software remains free for everyone to use, modify, and redistribute.



1. _____ software manages the hardware and provides a platform for application software.
2. _____ is an example for open-source software.
3. Programs designed for specific tasks like word processing are called _____
4. FSF was founded by -----.
5. Expand FOSS.
6. Distinguish between system software and application software.
7. Write the functions of an operating system.
8. Give two examples each for open source and free software.
9. Explain different freedoms related to software.
10. Explain the use of Device drivers.



11. What do you mean by utility program? Give any two examples.
12. Explain the types of software with examples.
13. Describe the role of utility software in a computer system.
14. Differentiate between open source and free software with examples.

Match the following

1. Operating System	(a) Mozilla Firefox
2. Utility Software	(b) Spreadsheet
3. Application Software	(c) Windows
4. Open Source	(d) Disk Cleanup

Activity

Install any one open source software (including device drivers) in a computer.

1.4 Data and Information

Data and information are extremely important in the modern world of Artificial Intelligence (AI). They serve as the fundamental basis for knowledge, decision-making, and innovation across all sectors, including business, science, government, and even in daily life. In our daily activities, we encounter large quantities of raw facts – such as numbers, words, figures, or symbols. When these raw facts are scattered or unorganized, they lack specific meaning or context. However, they become useful once they are organized, structured, and processed. Computers are widely used for data processing because of their inherent speed, accuracy, and reliability. A computer system treats everything as data such as characters, pictures, audio, video, documents, and instructions. Most people use the terms data and information interchangeably, even though they are actually quite different. You will understand the critical distinction between data and information, the various stages of data processing, and the essential role computers play in this transformation. The following are the different stages of data transformation depicted in the pyramid diagram.



Pyramid diagram of data transformation

Data

Data is a raw, unorganized fact or a set of facts, such as numbers, text, images, audios, videos, etc. Data on its own has no complete meaning. For example, exam scores of a student, like 85, 92, 78, 95, are data.

Information

Processed data is called *information*. From the previous example, we can generate information such as total score or average score by processing the scores.

Knowledge

Knowledge is the ability to use information to make decisions. For example, by comparing a student's marks with the class average, we can understand whether the student needs improvement.

Intelligence

Intelligence is the ability to apply knowledge to solve problems in different situations. To improve the scores of students, apply strategies such as grouping them based on their learning styles and giving them different types of homework.

Wisdom

Wisdom is the ability to use knowledge and experience to make good decisions in real life situations. For example, a student notices that their maths score is low in two exams. Instead of feeling upset, he decide to change the study plan by practice daily, ask the teachers help, and revise weak topics.

1.4.1 Data Processing

Data processing is a series of activities carried out to convert raw data into meaningful information. It includes collection, manipulation and transformation of data into information. It involves a systematic set of operations performed on data to achieve a desired outcome. This process can be done manually or electronically.

i) Manual Data Processing (MDP)

It is performed using human effort with simple tools like a pen and paper. This method is slow and prone to errors.

ii) Electronic Data Processing (EDP)

It is the most common method today, which uses electronic devices like computers. It is highly efficient, fast, and accurate.

1.4.2 Stages of Data Processing

The data processing involves different stages that transform raw data into meaningful information. These stages are very important in decision-making. Modern data processing has evolved from simple 'collect-input-process-storage-output' cycles into a complex, continuous process. Nowadays, the concept is defined by real-time intelligence, cloud architecture, and AI-driven automation. Data processing has evolved from a rigid, manual, and localised activity into an automated, global, and real-time operation. Instead of waiting for reports at the end of the day, modern systems process data as it happens, allowing the environment to respond to events in milliseconds. Stages of data processing are shown in Figure 1.9.

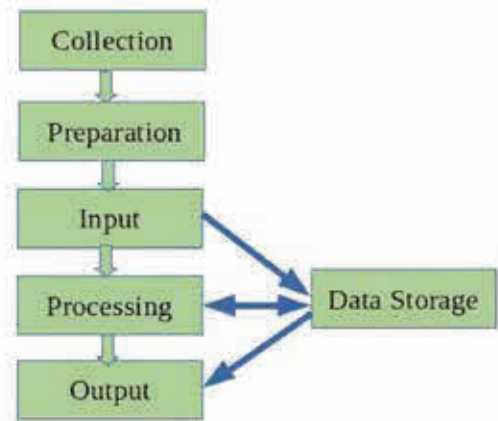


Figure 1.9. Stages of data processing

i. Collection

This is the first and most critical stage. Gathering raw data from various sources is called data collection. Data is collected from sensors, surveys, transaction logs or physical documents. This stage ensures that the data gathered is accurate and complete. For example, collecting marks of students, recording daily sales or registering patients in a hospital.

ii. Preparation

Raw data often contains errors or unnecessary information. In this stage, data is 'cleaned and organized'. The main tasks of this stage are removing duplicates, correcting errors, and filtering out irrelevant facts (often called "Garbage In, Garbage Out" or GIGO). Here we ensure that only high-quality data moves to the next stage.

iii. Input

It is the activity of entering the collected data into the computer system using input devices and other strategies. The cleansed data is converted into a machine-readable format and entered into the computer system. For example, entering sales figures using a keyboard, barcode scanner or by importing digital files.

iv. Processing

This is the stage where the computer actually works on the data. The CPU performs various operations based on instructions.

Manipulating the input data according to given instructions is known as processing. Processing may involve calculations, comparisons, sorting, classifications, or summarising. For example, calculating total sales or generating grades from exam scores.

v. Storage

Storage is the mechanism that holds the data for each transformation phase in data processing. The processed data and results are stored here for future use. Data can be stored temporarily in primary memory (RAM) or permanently in secondary storage (hard disks, SSDs). Nowadays the storage types like cloud, database, archives, etc. are also in use.

vi. Output

Presenting the information in a human-readable form using output devices is called the output. This is the real stage to present the “Information” in a way that users can easily interpret for decision making. For example, displaying the reports, graphs, charts, tables on the monitor, printing an invoice or presenting audio/video.

1.4.3 Computer as a Data Processing Machine



Computer - a data processing machine

The different functional units of computer are used to perform the activities involved in data processing. Inputting can be done with keyboard/mouse, to store the data use devices like HDD, CD, Pen drive, etc., use CPU for processing and display output using monitor, projector, etc. Today computers are widely used for data processing because of their tremendous features. Some of which are the following.

- Speed** : Computers can process millions of instructions per second.
- Accuracy** : Errors are minimal if correct instructions and data are given.
- Storage** : Computers can store large volumes of data.
- Diligence** : The ability to work for hours without getting tired or bored.
- Automation** : Once a program is given, computers can perform tasks automatically.
- Versatility** : The ability to perform different kinds of work. It is used in every walk of human life.



1. Raw facts and figures are called _____.
2. Processed data is called -----
3. _____ stage converts data into meaningful information.
4. Define data and information. Give examples.
5. List the different stages of data processing.
6. Computers are called data processing machines. Why ?
7. Differentiate between data and information.
8. Explain the stages of data processing.
9. Briefly explain the features of a computer system.
10. What are the different types of data processing?
11. Explain manual and electronic data processing.

Match the following

1. Data	(a) Organised and meaningful
2. Information	(b) Raw facts and figures
3. Input	(c) Keeping processed results
4. Storage	(d) Entering data

Activity/Case study

List and write the steps involved in plus one admission procedure (SWS) through hscap.

1.5 Green Computing

Computers and electronic devices have become an integral part of our daily life. However, their tremendous use has created two major concerns-environmental impact of computing and electronic waste (e-Waste) management. By the end of this session, you will learn about green computing and the problems posed by e-waste, as well as the various methods for its disposal.

To reduce harmful effects on the environment, the concepts of Green Computing and proper e-Waste disposal have become very important. Green computing, also known as green IT, is the practice of designing, manufacturing, using, and disposing of computers and related subsystems (like monitors, printers, and



Green Computing

storage devices) in an environment (eco) friendly and energy-efficient way. The goal is to minimize the negative impact of technology on the environment. Many corporate IT departments have green computing initiatives to reduce the environmental effect of their IT operations. Computers and electronic devices are to be used responsibly to support sustainable development. Through Green Computing and safe disposal of e-Waste, we can ensure that technology benefits society without harming the environment.

The key principles of green computing include

Energy Efficiency: Energy efficiency is the practice of using less power for devices and data centres. This involves using energy-saving hardware and implementing power management features.

Eco-Friendly Materials: Designing and manufacturing devices with biodegradable materials that are less harmful to the environment and which can be recycled.

Waste Reduction: Extending the lifespan of devices to reduce the amount of electronic waste produced. This is a key focus area.

Responsible Disposal: Ensuring that old and obsolete equipment is handled and disposed properly and safely.

1.5.1 E-waste

E-waste, or electronic waste, refers to discarded electrical or electronic devices such as computers, mobile phones, televisions,



E-waste

printers, and other gadgets. As technology rapidly evolves, consumers and manufacturers frequently upgrade devices like smart phones, laptops, and televisions, creating a massive amount of e-waste.

The main problem with e-waste is its composition. It contains both valuable materials like gold, silver, copper, and palladium, and hazardous substances such as lead, mercury, cadmium,

bromine and arsenic. When e-waste is improperly disposed of, the toxic substances can dissolve into the soil and groundwater, contaminating ecosystems and posing serious health risks to humans and animals.

1.5.2 E-waste Disposal Methods

Responsible disposal of e-waste is crucial for protecting the environment and human health. Here are the main e-waste disposal methods, including both safe and harmful or unsafe.

i) Safe Methods

Reuse: If a device is still functioning after its long term usage, sell it, or give it to someone (Schools, Individuals in need, etc.) who can use it.

Recycle: For devices that are no longer working, ensure they are sent to a certified e-waste recycling facility. These facilities can safely dismantle devices, separate hazardous materials, and recover valuable metals and plastics for use in new products.

Manufacturer Take-Back Programmes: Many electronics companies (e.g., Apple, Dell and HP) offer free return trade-in programme.

ii) Unsafe Methods

Land filling: E-waste is simply buried in landfills, where its toxic chemicals can contaminate the surrounding environment.

Incineration: Burning e-waste at high temperatures reduces its volume but can release harmful toxins and gases into the atmosphere.

Acid Baths: In some informal recycling operations, components are submerged in strong acids to dissolve metals. This is highly dangerous and produces toxic chemical waste that is often improperly disposed of, polluting local water sources.

Exporting to Developing Countries: Illegal or unethical export of e-waste to countries with poor regulations leads to environmental pollution and health hazards for workers handling e-waste without protection.



E-Waste disposal

1.5.3 What Should Be Done

To protect ecosystems from e-waste damage, we need to

Recycle electronic devices responsibly via certified recyclers.

Reduce e-waste by buying less or repairing devices.

Proper disposal of e-waste with drop-off centres or manufacturer take-back.

Educate people to handle the e-waste safely.



1. The eco-friendly use of computers is called _____.
2. Discarded electronic devices are known as _____.
3. Toxic substances like mercury and lead are found in _____.
4. Define Green Computing. Write about any two green computing practices .
5. What are the health hazards caused by e-Waste?
6. Distinguish between reuse and recycling.
7. What are the key principles of green computing?
8. Why do we protect our ecosystems ? Explain.
9. Explain the concept of e-Waste and its problems.
10. Discuss the disposal methods of e-Waste with examples.
11. How does Green Computing contribute to environmental protection?

Match the following

1. Green Computing	(a) Discarded computers
2. e-Waste	(b) Energy-efficient use of technology
3. Recycling	(c) Producer responsibility
4. Take-back programmes	(d) Extracting valuable materials from old devices

Activity

Collect the e-wastes in your school and segregate them into reusable, recyclable, and unusable categories and suggest appropriate e-waste disposal methods for each of them.

◆———— Summary ———◆

A computing system is a programmable electronic machine that transforms raw data into information. Most modern systems follow the Von Neumann architecture, which utilises a Central Processing Unit (CPU) to fetch and execute instructions stored alongside data in a single memory unit. Data processing has evolved significantly from early manual tools like the Abacus to high-speed electronic machines which are capable to handle complex tasks in fractions of a second. Beyond hardware, the system depends on software, which is broadly classified into system software

and application software. Free and Open-Source Software (FOSS) is a vital part of this ecosystem, granting users the freedom to run, study, modify, and redistribute the source code. In contrast, closed-source or proprietary software keeps its source code confidential and freedoms are restricted to the users. The rapid pace of development means that new technologies emerge quickly which often leads to significant environmental challenges such as the accumulation of e-waste. E-waste consists of discarded devices like old computers and smart phones that can release toxic substances if not handled correctly. To tackle this situation effectively, green computing practices emphasis on energy efficiency and the responsible recycling or reuse of electronic components to protect ecosystems. As a whole this chapter covers the technological developments from historical roots to the modern organization of computing systems.



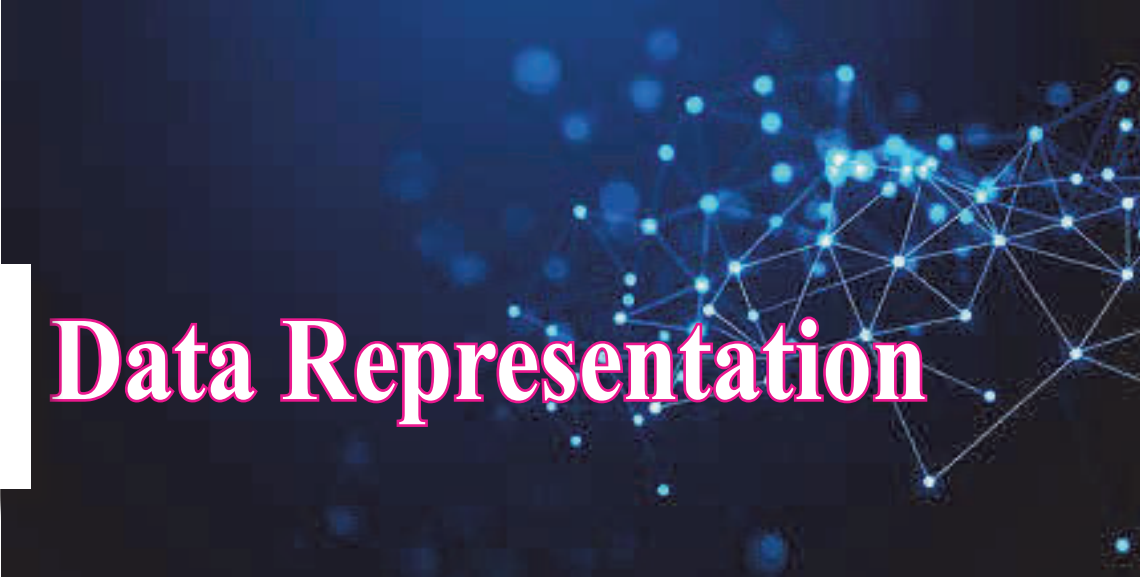
1. ROM stands for _____ .
2. Data processing converts raw data into _____.
3. Removing duplicate records is done during _____ stage.
4. Which unit acts as a 'traffic policeman' by directing the flow of data between functional units?
 - a) Arithmetic Logic Unit b) Accumulator c) Control Unit d) Status Register
5. In the memory hierarchy, which unit is exactly equal to half a byte?
 - a) Bit b) Nibble c) Kilo byte d) Peta byte
6. Which of the following is considered a "hardcopy" output device?
 - a) Monitor b) Projector c) Printer d) Speaker
7. Which organization was founded by Richard Stallman in 1985 to promote software ethics?
 - a) OSI b) FSF c) KITE d) GNU Project
8. Which stage of data processing removes errors and duplicates?
 - a) Collection b) Input c) Preparation d) Output
9. Which unit of the computer performs processing?
 - a) Input unit b) Output unit c) Storage unit d) CPU
10. "Garbage In, Garbage Out (GIGO)" is related to which stage?
 - a) Collection b) Preparation c) Storage d) Output
11. Which of the following is a hazardous substance found in e-waste?
 - a) Oxygen b) Mercury c) Nitrogen d) Helium
12. Which is a safe method of e-waste disposal?
 - a) Incineration b) Land filling c) Recycling d) Acid bath

13. Manufacturer take-back programmes are offered by
a) Schools b) Hospitals c) Electronics companies d) Farmers
14. Distinguish between General-purpose and Special-purpose application software.
15. Explain the hierarchy and relationship between the CPU, Cache memory, and RAM.
16. How does Cache memory improve CPU utilization?
17. Compare Open source and Closed source software.
18. What is the primary function of a “Device Driver”?
19. Give any two examples of “Ambient gadgets”.
20. What are the three levels of Cache memory?
21. List the different functions of an Operating System.
22. Explain the modern Input strategy and the “Always-Available” Concept.
23. Explain the features of computer system.
24. Define data processing.
25. Write the advantage of electronic data processing.
26. Write any two differences between MDP and EDP.
27. List the stages of data processing.
28. Write any two advantages of computers in data processing.
29. What is meant by storage in data processing?
30. Give two examples of data collection.
31. Explain Manual Data Processing and Electronic Data Processing with examples.
32. Explain any four features of computers as a data processing machine
33. Explain the role of a computer as a data processing machine.
34. Draw and explain the Stages of Data Processing.
35. Define e-waste.
36. Name any two hazardous substances found in e-waste.
37. Give two examples of an electronic device that becomes e-waste.
38. Write any one objective of green computing.
39. What is a manufacturer take-back programme?
40. List any two principles of green computing.
41. Write any two problems caused by improper e-waste disposal.
42. Write any two safe methods of e-waste disposal.
43. What are the hazardous substances present in e-waste?
44. Explain green computing in detail. Discuss its key principles.
45. Describe the safe and unsafe methods of e-waste disposal.



2

Data Representation



“There are 10 kinds of people in the world: those who understand binary numerals, and those who don’t”

Ian Stewart.

2.1 Introduction

Data representation is the basic concept of computer science that explains how computers store, process, and communicate information. Since computers can understand only binary digits (0s and 1s), all types of data, including numbers, characters, images, audio, and video must be converted into a binary form. This chapter introduces the various data representation techniques used for numbers, characters and multimedia file format.

2.2 Number System

A number system is a way of writing and representing numbers using a set of symbols or digits. It is a method of expressing numbers using a systematic structure based on value and base. The number of symbols used is called the *base or radix* of the number system.

Types of Number Systems

Various number systems classified on the basis of base value are as follows.

- Decimal number system (Base - 10)
- Binary number system (Base - 2)
- Octal number system (Base - 8)
- Hexadecimal number system (Base - 16)

These number systems are explained in subsequent sections.

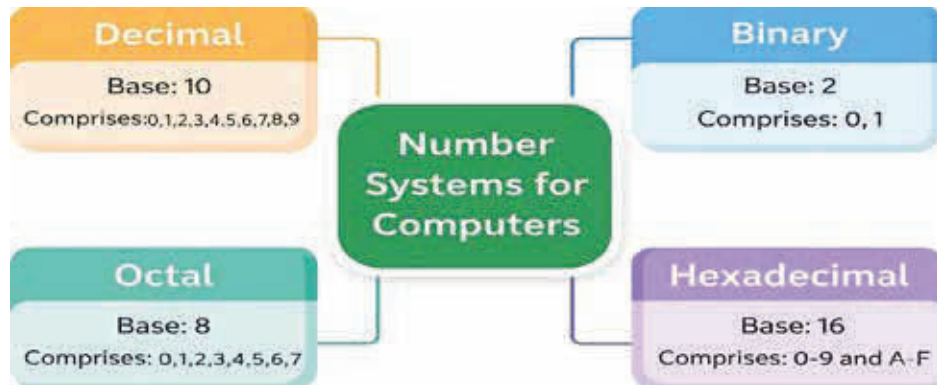


Figure 2.1 - Number systems

a. Decimal Number System

The word “decimal” comes from the Latin word *decem*, which means ten. This system uses ten distinct symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9, hence the base value of the decimal number system is 10. Every number is represented using a combination of these digits. The key to understanding this system is the concept of place value.

Consider two decimal numbers: 123 and 321

$$123 = 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0$$

$$321 = 3 \times 10^2 + 2 \times 10^1 + 1 \times 10^0$$

In the number 123, the digit 1 is in the hundreds place, so its place value is $10^2=100$.

In the number 321, the digit 1 is in the ones place, so its place value is $10^0=1$.

Thus, the value of a digit depends on its position in the number. A number system in which the value of a digit depends on its position is called a *positional number system*. Each position represents a power of 10 (base), starting from 10^0 (which is 1) on the rightmost as shown below.

Consider the example 534 can be expanded as below.

Place Value	10^2	10^1	10^0
Digits	5	3	4

$$= 5 \times 10^2 + 3 \times 10^1 + 4 \times 10^0$$

$$= 5 \times 100 + 3 \times 10 + 4 \times 1$$

$$= 500 + 30 + 4$$

$$= 534$$

In the above example, the digit 5 has the maximum place value, $10^2=100$, and the digit 4 has the minimum place value, $10^0=1$. The rightmost digit in a decimal number is called *Least Significant Digit (LSD)*, which has the smallest place value and the leftmost digit in a decimal number is called *Most Significant Digit (MSD)* which has the largest place value.

b. Binary Number System

The binary number system uses only two symbols, 0 and 1. Each symbol is known as a *binary digit* or a *bit*. It is also known as the base-2 number system, where the positional digits are powers of 2. Binary numbers are often subscripted with 2 in order to indicate the base. If the base not specified, it is usually assumed to be 10 (decimal). The numbers are represented by creating a sequence of bits.

For example: $(1001011)_2$

The rightmost bit in a binary number is called *Least Significant Bit (LSB)*. The leftmost bit in a binary number is called *Most Significant Bit (MSB)*.

Consider the example $(1001011)_2$ can be expanded as shown below.

Positional value	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Binary Number	1	0	0	1	0	1	1

$$= 1 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$

$$= 1 \times 64 + 0 \times 32 + 0 \times 16 + 1 \times 8 + 0 \times 4 + 1 \times 2 + 1 \times 1$$

$$= 64 + 0 + 0 + 8 + 0 + 2 + 1$$

$$= 75$$

Some popular relevance and applications of binary numbers are:

- **Computer Language:** Binary is the native language of computers. All data and instructions are represented using binary codes (0s and 1s).
- **Digital Electronics:** Binary is used to represent digital signals in electronic circuits, making it the backbone of digital technology
- **Efficient Data Storage:** Binary is used to store data in computers, making it an essential part of modern computing
- **Encryption:** Securing data by converting information into binary form.

c. Octal Number System

The octal number system is a base-8 number system that uses 8 digits: 0, 1, 2, 3, 4, 5, 6, and 7. Octal is more compact than binary, making it easier to read. It also converts easily to/from binary, which made octal useful in early computing. Octal numbers are used to represent file permissions in Unix-like operating systems, represent memory addresses in some computer systems and in embedded systems like microcontrollers, for programming and debugging.

Consider the example $(123)_8$ can be expanded as shown below.

Positional Value	8^2	8^1	8^0
Octal Number	1	2	3

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

$$= 1 \times 64 + 2 \times 8 + 3 \times 1$$

$$= 64 + 16 + 3$$

$$= 83$$

d. Hexadecimal Number System

The number system uses 16 distinct symbols: digits 0-9 and letters A-F, where A represents 10, B represents 11, and so on, up to F, which represents 15, hence the base value is 16. This system is particularly useful in computing because it provides a human-friendly way to interact with binary values. The table 2.1 shows the hexadecimal digits and their decimal equivalents.

Table 2.1 - The hexadecimal digit and their equivalent decimal numbers

Decimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Hexadecimal	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Consider the example $(258)_{16}$ can be expanded as shown below.

Positional Value	16^2	16^1	16^0
Hexadecimal Number	2	5	8

$$= 2 \times 16^2 + 5 \times 16^1 + 8 \times 16^0$$

$$= 2 \times 256 + 5 \times 16 + 8 \times 1$$

$$= 512 + 80 + 8$$

$$= 600$$



1. What do you mean by number system? List any four number systems.
2. Define the term base (radix) of a number system.
3. What are MSB and LSB in binary number system?
4. Why do we use binary number system in computers?
5. Write a short note on importance of hexadecimal number system.

2.3 Number System Conversions

Number system conversion is the process of changing a number from one base to another.

It acts as a bridge between human understanding and computer processing.

2.3.1 Decimal to Binary Conversion

1. Divide the number by 2 (base of Binary Number system) repeatedly until we get zero, noting the remainder in each division.
2. Read the remainders from bottom to top to get the binary equivalent of the decimal number.

Example: Convert $(35)_{10}$ to binary.

2	35	
2	17	1
2	8	1
2	4	0
2	2	0
2	1	0
0		1

$$(35)_{10} = (100011)_2$$

Example: Convert $(59)_{10}$ to binary.

2	59	
2	29	1
2	14	1
2	7	0
2	3	1
2	1	1
0		1

$$(59)_{10} = (11101)_2$$

2.3.2 Binary to Decimal Conversion

1. Multiply each digit with its respective weighted power of 2.
2. Calculate its sum.

Example: Convert $(101101)_2$ to decimal number.

Positional Value	2^5	2^4	2^3	2^2	2^1	2^0
Bits	1	0	1	1	0	1

$$\begin{aligned}
 &= 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \\
 &= 32 + 0 + 8 + 4 + 0 + 1 \\
 &= (45)_{10}
 \end{aligned}$$

Example: Convert $(110101)_2$ to decimal number.

Positional Value	2^5	2^4	2^3	2^2	2^1	2^0
Bits	1	1	0	1	0	1

$$\begin{aligned}
 &= 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \\
 &= 32 + 16 + 0 + 4 + 0 + 1 \\
 &= (53)_{10}
 \end{aligned}$$

2.3.3 Octal to Binary Conversion

1. Convert each octal digit to 3 bit binary equivalent using the conversion table.
2. Combine all the 3-bit groups to form the binary number.

Table 2.2 shows the hexadecimal digits and their binary equivalents.

Octal digit	0	1	2	3	4	5	6	7
Binary equivalent	000	001	010	011	100	101	110	111

Example: Convert $(523)_8$ to binary number.

5	2	3	(Octal digit)
↓	↓	↓	
101	010	011	(3 bit Binary)

$(523)_8 = (101010011)_2$

Example: Convert $(7046)_8$ to binary number.

7	0	4	6	(Octal digit)
↓	↓	↓	↓	
111	000	100	110	(3 bit Binary)

$$(7046)_8 = (111000100110)_2$$

2.3.4 Binary to Octal Conversion

1. Group the bits into sets of 3, starting from the LSB. Attach sufficient 0s to the left to make leftmost group contain 3 bits.
2. Convert each group of 3 bits to its octal equivalent using the conversion table.
3. Combine the octal digits to get the final result.

Example: Convert $(101101010)_2$ to octal number.

Group the given binary number 101101010 into a set of 3 bits from right as shown below

<u>101</u>	<u>101</u>	<u>010</u>
↓	↓	↓
5	5	2

$$(101101010)_2 = (552)_8$$

Example: Convert $(1110010101)_2$ to octal number.

Additional bits are added to make the set as group of 3 bits	<u>001</u>	<u>110</u>	<u>010</u>	<u>101</u>
	↓	↓	↓	↓
	1	6	2	5

$$(1110010101)_2 = (1625)_8$$

2.3.5 Hexadecimal to Binary Conversion

1. Convert each hexadecimal digit to 4 bit binary equivalent using the conversion table.
2. Combine all the 4-bit groups to form the binary number.

Table 2.3 shows the hexadecimal digits and their binary equivalents.

Example: Convert $(9AF)_{16}$ to binary number.

9	A	F	(Hexadecimal digit)
↓	↓	↓	
1001	1010	1111	(4 bit Binary)

$(9AF)_{16} = (100110101111)_2$

Example: Convert $(12D)_{16}$ to binary number.

1	2	D	(Hexadecimal digit)
↓	↓	↓	
0001	0010	1101	(4 bit Binary)

$(12D)_{16} = (000100101101)_2$

It can be written as $(100101101)_2$ by trimming leading zeros.

Table 2.3 - Binary equivalent of hexadecimal

Hexa-decimal	Binary equivalent
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

2.3.6 Binary to Hexadecimal Conversion

1. Group the bits into sets of 4, starting from the LSB. Attach sufficient 0s to the left to make leftmost group contain 4 bits.
2. Convert each group of 4 bits to its hexadecimal equivalent using conversion table.
3. Combine the hexadecimal digits to get the final result.

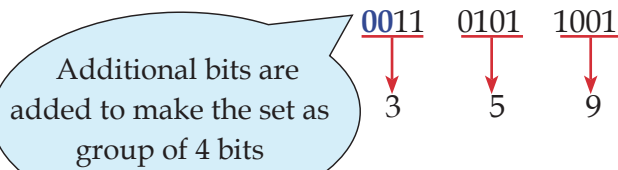
Example: Convert $(100101111011)_2$ to hexadecimal number.

Group the given binary number 100101111011 into a set of 4 bits from right as shown below

<u>1001</u>	<u>0111</u>	<u>1011</u>
↓	↓	↓
9	7	B

$$(100101111011)_2 = (97B)_{16}$$

Example: Convert $(1101011001)_2$ to hexadecimal number.



$$(1101011001)_2 = (359)_{16}$$



1. Convert the following decimal numbers into binary equivalent.
 - a. $(45)_{10}$ b. $(65)_{10}$ c. $(78)_{10}$ d. $(123)_{10}$
2. Convert the following binary numbers into decimal and hexadecimal equivalent.
 - a. $(10110101)_2$ b. $(10101101)_2$ c. $(110101)_2$ d. $(10001)_2$
3. Find missing terms in the following series.

$(1010)_2, (1100)_2, (1110)_2, \underline{\hspace{1cm}}, \underline{\hspace{1cm}}.$
4. Convert the following binary numbers into octal and hexadecimal equivalent.
 - a. $(101111010)_2$ b. $(1111001)_2$ c. $(101010101)_2$ d. $(1000101)_2$

The **binary number system** (base-2) is the foundation of all modern computers because it perfectly matches how electronic hardware works. Computers are built using millions (or billions) of tiny electronic switches called **transistors**, which can be in only two states: **ON** (high voltage, represented as 1) or **OFF** (low voltage, represented as 0). This two-state nature makes binary the simplest, most reliable and error-resistant way to represent data.

Using more states (like decimal's 10 digits) would require precise voltage levels, which are hard to maintain accurately due to electrical noise, heat, or signal degradation—leading to frequent errors. Binary avoids this: a signal is clearly “on” or “off”.

2.4 Binary Addition

Computers do all their calculations using the binary number system, which contains only two digits 0 and 1. Just like we add decimal numbers in everyday life, computers add binary numbers when performing arithmetic operations.

Binary addition follows its own set of rules because the base of the binary system is 2, not 10.

Understanding how binary numbers are added helps us to understand the internal working of computers, logic circuits, and digital electronics. The rules for adding two bits are given in Table 2.4

Table 2.4 -Binary addition rules

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Example: Adding 1101_2 and 0011_2

$$\begin{array}{r}
 1\ 1\ 1\ 1 \quad (\text{carry}) \\
 1\ 1\ 0\ 1 + \\
 \underline{0\ 0\ 1\ 1} \\
 1\ 0\ 0\ 0\ 0
 \end{array}$$

Example: Adding 1111_2 and 1101_2

$$\begin{array}{r}
 1\ 1\ 1\ 1 \quad (\text{carry}) \\
 1\ 1\ 1\ 1 + \\
 \underline{1\ 1\ 0\ 1} \\
 1\ 1\ 1\ 0\ 0
 \end{array}$$

2.5 Numeric Representation

We are writing numbers with or without fractional part. Numbers without a fractional part are called *integers* and numbers with a fractional part are called *floating point numbers*.

In computers, integers and floating point numbers are represented differently. Let us go to the details.

2.5.1 Integer Representation

There are three representation schemes for integers:

1. Sign and Magnitude representation.
2. 1's Complement representation.
3. 2's Complement representation.

A group of bits of fixed-size that are handled as a unit by a processor is called a *word*.

The number of bits in a word is called *word length*. Some popular word lengths are 8, 16, 32, 64 etc.

The following data representation methods are based on 8 bit word length.

a. Sign and Magnitude Representation

The sign and magnitude method is a simple technique used to represent both positive and negative numbers in binary form. In this method:

- The leftmost bit (MSB) represents the sign of the number.
- 0 indicates positive number and 1 indicates negative number.
- The remaining 7 bits represent the magnitude (absolute value) of the number.

To represent a number in sign and magnitude method,

1. Convert the absolute value (magnitude) of the number to 7 bit binary.
2. Determine the sign of the number (positive or negative).
3. Add the sign bit (0 for positive, 1 for negative) at MSB position.

Note: If the binary value of the magnitude is less than 7 bits, add sufficient zeros to the left to make it a 7 bit number

Example: Represent -25 in sign and magnitude using 8 bits.

1. Convert the magnitude 25 to binary : 0 0 1 1 0 0 1
2. Determine the sign of value : Negative (MSB=1)
3. Add the sign bit as MSB.

$$\begin{array}{c} : \boxed{1} \boxed{0011001} \\ \downarrow \qquad \downarrow \\ \text{sign} \quad \text{magnitude} \end{array}$$

b. 1's Complement Representation

The 1's complement method is used by computers to represent negative numbers in binary. A negative number is formed by changing all 0s to 1s and all 1s to 0s in the binary form of the positive number.

To represent a negative number in one's complement form,

1. Convert the number to 8 bit binary (true form).
2. Replace all 0s with 1s and 1s with 0s.

Example: Represent -87 in 1's complement form.

$$\begin{array}{ll} \text{8 bit binary of 87} & = 01010111 \\ \text{1's complement representation of -87} & = 10101000 \end{array}$$

c. 2's Complement Representation

The 2's complement method is the most commonly used method by computers to represent negative integers in binary form. It is simple, efficient, and avoids many problems found in other methods.

To represent a negative number in 2's complement form,

1. Convert the absolute value of the number to 8 bit binary.
2. Find one's complement form.
3. Add 1 to the above.

Example: Represent -25 in 2's complement form.

$$\text{8 bit binary of 25} = (00011001)_2$$

$$\text{1's complement} = 11100110 +$$

$$\text{2's complement} \quad \begin{array}{r} 1 \\ \hline (11100111)_2 \end{array}$$

Note: Complement methods are used only for representing negative integers.

2.5.2 Floating Point Representation

A number which consists of integer part and fractional part is called a floating point number.

A floating point number has two parts:

1. **Mantissa:** This is the actual number, like 13.4. In binary floating-point representation, the mantissa is usually normalized to the form 0.134.
2. **Exponent:** It tells us how many places the decimal point (binary point) should move. It works like scientific notation. In decimal numbers, we use powers of 10, but computers use powers of 2.

For example consider the number 13.4 we can write it as:

$$13.4 = 0.134 \times 10^2$$

Here 134 is the mantissa, 2 is the exponent. In the case of binary it will be represented in binary format.



1. Name the common integer representation methods.
2. Represent the following numbers in sign and magnitude method.
 - a. -19 b. +49 c. -97 d. +127
3. Represent the following numbers in 1's complement method.
 - a. -24 b. -69 c. -100 d. -127
4. Represent the following numbers in 2's complement method.
 - a. -33 b. -71 c. -111 d. -127

2.6 Character Representation

Some of the character representation methods are discussed in this section.

a. ASCII

ASCII (American Standard Code for Information Interchange) is a character encoding standard that represents characters, such as letters, numbers, and symbols, using 7-bit binary codes (128 characters). ASCII was first published in 1963 by the American Standards Association (ASA, renamed ANSI- American National Standards Institute in 1969). It was developed to standardize the representation of characters in computers and communication devices. ASCII was widely adopted and became a standard for character encoding. IBM popularized 8-bit character encodings, commonly referred to as Extended ASCII.

Merits and Demerits of ASCII

Merits:

- Simple and efficient representation
- Widely supported and compatible

Demerits:

- It has a limited character set and does not adequately support multiple languages, special characters, or symbols.

b. Unicode

Since original ASCII is only 7-bit (128 characters) and extended ASCII is 8-bit (256 characters), it cannot represent all the characters and symbols in different scripts. Unicode was introduced to resolve this crisis. It aims to provide a standard character encoding scheme, which is universal, efficient, uniform and unambiguous.

The Unicode standard is maintained by a non-profit organization called the Unicode Consortium. It was originally designed with a 16-bit goal (called UCS-2 or Unicode Character Set - 2 byte), which can represent up to 65,536 characters. It has since been expanded to more than 16 bits; Unicode defines up to 1,114,112 code points and is commonly encoded using UTF-8, UTF-16, or UTF-32. Table 2.5 shows the comparison between various character representation methods.

Merits of Unicode

- Universal Language Support
- Platform Independence

- Simplified Application Development
- Efficient Data Sharing

Table 2.5: The comparison between various character representation methods

Encoding Method	Bits	Character Range	Used For	Limitation
ASCII	7 bits	$2^7 = 128$ characters	English text, control codes	Limited characters
Extended ASCII	8 bits	$2^8 = 256$ characters	European characters	Not global
Unicode	16-32 bits	2^8 to 2^{32} characters	Global language support	Requires more storage

2.7 Multimedia File Formats

2.7.1 Image File Formats

An image is a visual representation of something, it can be a photo, drawing, or graphic displayed on a screen. In computers, an image is represented digitally as a collection of tiny dots called *pixels*. Each pixel has a specific color and brightness value, and when arranged together, they form a complete image. For example, a High Definition (HD) image with a resolution of 1920×1080 has 2,073,600 pixels.

Since computers can understand only binary numbers (0s and 1s), each pixel's color and brightness are converted into binary data. Colors are represented using the RGB color model (Red, Green, Blue), where each component has a value between 0 and 255. Each pixel's color is stored using 24 bits (8 bits each for R, G, and B).

A digital image consists of various sections like the

Header section which contains information like file format, image width, height, color depth, and compression type.

Pixel Data, the actual binary data representing the color values of pixels.

Metadata, data about the data, which is the additional details like date, camera settings, author name, GPS location, etc.

Image file formats are almost like the different languages we speak, they are the means for saving and displaying images. The data can be stored in uncompressed form or compressed form to reduce the file size. Normally, the image data is stored in compressed form.

Compression is of two types,

Lossy compression: Removes certain data and achieves significant reduction in file size.

Lossless compression: Reduces file size by removing redundant data without discarding any information from the original image. The original image can be reconstructed by decompression and preserves image quality with no data loss.

Depending on the application, images are stored in various file formats. Some of the popular image file formats are listed as follows.

i) BMP (Bitmap Image File)

- Developed by Microsoft.
- Stores pixel data without compression.
- Large size but high quality.
- Used for printing and editing.
- Used in Windows applications and image editing where detail is crucial.

ii) JPEG / JPG (Joint Photographic Experts Group)

- Uses lossy compression.
- Good balance between quality and size.
- Common for photos and web images.

iii) PNG (Portable Network Graphics)

- Uses lossless compression.
- Supports transparency.
- Ideal for logos and web graphics.

iv) GIF (Graphics Interchange Format)

- Limited to 256 colors.
- Supports animation and transparency.
- Used for web animations, memes, and short graphics.

v) TIFF (Tagged Image File Format)

- Used by professionals.
- Supports both lossy and lossless compression.
- High quality and large file size.

Table 2.6 shows the comparison between various image file formats.

Table 2.6: Image file formats comparison

Format	Compression Type	Transparency	Animation	Image Quality	File Size	Common Use
BMP	Uncompressed	No	No	Excellent	Very Large	Windows apps, raw image storage
JPEG	Lossy	No	No	Good	Small	Photos, web images
PNG	Lossless	Yes	No	Excellent	Moderate	Logos, transparent images
GIF	Lossless	Yes	Yes	Low (256 colours)	Small	Animations, web graphics
TIFF	Lossless / Lossy	Yes	Yes	Excellent	Very Large	Professional design

2.7.2 Audio File Formats

Audio refers to any sound that can be heard, such as speech, music, or ambient noises. In computing, audio represents sound that has been converted into digital form so that computers can store, process, and transmit it. Since computers can only understand binary, natural sound waves known as analog signals are converted into digital signals and this process is called *digitization*.

Some of the popular audio file formats are listed below,

i) WAV (Waveform Audio File Format)

- Developed by Microsoft.
- WAV stores audio in uncompressed form, ensuring very high quality but large file sizes.
- It's ideal for professional editing and recording.

ii) MP3(MPEG-1 Audio Layer 3)

- Uses lossy compression to remove inaudible sounds, balancing quality and file size.
- Widely supported across devices.
- Common for portable music players, online sharing, and general listening.

iii)AAC(Advanced Audio Coding)

- An improved version of MP3 offering better sound quality at the same bit rate.
- Commonly used in streaming platforms.

iv) FLAC (Free Lossless Audio Codec)

- Lossless compression, reducing file size without affecting the audio quality.
- Preferred for archiving and high-fidelity playback.

v) OGG (Ogg Vorbis)

- OGG is actually a container format, and Vorbis is the audio compression format commonly used inside it.
- An open-source format that provides good compression efficiency.
- Used in open-source applications and interactive games.

vi) WMA (Windows Media Audio)

- Microsoft's format offers both lossy and lossless options.
- Works best within Windows applications.
- Table 2.7 shows the comparison between various audio file formats.

Table 2.7: Audio file formats comparison

Format	Compression Type	Quality	File Size	Common Uses	Special Features
WAV	Uncompressed	Excellent	Very Large	Studio recording, editing	Original sound data
MP3	Lossy	Good	Small	Music, streaming	Widely compatible
AAC	Lossy	Better than MP3	Small	Streaming, Apple devices	High efficiency
FLAC	Lossless	Excellent	Medium	Archiving audio editing	Retains quality
OGG	Lossy	Good	Small	Gaming, streaming	Open - source
WMA	Lossy/Lossless	Good - Excellent	Small - Medium	Window apps	Microsoft format

2.7.3 Video File Formats

A video is a sequence of moving images, often combined with audio, that create the illusion of motion when displayed rapidly one after another. In other words, a video is made up of a series of still images (called frames) moved quickly, usually 24 to 60 frames per second (fps) along with sound. For example, when you are watching a movie, you are actually viewing thousands of images displayed in quick succession, which our eyes perceive as smooth motion.

Just like images and audio, a video is also converted into digital form before a computer can store or process it. Each frame is stored digitally, often in RGB color format. Videos are compressed to reduce file size and audio tracks are synchronized with video frames to create a complete audiovisual experience. The compressed video and audio streams are stored together in a single file container (e.g., MP4, AVI, MOV). Table 2.8 shows the comparison between various video file formats.

Table 2.8: Video file formats comparison

Format	Developer	Compression Type	Quality	File Size	Compatibility	Common Uses
AVI	Microsoft	Lossy/ Lossless	Excellent	Very Large	Windows	Editing, archiving
MP4	MPEG	Lossy	High	Small	Universal	Web, mobile, streaming
MKV	Open Source	Lossy/ Lossless	Very High	Medium	Platform-dependent / Wide software support	Movies, HD videos
MOV	Apple	Lossy	Excellent	Large	macOS	Professional editing
FLV	Adobe	Lossy	Good	Small	Browser dependent / Declining platform support	Old online streaming
WMV	Microsoft	Lossy	Good	Small	Windows	Media playback
WebM	Google	Lossy	High	Small	Browsers	HTML5 streaming

◆ Summary ◆

This chapter explains how computers store and process information using binary digits (0 and 1). It introduces the concept of number systems such as decimal, binary, octal and hexadecimal. The importance of the binary number system is emphasized, as it forms the foundation of computer hardware and software. We discussed number system conversions between decimal, binary and hexadecimal forms using systematic methods. Binary addition and its rules are explained to show how arithmetic operations are performed inside a computer. Different methods of number representation—including sign and magnitude, 1's and 2's complement mantissa-exponent—are discussed. The chapter explains character representations such as ASCII, Unicode and discusses multimedia file formats and compression techniques used in computing.



1. Data representation in computers primarily uses:
 - a) Decimal b) Octal c) Binary d) Hexadecimal
2. The base of a number system is also known as:
 - a) Digit b) Place value c) Radix d) Weight
3. The base of the hexadecimal number system is _____.
 - a) 2 b) 8 c) 10 d) 16
4. Which number system uses digits 0–7?
 - a) Binary b) Decimal c) Octal d) Hexadecimal
5. Which number system uses letters A–F?
 - a) Binary b) Decimal c) Octal d) Hexadecimal
6. Each octal digit corresponds to how many binary bits?
 - a) 2 b) 3 c) 4 d) 8
7. Decimal to binary conversion uses repeated division by _____.
 - a) 10 b) 8 c) 2 d) 16
8. Binary to hexadecimal conversion groups bits in sets of
 - a) 2 b) 3 c) 4 d) 8
9. In binary addition, $1 + 1$ results in _____.
 - a) 1 b) 10 c) 11 d) 0
10. Which integer representation is most widely used by computers?
 - a) Sign and magnitude b) 1's complement c) 2's complement

11. ASCII uses how many bits?
a) 5 b) 7 c) 8 d) 16
12. Which format supports transparency?
a) BMP b) JPEG c) PNG d) None of these
13. Which audio format is uncompressed?
a) MP3 b) AAC c) WAV d) OGG
14. Which format is universally compatible?
a) AVI b) MOV c) MP4 d) MKV
15. Name the basic language understood by a computer.
16. What does word length mean?
17. Which bit represents the sign in sign-magnitude form?
18. Name the two main parts of a floating-point number.
19. Define data representation.
20. Explain the positional number system with an example.
21. Explain the decimal number system with place values.
22. Explain decimal to binary conversion with an example.
23. Explain the rules of binary addition.
24. Why is binary addition important in computers?
25. Explain sign and magnitude representation.
26. Describe 1's complement method with an example.
27. Explain 2's complement method and its advantages.
28. Expand ASCII.
29. What is ASCII? Explain the merits and demerits of using it.
30. Why was Unicode introduced? Explain its advantages over ASCII.
31. Explain lossy and lossless compression techniques with suitable examples.
32. Compare ASCII, Extended ASCII, and Unicode.
33. Discuss any five image file formats.
34. Explain digitization of audio signals.
35. Compare different audio file formats with examples.
36. Compare any three video file formats.
37. Discuss the importance of multimedia file formats in modern computing.



3

Programming Concepts



“The function of a good software is to make the complex appear to be simple”

Edward Bernard

3.1 Introduction

We have already explored different components of the computer system (hardware, software, and users) and understood how they collaborate to perform the tasks. This chapter shifts the focus to software from the perspective of a creator: the programmer.

In simple terms, software is a collection of computer programs, which are sets of instructions written in a language that a computer can understand and execute to solve specific problems. Because computers lack the ability to make assumptions, they require precise, step-by-step instructions to carry out even the simplest operations. When these instructions are organized logically, they enable the machine to process data and produce the desired results.

However, writing a computer program is not merely about typing commands. It requires a clear understanding of the problem, the design of a logical solution, and the translation of that solution into a programming language. This holistic process is known as Program Development. Mastering these skills allows us to break down complex tasks into smaller, manageable sub-tasks. Before we can write efficient software, we must first understand the fundamental approach to solving problems using a computer.

3.2 Problem Solving Using Computers

We solve problems every day. Whether it is solving a math puzzle, planning a travel route, or even making a cup of tea, we naturally break the task into smaller steps to get the result we want.

When we use a computer to solve a problem, the approach is very similar. However, there is one major difference: humans have common sense, but computers do not. A computer is just a machine. It is incredibly fast and never gets tired, but it cannot think or guess. It does not know how to solve a problem unless we give it specific, detailed instructions.

Problem solving using computers is a real-world problem and breaking it down into a list of very simple steps that the computer can follow without confusion. We act as the planner, figuring out the solution in our mind, and then direct the computer to execute that plan.

3.3 Phases in Programming

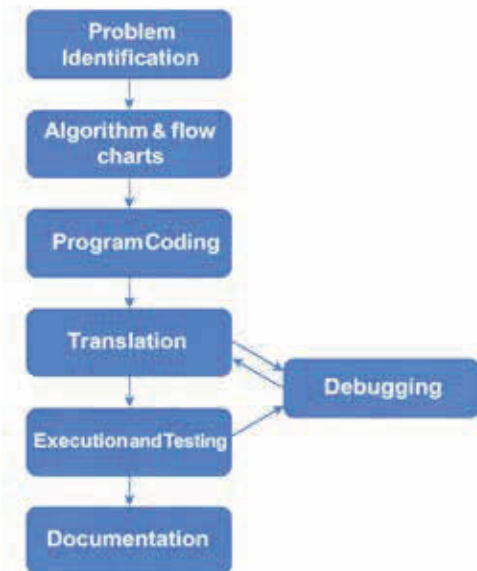


Figure: 3.1 - Phases of Programming

Writing a computer program is just like constructing a building. You cannot simply start stacking bricks to construct a building; you need a plan first. Similarly, in programming, we cannot just start typing code. We need to follow a clear, step-by-step process to ensure that our program works correctly. This process involves two main types of thinking:

- **Analytical Thinking:** Breaking a big problem into smaller, easier-to-handle parts to understand it better.
- **Creative Thinking:** Finding smart and efficient ways to solve those parts.

To guide us through this, we follow a specific cycle called the phases of programming. These phases act as a road map, from the initial problem statement to a finished, working program.

The key phases are:

1. Problem Identification
2. Preparing Algorithms and Flowcharts
3. Coding the Program
4. Translation
5. Debugging
6. Execution and Testing
7. Documentation

To illustrate these phases, we will consider a specific scenario: Develop a program to calculate and display the total marks and percentage of mark of a student based on scores in six subjects.

3.3.1 Problem Identification

Problem identification is the foundation of the entire programming process. It is the phase where we clearly define the gap between the current state (the problem) and the desired state (the solution). Before we instruct the computer how to do something, we must first clearly understand what needs to be done.



Figure: 3.2 - Problem Identification

This process involves analyzing the problem statement to filter out unnecessary details and focusing on the core requirements. To identify a problem effectively, we specifically look for three key components:

1. What raw data or facts do we have?
2. What is the final result we want the computer to produce?
3. How do we transform the data into result?

By explicitly defining these three aspects, we ensure that our program will solve the problem.

Let us apply this process to our example. Our initial idea is: Calculate total marks for a student. This is too vague for a computer. To make it precise, we break it down into its three core components immediately:

1. *Data Needed:* The computer requires marks (out of 100) for six different subjects. We can label them as M1, M2, M3, M4, M5 and M6.
2. *Desired Result:* The user expects to see the total marks and the percentage of marks.
3. *The process:* We need to define the calculations required to get the output.
 - $\text{Total} = M1 + M2 + M3 + M4 + M5 + M6$
 - $\text{Percentage} = (\text{Total} / 600) \times 100$

Final Problem Statement : Based on this analysis, we can write a clear definition of our problem:

Create a program that accepts mark of six subjects as input, calculates their sum and percentage using standard formulas, and displays both results.

Since the problem is clear, logically sound, and solvable, we are ready to move to the next phase: Preparing algorithms and flowcharts.

3.3.2 Algorithms and Flowcharts



Figure: 3.3 - Preparing Algorithms and Flowcharts

Once the problem is identified, we might be tempted to jump straight to the computer and start typing. However, this is like trying to build a house without a blueprint—it often leads to mistakes and confusion. Before writing a single line of code, a programmer must design a logical plan.

This planning phase helps us to organize our thoughts and verify our logic without getting distracted by the strict grammar rules of programming languages. To create this plan, we use two powerful tools: Algorithms and flowcharts.

a. Algorithm



Figure 3.4 - Muhammad Ibn Musa Al-Khwarizmi

An algorithm is a step-by-step procedure to solve a problem. Think of it like a recipe for cooking or the path to a friend's house. Before you attempt the task, you write down the steps to ensure you don't miss anything. The word *Algorithm* comes from the name of a famous Persian mathematician, Muhammad Ibn Musa Al-Khwarizmi, who lived around 820 AD. He is often called the father of algebra. Long before computers existed, he developed methods to solve mathematical problems using sequential steps.

We use algorithms every day without realizing it. When you get ready for school, you follow a specific order: wake up, brush your teeth, wear the uniform, and pack your bag. That routine is an algorithm! If you skip a step or change the order (like putting on shoes before socks), the result might not be correct. In programming, writing an algorithm is the planning stage. Since computers need precise instructions, we first write the solution in plain English (the algorithm) to check our logic. Once the algorithm is clear and correct, writing computer code becomes much easier.

Guidelines for Writing Algorithms

Writing an algorithm is like telling a story that explains how to solve a problem. At this stage, you do not need to use computer language, but the explanation must be clear and easy to understand. Before writing the steps of the algorithm, it is important to understand how the information will be handled. The guidelines for writing algorithms is given below.

1. Storing Information: Imagine you are cooking. You cannot just hold water, sugar, and flour in your hands all at once; you need bowls and jars to keep them organized. Computers are of the same situation. They need a place to store data while they work on it. We give unique names to these storage spaces (like Marks, Age, or Result) so that we can find them later.

2. Writing the Steps : Use simple language to describe the actions.

- *Getting Information (Input):* When you want the computer to ask for information and store it in one of the containers, use words like Input, Read, or Get.

Example: Input Number1 (This tells the computer: Ask for a number and put it in the box named Number1).

- *Doing the Work (Process):* When you need to change data or do calculation, use action words like Calculate, Add, or Set.

Example: Calculate Total = Number1 + Number2 (This tells the computer: Add the numbers inside the boxes Number1 and Number2, and put the answer in a new box named Total).

- *Showing the Answer (Output):* When the process is done and you want to show the result to the user, use words like Print, Display, or Show.

Example: Print Total (This tells the computer: Show the user what is inside the box named Total).

3. General Rules for the Algorithms

- **Start and Stop:** Every algorithm must have a specific beginning (Start) and ending (Stop).
- **Be Clear (Unambiguous):** Do not use confusing words. Instead of saying *Do the math*, say exactly what math to do (example, Add A and B).
- **Simple Steps:** Keep each step short. One action per step is best.
- **Must End:** The algorithm cannot go on forever; it must reach a finish line.

Real-Life Example: The Algorithm of Making Lime Juice

To understand the structure of an algorithm, let's look at a simple everyday task: making lime juice. Even though this isn't a computer program, it follows the exact same logic of data processing.



Figure 3.5- Making Lime Juice

- Step 1 : Start Explanation - Every algorithm needs a clear beginning.
- Step 2 : Get Lime, Sugar, Water, Ice - This is the Input step. We gather the specific ingredients (variables) we need.
- Step 3 : Cut Lime - This is a Process step. We are acting on the raw input (lime) to prepare it.
- Step 4 : Squeeze Lime into Glass - This is a Process step. We are extracting the core part of our input (the juice).
- Step 5 : Add Sugar and Water - This is a Process step. We are combining different inputs to transform them.
- Step 6 : Stir Well - This is a Process step. We are mixing everything to finalize the result.
- Step 7 : Serve Juice - This is the Output step. We are presenting the final result.
- Step 8 : Stop - Every algorithm must have a clear ending.

This example demonstrates the core concept of any computer algorithm:

1. Input: Just as we cannot make juice without ingredients, a computer cannot work without data.
2. Process: Just as we must squeeze and stir in the right order, a computer needs a set of sequential instructions to process that data.
3. Output: Just as we get a refreshing drink at the end, the computer gives us useful information (the solution).

Once we understand how to write algorithm for simple tasks, we can easily write them for complex computer programs.

Example 1: Write an algorithm to find the area and perimeter of a rectangle.

Problem: Calculate area and perimeter of a rectangle. Given length and breadth.

Step 1: Start

Step 2: Input Length, Breadth

Step 3: Perimeter = $2 \times (\text{Length} + \text{Breadth})$

Step 4: Area = Length $\times$ Breadth

Step 5: Print Perimeter, Area

Step 6: Stop

Flow of Control

Consider your daily routine at school. Most of the time, you follow a predictable order. When the bell rings, step into the classroom, take your seat, and open your book. However, your schedule is dynamic. Sometimes, you face a decision based on the situation. If it is raining during the sports period, stay in class; otherwise, go to the playground. At other times, you must repeat a specific action, such as solving five math problems or running three laps around the ground. Think about your daily routine at school. Most of the time, you follow a fixed schedule. Computer programs work the same way. They do not always run in sequential order. Sometimes they need to make decisions or repeat certain tasks. The path through which computer takes the instructions is called the *Flow of control*. We need different ways to control this flow because real-world problems are complex. We cannot solve everything with a simple list of steps. To handle choices and repetition, we use three primary types of control flow:

1. **Sequential Control Flow (The Straight Line):** Sequential flow is the simplest way a computer follows instructions. Imagine reading a book; you read the first sentence, then the second, and so on until the end. You do not skip pages or read backwards. Similarly, in sequential control flow, the computer executes every instruction in the exact order it is written, from the top to the bottom. It performs Step 1, finishes it, moves to Step 2, finishes it, and continues this pattern until the program ends. A real-life example is baking a cake. You must mix the ingredients first, then pour the batter into a pan, and finally put it in the oven. You cannot change this order or skip a step, or the cake will not turn out right. The above *Example1* illustrates the sequential control flow.
2. **Branching or Selection (The Crossroad):** In many situations, a straight path is not enough because we need to make decisions based on the current situation. Branching, also known as *Selection*, allows the program to choose between different paths. The computer asks a question or checks a specific condition (like 'Is it raining?'). If the answer is 'Yes' (True), it follows one set of instructions. If the answer is 'No' (False), it follows a different set. This is similar to reaching a crossroad or a traffic light. If the light is green, move the vehicle forward. If the light is red, stop the vehicle. In an algorithm, we represent this using 'If... Then... Else' logic. This capability allows computer programs to be smart and react differently to different inputs.

Example 2: Write an algorithm to find the highest of two different scores.

Problem: Compare two test scores (M1, M2) and displays the higher one.

Step 1: Start.

Step 2: Input M1, M2

Step 3: If $M1 > M2$, then Go to Step 4 else Goto Step 5

Step 4: Print "M1 is higher". Goto Step 6

Step 5: Print "M2 is higher".

Step 6: Stop.

3. **Iteration or Looping (The Cycle):** Often, a task requires doing the same thing multiple times. Instead of writing the same instruction over and over again, programmers use Iteration, commonly called *Looping*. A loop allows a set of instructions to be repeated as long as a certain condition remains true. Once the condition is false, the loop stops. This makes programs much shorter and more efficient. Think about walking up a long flight of stairs. You do not think *lift foot, place foot* for every single step individually. Instead, you repeat the action of *stepping up* until you reach the top floor. In algorithms, we use terms like Repeat or While to indicate this cycle, such as Repeat adding and stirring sugar until the tea is sweet enough.

Example 3: Write an algorithm to display the first 10 multiples of a number.

Problem: Print the first 10 multiples of a number N.

Step 1: Start

Step 2: Read N

Step 3: Set A to 1

Step 4: Calculate M as $A \times N$

Step 5: Display M

Step 6: Increase A by 1

Step 7: Repeat Steps 4 to 6 while A is less than or equal to 10

Step 8: Stop

Example 4: Algorithm - Student Result Management.

Now, let us put everything we have learned into practice. We have

already identified the problem in the previous section : we need to take the marks of six subjects, find the total mark, and calculate the percentage. Writing this algorithm is just like writing down the steps to solve a mathematical problem. We need to instruct the computer to read numbers, apply the formulas and display the answers.

The Logic:

1. Start: Begin the process.
2. Get Data: Ask for the marks of all six subjects.
3. Calculate: Add all the marks together to find the Total.
4. Define Constant: We need to know the maximum possible marks (e.g. 600) to calculate the percentage.
5. Calculate (Division/Multiplication): Apply the standard percentage formula.
6. Show Results: Display the calculated Total and Percentage.
7. Stop: End the process.

The algorithm:

Step 1: Start.

Step 2: Input M1, M2, M3, M4, M5, M6.

Step 3: $\text{Total} = M1 + M2 + M3 + M4 + M5 + M6$.

Step 4: $\text{Max} = 600$.

Step 5: $\text{Percentage} = (\text{Total} / \text{Max}) \times 100$.

Step 6: Display Total, Percentage.

Step 7: Stop.

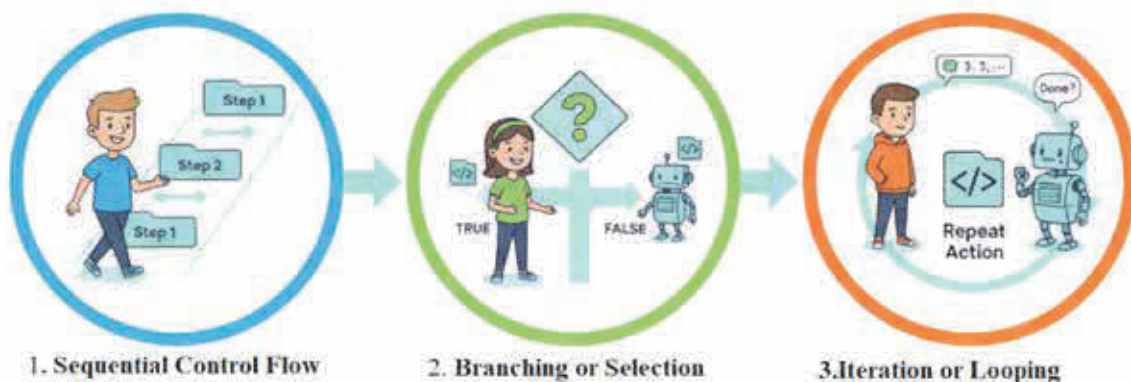


Figure 3.6

Benefits of Using Algorithms

1. **Efficient Problem-Solving:** Algorithms help us handle large or difficult problems by breaking them into smaller, easy-to-manage steps. Instead of finding solution to the whole problem at once, we focus on solving one small step at a time.
2. **Improved Accuracy:** Since an algorithm follows a specific, step-by-step order, it reduces the chance of making mistakes. If the logic of your algorithm is correct, the final result will always be correct.
3. **Scalability:** A good algorithm works well for small tasks and large tasks alike. For example, the logic used to find the average marks of one class can easily be used to find the average for the entire school.
4. **Foundation of Programming:** Algorithms act as the blueprint for your code. Once you have a clear algorithm written in plain English, you can easily develop a computer program using a programming language like Python, C++, etc.
5. **Clarity:** Writing an algorithm forces us to think clearly and logically. It helps us to spot errors in our thinking or missing steps before we start coding.

b. Flowchart

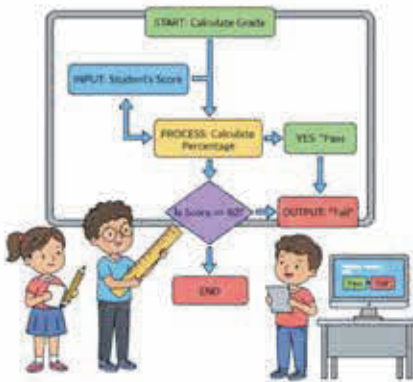


Figure 3.7
Flowchart preparation

We have learned that an algorithm is a step-by-step set of instructions written in plain language. However, sometimes reading a long list of textual steps can be confusing, especially when the problem involves making choices or repeating tasks. This is where a flowchart comes in handy.

A *flowchart* is a pictorial representation of an algorithm. It basically converts the written steps of an algorithm into a diagram. Just as a map helps you navigate a route better than a list of written directions, a flowchart helps programmers visualize the flow of control in a program. Hence

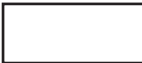
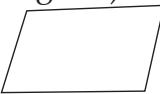
a flowchart is often easier to understand and analyze than a text-based algorithm. It shows clearly where the program starts, what decisions are made, and how the logic moves from one step to another.

It allows programmers to spot logical errors quickly before they start writing the actual code. To ensure that everyone understands

these diagrams in the same way, flowcharts use standard geometric shapes (like rectangles, diamonds, and ovals) to represent different types of instructions. These shapes are connected by arrows to show the direction of the flow.

Standard Flowchart Symbols

Flowcharts rely on a set of universal symbols to communicate logic clearly. Each shape represents a specific type of action or step. Using these standard symbols ensures that anyone reading the flowchart can understand the process without confusion.

Symbol	Name	Description
Terminator (Rectangle with Rounded Ends) 	Start / End	Marks the beginning or end of a process.
Process (Rectangle) 	Process / Action Step	Represents a process, task, or action to be performed.
Decision (Rhombus or Diamond) 	Decision Point	Indicates a decision or branching point (Yes/No or True/False).
Input/Output (Parallelogram) 	Input / Output	Represents input to or output from a process (e.g, user input, data display)
Arrow 	Flow line	Shows the direction of flow between steps.
Connector 	Connection/Link	To connect or link between different sections of flowcharts.

Advantages of Flowcharts

Flowcharts are not just drawings; they are powerful tools that make the entire programming process smoother. Their primary benefits include:

1. **Better Communication:** It is often said that a picture is worth a thousand words. A flowchart acts as a visual map of your program. It is much easier to explain the logic of a complex problem to other programmers (or even non-programmers) using a diagram than by reading out long lines of text or code. It serves as a universal language for logic.
2. **Effective Analysis:** By visually mapping out the flow of data, we can analyze the problem more effectively. It allows us to see the big picture and spot logical errors, dead ends, or missing steps before we even start writing code. This early detection saves a lot of time during the debugging phase.
3. **Proper Synthesis:** Large software systems are rarely written as one giant block of code. They are usually built by combining many smaller, independent programs (modules). Flowcharts help us visualize how these different modules connect and interact with each other, ensuring the entire system works together harmoniously.
4. **Efficient Coding:** Once a flowchart is finalized, the difficult task of thinking through the logic is mostly complete. The actual coding phase becomes a simple task of translation. The flowchart acts as a blueprint or a road map, guiding the programmer from the start to the end, ensuring no crucial step is forgotten.

Limitations of Flowcharts

Despite their significant benefits, flowcharts have practical limitations that programmers should be aware of:

- **Time-Consuming:** Drawing a neat, readable flowchart with the correct symbols and proper spacing takes significantly more time than writing a simple text-based algorithm. For very large or complex programs, creating a detailed flowchart can be a slow and laborious process.
- **Difficult to Modify:** Programs often change. If the logic of the solution changes (for example, adding a new decision branch), the flowchart often needs to be completely redrawn to accommodate the new shapes and arrows. Unlike text, which can be easily edited or inserted, rearranging a diagram is often difficult.

Flowchart Examples. Let us visualize the algorithms we wrote earlier. These diagrams show exactly how the computer moves from one instruction to the next.

1. **Sequential Flowchart:** Area and perimeter. This flowchart follows a linear or sequential structure. The flow enters at the top and leaves at the bottom, visiting every step exactly once. There are no detours.

- *How it works:* The arrow points directly from the Start to Input, then to the Process (calculations), then to Output (Print), and finally to Stop. The computer has no choices to make; it just follows the path.

2. **Branching Flowchart:** Highest of two scores. (This flowchart introduces a decision. Notice how the path splits into two branches).

- How it works: The flow reaches the diamond shape (the Decision). Here, a question is asked: Is $M1 > M2$?
 - If the answer is Yes, the flow follows the path to print M1 as the highest score.
 - If the answer is No, the flow follows the path to print M2 as the highest score.
- Essentially, the flow cannot go down both paths. It must choose one based on the data. Finally, both paths merge back together to reach the Stop symbol.

3. **Iteration Flowchart:** First 10 multiples of a number. This flowchart shows a loop. Notice the arrow that goes backwards (upwards) to a previous step.

How it works:

1. The computer calculates and prints the multiple.
2. It increases the counter A by 1.
3. It reaches the decision diamond: Is A less than or equal to 10?
4. The Loop: As long as the answer is Yes, the arrow sends the flow back up to the calculation step. This causes the computer to repeat the work.
5. The Exit: When A becomes 11, the answer becomes No, and the flow finally breaks out of the loop and goes to Stop.

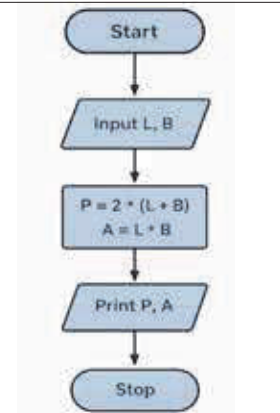


Figure 3.8
Sequential Flowchart

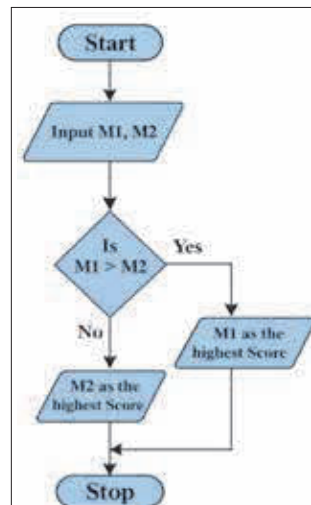


Figure 3.9
Branching Flowchart

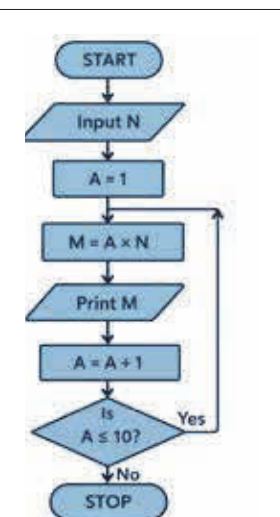
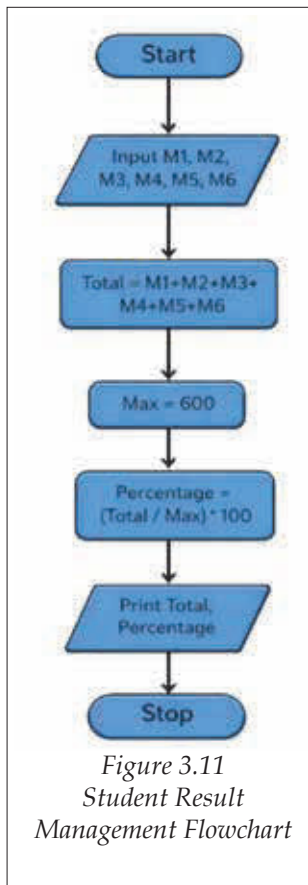


Figure 3.10
Iteration Flowchart

4. Applied Example: Student Result Management Flowchart



Finally, let us draw the flowchart for our main problem: calculating the total marks and percentage of marks of a student. This brings together the concepts of sequential flow and data processing.

Explanation of the Flow:

- Start: The program begins.
- Input: The computer asks for the marks of all six subjects (M1 through M6). Without this data, no calculation can happen.
- Process (Calculations):
 1. First, the computer adds all the marks together to find the Total.
 2. Next, it sets the Maximum marks to 600 (assuming each subject is out of 100).
 3. Then, it uses the percentage formula to calculate the Percentage.
- Output: The computer displays the final Total and Percentage to the user.
- Stop: The program has finished its task and ends.



1. Who is known as the father of algebra and the concept of algorithms?

a) Charles Babbage	b) Alan Turing
c) Muhammad Ibn Musa Al-Khwarizmi	d) Blaise Pascal
2. Which symbol in a flowchart represents a decision or branching point?

a) Rectangle	b) Oval	c) Parallelogram	d) Diamond
--------------	---------	------------------	------------
3. Define Algorithm.
4. What is an Algorithm? Explain its characteristics and benefits using a real-life example.
5. Describe the standard symbols used in Flowcharts.
6. Write an algorithm to find simple interest.
7. List the advantages of Flowcharts

3.3.3 Coding

After we have carefully designed our algorithm and visualized it with a flowchart, we have a solid plan. However, a plan on paper is not enough for a computer to run. The computer cannot read our English sentences or look at our diagrams to understand what we want. We must code our logic using a specific language that the computer can process. This step is called coding.

The Language Barrier - Humans vs. Computers:

To understand why we need coding, we must first look at the fundamental difference between how humans speak and how computers think.

- Computers are electronic machines. At the most basic level, they operate using electricity, which can either be ON or OFF. Because of this, the native language of a computer is Binary Code, consisting entirely of 0s (Off) and 1s (On). This is also known as *Machine Language*. For humans, writing complex instructions using only zeros and ones is incredibly difficult, slow, and prone to mistakes.
- Humans communicate using Natural Languages like English, Malayalam, Hindi etc. These languages are expressive and easy for us to understand, but they are often too vague and complex for a computer's strict logic.

To simplify programming, computer scientists developed High-Level Languages (HLL). These languages act as a bridge between human language and machine code. They allow programmers to write instructions using familiar English words (like print, input, if, and while) and standard mathematical symbols. This makes writing programs much easier and faster for humans.

The set of instructions that we write in a high-level language is called the *Source code*.

The transition from planning to development is known as Coding, which is essentially the process of translating your algorithm into source code. While the algorithm defines exactly what needs to be done and the flowchart visualizes the logical order of those steps, the code provides the specific commands required for the computer to execute the task.

There are many popular programming languages used today, such as C++, Java, and Python. Each language has its own grammar rules, known as syntax. In the upcoming chapters, we will focus on



Figure 3.12
Coding the program

Python. We chose Python because its syntax is very simple, clean, and readable, making it an excellent language for beginners to learn the art of programming.

Coding : Program to find the area and perimeter of a rectangle

```
L = float(input("Enter Length: "))
B = float(input("Enter Breadth: "))
P = 2 * (L + B)
A = L * B
print("Perimeter is:", P)
print("Area is:", A)
```

Coding of Student Result Management System, here is the code for our main project. Notice how closely it follows the steps we wrote in our algorithm earlier.

```
m1 = int(input("Enter Mark 1: "))
m2 = int(input("Enter Mark 2: "))
m3 = int(input("Enter Mark 3: "))
m4 = int(input("Enter Mark 4: "))
m5 = int(input("Enter Mark 5: "))
m6 = int(input("Enter Mark 6: "))
total = m1 + m2 + m3 + m4 + m5 + m6
max_marks = 600
percentage = (total / max_marks) * 100
print("Total Marks =", total)
print("Percentage =", percentage)
```

3.3.4 Translation

Let us consider the Python program just written for the student result management system. Even though it looks like a complete code, the computer can not run it immediately in this form. Python code is just a simple text, whereas computer understands only the machine language. So the program should be translated into machine language.

The program we have written is referred as source code. This source code is fed into a specialized software program called a *Language Translator* (or *Language Processor*). The translator takes our human-readable instructions and converts them into *Object code*

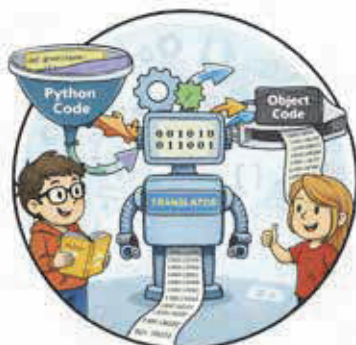


Figure 3.13 - Translation

(machine-readable binary code). Once the translation process is complete, the program is ready for execution.

There are two main translators used to convert high-level languages: the Compiler and the Interpreter.

1. **Compiler:** A compiler reads the entire source program at once, checks for errors, and translates it into a complete file of machine code known as object code. Once this object code is generated, the program can be executed multiple times without needing to be translated again. Languages such as C and C++ use compilers to create these standalone executable files.
2. **Interpreter:** An interpreter reads the source code one line at a time, translates it into machine language, and executes it immediately before moving to the next line. Unlike a compiler, it does not generate a separate object code file. If the interpreter encounters an error, it stops execution instantly at that line. This makes it easier to identify bugs during development. Popular languages like Python and JavaScript use interpreters.

Comparison between Compiler and Interpreter

Compiler	Interpreter
Translates the whole program at once	Translates and executes the program line by line
Generates a separate object code file	Does not generate an object code file
Scans the entire code and lists all errors at the end	Stops execution immediately when the first error is found

During this phase, the interpreter checks if we followed the syntax of the programming language. If there is any *error*, the translation will fail, and we must fix the code before proceeding. In the case of student result management program, the Python instructions constitute the source code. For the computer to calculate the marks, this source code is fed into a proper translation software (in this case, the Python interpreter). The interpreter processes these instructions and converts them into machine-readable object code. Once this conversion is successfully completed without errors, the object code is ready for execution, allowing the computer to perform the addition and division operations and display the final results.

3.3.5 Debugging

Programming is complex, and it is rare to produce perfect code on the first attempt. Mistakes are bound to happen; in the world of



Figure 3.14 - Debugging

computing, an error in a program is historically referred to as a *Bug*. The process of identifying, locating, and correcting these bugs is called *Debugging*.

Debugging is a continuous process that spans across different phases of programming, such as translation and execution. A program acts like a patient, and the programmer acts like a doctor diagnosing the errors. Generally, errors in the program are classified into three main categories. They are:

1. Syntax Errors: Just as English has grammar rules, programming languages have strict rules known as Syntax. If you violate these rules, the computer gets confused and cannot understand your instruction.

Syntax errors are typically caused by misspelling a keyword (such as typing 'prnt' instead of 'print'), forgetting essential punctuation like colons or commas, or referencing a variable that has not been declared. These errors are detected during the Translation phase, where the compiler or interpreter will refuse to execute the program. When this occurs, the translator usually identifies the specific line of code where the error was found, helping the programmer fix the issue before trying again.

2. Logical Errors: Logical errors are the most difficult bugs to detect. Here, the program runs perfectly without crashing, but it gives the wrong result. The computer is doing exactly what you instruct it to do, but your instruction was incorrect.

Logical errors occur when a programmer uses an incorrect formula, flawed logic, or mistakes the mathematical order of operations. Because the syntax is correct, these errors are not detected by the compiler or interpreter. Instead, they are found during testing by comparing the actual output to the expected result. For example, to find the average of 10 and 20, the correct formula is $(10+20)/2=15$. If the brackets are forgotten and written as $10+20/2$, the computer follows the standard order of operations, dividing first to get 10 and then adding 10. This results in an incorrect answer of 20, creating a logical error that the programmer must fix.

3. Run-time Errors: These errors occur while the program is successfully running, even if the code has no syntax or obvious logical flaws. They are typically triggered during the execution phase by unexpected events, such as invalid user input. A classic example is a 'ZeroDivisionError', which happens if a program tries to perform a mathematically impossible operation-like dividing 100 by zero-because

Syntax Error

The mistakes in the rules of the language (Grammar) that prevents the program to start.

Logical Error

The mistakes in the logic or steps that makes the program to give the wrong answer without stopping.

Runtime Error

The mistakes that happens while the program is running, causing it to stop suddenly.

a user entered '0' when prompted for a number. When this happens, the computer doesn't know how to proceed and the program crashes abruptly.

Debugging Student Result Management Program

Accuracy is the top priority for our student result management program. In a real-world grading system, errors can lead to serious consequences, such as incorrect reports or system crashes during data entry. To ensure the software is reliable and trustworthy, we must diligently fix any syntax or logical mistakes. By verifying that our calculations for totals and percentages are mathematically sound and that the program handles all user inputs safely, we create a professional tool that teachers and students can depend on. As we discussed earlier, the process of Debugging involves finding these specific mistakes and fixing them.

3.3.6 Execution and Testing

- Once the translation process is complete and the program is free of syntax errors, the computer generates the object code. We are now ready to run the program. This phase is known as execution and testing.
- Execution is simply running a program to process data, while testing is the intentional process of checking that program for errors. In short: execution proves the code works, but testing proves the code is correct by verifying its results under different conditions.
- Testing is essential because a program can run smoothly while still producing incorrect results due to hidden logical errors. It also ensures the software won't crash for different users due to unexpected run-time errors. By testing under various conditions, we identify and fix these issues before the software reaches the final user, ensuring the program is both accurate and reliable.
- To test our student result management program, we first enter normal marks (like 50 to 100) and check if the total (450) and percentage (75%) match our manual calculations. Next, we test the boundaries by entering 0 or 100 for all subjects to ensure the program handles minimum and maximum values correctly. Finally, we test for human error: if a teacher accidentally types the word *Fifty* instead of the number 50, the program might crash with a run-time error. This shows us that while the basic logic works, the software needs extra code to handle invalid typing. Only after passing these rigorous tests—confirming the program is robust and accurate—is the software considered complete and ready for documentation.



Figure 3.15
Execution and Testing

3.3.7 Documentation



Figure 3.15
Documentation

Developing a program does not end with testing. Even a perfect program is useless if no one understands how to use it or how it was built. Therefore, the final and ongoing phase of software development is documentation, which ensures the software remains accessible and maintainable for both users and future developers. *Documentation* is the process of creating detailed written records that explain how a program works, how to use it, and how it was built. It is essential for three main reasons: it acts as a memory aid for the original programmer, it enables teamwork by allowing others to understand the code without constant explanations, and it ensures easy maintenance,

allowing future developers to upgrade or fix the system years later. Without these records, even the best software becomes difficult to manage and improve.

Documentation is categorized into two main types: Internal documentation and External documentation. In, internal documentation, which consists of comments written directly in the source code to explain logic to other programmers. External documentation includes resources like user and technical manuals, which provide installation and usage guides for non-programmers. Proper documentation provides three key benefits: it simplifies maintenance by making bugs easier to fix, improves usability so users can operate the software effectively, and ensures continuity by allowing new developers to manage the project if the original team leaves.

To finish our student result management program, we must document it properly:

- **Internal:** Inside our Python code, we should add comments to explain that the variable Total sums up six specific subjects and the variable max_marks is set to 600 as each subject carries maximum of 100 marks. This helps any future programmer who wants to change the maximum marks to 1000.
- **External:** We should create a simple one-page *User Guide* for the school teachers. This guide would explain:
 - How to open the program.
 - That the program will accept only numbers (not words) for marks.
 - How to read the final mark list displayed on the screen.

With this final step, our software is not just a working code file; it is a complete, professional product ready for use in a school.

3.4 Programming Paradigms

3.4.1 Approaches to Programming

Imagine you want to travel from your home to school. You have several options: by walk, ride a bicycle, take a school bus, or perhaps your parents drop you off in a car. All these methods achieve the same goal-getting you to school-but the approach is different. The “bus approach” requires waiting at a stop, while the “walking approach” requires physical effort and time management.

Similarly, in computer application, there is no single “correct” way to write a program. Over the years, programmers have developed different styles or philosophies for solving problems. These different approaches are called programming paradigms.

Programming paradigm is a fundamental style of computer programming that guides how the programmer thinks about and structures the code.

Programming paradigms are essential because they provide a blueprint for solving complex problems. By defining how code is organized and how data flows, a paradigm helps developers choose the most efficient strategy-whether that is a step-by-step procedural approach or a data-focused object-oriented one. Paradigms turn disorganized code into a structured, high-quality system that is easier to fix and expand.

3.4.2 Common Programming Paradigms

A programming paradigm is a style or method of programming that provides a way to structure and organize programs. The commonly used programming paradigms are procedure oriented programming (POP) and object oriented programming (OOP).

Procedure oriented programming is a method that treats a program like a step-by-step recipe, where a large task is broken down into a sequence of smaller instructions or procedures. Much like the process of making tea-where you must boil water, add leaves, and then pour in a specific order-the computer executes these functions one after another from top to bottom. While this approach makes code easier to organize and reuse through simple languages like C or Pascal, it can be risky for complex projects; since data is often shared across the entire program, one step might accidentally interfere with another, leading to errors.

Object oriented programming is a design approach that models software after real-world entities by organizing code into Objects. Each object acts as a self-contained unit that bundles together Data (attributes like a car’s color or fuel level) and Methods (actions like starting the engine or accelerating). By using languages such as Python, Java or C++ developers can securely hide data within these objects to prevent accidental errors, making it highly effective for managing complex systems like video games or operating systems. While OOP can be more difficult for beginners to learn and often requires more initial code to set up, it results in a modular and readable structure that mirrors the real world.

◆———— Summary ———◆

The chapter programming concepts explains how computer software is developed step by step. It shows how a person moves from just using a computer to creating programs. Computers do not have common sense, so they need clear and logical instructions to solve problems. This program development involves several key phases, starting with problem identification, where a programmer defines the necessary inputs, desired outputs, and the processes required to transform data. Before any coding begins, a logical plan is designed using algorithms written step-by-step procedures and flowcharts, which are visual diagrams that represent the logic using standard geometric symbols. These tools help manage the flow of control, whether it is sequential, involves branching (decisions), or requires iteration (looping). Once a plan is finalized, it is translated into coding using high-level languages like Python, which are then converted into machine-readable object code by language Translators such as compilers or interpreters. Throughout this process, debugging is essential to identify and correct syntax, logical, and run-time errors. The final stages involve execution and testing, where the program is run with various test cases including normal, boundary, and erroneous data to ensure it is reliable and accurate before reaching the end-user. Finally, Documentation serves as a vital record of the program's design and functionality for future maintenance and updates.



1. Which is the first phase in the programming life cycle?
a) Coding b) Debugging c) Problem Identification d) Documentation
2. Which type of translator converts the entire source code into object code at once?
a) Interpreter b) Compiler c) Assembler d) Debugger
3. “Division by Zero” Identify the type of error?
a) Syntax Error b) Logical Error c) Run-time Error d) Compilation Error
4. The detailed process of finding and fixing errors in a program is called:
a) Translation b) Execution c) Debugging d) Documentation
5. Which type of documentation is written directly inside the source code for programmers?
a) User Manual b) Technical Guide
c) Internal Documentation (Comments) d) External Documentation
6. Differentiate between a Compiler and an Interpreter.
7. List the three types of programming errors. Briefly explain any one of them.
8. Explain the difference between execution and testing of a program.
9. Why is Documentation important for the maintenance of a software?
10. Explain the different phases of programming in detail.
11. Compare procedural and object oriented programming paradigms.



4

Elements of Python



“Programming isn’t about what you know; it’s about what you can figure out.”

Chris Pine

4.1 Introduction

Programming languages allow humans to communicate with computer systems by writing instructions that the computer can interpret and execute. This chapter gives a brief overview of the Python programming language. Python is an interpreted, dynamically-typed, and easy-to learn programming language used worldwide by beginners and experts. It was created by Guido van Rossum in 1991. Python supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is used for Web Development, Game Development, Data Science and Analysis, Artificial Intelligence, Internet of Things (IOT), Cybersecurity, Machine Learning, Scientific Computing etc. It is an interpreted language, i.e. Python code will be executed line by line. It can be executed on various operating systems, making Python portable and platform-independent.

4.1.1 How To Install?

Python is a free and open-source programming language that can be easily installed in different operating systems such as Windows, macOS and Linux. The latest version of Python can be downloaded from the official web site <https://www.python.org>. Python package includes Python programming language interpreter, and a basic code editor.

We can write and execute Python programs using Python interpreter, Integrated Development Environment or online Python

interpreter. The Python interpreter is also called Python Shell. In the Python shell, the prompt `>>>` indicates that Python is ready to accept commands. The symbol `>>>` is called Python prompt. A sample screen of Python interpreter is shown in the Figure 4.1

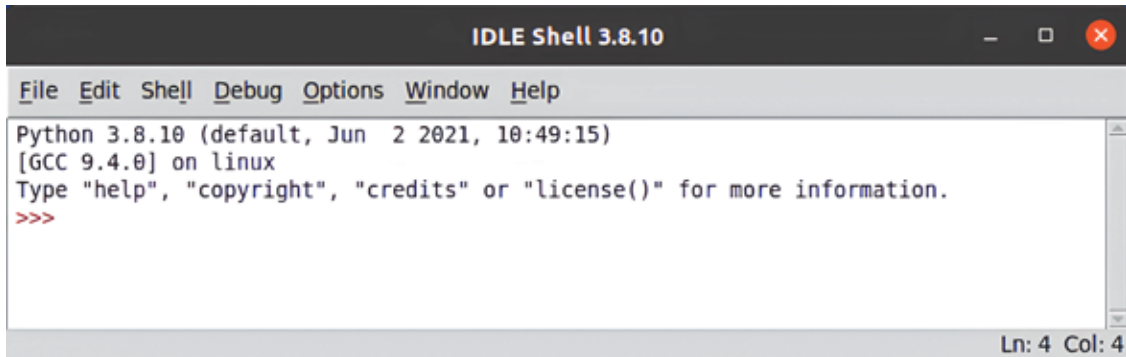


Figure 4.1 - Python Interpreter or Python Shell

4.1.2 Python Program Execution

There are three ways to write and run Python programs.

(i) Interactive Mode

The Python shell is an interactive environment that allows us to write Python code and execute it immediately. REPL stands for Read-Eval-Print-Loop, Python's interactive shell where you write, execute, and test code line-by-line with immediate feedback. Whenever we see this symbol `>>>`, it means that Python is ready to receive instructions. When we type a Python statement and press enter, immediately the interpreter executes the instruction and displays the output as shown in Figure 4.2. In this method, we cannot save the statements for future use. Interactive mode is used for testing single-line statements.

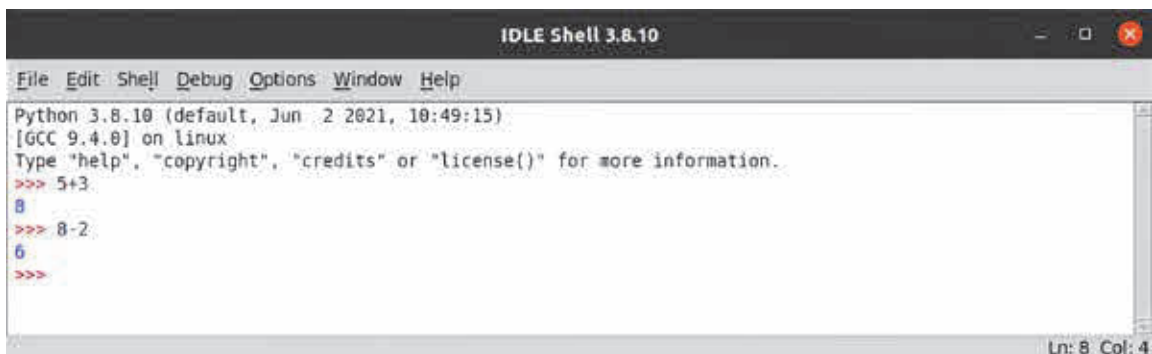


Figure 4.2 - Python Interpreter in interactive mode

(ii) Scripting Mode

In scripting mode, we can write a Python program with multiple Python statements and then save it as a file with `.py` extension. Programs can be created using any one of the text editors. Python has an integrated environment containing built-in editor, debugger, interpreter, program executor etc. called Integrated Development and Learning Environment (IDLE). Use the menu option **File** → **NewFile** to open a new editor window, then write the program and save it as `sample.py` as shown in the Figure 4.3. The python program can execute from the IDLE menu **Run** → **Run Module(F5)** as shown in Figure 4.4. The output displays on a shell as shown in Figure 4.5. Some popular Python IDEs/Editors are Geany, PyCharm, Jupyter Notebook, Thonny, VS Code, Spyder, etc.



Figure 4.3 - Python Program in Script Mode.

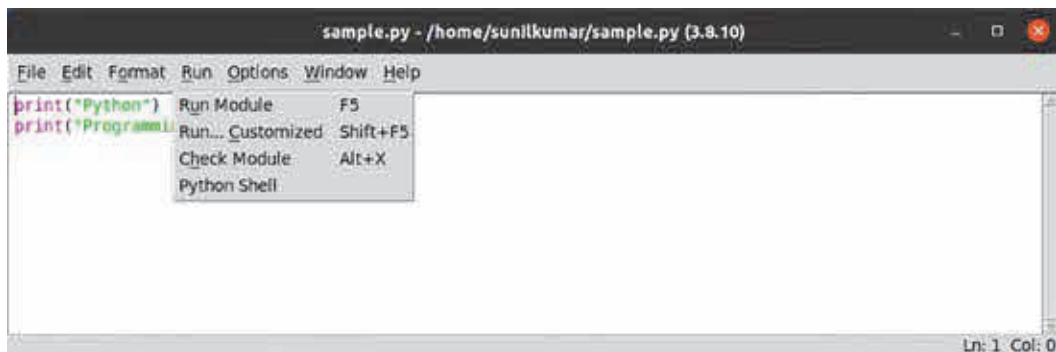


Figure 4.4 - Execution of Python Program in Script Mode

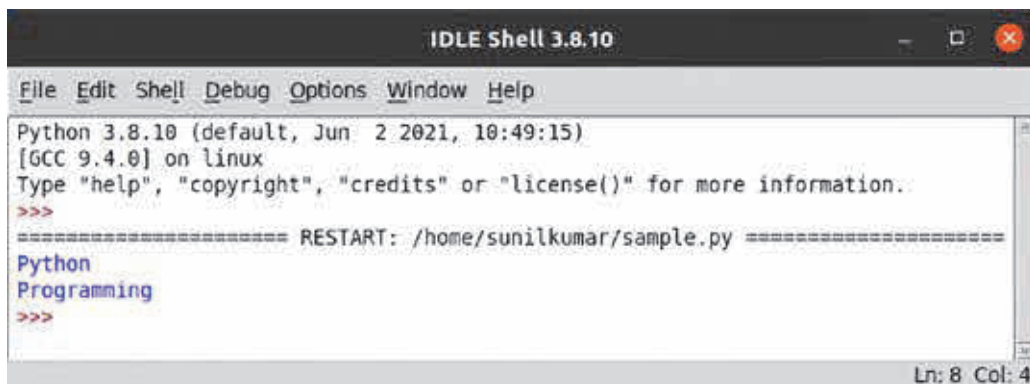


Figure 4.5 - Output of Python Program in Script Mode

(iii) Online Platform Mode

In an online platform, Python code is written and executed in a web-based interface (no local installation needed). It allows users to write Python code and see the output directly in a web browser. Google Colab, Replit, Kaggle Notebooks, Deepnote, Binder, etc. are popular online platforms for Python programming.

Advantages of Online Python Programming

- No installation required
- Program is written in browser
- Output is displayed instantly
- Works on any operating system

Limitations of Online Python Programming

- Internet connection required
- Limited file handling
- Not suitable for large projects

4.2 Python Comments

Python comments are used to explain notes within the program code, making it easier for programmers to read and understand. They are used for internal documentation. Python interpreter completely ignores the comments during the execution of the program.

There are two types of comments in Python:

1. Single-line Comment

A single-line comment begins with # (hash symbol) character. Anything written after # is considered as a comment.

Program 4.1 Python program to illustrate single-line comment.

```
# Simple program
print("Welcome to Python")
```

Output

Welcome to Python

First line of program starting with # is a comment. Python interpreter ignores that line while executing the program.

2. Multi-line Comment

Python does not have a separate multi-line comment symbol. However, a string enclosed in a triple single quote (""") or triple double quote ("""") can serve as a multi-line comments if it is not assigned to a variable or used in code.

Program 4.2 Python program to illustrate multi- line comment.

```
"""Program 4.2
Multi-line Comment"""
print("Welcome to Python")
```

Output

Welcome to Python

The first two lines within the triple single quote are considered as comments and the Python interpreter ignores the line while executing the program.

4.3 Python Keywords

A keyword in Python is reserved word that has a special meaning in the language and is used to define a syntax and structure of a Python program. All Python keywords except *None*, *True* and *False* are written in lowercase. Some Python keywords are listed in Table 4.1.

Table 4.1 Python Keywords

False	await	else	import	pass
None	break	elif	in	raise
True	class	except	is	return
and	continue	finally	lambda	try
as	def	for	nonlocal	while
assert	del	from	not	with
async	if	global	or	yield

4.4 Identifiers

In the Python language, an Identifier is a name used to define a variable, function, class, or module. It helps programmers to identify and differentiate them from other entities. Identifiers can be any

length; however, they are usually kept short and meaningful. The rules for naming Python identifiers are as follows:

- Only letters (A-Z, a-z), digits (0-9) or underscore are allowed.
- Must not start with a digit.
- Python keywords cannot be used.
- Whitespaces and special character are not allowed.
- Identifiers are case-sensitive, i.e. Grade, grade and GRADE are all different identifiers.

Some valid and invalid identifiers are listed below.

Valid identifiers:

- totalamount
- score1
- value
- basic_pay

Invalid identifiers:

- user@name - contain special character @
- return - Python keyword
- total value - contain whitespace
- 2A - start with digit

4.5 Variables

Variables are symbolic names used to store data that may change value during program execution. Variables can store integers, floats, strings, lists, dictionaries, etc.

Example 4.1

```
rollno=12          # rollno is an integer
name="Arun"        # name is a string
price=85.5         # price is floating point
```

A Python variable can be defined by assigning a value using the assignment operator (=). Python variables are dynamically typed, so there is no need to explicitly declare them before use. The data type of a variable changes automatically when it is reassigned to a different type of value, as illustrated in the example below:

```
n=10           # n is an integer
n="Hello"      # now n is string
n=True         # now n is Boolean
```

4.6 Data Types

Data type is a classification or grouping of data values that specifies what kind of value a variable can hold and which operation can be performed on it. Python includes a range of built-in data types that are classified into several categories. Python data types are listed in Figure 4.6.

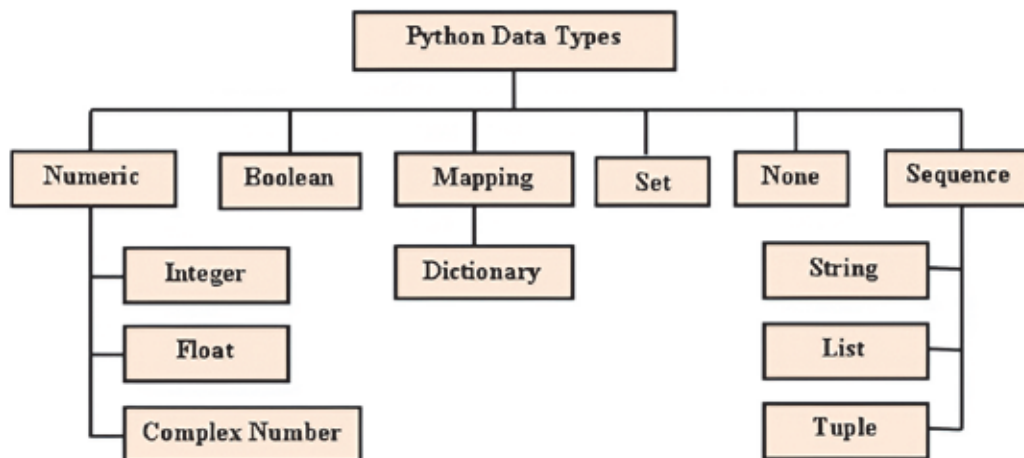


Figure 4.6 - Classification of Python Data Types

4.6.1 Numeric

A Python numeric value can be an integer, floating point and complex numbers.

- **Integer:** It contains positive or negative whole numbers without decimal part. Python numbers belong to *int* class. e.g.: 27, -36, 412, 0
- **Float:** Floating-point values are numbers that contain a fractional part. They are used to represent real numbers, such as 3.5, 0.0, or -12.75. It belongs to the *float* class.
- **Complex number:** Complex numbers are of the form $a + bj$, where a is the real part, b is the imaginary part. Python complex numbers belong to *complex* class. Use the `complex(x, y)` functions to create a complex number with real part x and imaginary party.

The following examples illustrates the data type of numeric using the built-in function `type()`.

Example 4.2

```
>>>mark=60
>>>type(mark)
<class 'int'>
>>>price=35.5
>>>type(price)
<class 'float'>
>>>x=complex(2 , 5)
>>>print(x)
(2 + 5j)
>>>type(x)
<class 'complex'>
```

4.6.2 Boolean

A Boolean data type has one of the two logical values - True or False. It belongs to the class *bool*.

Example 4.3

```
>>>a=True
>>>b=False
>>>type(a)
<class 'bool'>
>>>print(b)
False
```

4.6.3 Sequence

Sequence is an ordered group of value arranged one after another and each value has a fixed position (index) and accessed using that index. Python sequences are three types. String, List and Tuple.

- **String:** Python string is a sequence of Unicode characters. It can be created using single quotes or double quotes. There is no character data type in Python. Single character is a string of length one. Individual characters of a string can be accessed by index, and a negative index represents a position from the end of the string. Index starts from 0. Python string belongs to *str* class. e.g. "Delhi", '45.62'. Python strings are immutable.

In Python, objects are classified into mutable and immutable types based on whether their values can be changed after creation.

Mutable objects: It is an object whose value can be changed after is created. Eg, List, Set, Dictionary

Immutable objects: An immutable objects is and object whose value cannot be modified after creation. Eg. Numeric, String, Boolean, Tuple.

Example 4.4

```
>>>s1="Programming"
>>>type(s1)
<class 'str'>
>>>print(s1[0])
P
>>>print(s1[4])
r
>>>print(s1[-1])      #last character in the string
g
```

- **Lists:** A list is an ordered sequence of items, separated by commas and enclosed in square brackets []. The Python Lists belong to the *list* class. We can access the list items using index numbers. Lists are mutable object.

Example 4.5

```
>>>x=[10, "Delhi", 65.5, 'A2', 40]
>>>type(x)
<class 'list'>
>>>print(x)
[10, 'Delhi', 65.5, 'A2', 40]
>>>print(x[0])
10
>>>print(x[3])
A2
>>>print(x[-1])
40
>>>print(x[-2])
A2      # x[3] and x[-2] are the same position.
```

- **Tuple:** Tuple is an ordered collection of Python objects. The only difference between a tuple and a list is that tuples are immutable. Tuples can contain any number of elements and of any data types separated by comma and enclosed within parenthesis (). It belongs to the class *tuple*.

Example 4.6

```
>>>t1=(40,"Computer", 65.5, 'Phone')
>>> type(t1)
<class 'tuple'>
>>>print(t1)
(40,'Computer', 65.5, 'Phone')
>>>print(t1[0])
40
>>>print(t1[1])
Computer
>>>print(t1[-1])
Phone
```

4.6.4 None

None represents the absence of a value. It is used to indicate nothing, null, or empty result. It belongs to the class *NoneType*.

Example 4.7

```
>>>x=None
>>>type(x)
<class 'NoneType'>
>>>print(x)
None
```

4.6.5 Set

Set is a collection of unordered, unique elements. It is mainly used to store multiple items in a single variable without duplicates and to perform mathematical set operations like union, intersection, and difference. Set is mutable object. Set elements are separated with comma and enclosed within braces {}. Sets are unordered collections, so the elements have no defined order, they cannot be accessed by index. It belongs to the class *set*. *set()* creates an empty set.

Example 4.8

```
>>>x={10,20,30}
>>>type(x)
<class 'set'>
>>>print(x)
{10, 20, 30}
```

Example 4.9

```
>>>x={10,20,30,20}
>>>print(x)
{10, 20, 30}                # Eliminate duplicate elements
```

4.6.6 Mapping

Mapping is an unordered data type in Python. The most common mapping data type in Python is dictionary.

Dictionary

A Python dictionary stores data as key-value pairs and allows efficient lookup or modification of values based on their keys. The items are enclosed within braces {}. Values in a dictionary can be any data type and can be accessed using their corresponding keys. We can add and remove key-value pairs in a dictionary. Values in the dictionary can be accessed using the key enclosed square bracket []. The python dictionary data type belongs to class *dict*. The dictionary object is mutable. Note that {} creates an empty dictionary.

Example 4.10

```
>>>mark={'Diya': 70, 'Biju': 65, 'Thomas' : 58}
>>>type(mark)
<class 'dict'>
>>>print(mark['Biju'])        # Key Biju, Value 65
65
>>>mark['Fathima']=75        # Adding a new key value pair
>>>print(mark)
{'Diya': 70, 'Biju': 65, 'Thomas': 58, 'Fathima': 75}
>>>del mark['Diya']          # Removing a key value pair
>>>print(mark)
{'Biju': 65, 'Thomas': 58, 'Fathima': 75}
```

4.7 Operators

Operators are special symbols that carry out operations on values or variables. The various types of Python operators are

1. Arithmetic Operators
2. Assignment Operators
3. Relational Operators
4. Logical Operators
5. Identity Operators
6. Membership Operators

4.7.1 Arithmetic Operators

Arithmetic operators are used to perform basic mathematical operations like addition, subtraction, multiplication, division, etc. Table 4.2 illustrates the arithmetic operators in Python.

Table 4.2 Arithmetic Operators in Python

Operator	Name of Operator	Description	Example	Python Code
+	Addition	Add two numbers	$5 + 2 = 7$	<pre>>>>x=5 >>>y=2 >>>x+y 7</pre>
	Concatenation	Add two strings	'New' + 'Delhi' = 'NewDelhi'	<pre>>>>x='New' >>>y='Delhi' >>>x+y 'NewDelhi'</pre>
-	Subtraction	Subtracts two numbers	$5 - 2 = 3$	<pre>>>>x=5 >>>y=2 >>>x-y 3</pre>
*	Multiplication	Multiplies two numbers	$5 * 2 = 10$	<pre>>>>x=5 >>>y=2 >>>x*y 10</pre>
		String Integer multiplication (you can multiply a string and an integer only, cannot multiply two strings or a string by a float)	'Hi' * 3 = 'HiHiHi'	<pre>>>>x='Hi' >>>y=3 >>>x*y 'HiHiHi'</pre>

/	Division	Divides two numbers	$5 / 2 = 2.5$	<pre>>>>x=5 >>>y=2 >>>x/y 2.5</pre>
//	Floor Division or Integer Division	Returns floored integer quotient (floored result of division)	$5/2 = 2$ $-5/2=-3$ (Negative flooring)	<pre>>>>x=5 >>>y=2 >>>x//y 2 >>>x=-5 >>>y=2 >>>x//y -3</pre>
%	Modulus	Returns remainder after division	$5 \% 2 = 1$	<pre>>>>x=5 >>>y=2 >>>x%y 1</pre>
**	Exponentiation	Raise the base to power of the exponent	$5**2 = 5^2=25$	<pre>>>>x=5 >>>y=2 >>>x**y 25</pre>

4.7.2 Assignment Operators

Assignment operator used to change or assign the value on its right side to the operand. Table 4.3 illustrates the assignment operators in Python.

Table 4.3 Assignment Operators in Python

Operator	Name of Operator	Example	Python Code
=	Assignment	$a=10$	<pre>>>>a=10 >>>print(a) 10</pre>
+=	Addition Assignment	$a+=5$ same as $a=a+5$	<pre>>>>a=10 >>>a+=5 >>>print(a) 15</pre>
-=	Subtraction Assignment	$a-=5$ same as $a=a-5$	<pre>>>>a=10 >>>a-=5 >>>print(a) 5</pre>
=	Multiplication Assignment	$a=5$ same as $a=a*5$	<pre>>>>a=10 >>>a*=5 >>>print(a) 50</pre>
/=	Division Assignment	$a/=5$ same as $a=a/5$	<pre>>>>a=10 >>>a/=5 >>>print(a) 2.0</pre>

//=	Floor Division Assignment	a//5 same as a=a//5	>>>a=10 >>>a//5 >>>print(a) 2
%=	Modulus Assignment	a%=5 same as a=a%5	>>>a=10 >>>a%=5 >>>print(a) 0
=	Exponent Assignment	a=5 same as a=a**5 =a ⁵	>>>a=10 >>>a**=5 >>>print(a) 100000

4.7.3 Relational Operators

Relational operators are used to compare two values or variables and return Boolean values True or False. It is also known as comparison operators. Table 4.4 illustrates the relational operators in Python.

Table 4.4 Relational Operators in Python

Operator	Name of Operator	Example	Python Code
==	Equal	8 == 5 is False	>>>a=8 >>>b=5 >>>print(a==b) False
!=	Not equal	8 != 5 is True	>>>a=8 >>>b=5 >>>print(a!=b) True
<	Less than	8 < 5 is False	>>>a=8 >>>b=5 >>>print(a<b) False
>	Greater than	8 > 5 is True	>>>a=8 >>>b=5 >>>print(a>b) True
<=	Less than or equal	8 <= 5 is False	>>>a=8 >>>b=5 >>>print(a<=b) False
>=	Greater than or equal	8 >= 5 is True	>>>a=8 >>>b=5 >>>print(a>=b) True

4.7.4 Logical Operators

Logical operators are used to combine one or more conditions and return True or False result. Table 4.5 illustrates the logical operators in Python.

Table 4.5 Logical Operators in Python

Operator	Name of Operator	Description	Example assumptions: a=20, b=10	Python Code
and	Logical AND	True only if both relational expressions are True	a > b and b < a is True a < b and b < a is False	>>>a=20 >>>b=10 >>>print(a>b and b<a) True >>>print(a<b and b<a) False
or	Logical OR	True if at least one relational expression is True	a > b or b > a is True a < b or b > a is False	>>>a=20 >>>b=10 >>>print(a>b or b>a) True >>>print(a<b or b>a) False
not	Logical NOT	True if the relational expression is False, and False if it is True.	not (a < b) is True not (b < a) is False	>>>a=20 >>>b=10 >>> print(not (a<b)) True >>> print(not (b<a)) False

4.7.5 Identity Operators

In Python, Identity operators are used to check whether two variables or objects refer to the same object in memory. Python includes two identity operators: *is* and *is not*, see in Table 4.6.

Table 4.6 Identity Operators in Python

Operator	Description	Example	Python Code
is	Returns True, if two variables are the same object	x is y	<pre>>>>x=[1,2,3] >>>y=[1,2,3] >>>print(x is y) False #Because x and y are different objects. >>>z=x >>>print(x is z) True #Because x and z refer to the same object in memory</pre>
is not	Returns True if two variables are not the same object (False if they are the same)	x is not y	<pre>>>>x=[1,2,3] >>>y=[1,2,3] >>>print(x is not y) True #Because x and y are different objects. >>>z=x >>>print(x is not z) False #Because x and z refer to the same object in memory</pre>

4.7.6 Membership Operators

Python membership operators are used to check whether a value or variable is found in a sequence. Python includes two membership operators: *in* and *not in*; they are described in Table 4.7.

Table 4.7 Membership Operators in Python

Operator	Description	Example	Python Code
in	Returns True if a value or variable is found in a sequence	10 in a	<pre>>>>a=[10,20,30] >>>print(10 in a) True >>>print(50 in a) False</pre>
not in	Returns True if a value or variable is not found in a sequence	P not in text	<pre>>>>text='Python' >>>print('P' not in text) False >>>print('k' not in text) True</pre>

4.8 Precedence of Operators

An expression is a combination of values, variables and operators that can be evaluated to produce another value. In any programming language, an expression is evaluated as per the precedence of its operators. If there is more than one operator in an expression, their precedence decides which operation will be performed first. This is listed in the Table 4.8.

Precedence level 1 is highest priority

Table 4.8 Precedence of Operators in Python

Precedence Level	Operator(s)	Description
1	()	Parentheses are used to group expressions and override the usual precedence order.
2	**	Exponentiation
3	+, -	Unary plus, unary minus
4	*, /, //, %	Multiplication, division, floor division, and modulus
5	+, -	Addition, Subtraction
6	==, !=, >, <, >=, <=, is, is not, in, not in	Relational, identity, and membership operator
7	not	Logical NOT
8	and	Logical AND
9	or	Logical OR

- $40 + 50 * 3$ # precedence of * is more than +
 $40 + 150$
 190
- $(30 - 20) * 5$ # parenthesis () change the normal precedence.
 So subtraction is evaluated
 $10 * 5$ #before multiplication
 50
- $60 / 5 - (6 + 2)$ # parenthesis has higher precedence
 $60 / 5 - 8$ # precedence of / is more than -
 $12.0 - 8$
 4.0

4.9 Input and Output Statements

In Python, input and output statements are used to take input from the keyboard and display output on the screen. The built-in functions `input()` and `print()` are used for this purpose.

4.9.1 Input Statement.

The `input()` function is used to receive data from the keyboard. The function always returns a string value.

Syntax

```
variable=input("Prompt message")
```

Example 4.11

```
name=input("Enter your name : ")
print("Entered Name : " + name)
```

Output

Enter your name : Akhil

Entered Name : Akhil

4.9.2 Output Statement.

The `print()` function is used to display data on the output screen. It can print strings, numbers, or values of variables.

Syntax

```
print("Display message")
```

Example 4.12

Write a Python program to add two integers.

```
num1=int(input("Enter first number : "))
num2=int(input("Enter second number : "))
total=num1 + num2
print("Total :",total)
```

Output

Enter first number : 10

Enter second number : 20

Total : 30

◆———— Summary ———◆

This chapter introduced the fundamental concepts of the Python programming language, focusing on its basic structure, features, and core elements required for beginners. It explained how Python works, how to write simple programs, highlighting the building blocks like identifiers, variables, data types, expression, operators, and input/output functions with clear rules and examples.

- Python: A high-level, interpreted, and easy-to-learn programming language.
- Interpreter: A program that executes Python code line by line.
- REPL: Read-Eval-Print Loop used in interactive mode.
- IDLE: Integrated Development and Learning Environment for Python.
- Comment: Text in a program ignored by the interpreter, used for explanation.
- Keyword: A reserved word in Python with a predefined meaning.
- Identifier: Name used to identify variables, functions, classes, or modules.
- Variable: A symbolic name that stores data values.
- Dynamic Typing: Feature of Python where variable types are decided at runtime.
- Data Type: Classification of data that determines the kind of value a variable can hold.
- Boolean: A data type for True/False values.
- List: An ordered, mutable collection of elements.
- Tuple: An ordered, immutable collection of elements.
- Set: An unordered collection of unique elements.
- Dictionary: A collection of key-value pairs.
- Operator: A symbol that performs an operation on variables or values.
- Expression: Combination of values, variables, and operators.
- Operator Precedence: Order in which operators are evaluated in an expression.
- input(): Function used to accept input from the user.
- print(): Function used to display output on the screen.



1. The Python programming language is developed by _____.
2. The symbol >>> represents
 - a) Output symbol
 - b) Python prompt
 - c) Comment symbol
 - d) Error indicator
3. Single-line comments in Python begin with _____.
4. Python programs are saved with _____ extension.
5. The operator // is used for
 - a) Division
 - b) Modulus
 - c) Floor division
 - d) Exponentiation
6. The symbol used for single-line comments is _____.
7. Which data type stores key-value pairs?
8. Which collection is unordered?
 - a) List
 - b) Tuple
 - c) Set
 - d) String
9. The function used to display output is _____.
10. Which operator is used to check object identity?

11. Define Python programming language. Mention any two features of Python.
12. What is interactive mode in Python? Write one advantage of it.
13. What is scripting mode? How is it different from interactive mode?
14. What are comments in Python? Why are they used?
15. What are Python keywords? Give any four examples.
16. Define identifiers. Write rules for naming identifiers.
17. What is a variable in Python? Explain dynamic typing.
18. Define data type. Name any four built-in Python data types.
19. What is a set? Mention any two characteristics of a set.
20. What is a dictionary in Python?
21. Explain the classification of Python data types with suitable examples.
22. Describe logical operators in detail with examples.
23. Explain Python identifiers. List rules for naming identifiers.
24. Explain arithmetic and assignment operators with examples.
25. Explain sets and dictionary in Python with examples.
26. Write a Python program to find the sum of two numbers.
27. Write a Python program to calculate the average of three numbers.
28. Write a Python program to demonstrate arithmetic operators.

Identify errors in the following code:

1. `name=input(Enter your name)`
`print("Hello" name)`
2. `x=10`
`y="20"`
`total= x + y`
3. `1stNumber=50`
`second number=60`
`print(1stNumber + second number)`
4. `list=(10,20,30)`
`list[1]=50`
`print(list)`
5. `a=5`
`b=2`
`result=a** b`
`print("Result is "+result)`

Write the output of the following code.

1. `a=5`
`b=2`
`print(a % b)`
2. `x="Hello"`
`print(x *3)`
3. `print(10>5and5<3)`
4. `fruits=["apple","banana","cherry"]`
`print("banana"in fruits)`
5. `a=10`
`b=a`
`print(a is b)`



5

Python Control Structures and Data Handling



“Programs must be written for people to read, and only incidentally for machines to execute.”

Harold Abelson

5.1 Introduction

We have already studied Python elements. Python sequences, including lists, tuples and sets, have also been introduced. For the efficient utilization of Python elements and Sequences, control structures are essential. Control structures determine the flow of execution of instructions. They allow decision-making, looping and branching based on conditions. Python also provides powerful data structures to store and manipulate collections of data efficiently. This chapter introduces control statements and explores advanced data types like lists, tuples, sets and dictionaries.

5.2 Control Structures

Control structures are methods that dictate the flow of execution in a program. They answer the question: “What should happen next in a program?”. Normally, a program executes instructions one after another in the order they appear. By using conditional statements and looping statements, the order of execution of instructions can be controlled and managed according to the requirement. Statements or Instructions in a program which are executed together are called code blocks. In Python, indentation is the only way to define code blocks such as conditional statements and Looping Statements. Indentation is the use of whitespace (spaces or tabs) at the beginning of a line to define the structure and grouping of code blocks. It is mandatory and directly affects how the code executes.

5.2.1 Conditional Statements

Conditional statements allow the program to make decisions based on certain conditions. They are the backbone of logical decision-making and flow control in any programming language. They determine how and when different parts of a program are executed. Major applications of control structures in programming are listed below.

- Decision Making
- Flow Control
- Iterations
- Improved Efficiency and Flexibility
- Real-World Applications

In short, conditional statements give a program its intelligence and adaptability, making it capable of solving real-world problems efficiently. Mainly used conditional statements in Python are,

- a. if-elif-else statement
- b. match-case Statement

a. *if-elif-else* statement

The *if-elif-else* control structure is used to make decisions in a Python program based on conditions. It allows the program to evaluate multiple expressions and execute a block of code depending on which condition is true.

- The *if* statement checks the first condition.
- If the *if* condition is false, the *elif* (else if) statement checks another condition.
- If neither of the *if* or *elif* conditions is true, the *else* block is executed.
- Multiple *elif* branches are allowed.
- Only one *else* branch is allowed, and it must come last.
- Usage of *elif* or *else* is NOT compulsory with every *if-elif-else* control structure.

This structure helps in writing clean and readable decision-making code.

Syntax

if condition:

code block/statement

elif condition:

code block/statement

else:

code block/statement

Example 5.1

```
marks = int(input("Enter marks: "))
```

```
if marks >= 90:
```

```
    print("Grade A+")
```

```
elif marks >= 80:
```

```
    print("Grade A")
```

```
elif marks >= 70:
```

```
    print("Grade B+")
```

```
elif marks >= 60:
```

```
    print("Grade B")
```

```
elif marks >= 50:
```

```
    print("Grade C+")
```

```
elif marks >= 40:
```

```
    print("Grade C")
```

```
elif marks >= 30:
```

```
    print("Grade D+")
```

```
else:
```

```
    print("Grade D")
```

```

Example 5.1.py - /home/python/Ex...
File Edit Format Run Options Window Help

mark = int(input("Enter marks: "))
if mark >= 90:
    print("Grade A+")
elif mark >= 80:
    print("Grade A")
elif mark >= 70:
    print("Grade B+")
elif mark >= 60:
    print("Grade B")
elif mark >= 50:
    print("Grade C+")
elif mark >= 40:
    print("Grade C")
elif mark >= 30:
    print("Grade D+")
else:
    print("Grade D")

Ln: 1 Col: 0

IDLE Shell 3.8.10
File Edit Shell Debug Options Window Help

Python 3.8.10 (default, Jun 2 2021, 10:49:
15)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "lic
ense()" for more information.
>>>
===== RESTART: /home/python
/Example 5.1.py =====
Enter marks: 91
Grade A+
>>>

Ln: 8 Col: 4

```

Example 5.1

In this example, the program evaluates the marks and prints the appropriate grade. The *if-elif-else* structure ensures only one block is executed, even if multiple conditions are true.

b. Match-Case Statement in Python

Python introduced the *match-case* statement in version 3.10 as part of its structural pattern matching feature. It serves as an alternative to the traditional *switch* statement found in other programming languages like C++, Java or JavaScript.

The *match-case* structure allows a program to compare a value, termed pattern, against multiple cases and execute code based on the first matching case. It improves readability and simplifies complex conditional logic. In Python, a pattern means an arrangement of characters, numbers or symbols.

- *match* evaluates the variable.
- Each *case* checks for a specific pattern.
- The *_* (underscore) acts as a default case.

Syntax

match variable:

```
case pattern1:
    # code block
case pattern2:
    # code block
case _:
    # default block
```

Example 5.2

```
day = "Monday"
match day:
    case "Monday":
        print("Start of the week")
    case "Friday":
        print("Weekend is near")
    case _:
        print("Regular day")
```

This structure is especially powerful because it supports complex patterns, such as matching tuples, lists and even conditions within patterns.

Benefits of match-case statement:

- Cleaner and more readable than long *if-elif-else* chains.
- Supports advanced pattern matching.
- Enhances code maintainability.

5.2.2 Looping Statements

Looping is a fundamental concept in programming that allows a set of instructions to be executed repeatedly until a certain condition is met. It helps in performing repetitive tasks efficiently, reducing the need to write the same code multiple times.

A loop works by checking a condition before or after executing a block of code. If the condition is true, the statements inside the loop are executed. This process continues until the condition becomes false, at which point the loop stops automatically.

Loops are commonly used for tasks such as processing collection of data, performing calculations multiple times or controlling the flow of a program. They make programs shorter, faster and easier to maintain.

Looping is an essential feature of programming because it enables automation, iteration and logical control in program design. It is widely used in areas like data processing, simulations and algorithm development. In summary, looping helps programmers write efficient and flexible programs that can handle repetitive tasks automatically.

(i) *for* loop

In Python, one of the most used loops is *for* loop.

A *for* loop in Python is used to iterate over a sequence (like a list or string) and perform actions on each item in that sequence. It's like saying, "do something for each item in this group."

Syntax

for loop_variable in sequence:

do required operation with loop_variable.

- loop_variable is a variable that takes the value of each element in the sequence, one at a time.
- sequence is the collection that is being iterated over.

- The code block inside the loop executes once for each item in the sequence.

How the for Loop Works – Step by Step

Let's break down the process:

1. Python looks at the first item in the sequence.
2. It assigns that item to the loop variable.
3. It runs the code inside the loop.
4. It moves to the next item and repeats.
5. When all items are processed, the loop ends.

Looping Through Strings

Strings are sequences of characters, so you can loop through them too. In the following program, you can see the for loop is executing for all characters of the string.

Example 5.3

```
word = "hello"
for letter in word:
    print(letter)
```

Output

h
e
l
l
o

Example 5.3

Each character in the string is printed on a new line.

Using range() with for loops

Python provides a built-in function called *range()* that generates a sequence of numbers. It's often used with *for* loops when you want to repeat something a specific number of times.

Syntax of range()

range(start, stop, step)

- start: where the sequence begins (default is 0)
- stop: where the sequence ends (not inclusive)
- step: how much to increase each time (default is 1)

Using range()

Example 5.4

```
for i in range(1, 6):
```

```
    print(i)
```

Output:

```
1
2
3
4
5
```

```

Example 5.4.py - /ho...
File Edit Format Run Options
Window Help
for i in range(1, 6):
    print(i)

Ln: 1 Col: 0

IDLE Shell 3.8.10
Python 3.8.10 (default, Jun 2 2021, 10:49:15)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /home/python/Example 5.4.py =====
1
2
3
4
5
>>>

```

Example 5.4

This loop prints numbers from 1 to 5.

Nested for loops

You can place one for loop inside another. This concept is called Nesting. It's like having a clock where:

- The *hour hand* moves slowly (outer loop)
- The *minute hand* moves faster (inner loop) - it completes a full circle for each hour

Using nested loop

Example 5.5

```
for i in range(1, 4):
```

```
    for j in range(1, 4):
```

```
        print(i, "*", j, "=", i * j)
```

Output:

```
1 * 1 = 1
1 * 2 = 2
1 * 3 = 3
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
```

```

Example 5.5.py - /home/python...
File Edit Format Run Options Window Help
for i in range(1, 4):
    for j in range(1, 4):
        print(i, "*", j, "=", i * j)

Ln: 1 Col: 0

IDLE Shell 3.8.10
Python 3.8.10 (default, Jun 2 2021, 10:49:15)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /home/python/Example 5.5.py =====
1 * 1 = 1
1 * 2 = 2
1 * 3 = 3
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
>>>

```

Example 5.5

Controlling the loop: *break* and *continue*

break statement

Used inside loops to immediately stop the loop when a certain condition is met. This prevents unnecessary iterations and makes programs more efficient.

continue statement

Skips the current iteration and moves to the next one, allowing selective execution of loop code without stopping the loop entirely.

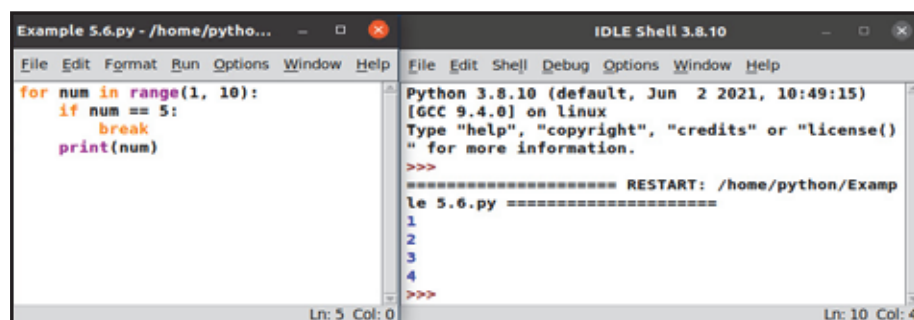
Using *break* statement

Example 5.6

```
for num in range(1, 10):
    if num == 5:
        break
    print(num)
```

Output:

1
2
3
4



Example 5.6

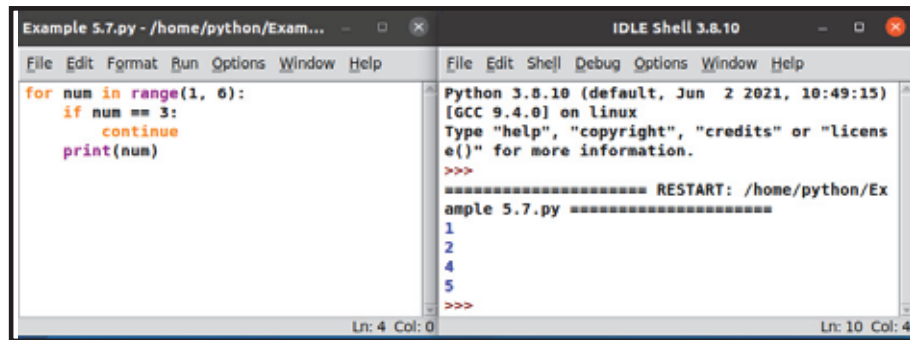
Using *continue* statement

Example 5.7

```
for num in range(1, 6):
    if num == 3:
        continue
    print(num)
```

Output:

1
2
4
5



```

Example 5.7.py - /home/python/Exam...
File Edit Format Run Options Window Help
for num in range(1, 6):
    if num == 3:
        continue
    print(num)
Ln: 4 Col: 0

IDLE Shell 3.8.10
File Edit Shell Debug Options Window Help
Python 3.8.10 (default, Jun 2 2021, 10:49:15)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /home/python/Example 5.7.py =====
1
2
4
5
>>>
Ln: 10 Col: 4

```

Example 5.7

Example 5.8

Counting vowels in a string

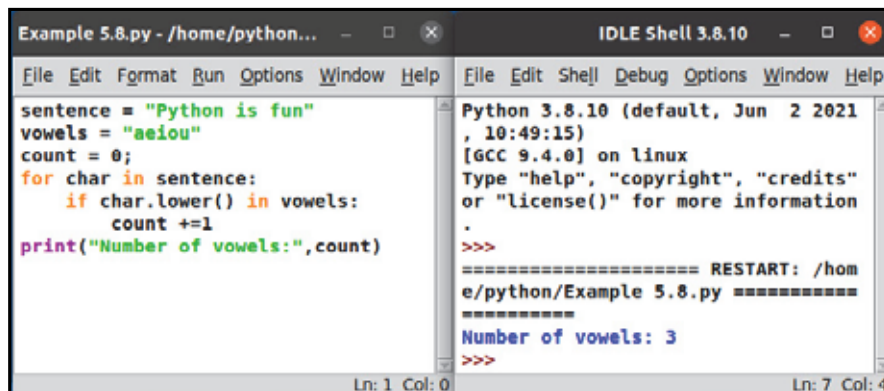
```

sentence = "Python is fun"
vowels = "aeiou"
count = 0
for char in sentence:
    if char.lower() in vowels:
        count += 1
print("Number of vowels:", count)

```

Output:

Number of vowels: 3



```

Example 5.8.py - /home/python...
File Edit Format Run Options Window Help
sentence = "Python is fun"
vowels = "aeiou"
count = 0;
for char in sentence:
    if char.lower() in vowels:
        count +=1
print("Number of vowels:",count)
Ln: 1 Col: 0

IDLE Shell 3.8.10
File Edit Shell Debug Options Window Help
Python 3.8.10 (default, Jun 2 2021, 10:49:15)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /home/python/Example 5.8.py =====
Number of vowels: 3
>>>
Ln: 7 Col: 4

```

Example 5.8

for loop with else

Python allows an else clause with for loop. It runs after the loop finishes normally.

Example 5.9

```

for i in range(3):
    print("Loop:", i)
else:
    print("Loop finished")

```

Output:

```

Loop: 0
Loop: 1
Loop: 2
Loop finished

```

```

Example 5.9.py - /home/python/...
File Edit Format Run Options Window Help
for i in range(3):
    print("Loop:", i)
else:
    print("Loop finished")
Ln: 3 Col: 5

IDLE Shell 3.8.10
Python 3.8.10 (default, Jun 2 2021, 10:49:15)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /home/python/Example 5.9.py =====
Loop: 0
Loop: 1
Loop: 2
Loop finished
>>>
Ln: 10 Col: 4

```

Example 5.9

Summary

- for loops are used to iterate over sequences.
- You can loop through lists, tuples, strings, dictionaries, sets and use range().
- Use break to stop early and continue to skip current iteration.
- You can use nested loops and an else clause, if necessary.

(ii) while loop

A while loop in Python is used to execute a block of code repeatedly based on specified condition. The condition is evaluated before each iteration and if it returns True, the loop body is executed. Once the condition becomes False, the loop terminates.

Syntax

while condition:

block of code

- **condition:** A Boolean expression that is evaluated before each iteration.
- **block of code:** One or more statements that are executed repeatedly as long as the condition is true.

Basic while loop

```

Example 5.10.py - /home/pytho...
File Edit Format Run Options Window Help
count = 1
while count <= 5:
    print("Count is :", count)
    count += 1
Ln: 1 Col: 0

IDLE Shell 3.8.10
Python 3.8.10 (default, Jun 2 2021, 10:49:15)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /home/pytho.../Example 5.10.py =====
Count is : 1
Count is : 2
Count is : 3
Count is : 4
Count is : 5
>>>
Ln: 11 Col: 4

```

Example 5.10

Example 5.10

```

count = 1
while count <= 5:
    print("Count is:", count)
    count += 1

```

Output:

```

Count is: 1
Count is: 2
Count is: 3
Count is: 4
Count is: 5

```

In this example, the loop runs five times, incrementing the count variable each time. When count becomes 6, the condition $\text{count} \leq 5$ evaluates to False and the loop terminates.

Infinite loops

An infinite loop occurs when the condition never becomes False. This can happen if the loop control variable is not updated correctly.

Example 5.11

```
i = 1
while i < 5:
    print("i =", i)    #Missing increment
                      #statement in the block
```

This loop will run forever because i is never incremented, so $i < 5$ always remains True.

Example 5.11

while loop with break statement

The *break* statement is used to exit this while loop prematurely, even if the condition is still True.

Example 5.12

```
i = 1
while i <= 10:
    print(i)
    if i == 5:
        break
    i += 1
```

Example 5.12

Output:

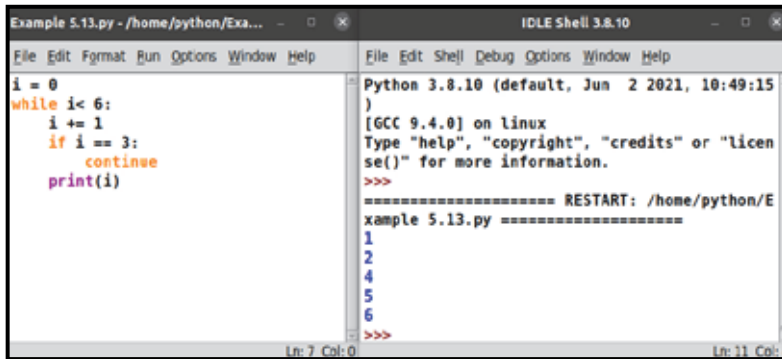
```
1
2
3
4
5
```

The loop stops when i equals 5 due to the *break* statement.

while loop with continue statement

The *continue* statement in the following while loop skips the current iteration and moves to the next one.

Example 5.13



```

i = 0
while i < 6:
    i += 1
    if i == 3:
        continue
    print(i)

```

```

Python 3.8.10 (default, Jun 2 2021, 10:49:15)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /home/python/E
xample 5.13.py =====
1
2
4
5
6
>>>

```

Example 5.13

```

i = 0
while i < 6:
    i += 1
    if i == 3:
        continue
    print(i)

```

Output:

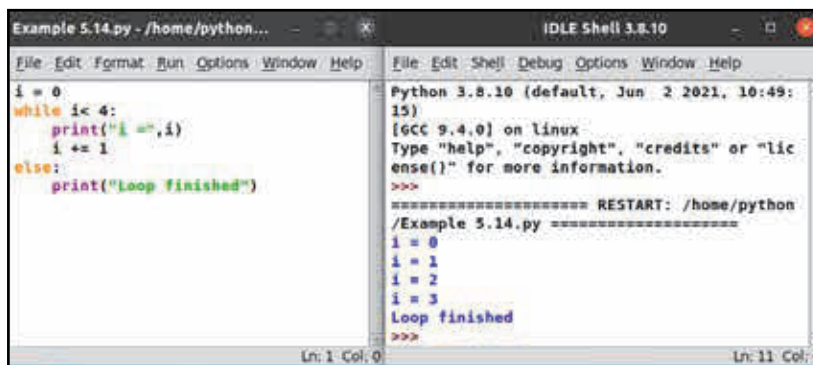
1
2
4
5
6

Here, when i is 3, the continue statement skips the print(i) line.

while loop with else clause

Python allows an else clause with a while loop. The else block is executed when the loop condition becomes False.

Example 5.14



```

i = 0
while i < 4:
    print("i =", i)
    i += 1
else:
    print("Loop finished")

```

```

Python 3.8.10 (default, Jun 2 2021, 10:49:15)
[6CC 9.4.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /home/python
/Example 5.14.py =====
i = 0
i = 1
i = 2
i = 3
Loop finished
>>>

```

Example 5.14

```

i = 0
while i < 4:
    print("i =", i)
    i += 1
else:
    print("Loop finished")

```

Output:

i = 0
i = 1
i = 2
i = 3
Loop finished

Practical applications of while loop

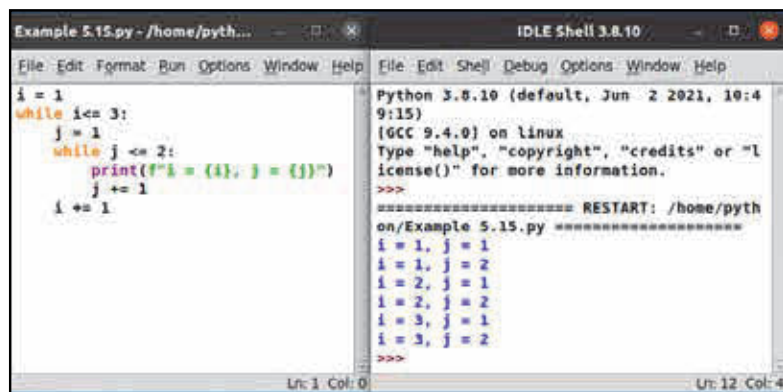
- User input validation: Repeatedly ask for input until valid data is entered.
- Menu-driven programs: Keep displaying options until the user chooses to exit.
- Game loops: Continuously run the game logic until the player quits.

Nested while loop

A *while* loop can be nested inside another *while* loop.

Example 5.15

```
i = 1
while i <= 3:
    j = 1
    while j <= 2:
        print(f"i = {i}, j = {j}")
        j += 1
    i += 1
```



```
Example 5.15.py - /home/pyth...
Python 3.8.10 (default, Jun 2 2021, 10:49:15)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /home/pyth
on/Example 5.15.py =====
i = 1, j = 1
i = 1, j = 2
i = 2, j = 1
i = 2, j = 2
i = 3, j = 1
i = 3, j = 2
>>>
```

Example 5.15

Output:

```
i = 1, j = 1
i = 1, j = 2
i = 2, j = 1
i = 2, j = 2
i = 3, j = 1
i = 3, j = 2
```

Tips for writing effective while loops

- Always ensure the loop condition will eventually become False
- Use *break* and *continue* judiciously to control loop flow.
- Avoid infinite loops unless intentionally required (e.g., servers, games).
- Use comments to clarify loop logic for better readability.

Exercise

1. Write a program to print numbers from 10 to 1 using a while loop.

2. Write a program that asks the user to enter a password until the correct one is entered.
3. Simulate a simple ATM withdrawal system using a while loop.

5.3 Lists

In Python, a *list* is a data structure used to store multiple items in a single variable. Lists are one of the four built-in data types in Python used to store collections of data. The other three are Tuple, Set and Dictionary. Among these, lists are more flexible and easier of use.

A *list* can contain elements of different data types such as integers, strings, floats or even other lists. Lists are ordered, mutable and allow duplicate values, making them ideal for a wide range of programming tasks. Python Lists are created using square brackets `[]`.

Characteristics of Lists

- **Ordered:** The items in a list have a defined order and this order will not change unless explicitly modified.
- **Mutable:** Lists can be changed after their creation. You can add, remove or modify elements.
- **Allow duplicates:** Same value can be repeated in a list.
- **Indexed:** Each item in a list has a unique index starting from 0.

Example 5.16

```
fruits = ["apple", "banana", "cherry"]
```

You can also create an empty list:

```
empty_list = []
```

Lists can contain mixed data types:

```
mixed = [1, "hello", 3.14, True]
```

Accessing List Elements

List elements can be accessed using their index.

Example 5.17

```
fruits = ["apple", "banana", "cherry"]
```

```
print(fruits[0])           # Output: apple
```

```
print(fruits[2])           # Output: cherry
```

Negative indexing allows access from the end:

```
print(fruits[-1])          # Output: cherry
```

```

Example 5.17.py - /home/python/...
File Edit Format Run Options Window Help
fruits = ["apple","bannana","cherry"]
print(fruits[0])
print(fruits[2])
print(fruits[-1])

IDLE Shell 3.8.10
Python 3.8.10 (default, Jun 2 2021, 10:49:15)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /home/python/
Example 5.17.py =====
apple
cherry
cherry
>>>

```

Example 5.17

Slicing lists

Slicing is used to access a range of elements from a list.

Syntax

`list[start:stop:step]`

Example 5.18

```
numbers = [10, 20, 30, 40, 50]
```

```
print(numbers[1:4])          # Output: [20, 30, 40]
```

```

Example 5.18.py - /home/python/...
File Edit Format Run Options Window Help
numbers = [10, 20, 30, 40, 50]
print(numbers[1:4])

IDLE Shell 3.8.10
Python 3.8.10 (default, Jun 2 2021, 10:49:15)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /home/python/Example 5.18.py =====
[20, 30, 40]
>>>

```

Example 5.18

Modifying lists

Lists are mutable, so their elements can be changed.

Example 5.19

```
fruits = ["apple", "banana", "cherry"]
```

```
fruits[1] = "blueberry"
```

```
print(fruits)          # Output: ['apple', 'blueberry', 'cherry']
```

```

Example 5.19.py - /home/python/...
File Edit Format Run Options Window Help
fruits = ["apple","bannana","cherry"]
fruits[1] = "blueberry"
print(fruits)

IDLE Shell 3.8.10
Python 3.8.10 (default, Jun 2 2021, 10:49:15)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /home/python/Example 5.19.py =====
['apple', 'blueberry', 'cherry']
>>>

```

Example 5.19

Adding elements to a list

1. `append()`: Adds an item to the end of the list.
`fruits.append("orange")`
2. `insert()`: Adds an item at a specific index.
`fruits.insert(1, "grape")`
3. `extend()`: include elements from another list into the existing list.
`fruits.extend(["kiwi", "melon"])`

Removing elements from a list

1. `remove()`: Removes the first occurrence of a value.
`fruits.remove("banana")`
2. `pop()`: Removes an item at a given index.
`fruits.pop(2)`
3. `clear()`: Removes all items from the list.
`fruits.clear()`

List length

We can use `len()` to find the number of items in a list.

```
print(len(fruits))
```

Looping through a list

Using *for* loop:

```
for fruit in fruits:  
    print(fruit)
```

Using *while* loop:

```
i = 0  
while i < len(fruits):  
    print(fruits[i])  
    i += 1
```

Checking membership

We can use *in* and *not in* to check if an item exists in a list.

```
if "apple" in fruits:  
    print("Apple is in the list")
```

List Methods summary

Method	Description
append()	Adds an item to the end
insert()	Inserts item at specified index
extend()	Adds multiple items
remove()	Removes first occurrence of item
pop()	Removes item at index
clear()	Removes all items
index()	Returns index of first occurrence
count()	Returns number of occurrences
sort()	Sorts the list in ascending order
reverse()	Reverses the list
copy()	Returns a shallow copy of the list

Student marks

Examples 5.20

```
marks = [85, 90, 78, 92]
average = sum(marks) / len(marks)
print("Average marks:", average)
```

```
Example 5.20.py - /home/python/Exa...
marks = [85, 90, 78, 92]
average = sum(marks) / len(marks)
print("Average marks:", average)

Python 3.8.10 (default, Jun 2 2021, 10:49:15)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /home/python/Example 5.20.py =====
Average marks: 86.25
>>>
```

Example 5.20

Shopping list

Example 5.21

```
shopping = []
shopping.append("milk")
shopping.append("bread")
shopping.append("eggs")
print(shopping)
```

```
Example 5.21.py - /home/python...
shopping = []
shopping.append("milk")
shopping.append("bread")
shopping.append("eggs")
print(shopping)

Python 3.8.10 (default, Jun 2 2021, 10:49:15)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /home/python/Example 5.21.py =====
['milk', 'bread', 'eggs']
>>>
```

Example 5.21

Removing duplicates

Example 5.22

```
items = ["apple", "banana", "apple", "orange"]
unique_items = []
```

for item in items:

if item not in unique_items:

unique_items.append(item)

print(unique_items)

```

Example 5.22.py - /home/python/Examp...
File Edit Format Run Options Window Help
items = ["apple", "banana", "apple", "orange"]
unique_items = []
for item in items:
    if item not in unique_items:
        unique_items.append(item)
print(unique_items)

Python 3.8.10 (default, Jun 2 2021, 10:49:15)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or
"license()" for more information.
>>>
===== RESTART: /home/
python/Example 5.22.py =====
>>>
['apple', 'banana', 'orange']
>>>
Ln: 6 Col: 19
Ln: 7 Col: 4

```

Example 5.22

List comprehension

List comprehension provides a concise way to create lists.

Example 5.23

squares = [x*x for x in range(1, 6)]

print(squares) # Output: [1, 4, 9, 16, 25]

```

Example 5.23.py - /home/python...
File Edit Format Run Options Window Help
squares = [x*x for x in range(1, 6)]
print(squares)

Python 3.8.10 (default, Jun 2 2021, 10:49:15)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or
"license()" for more information.
>>>
===== RESTART: /home/py
thon/Example 5.23.py =====
>>>
[1, 4, 9, 16, 25]
>>>
Ln: 3 Col: 9
Ln: 7 Col: 4

```

Example 5.23

Nested lists

Similar to Loops, Lists can contain other lists.

```

Example 5.24.py - /home/python...
File Edit Format Run Options Window Help
matrix = [[1,2],[3,4],[5,6]]
print(matrix[1][0])

Python 3.8.10 (default, Jun 2 2021, 10:49:15)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits"
or "license()" for more information
>>>
===== RESTART: /hom
e/python/Example 5.24.py =====
>>>
3
>>>
Ln: 3 Col: 0
Ln: 7 Col: 4

```

Example 5.24

matrix = [[1, 2], [3, 4], [5, 6]]

print(matrix[1][0]) # Output: 3

Example 5.24

Exercises

1. Create a list of 5 fruits and print them using a loop.
2. Write a program to find the largest number in a list.
3. Create a list of numbers and remove all even numbers.
4. Use list comprehension to generate a list of cubes from 1 to 10.
5. Create a nested list and access specific elements.

5.4 Tuples

In Python, a *tuple* is a built-in data type used to store multiple items in a single variable. Tuples are like lists in many ways, but with one key difference: tuples are immutable, meaning their contents cannot be changed once created. This immutability makes tuples ideal for storing fixed collections of data, such as coordinates, configuration settings or constant values. Tuples are especially useful when you want to ensure that data remains unchanged throughout the program.

Creating Tuples

Tuples are defined using parentheses (), with elements separated by commas.

```
fruits = ("apple", "banana", "cherry")
```

```
numbers = (10, 20, 30)
```

You can also create a tuple without parentheses:

```
colors = "red", "green", "blue"
```

Empty tuple

```
empty = ()
```

Single-element tuple:

To create a tuple with one item, include a trailing comma:

```
single = ("apple",)
```

Accessing Tuple Elements

Tuple elements are accessed using their index.

Examples 5.25

```
fruits = ("apple", "banana", "cherry")
```

```
print(fruits[0]) # Output: apple
```

```
print(fruits[-1]) # Output: cherry (last item)
```

```

Example 5.25.py - /home/python...
File Edit Format Run Options Window Help
fruits = ("apple","banana","cherry")
print(fruits[0])
print(fruits[-1])

Python 3.8.10 (default, Jun 2 2021, 10:49:15)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /home/python/Example 5.25.py =====
>>>
apple
cherry
>>>
Ln: 5 Col: 0
Ln: 8 Col: 4

```

Example 5.25

Slicing Tuples

Slicing allows you to extract a portion of the tuple.

Examples 5.26

```
numbers = (10, 20, 30, 40, 50)
```

```
print(numbers[1:4]) # Output: (20, 30, 40)
```

You can also use step values:

```
print(numbers[::2]) # Output: (10, 30, 50)
```

```

Example 5.26.py - /home/python...
File Edit Format Run Options Window Help
numbers = (10, 20, 30, 40, 50)
print(numbers[1:4])
print(numbers[::2])

Python 3.8.10 (default, Jun 2 2021, 10:49:15)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /home/python/Example 5.26.py =====
>>>
(20, 30, 40)
(10, 30, 50)
>>>
Ln: 1 Col: 1
Ln: 8 Col: 4

```

Example 5.26

Tuple operations

Although tuples are immutable, you can perform several operations:

1. Concatenation

```

Example 5.27.py - /home/python...
File Edit Format Run Options Window Help
a = (1,2)
b = (3, 4)
c = a + b
repeat = ("Hi",) * 3
print(repeat)

Python 3.8.10 (default, Jun 2 2021, 10:49:15)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /home/python/Example 5.27.py =====
>>>
(1, 2, 3, 4)
('Hi', 'Hi', 'Hi')
>>>
Ln: 6 Col: 13
Ln: 1 Col: 1

```

Example 5.27

Examples 5.27

```
a = (1, 2)
```

```
b = (3, 4)
```

```
c = a + b
```

```
print(c) # Output: (1, 2, 3, 4)
```

2. Repetition

```
repeat = ("Hi",) * 3
```

```
print(repeat) # Output: ('Hi', 'Hi', 'Hi')
```

3. Membership testing:

```
fruits = ("apple", "banana", "cherry")
```

```
if "apple" in fruits:
```

```
    print("Apple is present")           # Output: Apple is present
```

Measuring tuple length

Use len() to find the number of items:

```
print(len(fruits))                     # Output: 3
```

Looping through tuples**Using for loop:**

```
for fruit in fruits:                   # Output: apple
    print(fruit)                       banana
                                       cherry
```

Using while loop:

```
i = 0
while i < len(fruits):                 # Output: apple
    print(fruits[i])                   banana
    i += 1                             cherry
```

Tuple methods

Tuples have limited built-in methods due to their immutability.

Method	Description
count()	Returns the number of times a value appears
index()	Returns the index of the first occurrence

Example 5.28

```
numbers = (1, 2, 3, 2, 2)
print(numbers.count(2)) # Output: 3
print(numbers.index(3)) # Output: 2
```

```
Example 5.28.py - /hom...  IDLE Shell 3.8.10
File Edit Format Run Options  File Edit Shell Debug Options Window Help
Window Help
Python 3.8.10 (default, Jun 2 2021, 10:4
9:15)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "l
icense()" for more information.
>>>
===== RESTART: /home/pyth
on/Example 5.28.py =====
3
2
>>>
```

Example 5.28

5.5 Sets

In Python, a set is a built-in data type used to store multiple items in a single variable. Unlike lists or tuples, sets are unordered, unindexed and do not allow duplicate values. Sets are particularly useful when you need to store unique elements and perform operations

like union, intersection and difference—similar to mathematical sets. Sets are especially useful for filtering duplicates, checking membership and performing fast operations on large datasets.

Characteristics of sets

Understanding the properties of sets helps in using them effectively:

- **Unordered:** The items in a set do not have a guaranteed order and may appear in a different order each time they are accessed.
- **Unindexed:** Set elements cannot be accessed using indices like lists or tuples.
- **No duplicates:** Sets automatically remove duplicate values.
- **Immutable elements:** The elements of a set must be immutable (e.g., numbers, strings, tuples). You cannot include lists or dictionaries as set elements.
- **Mutable container:** While the elements are immutable, the set itself is mutable—you can add or remove items.

Creating sets

Sets are created using curly braces {} or the set() constructor.

```
fruits = {"apple", "banana", "cherry"}
```

```
numbers = {1, 2, 3, 4}
```

Using set() constructor:

```
colors = set(["red", "green", "blue"])
```

Empty set:

```
empty = set()          # NOT empty = {} (which creates a dictionary)
```

Duplicate removal:

```
items = {"apple", "banana", "apple"}
```

```
print(items)          # Output: {'apple', 'banana'}
```

Accessing set elements

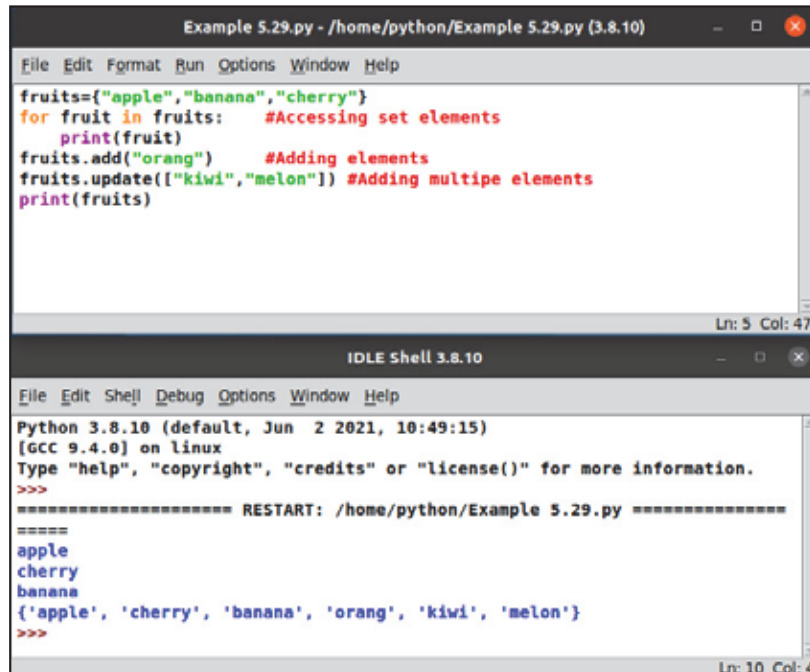
Since sets are unordered and unindexed, you cannot access elements by position. Instead, you can loop through the set:

Examples 5.29

```
fruits = {"apple", "banana", "cherry"}
```

```
for fruit in fruits:
```

```
    print(fruit)
```



```

Example 5.29.py - /home/python/Example 5.29.py (3.8.10)
File Edit Format Run Options Window Help
fruits={"apple","banana","cherry"}
for fruit in fruits:    #Accessing set elements
    print(fruit)
fruits.add("orange")    #Adding elements
fruits.update(["kiwi","melon"]) #Adding multiple elements
print(fruits)

Ln: 5 Col: 47

IDLE Shell 3.8.10
File Edit Shell Debug Options Window Help
Python 3.8.10 (default, Jun 2 2021, 10:49:15)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /home/python/Example 5.29.py =====
>>>
apple
cherry
banana
{'apple', 'cherry', 'banana', 'orang', 'kiwi', 'melon'}
>>>

Ln: 10 Col: 4

```

Example 5.29

Adding Elements to a Set

1. **add()** – Adds a single element

```
fruits.add("orange")
```

2. **update()** – Adds multiple elements

```
fruits.update(["kiwi", "melon"])
```

Removing elements from a set

1. **remove()** – Removes a specific item; raises error if not found

```
fruits.remove("banana")
```

2. **discard()** – Removes item without error if not found

```
fruits.discard("banana")
```

3. **pop()** – Removes a random item

```
fruits.pop()
```

4. **clear()** – Removes all items

```
fruits.clear()
```

Measuring set size

Use **len()** to find the number of items:

```
fruits = {"apple", "banana", "cherry"}
```

```
print(len(fruits))          # Output: 3
```

Looping through sets

```
fruits = {"apple", "banana", "cherry"}    # Output: apple
for item in fruits:                        banana
    print(item)                            cherry
```

Membership testing

Use in and not in:

```
fruits = {"apple", "banana", "cherry"}
if "apple" in fruits:
    print("Apple is present")    # Output: Apple is present
```

Set operations

Python sets support standard mathematical set operations including Union, Intersection etc.

1. Union – Combines elements from both sets

```
Example 5.30.py - /h... IDLE Shell 3.8.10
File Edit Format Run Options
Window Help
a = {1, 2, 3}
b = {3, 4, 5}
print(a | b)
print(a & b)
print(a - b)
print(a ^ b)

Python 3.8.10 (default, Jun 2 2021, 10:49:15)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or
r "license()" for more information.
>>>
===== RESTART: /home/
python/Example 5.30.py =====
>>>
{1, 2, 3, 4, 5}
{3}
{1, 2}
{1, 2, 4, 5}
>>>
```

Examples 5.30

```
a = {1, 2, 3}
b = {3, 4, 5}
print(a | b) # Output: {1, 2, 3, 4, 5}
```

2. Intersection – Common elements

```
print(a & b) # Output: {3}
```

Example 5.30

3. Difference – Elements in a but not in b

```
print(a - b)    # Output: {1, 2}
```

4. Symmetric Difference – Elements in either set but not both

```
print(a ^ b)    # Output: {1, 2, 4, 5}
```

Set Methods Summary

Method	Description
add()	Adds a single element
update()	Adds multiple elements
remove()	Removes specified element (error if not found)
discard()	Removes specified element (no error)
pop()	Removes a random element
clear()	Removes all elements

union()	Returns union of sets
intersection()	Returns common elements
difference()	Returns elements in one set but not the other
symmetric_difference()	Returns elements in either set but not both
issubset()	Checks if one set is a subset of another
issuperset()	Checks if one set is a superset of another
isdisjoint()	Checks if sets have no common elements

Removing duplicates from a list

```
names = ["Alice", "Bob", "Alice", "Charlie"]
```

```
unique_names = set(names)
```

```
print(unique_names)      # Output: {'Alice', 'Bob', 'Charlie'}
```

Set operations

Example 5.31

```
a = {"apple", "banana", "cherry"}
```

```
b = {"banana", "kiwi", "melon"}
```

```
print("Union:", a.union(b))
```

```
print("Intersection:", a.intersection(b))
```

```
print("Difference:", a.difference(b))
```

Output:

```
Union: {'cherry', 'kiwi', 'apple', 'melon', 'banana'}
```

```
Intersection: {'banana'}
```

```
Difference: {'cherry', 'apple'}
```

Checking subsets

The screenshot shows a Python IDE window titled 'Example 5.31.py - /home/python/Example 5.31.py (3.8.10)'. The code defines two sets, 'a' and 'b', and prints their union, intersection, and difference. The output window shows the results: Union: {'cherry', 'kiwi', 'apple', 'melon', 'banana'}, Intersection: {'banana'}, and Difference: {'cherry', 'apple'}.

```

a = {"apple", "banana", "cherry"}
b = {"banana", "kiwi", "melon"}
print("Union:", a.union(b))
print("Intersection:", a.intersection(b))
print("Difference:", a.difference(b))

```

```

Python 3.8.10 (default, Jun 2 2021, 10:49:15)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license()" for
more information.
>>>
===== RESTART: /home/python/Example 5.
31.py =====
Union: {'cherry', 'kiwi', 'apple', 'melon', 'banana'}
Intersection: {'banana'}
Difference: {'cherry', 'apple'}
>>>

```

Example 5.31

Example 5.32

```
x = {1, 2}
```

```
y = {1, 2, 3, 4}
```

```
print(x.issubset(y)) # Output: True
```

The screenshot shows a Python IDE window titled 'Example 5.32.py - /home/python/Example 5.32.py (3.8.10)'. The code defines two sets, 'x' and 'y', and prints whether 'x' is a subset of 'y'. The output window shows the result: True.

```

x = {1, 2}
y = {1, 2, 3, 4}
print(x.issubset(y))

```

```

Python 3.8.10 (default, Jun 2 2021, 10:49:15)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license()" for
more information.
>>>
===== RESTART: /home/python/Example 5.32.py =====
True
>>>

```

Example 5.32

Exercise

1. Create a set of 5 fruits and then add a new fruit.
2. Write a program to remove duplicates from a list using a set.
3. Perform union and intersection on two sets of numbers.
4. Check if one set is a subset of another.
5. Create a set and then remove an element using discard().

5.6 Dictionaries

In Python, a dictionary is a built-in data type that allows you to store and manage data in the form of key-value pairs. Unlike lists and tuples that use numeric indices to access elements, dictionaries use unique keys to access corresponding values. This makes dictionaries ideal for representing structured data, such as student records, contact information or configuration settings.

Dictionaries are also known as associative arrays or hash maps in other programming languages. They are highly efficient for lookup, insertion and deletion operations, making them one of the most powerful and flexible data structures in Python.

Characteristics of Dictionaries

- **Key-Value pairs:** Each item in a dictionary is a pair consisting of a key and its associated value.
- **Unordered (before Python 3.7):** Dictionaries were unordered before Python 3.7. From Python 3.7 onwards, dictionaries preserve the insertion order.
- **Mutable:** Dictionaries can be modified after creation—keys and values can be added, updated or removed.
- **Unique keys:** Keys must be unique within a dictionary. If a key is repeated, the last value assigned to it will overwrite the previous one.
- **Immutable keys:** Keys must be of an immutable data type, such as strings, numbers or tuples. Lists and dictionaries cannot be used as keys.
- **Values can be any type:** Values can be of any data type—strings, numbers, lists, other dictionaries, etc.

Creating Dictionaries

Dictionaries are created using curly braces {} or the dict() constructor.

Using curly braces

Example 5.33

```
student = {
    "name": "Aryananda",
    "age": 17,
    "class": "12A"
}
print(student)
```

#Output: {'name': 'Aryananda', 'age': 17, 'class': '12A'}

The screenshot shows two windows from the IDLE 3.8.10 environment. The left window, titled 'Example 5.33.py - /home/pytho...', contains the following code:

```
student = {"name": "Aryananda",
           "age": 17,
           "class": "12A"}
print(student)
```

The right window, titled 'IDLE Shell 3.8.10', shows the output of the script after execution:

```
Python 3.8.10 (default, Jun 2 2021, 10:49:15)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license()"
>>>
===== RESTART: /home/python/Exam
ple 5.33.py =====
{'name': 'Aryananda', 'age': 17, 'class': '12A'}
>>>
```

Example 5.33

Using dict() Constructor

Example 5.34

```
student = dict(name="Sreerag", age=17, class_="CSE")
```

Note: When using dict(), keys must be valid identifiers (no spaces or special characters).

The screenshot shows two windows from the IDLE 3.8.10 environment. The left window, titled 'Example 5.34.py - /home/pytho...', contains the following code:

```
student = dict(name = "Sreerag",
               age = 17,
               class_ = "CSE")
print(student)
```

The right window, titled 'IDLE Shell 3.8.10', shows the output of the script after execution:

```
Python 3.8.10 (default, Jun 2 2021, 10:49:15)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license()"
>>>
===== RESTART: /home/python/Exa
mple 5.34.py =====
{'name': 'Sreerag', 'age': 17, 'class_': 'CSE'}
>>>
```

Example 5.34

Accessing Dictionary Elements

You can access values using their keys. An example is given below.

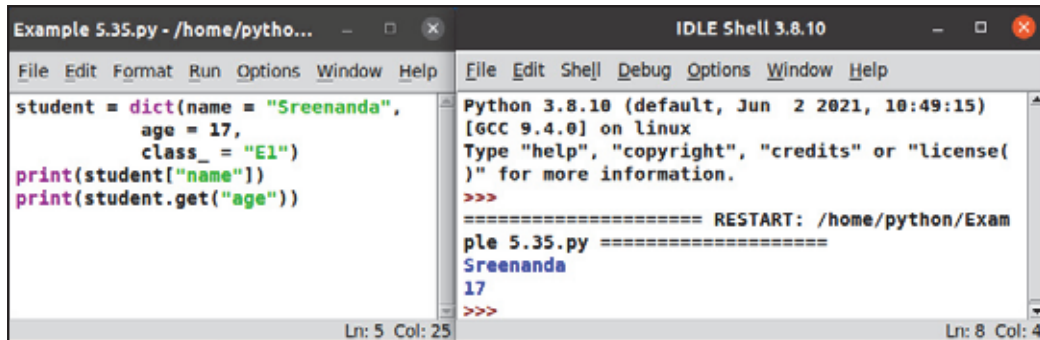
Example 5.35

```
student = dict(name="Sreenanda", age=17, class_="E1")
print(student["name"])
```

Output: Sreenanda

```
print(student.get("age"))           # Output: 17
```

The get() method is safer because it returns None if the key does not exist, whereas using square brackets will raise a KeyError.



```

Example 5.35.py - /home/pytho...
File Edit Format Run Options Window Help
student = dict(name = "Sreenanda",
               age = 17,
               class_ = "E1")
print(student["name"])
print(student.get("age"))

Ln: 5 Col: 25

IDLE Shell 3.8.10
File Edit Shell Debug Options Window Help
Python 3.8.10 (default, Jun 2 2021, 10:49:15)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license(
)" for more information.
>>>
===== RESTART: /home/python/Exam
ple 5.35.py =====
Sreenanda
17
>>>

Ln: 8 Col: 4

```

Example 5.35

Modifying Dictionary Elements

You can update the value of an existing key or add a new key-value pair.

Example 5.36

```
student = dict(name="Sreelakshmi", age=17, class_="XI")
```

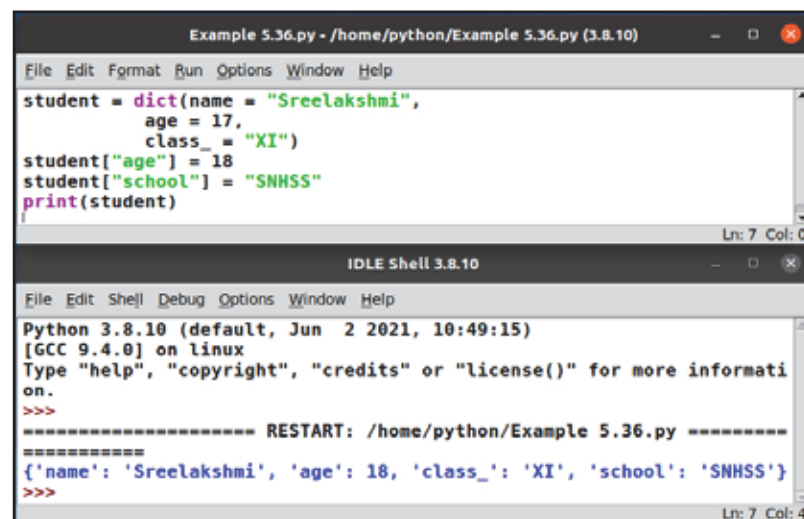
```
student["age"] = 18           # Update
```

```
student["school"] = "SNHSS" # Add
```

```
print (student)
```

Output:

```
{'name': 'Sreelakshmi', 'age': 18, 'class_': 'XI', 'school': 'SNHSS'}
```



```

Example 5.36.py - /home/python/Example 5.36.py (3.8.10)
File Edit Format Run Options Window Help
student = dict(name = "Sreelakshmi",
               age = 17,
               class_ = "XI")
student["age"] = 18
student["school"] = "SNHSS"
print(student)

Ln: 7 Col: 0

IDLE Shell 3.8.10
File Edit Shell Debug Options Window Help
Python 3.8.10 (default, Jun 2 2021, 10:49:15)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license()" for more informati
on.
>>>
===== RESTART: /home/python/Example 5.36.py =====
{'name': 'Sreelakshmi', 'age': 18, 'class_': 'XI', 'school': 'SNHSS'}
>>>

Ln: 7 Col: 4

```

Example 5.36

Removing Elements

1. pop(key)

Removes the item with the specified key.

Example 5.37

```
student = dict(name="Devarag", age=17, class_="XI")
student.pop("class_")
```

2. popitem()

Removes the last inserted item (Python 3.7+).

```
student.popitem()
```

3. del

Deletes a key-value pair.

```
del student["age"]
```

4. clear()

Removes all items from the dictionary.

```
student.clear()
```

```

Example 5.37.py - /home/pytho...
File Edit Format Run Options Window Help
student = dict(name = "Devarag",
               age = 17,
               class_ = "XI")
student.pop("class_")
del student["age"]
print(student)
student.clear()
print(student)

Python 3.8.10 (default, Jun 2 2021, 10:49:15)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license()"
>>>
===== RESTART: /home/python/Examp
le 5.37.py =====
{'name': 'Devarag'}
{}
>>>

```

Example 5.37

Dictionary Methods Summary

Method	Description
get(key)	Returns the value for the key
keys()	Returns a view of all keys
values()	Returns a view of all values
items()	Returns a view of all key-value pairs
update(dict2)	Updates the dictionary with another dictionary
pop(key)	Removes the item with the specified key
popitem()	Removes the last inserted item
clear()	Removes all items
copy()	Returns a shallow copy of the dictionary
setdefault()	Returns value of a key; sets default if not found

Comparison of List, Tuple, Set, and Dictionary in Python

Feature / Property	List	Tuple	Set	Dictionary
Definition	Ordered collection of items	Ordered, immutable collection	Unordered collection of unique items	Unordered collection of key-value pairs
Syntax	[] (square brackets)	() (parentheses)	{ } or set()	{key: value} (curly braces)
Mutable	Yes	No	Yes	Yes
Ordered (Python 3.7+)	Yes	Yes	No	Yes
Indexed	Yes (0-based indexing)	Yes (0-based indexing)	No	No (access via keys)
Allows Duplicates	Yes	Yes	No	Keys: No, Values: Yes
Element Access	By index	By index	Not allowed	By key
Use Case	General-purpose collection	Fixed data, read-only collections	Unique items, set operations	Structured data with key-value mapping
Common Methods	append(), insert(), pop()	count(), index()	add(), remove(), union()	get(), keys(), values(), items()
Performance	Slower than tuple for iteration	Faster than list for iteration	Fast membership testing	Fast lookup by key
Nested Structure	Yes	Yes	Yes (but elements are be immutable)	Yes (values can be any data type)
Memory Usage	More than tuple	Less than list	Less than list (no duplicates)	Depends on number of key-value pairs
Example	["a", "b", "c"]	("a", "b", "c")	{"a", "b", "c"}	{"name": "Anu", "age": 17}

Summary

This chapter introduces the fundamental control structures of Python, enabling students to manage program flow using conditional and looping statements. It explains decision-making constructs such as if-elif-else and match-case, along with iterative structures like for and while loops, supported by clear examples. The chapter also explores essential Python data structures—lists, tuples, sets and dictionaries—highlighting their properties and practical uses. By combining control logic with data handling, students gain the ability to write structured, efficient and readable programs. Overall, the chapter lays a strong foundation for problem-solving and logical thinking in Python programming.



1. How many else blocks can be used in an if structure?
2. Which symbol represents default case in match-case?
 - a) * b) # c) _ d) %
3. What is the default start value of range()?
4. Which statement stops the loop immediately?
 - a) continue b) break c) stop d) exit
5. What is the index of the first element in a list?
6. Which method removes all elements from a list?
7. Which data structure does not allow duplicates?
 - a) List b) Tuple c) Set d) Dictionary
8. What does a & b represent in sets?
 - a) Union b) Intersection c) Difference d) Symmetric difference
9. The _____ method safely retrieves a value from a dictionary.
10. The method used to remove all items from a dictionary is _____
11. Write the syntax of if-elif-else statement.
12. Explain the role of indentation in Python.
13. Differentiate between for loop and while loop.
14. What is a nested loop?
15. Distinguish between break and continue.
16. What is an infinite loop?
17. Define list and mention any two characteristics.
18. What is list slicing? Give an example.
19. What is a tuple? Why is it called immutable?
20. Differentiate between list and tuple.
21. Define a set and mention its key features.
22. What are set operations? Name any two.

23. Define dictionary with an example.
24. How are dictionary elements accessed?
25. Differentiate between keys and values in a dictionary.
26. Explain the working of for loop with else using an example.
27. Explain union and intersection of sets with examples.
28. Explain different methods used in dictionaries.
29. Explain the working of match-case with an example.
30. Write a program to count vowels in a string.
31. Write a program to print numbers from 1 to 10 using while loop.
32. Write a program to print multiplication table using nested for loop.
33. Write a program to demonstrate list operations (append, remove, pop).
34. Write a program to remove duplicates using a set.
35. Write a program to create and update a dictionary.

6

Computer Networks

"The Internet is becoming the town square for the global village of tomorrow."

Bill Gates

6.1 Introduction

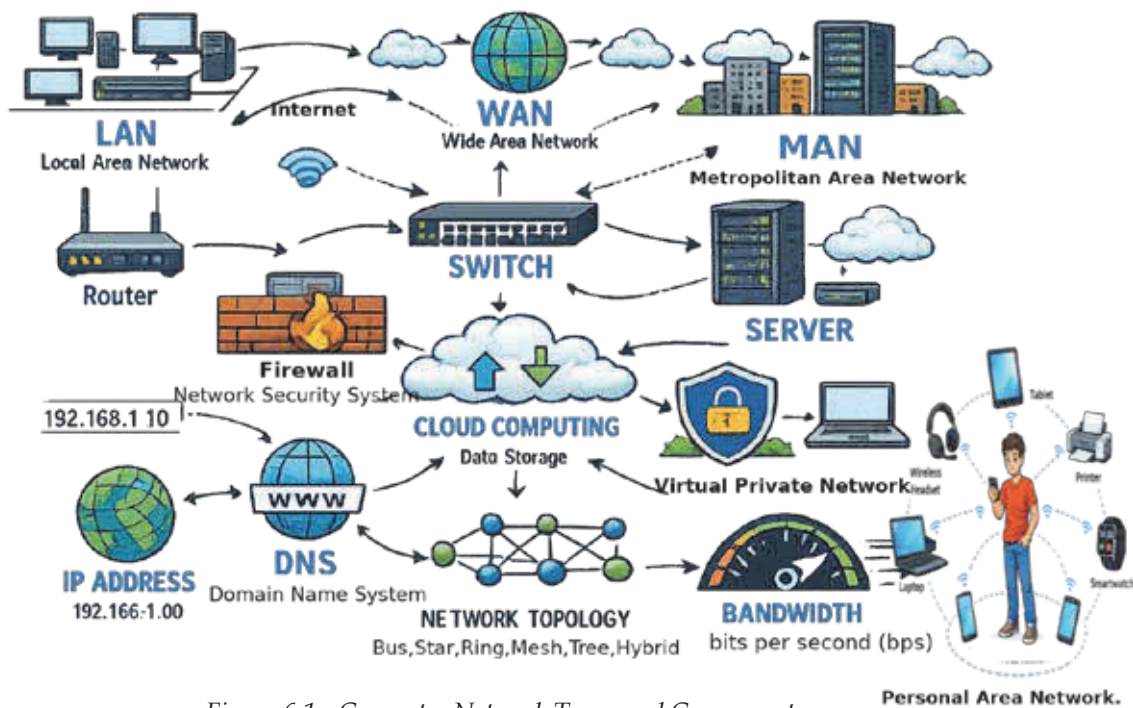


Figure 6.1 - Computer Network Types and Components

A network is created when two or more individuals or devices are connected, enabling them to communicate and share resources. Imagine sending a message to your friend in another city and receiving a reply within seconds. This is possible because of computer networks. Today's world is highly interconnected, with

information being generated, shared, and tracked globally in real time. This is possible because people, devices, and systems in the digital environment are connected in one way or another, forming a vast and continuous network. Computer networks help us to communicate, learn, work, and entertain ourselves. When we use the Internet, send emails, attend online classes, watch videos, or play online games, we are using computer networks. Even devices like mobile phones, printers, and smart TVs are connected through networks. In simple words, a computer network acts like a digital bridge that connects people and devices across the world, making communication fast, easy, and efficient.

We experience many kinds of networks in our daily life (Figure 6.2):

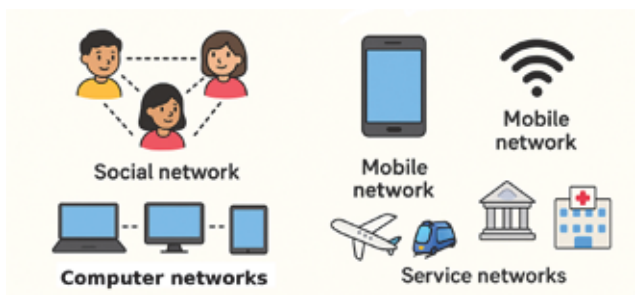


Figure 6.2 - Different Networks

- Social networks – friends and families are connected through relationships.
- Mobile networks – phones connected for calls, messages, and internet.
- Computer networks – computers sharing files, videos, and information.
- Service networks – airlines, railways, banks, and hospitals that connect people, places, and services.

A computer network is an interconnection among two or more computers or computing devices. Such interconnections allow computers to share data and resources among each other. A small network may link just a few computers in the same room, while a large network can connect thousands of devices across the world. The network size may vary from small to large depending on the number of connected computers. A computer network can include different types of hosts (also called nodes) like server, desktop, laptop, cellular phones, etc. Apart from computers, networks include networking devices like switch, router, modem, etc (Figure 6.1). Networking devices are used to connect multiple computers in different settings. The main goal of a network is to connect devices and people so that they can communicate, share resources, and work together easily.

6.2 Data Communication

Data Communication refers to the process of transmitting digital data between two or more devices through a communication medium such as cables or wireless signals. The main goal is to exchange information efficiently and accurately.

Basic Components:

1. **Sender:** The device that sends the data (Figure 6.3).
2. **Receiver:** The device that receives the data.
3. **Message:** The actual information or data to be transmitted.
4. **Medium (Transmission Channel):** The path through which data travels (e.g., cables, airwaves).
5. **Protocol:** A set of rules that governs data transmission (e.g., TCP/IP).

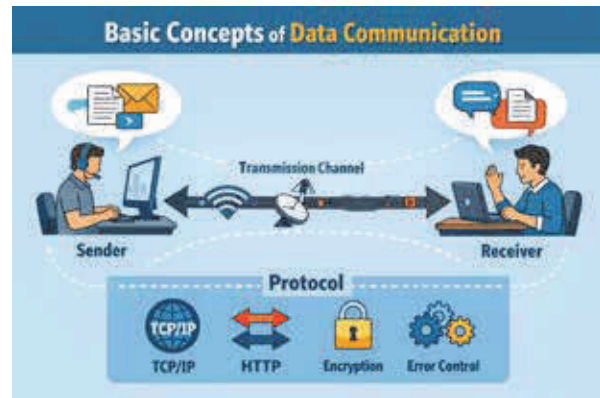


Figure 6.3 - Components of Data Communication

Key Terminologies:

- **Node:** A node in a computer network is any device that can send, receive, or process data. It serves as a connection point within the network, allowing communication between devices. A node can be a device such as computers (desktops, laptops), servers, printers, routers, switches, modems, smartphones and IoT (Internet of Things) devices. Each node has a unique network address (such as an IP address) that helps to identify it within the network. Nodes work together to share resources, transmit data, and maintain communication. In short, a node is any active device connected to a network that participates in data exchange.
- **Bandwidth:** The amount of data that can be transmitted over a network in a given time is called bandwidth. (Figure 6.4) for example: Consider a water pipe system.
- The pipe represents the communication channel.
- The water represents the data.
- The width of the pipe represents the bandwidth.

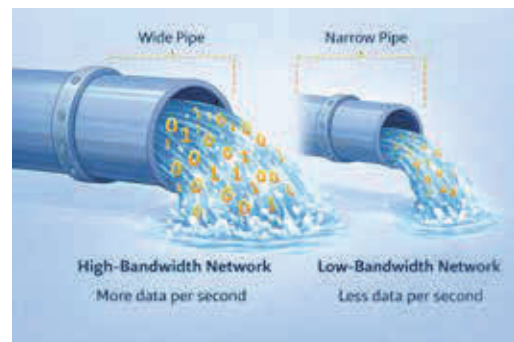


Figure 6.4 - Bandwidth

If the pipe is wide, more water can flow at the same time. This is similar to a high-bandwidth network, which can transmit more data per second. If the pipe is narrow, less water flows. This is like a low-bandwidth network, where data transfer is slower. Therefore, greater bandwidth allows more data to be transmitted in less time.



Figure 6.5 - Latency

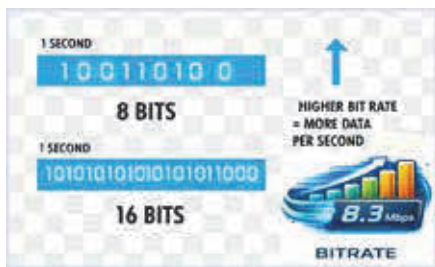


Figure 6.6 - Bit Rate

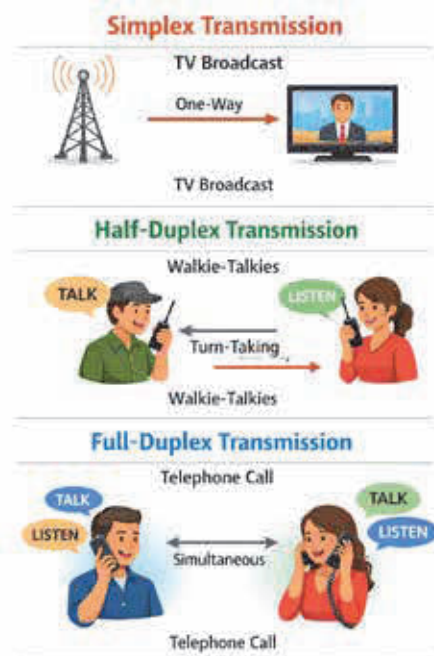


Figure 6.7 - Different Transmission Modes

- **Latency:** The time taken for data to travel from source to destination is called latency. Example: Imagine you're chatting with a friend online. You send a message, and there's a slight delay before your friend receives it. This delay is latency. In simple words, latency means delay in data transmission. (Figure 6.5)
- **Bit Rate:** It is the number of bits transmitted per second (bps). (Figure 6.6)
- **Transmission Modes:** The direction in which data flows between two communicating devices in a network. There are three main transmission modes (Figure 6.7):
 1. **Simplex:** Data flows only in one direction from sender to receiver. (one-way, e.g. Keyboard to computer, TV/ radio broadcast.)
 2. **Half Duplex:** Data can travel in both directions, but not simultaneously, one device sends while the other receives. (two-way, one at a time, e.g. Walkie-Talkies),
 3. **Full Duplex:** Data flows in both directions at the same time. (two-way simultaneously e.g. Mobile phones).

Effective data communication ensures that information is transmitted quickly, securely, and without loss or error between devices or networks. For communication, data in a network is divided into smaller packets. These packets are then carried over a network through a medium. Devices in a network can be connected using guided media such as cables/ Fiber optic cable (for wired communication) through which electrical

signals or light pulses propagate or unguided media such as free space (for wireless), through which electromagnetic (radio) waves propagate. In the context of data communication, interconnectivity of computing devices in a network allows us to exchange information simultaneously with many parties through email, websites, audio/video calls, etc. Network allows sharing of resources.

For example, Imagine you are preparing an assignment in the computer lab. All the computers in the lab are connected to one printer through a network. To take the printout of the assignment, you can directly give the print command, and the printer receives it through the network. This saves time, avoids extra work, and shows how computer networks make resources sharing simple and efficient.



1. Define a computer network.
2. What is a node in the network?
3. Give two examples for social networks.
4. Discuss how computer networks help in sharing resources efficiently.
5. Explain how computer networks connect people and devices in the modern world.
6. How does bandwidth affect data transmission speed?
7. Give an example for a real-life situation that uses different transmission modes.
8. How does a computer network improve communication and resource sharing in a school or office environment?
9. Using a diagram, explain the basic components of data communication and their functions.

6.3 Classification of Networks

Computer networks can be classified in different ways: based on the transmission media and geographical coverage.

6.3.1 Classification Based on Transmission Media

In a computer network, data is transmitted from one device to another through a channel. This channel is known as Transmission Media. It acts as a path between the sender and receiver through which data travels in the form of electrical signals, light, or electromagnetic waves. Transmission media are mainly classified into two categories.

A) Guided (Wired) Transmission Media:

Guided Transmission Media are physical paths used to transmit data signals from one device to another. In this type of media, signals

travel through cables, so the direction of transmission is guided by a solid medium. In wired networks, devices are connected using physical transmission media such as twisted pair cables, coaxial cables, and fiber optic cables (Figure 6.8). Guided transmission media are reliable, secure, and suitable for high-quality data transmission.

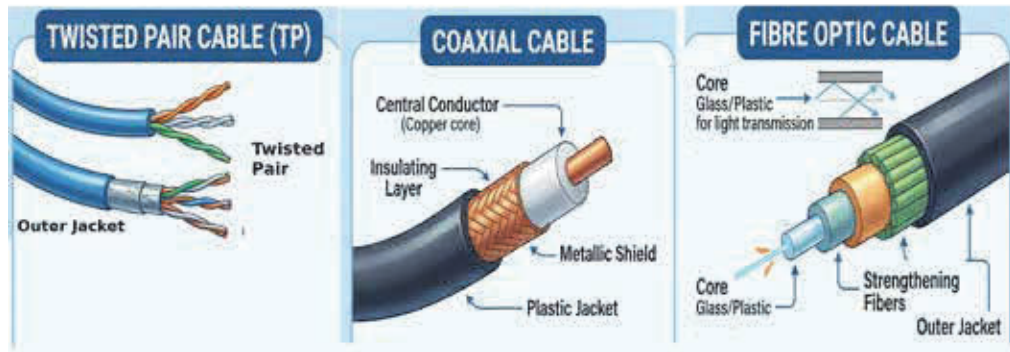


Figure 6.8 - Guided media

Table 6.1 Types of Guided Media

Type	Description	Features / Advantages	Limitations
1. Twisted Pair Cable	Consists of pairs of insulated copper wires twisted together to reduce interference. It is commonly used in telephone lines and local area networks (LANs).	• Inexpensive and easy to install • Suitable for short-distance communication.	• Prone to external interference and noise. • Limited bandwidth.
2. Coaxial Cable	Made of a central copper conductor surrounded by insulation, metallic shield, and outer cover. It is commonly used in cable TV and broadband connections.	• Better protection from noise. • Higher bandwidth than twisted pair.	• Thicker and harder to install • Costlier than twisted pair.
3. Optical Fiber Cable	Using light signals to transmit data through thin glass or plastic fibers	• Extremely high speed • Immune to electromagnetic interference • Suitable for long-distance communication	• Expensive • Fragile and difficult to install.

B) Unguided (Wireless) Transmission Media

Wireless networks connect devices without physical cables. Data is transmitted using electromagnetic signals (Figure 6.9). These networks provide greater flexibility and mobility, making them suitable for portable devices.



Figure 6.9 - Types of Unguided Media

Table 6.2 Types of Unguided Media

Type	Description	Features / Advantages	Limitations
1. Radio Waves	Used for wireless communication over long distances, such as FM radio, TV, and Wi-Fi.	• Can penetrate buildings • Supports mobility	• Prone to external interference • Limited security
2. Microwaves	High-frequency waves used for point-to-point communication between antennas.	• High data rate • Suitable for satellite and cellular communication	• Requires line-of-sight • Can be affected by weather conditions
3. Infrared Waves	Used for short-range communication like remote controls and wireless keyboards.	• Secure and simple • No interference with radio signals	• Limited range • Requires direct line-of-sight
4. Satellite Communication	Uses satellites orbiting the Earth to transmit data across large distances.	• Covers wide geographical areas • Useful for TV broadcasting and GPS	• Very costly • Delay in signal transmission.

Transmission media are the backbone of data communication in networks. The choice of medium depends on factors like distance, cost, data rate, and environment. For example, Twisted Pair Cable is used in small LANs (Local Area Network), Optical Fiber for high-speed backbone connections, and Wireless Media for mobile or remote communication.



1. Guided transmission media use _____ cables to transmit data
2. Optical fiber uses _____ signals for data transmission.
3. Radio waves can penetrate _____ and support mobility.
4. Microwave communication is suitable for _____ communication.
5. What are the two main ways to connect computers in a network?

6.3.2 Classification Based on Geographical Coverage

Based on the area they cover, networks are classified into the following types:

- A. PAN (Personal Area Network) connects devices within a very small area, typically around one person or in a room.
- B. LAN (Local Area Network) connects computers within a room, lab, or building.
- C. MAN (Metropolitan Area Network) covers a city.
- D. WAN (Wide Area Network) span over several countries or even the globe, the best example is the Banking Network / Internet

A. Personal Area Network (PAN)



Figure 6.10 - Personal Area Network

A Personal Area Network (PAN) is a type of computer network that connects devices within a very small area, typically around one person (Figure 6.10). It is designed for short-range communication, usually within a range of a few meters. PANs are commonly used to connect personal digital devices such as smartphones, tablets, laptops, printers, and wireless headsets.

PANs can be wired or wireless. A wired PAN might use USB cables to connect devices, while a wireless PAN (WPAN) uses technologies such as Bluetooth, Wi-Fi, or Infrared for data exchange. For example, when you connect your smartphone to a wireless earphone via Bluetooth, you are using a PAN.

The main advantage of a PAN is its convenience and mobility. It allows users to synchronize data, share files, and use devices together without the need for complex network setups. PANs are ideal for personal use, home offices, and mobile users who need to connect multiple devices on the go. In summary, a Personal Area Network provides an efficient and simple way to connect personal devices within a limited range, enabling quick communication, easy data sharing, and seamless device interaction in everyday life.

B. Local Area Network (LAN)

A Local Area Network (LAN) is a computer network that connects the server with devices like computers, mobile phones, tablet, printer etc within a small geographical area such as a room, building, school, or office (Figure 6.11). It allows users to share files, resources (like printers), and internet connections quickly and efficiently. LANs usually use Ethernet cables or Wi-Fi for connectivity, and they provide high-speed communication compared to larger networks. LAN is comparatively secure as only authentic users in the network can access other computers or shared resources. Users can print documents using a network printer, upload or 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 speed in LAN is quite high, and usually varies from 10 Mbps (called Ethernet) to 1000 Mbps (called Gigabit Ethernet). Ethernet is a set of rules that decides how the computers and other devices connect with each other through cables in a LAN. There are different types of LANs based on structure, purpose, and technology.



Figure 6.11 - Local Area Network

1. Wired LAN

A Wired Local Area Network (LAN) is a group of computers and peripheral devices connected within a limited geographical area, such as home, school or office building, using physical cables (Ethernet or Fiber Optics cables) to transmit data. For example: Computers in a computer lab is connected using Ethernet cables.

Advantages: High speed, reliable, secure connection and low latency.

2. Wireless LAN (WLAN)

A Wireless LAN (WLAN) Uses high-frequency radio waves instead of cables to transmit data. Devices connect to the network via a Wireless Access Point (WAP). The most common technology used for WLAN is Wi-Fi. For example: Laptops and smartphones are connected to a Wi-Fi router at home or in school.

Advantages: Mobility, scalability, easy installation and cost - efficient.

C. Metropolitan Area Network (MAN)



METROPOLITAN AREA NETWORK (MAN)

Figure 6.12 - Metropolitan Area Network

A Metropolitan Area Network (MAN) is a type of computer network that covers a larger area than a Local Area Network (LAN) (Figure 6.12). It Covers an area of 5 to 50 kilometers, usually spans in a city or a large campus, connecting multiple LANs within that area.

MANs are often used in businesses, government organizations, and educational institutions to share resources, exchange data, and provide high-speed internet connectivity.

They commonly use high-speed fiber optic cables or wireless connections to ensure fast and reliable communication across the metropolitan region. Data transfer rate in MAN generally ranges from 10 Mbps to 1 Gbps or higher, depending on the technology (fiber optic, wireless, etc.). Cable TV network, Public Wi-Fi hotspots or cable based broadband internet services are the examples for MAN.

D. Wide Area Network (WAN)

A Wide Area Network (WAN) is a large computer network that connects devices and smaller networks across vast geographical areas such as cities, countries, or continents (Figure 6.13). It is the largest form of network, much bigger than Local Area Networks (LANs) and Metropolitan Area Networks (MANs). WANs are essential for enabling communication, data transfer, and resource sharing over long distances. A WAN typically spans around thousands of kilometers. It uses various transmission media, including telephone lines, fiber optic cables, satellite links, and wireless systems. WANs are usually owned and maintained by Internet Service Providers (ISPs) or telecommunication companies. Data transfer speed is often slower than LANs or MANs due to distance, but with modern technologies, WANs can provide speed of several gigabits per second.

WANs allow organizations to connect offices worldwide, making communication and collaboration easier. They enable centralized data storage, resource sharing, and global access to applications. WANs support business growth by linking branches across the countries. The Internet, the most common example for WAN, demonstrates how these networks bring global connectivity. WANs are expensive to set up and maintain, requiring advanced infrastructure and skilled professionals to manage them. They are also more prone to security threats like hacking and data breaches. Additionally, long-distance communication can sometimes cause slower data transfer compared to LANs. The Internet is the largest WAN. Global banking networks, airline reservation systems, and multinational corporate networks are some examples for WAN. WANs are vital for connecting the world. Despite high costs and complexity, they provide the foundation for modern communication, making global connectivity possible.



Figure 6.13 - Wide Area Network

Table 6.3 Difference between WAN and Internet

WAN (Wide Area Network)	Internet
A network that connects computers and devices over a large geographical area	A global network of networks connecting millions of private and public networks
Limited to an organization or specific purpose	Worldwide, open to everyone
Owned and managed by a single organization (company, bank, government)	No single owner; managed collectively by ISPs and organizations
Restricted to authorized users only	Publicly accessible to all users
Bank branches connected across cities, Corporate offices in different countries are some examples	Websites, email, social media, cloud services are some examples
More secure as it is private	Less secure compared to private WANs
Expensive to set up and maintain	Relatively low cost for end users
Used for internal communication and data sharing	Used for global communication and information access



1. What are the different types of computer networks ?
2. Give two examples for wired connection media.
3. What type of signal is used in wireless communication?
4. Differentiate between wired and wireless networks.
5. Why is MAN called a “network of LANs”. Explain
6. How are networks classified based on the type of connection and geography.?
7. Imagine that you are setting up a PAN at home. What are the devices and technologies used for it?.
8. How does PAN improve personal productivity and convenience in a smart workspace environment.

6.4 Network devices

Network devices are the hardware components that connect computers and other electronic devices together to form a network. These devices help in data transmission, communication, and network management. Each device performs a specific role in ensuring that data travels efficiently between connected systems. To communicate data through transmission media and to configure networks with different functionality, we require devices like Modem, NIC, Hub, Switch, Repeater, Router, Gateway. Let us explore them in detail.

6.4.1 Modem (Modulator-Demodulator)

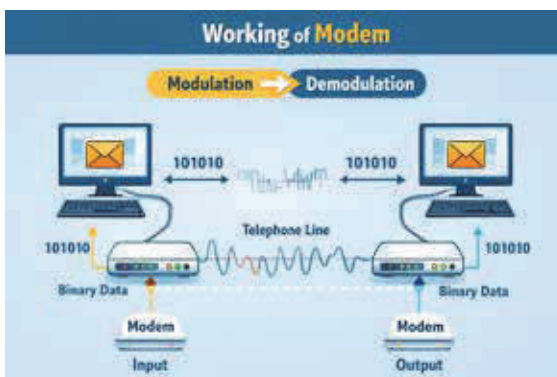


Figure 6.14 - Modem

The term “modem” stands for Modulator-Demodulator. It is a network device that allows digital computers to communicate over analog communication lines such as telephone lines (Figure 6.14). The modem plays a vital role in enabling Internet connectivity in homes, schools, and offices. Computers generate and understand digital signals (composed of 0s and 1s), while traditional telephone lines carry analog signals (continuous waves). The modem serves as a bridge between these two types of signals.

The modem performs two main operations:

1. Modulation:

When data is sent from a computer to the Internet, the modem converts the digital signals produced by the computer into analog signals suitable for transmission through the telephone lines.

2. Demodulation:

When data is received from the Internet, the modem converts the analog signals coming from the telephone line back into digital signals that the computer can understand.

Table 6.4 Type of Modem

Type of Modem	Connection Medium	Speed Range	Special Features / Specialities	Common Usage
Dial-up Modem	Telephone line (analog)	Up to 56 Kbps	Converts analog telephone signals to digital; requires dialling each time	Old Internet connections, basic email and browsing
DSL (Digital Subscriber Line) Modem	Telephone line (digital)	1–50 Mbps	Allow to use Internet and telephone simultaneously; always-on connection	Home broadband and small offices
Cable Modem	Coaxial cable (TV line)	10–200 Mbps	Uses cable TV network; faster than DSL; supports multiple users	Home and office broadband
Wireless Modem	Cellular network (3G/4G/5G)	10–100 Mbps	Portable; uses SIM card; connects through mobile data networks	Laptops, mobile hotspots, remote locations
Fiber Optic Modem (ONT – Optical Network Terminal)	Fiber optic cable	Up to 1 Gbps	Uses light signals; very high speed; stable and reliable connection	Modern fiber broadband and institutions
Satellite Modem	Satellite communication	1–50 Mbps	Works in remote areas without cable or telephone lines	Rural or remote Internet access

Thus, the modem ensures smooth communication between digital devices and analog communication channels. When you connect your home computer or Wi-Fi router to the Internet, the modem communicates with your Internet Service Provider (ISP). For example:

- In a broadband connection, your modem converts signals coming through ISP's cable line into digital form.
- In a fiber network, the Optical Network Terminal (ONT) acts as a modem.

6.4.2 Ethernet Card (Network Interface Card (NIC))

Ethernet card, also known as Network Interface Card is a network adaptor used to set up a wired or wireless network. It acts as an interface between the computer and the network. A Network Interface Card is a hardware component that allows a computer or device to connect to a network (such as a LAN or the internet). It provides the physical interface for data communication between the device and the network medium. Most modern computers, laptops, and even servers come with a built-in NIC on the motherboard. It is a printed circuit board mounted on the motherboard of a computer as shown in Figure 6.15. The Ethernet cable connects the computer to the network through NIC. Ethernet cards can support data transfer speed between 10 Mbps and 1 Gbps (1000 Mbps). Each NIC has a MAC (Media Access Control) address, which helps in uniquely identifying the computer on the network.



Figure 6.15 - Network Interface Card

6.4.3 Repeater

A repeater is a network device used to extend the coverage area of a network by amplifying and retransmitting signals. In data communication, when signals travel through cables or wireless channels, they gradually weaken (called attenuation). A repeater strengthens these weak signals and sends them forward without changing the original data.

- When data is transmitted over long distances, the signal loses strength.
- The repeater receives the weak signal, amplifies/regenerates it, and then retransmits it to the next segment of the network.
- A repeater does not filter or manage traffic, it only boosts signals.

6.4.4 Switch

A switch is a network device used to connect multiple computers or devices within a Local Area Network (LAN) (Figure 6.16). It plays a central role in a Local Area Network (LAN). A switch is called an intelligent device because it learns the MAC address of each device connected to its ports and stores them in a MAC address table. When data is received, the switch extracts the destination address from the data packet and checks the destination MAC address and forwards the data only to the specific port where the destination device is connected, instead of sending it to all ports. It can forward multiple packets at the same time. A switch does not forward the signals which are noisy or corrupted. It drops such signals and asks the sender to resend it.

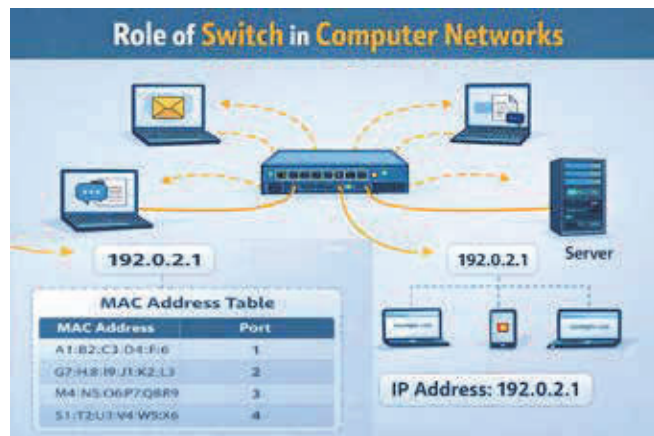


Figure 6.16 - switch

- Each device connected to a switch has a unique MAC (Media Access Control) address.
- When a device sends data, the switch reads the MAC address of both the sender and the receiver from the data packet .
- It then forwards the data only to the target device, not to every device in the network.
- This reduces network traffic and prevents data collisions.

Thus, switches make networks faster and more efficient.

An IP Address (Internet Protocol Address) is a unique numerical identifier assigned to each device connected to a network. It helps in identifying and locating devices so that data can be sent and received correctly over the Internet or a local network. IP addresses can be IPv4 (e.g., 192.168.1.1) or IPv6, which provides a much larger address space.

A **server** is a powerful computer that provides services, resources, and data to other computers in a network, called clients. It manages network resources and responds to requests sent by clients. Main roles of a server include: Data storage and sharing, Resource management, User authentication, Application hosting, Network security, Backup and recovery.



6.4.5 Router

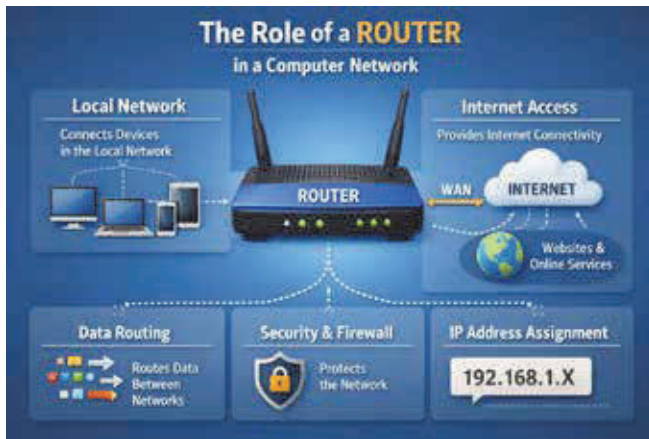


Figure 6.17 - Router

A router is a network device that connects multiple networks together (Figure 6.17). For example, connecting a Local Area Network (LAN) to the Internet. Every device in a network has an IP (Internet Protocol) address, and the router uses these addresses to forward data to the correct destination. The main function of a router is routing that is, receiving data packets, analysing destination IP address in

the packet header, and sending them to the appropriate network based on their destination IP address. A router can be wired or wireless. A wireless router can provide Wi-Fi access to smartphones and other devices. Usually, such routers also contain some ports to provide wired Internet access. Nowadays, 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.

6.4.6 Gateway

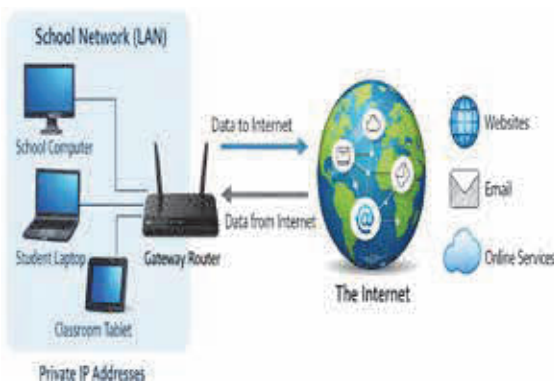


Figure 6.18 - Gateway

A gateway is a network device that connects two different networks that use different communication protocols. It acts as an entry and exit point (a “gate”) for data travelling between networks. In simple terms, a gateway works like a translator it receives data from one network, converts it if necessary, and sends it to another network in a format it can understand (Figure 6.18).

For example, when a Local Area Network (LAN) connects to the Internet (WAN), the gateway enables communication between two, even if they use different data formats, addressing schemes, or communication methods. At school/home, the router usually functions as the gateway, connecting your local devices (like computers, phones, and smart TVs) to the Internet Service Provider (ISP) and then to the wider Internet. Gateways are essential for data transfer, protocol conversion, and network

security, ensuring smooth communication between different systems and networks. A gateway can be implemented as software, hardware, or a combination of both. This is because a network gateway is placed at the edge of a network and the firewall is usually integrated with it.

Working of a Gateway

1. **Protocol Conversion:** The gateway translates data from one network's protocol to another (e.g., from LAN's TCP/IP to another system's proprietary protocol). This ensures that both networks can understand and exchange the data correctly.
2. **Data Handling:** When a device sends data to another network, the gateway receives the data packet, examines it, and converts it into a format that the destination network can interpret.
3. **Routing and Forwarding:** After conversion, the gateway forwards the data packet to the destination network using the correct route.
4. **Response Transmission:** When the destination device sends a response, the gateway again converts it back into the source network's format and delivers it to the original sender.



1. What are network devices?
2. Name any four common network devices.
3. What is demodulation?
4. What is attenuation?
5. Name the device that performs protocol conversion.
6. Explain the functions and importance of a Switch in a computer network.

Hub is a basic networking device used to connect multiple computers or devices in a Local Area Network. Hub acts as a central device where all computers in a network are connected using cables. When a hub receives data from one computer, it forwards (broadcasts) that data to all other connected devices, regardless of the intended recipient. Hub does not check MAC addresses, does not filter data, and does not decide the best path. It simply sends the signal to every port.



6.5 Network Topology

In a computer network, many devices such as computers, printers, and servers are connected to share data and resources. The arrangement or layout in which these devices (nodes) are connected to each other is called a Network Topology. It defines how data flows between devices and how each device communicates with the others in the network. It can be physical (actual layout of cables) or logical (the way data moves between devices). Number of computing devices are connected together to form a Local Area Network (LAN), and interconnections among millions of LANs form the Internet. Common network topologies are bus, star, ring, mesh and tree.

6.5.1. Bus Topology



Figure 6.19 - Bus Topology

Bus topology is a network structure in which all computers (nodes) are connected to a single central cable, called the bus (Main cable) (Figure 6.19). Data sent by any computer travels along this main cable and is received by all devices, but only the intended receiver accepts it. Terminators are placed at both ends of the cable to prevent signal reflection. When a device sends data, the signal moves in both directions along the bus. Every device checks the destination MAC/IP address in the data packet header, only the correct device processes the data, while others ignore it.



Firewall is a network security device or software that acts as a barrier between a trusted internal network (like your home or company network) and an untrusted external network (like the Internet). Its main job is to control and monitor incoming and outgoing network traffic based on predefined security rules. Essentially, it decides which traffic is allowed and the traffic to be blocked to protect the network from unauthorized access, hackers, malware, and other threats.

Table: 6.5 Advantages and Disadvantages of Bus Topology

Advantages	Disadvantages
Low installation cost	Failure of main cable stops the entire network
Easy to install and configure	Performance degrades as devices increase
Requires less cable	Difficult to locate faults

Suitable for small networks	Limited cable length and number of nodes
Easy to add new devices	High data collisions
Simple network structure	Low security (data broadcast to all nodes)

6.5.2. Star Topology

In a star topology, all computers are connected to a central device (hub or switch) (Figure 6.20). Each node has a dedicated connection to the hub/switch. The central node can be either a broadcasting device, means data will be transmitted to all the nodes in the network (hub), or a unicast device (switch) an intelligent device that identifies the destination using MAC addresses and forwards data only to the intended node. When a computer sends data, it first reaches the central hub or switch.



Figure 6.20 - Star Topology

Table 6.6 Advantages and Disadvantages of Star Topology

Advantages	Disadvantages
Easy to install and manage	Failure of central hub/switch affects the entire network
Easy to add or remove devices	Requires more cable than bus topology
Failure of one node does not affect others	Higher cost due to hub/switch
Better performance (less data collisions) and High security (data sent to specific node)	Central device dependency
Easy fault detection and troubleshooting	Limited scalability based on hub/switch capacity

6.5.3. Ring Topology

In this topology, all computers (nodes) are connected in a closed loop (ring) (Figure 6.21). Each device is connected to exactly two neighbouring devices, and data travels in one direction (or sometimes both directions in dual-ring networks).



Figure : 6.21 - Ring Topology

Working of Ring Topology

A small data packet called a token circulates around the ring.

- A device can send data only when it has the token
- The data travels from one node to the next until it reaches the destination
- This reduces data collisions

Table 6.7 Advantages and Disadvantages of Ring Topology

Advantages	Disadvantages
Data travels in an orderly manner	Failure of one node or cable affects the entire network
Reduces data collisions	Difficult to troubleshoot
Equal access to all nodes	Adding or removing nodes is difficult
Better performance than bus topology under heavy load	Network downtime during reconfiguration
Predictable data transfer	More expensive than bus topology
Suitable for small to medium networks	Less flexible compared to star topology

6.5.4. Mesh Topology

Mesh topology is a network structure in which each device (node) is connected to every other device in the network through dedicated



Figure : 6.22 - Mesh Topology

point-to-point links (Figure 6.22). This provides multiple paths for data transmission, making the network highly reliable. Data can travel through many possible routes from source to destination. If one link fails, data is automatically routed through an alternate path, ensuring uninterrupted communication.



SSL (Secure Sockets Layer) is a security protocol that helps protect data transmitted over the Internet. It ensures that the information sent between a web server and a web browser remains private and secure.

When a user connects to a secure website, the browser checks the site's SSL certificate. If the certificate is valid, an encrypted connection is established. The website address will begin with **https://** instead of **http://**, and a lock icon appears in the address bar.

Table: 6.8 Advantages and Disadvantages of Mesh Topology

Advantages	Disadvantages
High reliability and fault tolerance because each node has a dedicated link	Very expensive as it require large number of cables and ports
Failure of one link does not affect other connections	Complex installation and configuration
Provides high security and privacy since data travels through dedicated paths	Difficult to maintain and manage
Easy to detect and isolate faults	Requires more space for cables and hardware
Supports heavy data traffic without congestion	Not suitable for small networks
Good performance due to direct point-to-point connections	High power consumption

6.5.5. Tree Topology

Tree topology is a hierarchical network structure that combines the features of star and bus topologies (Figure 6.23). In this topology, computers are arranged in the form of a tree-like structure, with a main cable connecting multiple star networks. Each group of computers is connected to a hub or switch, forming a star. These hubs/switches are then connected to a main backbone cable.

*Figure : 6.23 - Tree Topology***Table 6.9 Advantages and Disadvantages of Tree Topology**

Advantages	Disadvantages
Easy to expand by adding more nodes or segments	Failure of the main backbone affects the entire network
Well-structured and hierarchical, making the management easier	More expensive due to extra cabling and networking devices
Suitable for large organizations and wide networks	Complex installation compared to bus and star topologies
Easy fault detection and isolation in individual segments	Requires careful configuration and maintenance
Supports scalability and network growth	Performance decreases as more nodes are added
Combines advantages of star and bus topologies	High dependency on central switch or main cable

6.5.6. Hybrid Topology

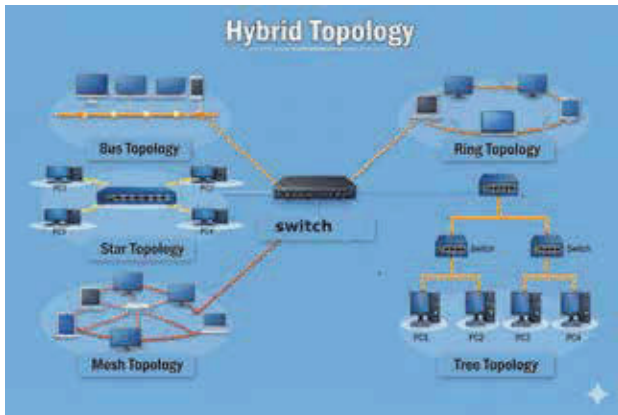


Figure 6.24 - Hybrid topology

Hybrid topology is a network structure that is formed by combining two or more different network topologies, such as star, bus, ring, or mesh (Figure 6.24). It is designed to take the advantage of the strengths of each topology while minimizing their weaknesses. Different segments of the network use different topologies.

For example, one department may use star topology, while another uses bus topology. These segments are interconnected using routers, switches, or main cable links, allowing smooth communication across the network.

Table: 6.10 Advantages and Disadvantages of Hybrid Topology

Advantages	Disadvantages
Highly flexible as it combines two or more different topologies	Very expensive as it require more devices and complex cabling
Reliable and fault tolerant; failure in one segment does not affect others	Complex design and implementation
Scalable and easy to expand according to network requirements	Requires skilled network administrators
Suitable for large organizations and enterprises	Difficult to manage and maintain
Good performance due to optimized use of different topologies	High maintenance cost
Can be customized to meet specific needs	Not suitable for small networks



1. The structure that defines how computers are connected in a network is called _____.
2. In a bus topology, all devices share a single _____.
3. The central device used in star topology is _____ or _____.
4. In a ring topology, data travels in _____ direction(s).
5. A mesh topology provides a dedicated _____ between each pair of nodes.
6. The tree topology is a combination of _____ topologies.

6.6 The Internet

The Internet is a global network that connects millions of computers and devices worldwide, allowing them to communicate and share information (Figure 6.25). It works by transferring data between devices through a series of interconnected networks using standard communication protocols like TCP/IP. When a user tries to open a website, his/her device (client) sends a request through an Internet Service Provider (ISP) to a web server that hosts the website. The request travels across several network devices such as routers, switches, and gateways that direct the data to its destination. Once the web server receives the request, it sends the required data (such as web pages, images, or videos) back to the user's browser in the form of data packets. These packets travel across multiple paths and are reassembled at the destination device to display the complete webpage. The Transmission Control Protocol (TCP) ensures that all packets arrive correctly, while the Internet Protocol (IP) manages addressing and routing. In short, the Internet functions as a massive system of connected networks that enables seamless communication, data sharing, and access to online resources worldwide.



Figure 6.25 - Internet

6.6.1 Protocols for Data Communication

Data communication refers to the exchange of data between two or more computing devices through some transmission medium. For successful communication, both sender and receiver must follow a set of rules that ensure accurate, reliable, and efficient data transfer. These rules are called Protocols and Standards. A network protocol defines how data is formatted, transmitted, and received over the network. Protocols ensure that computers around the world can understand and exchange information just like people use a common language to talk any one.

1. HTTP (Hyper Text Transfer Protocol)

HTTP (Hyper Text Transfer Protocol) is the standard communication protocol used for transferring web pages over the Internet. It defines how messages are formatted and transmitted between a web browser and a web server. When you type a web address beginning with `http://`, your browser uses HTTP to request a web page from the server, which then sends back the page content. Data is not

encrypted, so it's not secure for confidential information. HTTP is the communication protocol of the World Wide Web, enabling browsers and servers to exchange information across the Internet.

2. HTTPS (HyperText Transfer Protocol Secure)

HTTPS (HyperText Transfer Protocol Secure) is the secure version of HTTP used for transferring data between a web browser and a web server. It uses SSL (Secure Socket Layer) to encrypt data, ensuring that information like passwords, banking details, and personal data cannot be stolen or read by hackers. HTTPS keeps online communication safe and confidential, especially on websites that handle sensitive information (Online banking, shopping, email services, etc.).

Example: <https://onlinesbi.sbi.bank.in/>

3. FTP (File Transfer Protocol)

FTP (File Transfer Protocol) is a standard protocol used to transfer files between a client and a server over the Internet or a local network. It allows users to upload, download, rename, or delete files on a remote computer. Data is not encrypted in standard FTP, so it can be insecure. (Secure versions include FTPS and SFTP.)

4. TCP/IP (Transmission Control Protocol / Internet Protocol)

TCP/IP (Transmission Control Protocol/Internet Protocol) is a set of communication rules used to connect computers on the internet and other networks. It defines how data is divided into packets, transmitted, and reassembled at the destination. TCP ensures reliable data transfer by checking for errors, maintaining sequence, retransmitting lost packets and confirming delivery, while IP handles addressing and routing the packets to the correct destination. Together, they enable communication between different types of devices and networks. TCP/IP is the foundation of the Internet and supports many protocols like HTTP, FTP, SMTP, and DNS for various online services and applications.

5. SMTP (Simple Mail Transfer Protocol)

The Simple Mail Transfer Protocol (SMTP) is a standard communication protocol used to send and route emails over the Internet. It works between email clients and mail servers. SMTP ensures email messages are properly transferred to the recipient's mail server. SMTP uses port 25 by default and relies on TCP to provide reliable data transfer. When a user sends an email, SMTP transfers it from the sender's device to the outgoing mail server,

which then forwards it to the recipient's mail server. The recipient retrieves the email using protocols like POP3, IMAP. SMTP commands and responses help to manage the communication between mail servers, ensuring that each part of the email such as headers, sender, recipient, and contents are correctly delivered. SMTP is essential for email transmission, providing a fast, reliable, and standardized method of sending messages across networks.

A. POP3 (Post Office Protocol Version 3)

The Post Office Protocol Version 3 (POP3) is an Internet standard protocol used by email clients to retrieve messages from a mail server. It uses port 110 by default (or port 995 for secure connections). POP3 allows users to download emails from the server to their local computer, enabling them to read messages offline. When a user accesses their email through a POP3 client, the messages are typically downloaded and then deleted from the server, freeing up storage space. However, some configurations allow messages to remain on the server for backup or access from multiple devices. POP3 is simple and efficient, making it suitable for users who check email from a single device. Although newer protocols like IMAP offer more advanced features, POP3 remains a reliable and widely supported method for retrieving emails.

B. IMAP (Internet Message Access Protocol)

The Internet Message Access Protocol (IMAP) is an email protocol used by clients to access and manage messages stored on a mail server. It typically uses port 143 (or port 993 for secure connections). Unlike POP3, IMAP allows users to view and organize emails directly on the server without downloading them, enabling access from multiple devices such as computers, tablets, and smartphones. With IMAP, users can create folders, mark messages as read or unread, and synchronize changes across all devices connected to the same account. This makes it ideal for modern email usage where messages need to be available anytime and anywhere. IMAP ensures that emails remain stored on the server until the user deletes them, providing better management and backup. It is widely used by email services like Gmail, Outlook, and Yahoo Mail for efficient and flexible email access.

C. MIME Protocol (Multipurpose Internet Mail Extensions)

MIME is a protocol that enhances the standard email format to support the transmission of multimedia content such as images, audio, video, and application files. Originally, email systems

supported only plain ASCII text, which made it impossible to send non-textual data or files. With MIME, an email can contain multiple parts in a single message, such as text, images, and attachments, separated by boundaries. This allows modern email clients to display rich content consistently across platforms. MIME ensures compatibility, reliability, and versatility in email communication, making it essential for everyday internet use. Without MIME, sending multimedia content via email would not be feasible.

6. DNS (Domain Name System)

The Domain Name System (DNS) is a hierarchical and distributed naming system used to translate human-readable domain names (like `www.google.com`) into IP addresses (like `142.250.190.68`) that computers use to identify each other on the Internet (Figure 6.26). Without DNS, users would need to remember long numerical IP addresses to access websites. DNS works like a telephone directory of the Internet. When a user types a website address in a browser (`example.com`), the DNS starts by checking the local DNS cache to see if the IP address is already stored. If not found, the request goes to a DNS resolver, which contacts the root DNS server, then the Top-Level Domain (TLD) server and finally the authoritative DNS server that holds the actual IP address of the domain. Once the IP address is found, it is sent back to the user's device, allowing the browser to connect to the web server and display the requested website. This process happens in milliseconds. DNS makes Internet browsing simple, fast, and user-friendly by converting easy-to-remember names into machine-readable addresses automatically.

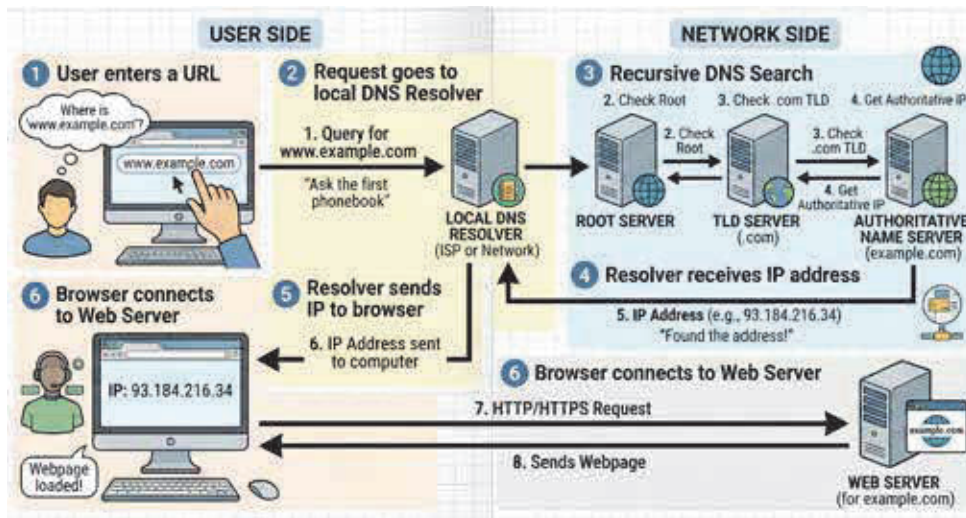


Figure 6.26 - Domain Name System

Table:6.11 Protocols for Data Communication

Protocol	Full Form	Purpose / Used For
HTTP	Hyper Text Transfer Protocol	The foundation of data exchange on the web; used to fetch HTML documents and resources.
HTTPS	Hyper Text Transfer Protocol Secure	The secure version of HTTP; uses encryption (SSL/TLS) to protect data like passwords and credit card info.
FTP	File Transfer Protocol	A standard way to transfer files between a client and a server over a network.
TCP/IP	Transmission Control Protocol / Internet Protocol	The fundamental suite of protocols that governs how data is packeted, addressed, transmitted, and routed across the internet.
SMTP	Simple Mail Transfer Protocol	Used specifically for sending and routing outgoing email from a client to a server.
POP3	Post Office Protocol version 3	Used to retrieve emails from a server. It usually downloads the mail to your device and deletes it from the server.
IMAP	Internet Message Access Protocol	A more modern way to retrieve email that synchronize messages across multiple devices, keeping them on the server.
MIME	Multi-Purpose Internet Mail Extensions	An extension that allows email to support non-text content like images, audio, video, and attachments.
DNS	Domain Name System	The "phone book" of the internet; it translates human-friendly URLs (like google.com) into IP addresses(like 142.250.190.68).

Network protocols are like rules that computers use to communicate. Each protocol has a specific purpose some send web pages, some transfer files, others handle emails. Together, they make the Internet efficient, secure, and reliable.

6.7 Network-Based Services

Network-based services are the various functions and resources provided over a computer network, especially the Internet (Figure 6.27). These services allow users and devices to communicate, share data, access resources, and perform online activities such as browsing, emailing, and file sharing. They are the foundation of modern digital communication and play



Figure : 6.27 - Network-Based Services

a vital role in education, business, entertainment, and government operations. Network-based services refer to the applications and processes that make use of a network (usually the Internet or a private network used within an organization) to deliver specific functions or information to users. Some common examples of network-based services include E-mail service, file transfer service, web service, video conferencing service, cloud storage service, social networking service etc.

Table 6.12 Network-based services

Network Service	Description	Purpose / Use	Example
E-mail Service	Allows users to send and receive electronic messages over a network	Communication and information sharing	Gmail, Outlook, Zoho Mail
File Transfer Service	Enables users to upload or download files from one computer to another	Sharing documents and data	FTP, Google Drive
Web Service (www)	Provides access to websites and web pages using browsers	Information access	Google, Wikipedia
Chat Service	Allows real-time text communication between users	Instant messaging	WhatsApp, Telegram
Video Conferencing Service	Supports live audio and video communication over a network	Online meetings and classes	Zoom, Google Meet
Remote Login Service	Allows users to access another computer remotely.	Remote system management	Telnet, SSH
Cloud Storage Service	Stores data on remote servers accessible via the Internet.	Data backup and storage	Dropbox, OneDrive
Streaming Service	Delivers audio or video content over the Internet without downloading.	Entertainment and learning	YouTube, Netflix
Online Transaction Service	Enables financial transactions through networks.	Banking and shopping	Online banking, Amazon
Social Networking Service	Helps people to share content and interact online.	Social interaction	Facebook, Instagram

Network-based services are essential in every field from education and healthcare to entertainment and commerce, making digital collaboration and information sharing faster and more reliable.



1. When you open a website, your device sends a request to a _____.
2. The full form of WWW is _____.
3. Each web page has a unique address called _____.
4. A school wants to conduct online classes with live interaction between teachers and students. Which network-based service should be used and why? Give one example.
5. A customer purchases products from an online shopping website and pays digitally. Which network-based service enables this activity and what is its purpose?

◆ ————— Summary ————— ◆

A computer network is a collection of interconnected devices that communicate with each other to share data and resources. This chapter begins with introducing to basic concepts of computer networks and the importance of data communication in today's digital world.

This chapter explains the components of data communication and different modes of data transmission across devices. It further classifies networks into PAN, LAN, MAN, and WAN based on their geographical spread and applications.

Students learn about transmission media, which are categorized into guided (wired) and unguided (wireless) media, along with their characteristics, advantages, and limitations. The role of essential network devices such as NIC, Hub, Switch, Router, Modem, and Firewall is discussed to show how data flows securely and efficiently across networks.

Different network topologies are explained with their structure, benefits, and drawbacks, helping students understand how computers are physically and logically connected. The chapter also introduces the Internet, its structure, and the role of the Domain Name System (DNS) in World Wide Web (WWW).

Finally, the chapter covers important network protocols and commonly used network-based services such as e-mail, file transfer, remote login, video conferencing, and cloud services, highlighting their significance in everyday life.

Overall, this chapter builds a strong foundation in computer networking concepts and prepares students to understand real-world communication systems.



1. A bank customer makes an online money transfer using a mobile app. Explain how data communication takes place in this process and why protocols are important for ensuring secure and accurate data transfer.
2. A teacher sends study materials from a laptop to students using an online platform. Identify the components of data communication involved (sender, receiver, message, transmission medium, and protocol) and explain the role of each in this situation.
3. A keyboard sends input signals to a computer, but the computer does not send data back through the keyboard. Which transmission mode is used in this case? Explain.
4. A company needs a fast, secure network connection between two buildings located close to each other and wants minimal signal interference. Which type of guided transmission media should be used and why?
5. A mobile user makes a video call while traveling in a moving vehicle without using any cables. Which type of unguided transmission media is used in this situation and why is it suitable?
6. A company has two office branches in different cities and wants them to communicate securely over the Internet as if they were on the same private network. Which network device is required to connect these networks and what role does it play?
7. A small office wants to set up a network where all computers are connected to a central device, and failure of one computer should not affect others. Which network topology is most suitable and why?
8. A military communication system requires a highly reliable network where data can still be transmitted even if one or more links fail. Which network topology should be used and explain how it ensures reliability.
9. A student accesses a secure website where personal information must be protected from unauthorized access. Which network protocol ensures secure communication in this situation and why is it important?



7

Cyber Security



“Information security is not a destination it’s a journey”

- Bill Cheswick

7.1 Introduction

In the 21st century, digital technology is an inevitable part of human life. We use digital devices right from early morning, starting with the alarm clocks on our mobile phones. We depend on computers and internet for online classes. Digital technology helps us to make digital payments. We use it for social media and online shopping. E-governance services also depend on computers and internet. Computers and internet influence almost every activity we do. A large part of modern life is based on digital connectivity. Therefore, the term *cyber space* has become very prominent. In today’s interconnected world, cyberspace plays a significant role in almost all aspects of life. Cyberspace refers to interconnected networks of computers and electronic systems. It is the environment where online communication, data exchange, and digital interactions take place. These include networks, hardware, software, data, and users.

In Kerala, digital literacy is relatively high, and initiatives such as Akshaya centres, e-District, K-FON and online education platforms have brought technology closer to ordinary citizens. However, along with convenience, technology also brings risks. Cyber criminals misuse digital systems to steal money, data and personal information. *Cyber security* is the practice of protecting computers, networks, devices and data from digital attacks, unauthorized access, damage, or theft. It involves using technologies, rules, and

safe practices to keep information secure while using the internet and digital devices. This makes cyber security one of the most important subjects for students to understand. This chapter introduces cyber security concepts, cyber crimes, preventive measures, ethics and laws in a structured and easy-to-understand manner.

7.2 Vulnerability in Cyber Space

Vulnerability in cyberspace refers to weaknesses or gaps in computers, networks, software or user practices that can be exploited by cyber attackers. These vulnerabilities make digital systems and data more exposed to cyber threats and security risks.

7.2.1 Causes of Vulnerability in Cyber Space

Vulnerabilities in cyberspace arise due to various factors that create weaknesses in digital systems, making them susceptible to attacks (Figure 7.1). They may come from software flaws, human mistakes, network setup errors and weak hardware. Some of the various causes and categories of vulnerability in cyber space are given below.



Figure: 7.1 - Causes of Vulnerability

7.2.1.1 Software Vulnerabilities

- **Bugs and coding mistakes:** Computers run on software (programs and apps), but those programs can have errors (called bugs) or flaws in their code. For example, a game or phone app might crash unexpectedly – that bug could let attackers sneak in.

- **Outdated Software:** If software is not updated, it will have security problems. Hackers can easily attack old software.
- **Digital footprints:** A digital footprint is the data trail created when a person uses digital devices and the internet. They show where you have been online. Some examples are search history (Google searches, browsing history, queries), social media activity (posts, likes, comments and profile info), cookies (small files tracking your browsing habits), IP Address, location data etc.

7.2.1.2 Human Factors

- **Excessive user privileges:** When users are given more access rights than necessary, a compromised account can cause serious damage, such as deleting data or changing system settings.
- **Weak passwords:** Easy passwords can be guessed quickly. Using the same password everywhere is also unsafe.
- **Lack of awareness:** When users do not know the basic cyber safety rules, they make mistakes which in turn cause attacks. People sometimes click on fake links or open unknown files. This allows hackers to enter the system.
- **Insider threats:** Users who already have access to the system may misuse it or make mistakes.

7.2.1.3 Network and Infrastructure Issues

- **Misconfigured firewalls and routers:** A firewall is like a security guard for a network. It controls incoming and outgoing network traffic. If firewall rules are weak, outdated or wrongly configured, attackers can bypass them and gain unauthorized access to the systems. Similarly, routers or network devices with default or open settings can be exploited. For instance, if a home router still has its factory password (A factory password is the default password that a device or software comes with from the manufacturer), attackers can easily take over the home network.
- **Poor security settings:** If security settings are not set properly, hackers can enter the network easily.
- **Untrusted devices or networks:** Using public Wi-Fi or unknown devices can expose data to hackers.
- **Misconfigured cloud storage:** Cloud storage becomes vulnerable when access rights are wrongly set (for example, public access enabled by mistake). It may lead to data leaks, unauthorized access, or data theft.

7.2.1.4 Hardware Vulnerabilities

- Outdated or unsupported devices: Just like software, hardware (physical devices) also needs updates. Old computers, phones or gadgets that never get security patches can be weak. An unsupported hardware can lead to security loopholes that hackers can exploit and older devices often lack modern security features, making them easy targets. Think of an old smart phone that can no longer install security updates – hackers know its old weaknesses.
- Insecure BIOS settings: The BIOS (Basic Input Output System) controls basic hardware operations. If BIOS passwords are not set or security features are disabled, attackers can change system settings, install malware or access data before the operating system starts.



1. How does human error lead to cyber attacks?
2. “Using untrusted networks can expose personal data.” Justify this statement.
3. Small files used by websites to track browsing habits are called ____
4. Explain weak passwords with an example.
5. What are bugs? How can they lead to cyber attacks?
6. What is meant by insider threat?
7. What is a digital footprint?
8. Which hardware component controls basic system operations before operating system loads?

7.3 Meaning and scope of cyber security

Along with many benefits of technology, involves serious dangers too. Some people misuse computers and internet to commit crimes such as stealing money, sensitive data and personal details. Because of these growing threats, it is very important to learn about cyber security and understand how to be safe in the digital world. Cyber security involves protecting the information and systems from malignant activities in cyber space. Cyber security is not only about computers. It is also about people, data and safe behavior in cyberspace.

Three basic principles of cyber security, known as the CIA Triad: (figure 7.2)

1. **Confidentiality:** Information should be accessible only to authorized persons.
2. **Integrity:** Information should not be altered without permission.
3. **Availability:** Information and services should be available when needed.

For example, when a student in Kerala logs into the Kerala Infrastructure and Technology for Education (KITE) portal to access digital textbooks, cyber security ensures that only the student can access their account (confidentiality), the content is not tampered with (integrity), and that website remains accessible during exam seasons (availability).

The three core goals of information security confidentiality, integrity and availability ensure that information is protected from unauthorized access, remains accurate and trustworthy and is available to authorized users when needed. To effectively achieve these goals, additional security objectives such as Authentication, Authorization, Non-repudiation and Access Control play a vital supporting role. *Authentication* is the process of verifying the identity of a user or system before granting access to resources. It answers the question “Who are you?”. For example, username and password, One-Time Password (OTP), biometrics such as fingerprint or face recognition. It is considered as an extended security goal because it ensures only legitimate users gain access to systems and data. *Authorization* determines what an authenticated user is allowed to do after logging into a system. For example, a student can view marks but cannot edit them; only an administrator can add or delete user accounts. It answers the question “What are you allowed to do?”. *Access Control Systems* are mechanisms that enforce authentication and authorization rules to protect resources. They ensure that only authorized users can read, modify or use digital resources. They work by verifying identity (authentication) and then granting permission (authorization) based on predefined rules. For example, a customer logs in using username, password, and OTP (authentication), the system allows



Figure 7.2 - CIA Triad

customer to view balance and transfer money but blocks access to admin functions (authorization). *Non-repudiation* ensures that a user cannot deny performing an action after it has occurred. Non-repudiation provides proof of origin and action, which is important in legal and financial systems like digital signatures on emails, online transaction records etc.



1. Which of the following is NOT part of the CIA Triad?
 - a) Confidentiality b) Integrity
 - c) Authentication d) Availability
2. When a server is hacked, the computer acts as
 - a) Tool b) Target c) Storage device
 - d) Communication medium
3. How is cyber security related to human behavior?

7.4 Importance of Cyber Security

The importance of cyber security cannot be ignored, as digital technology becomes a part of every aspect of our lives. India is a rapidly growing digital economy. Digital India initiatives have promoted online banking services, UPI payments, Aadhar based services, e-Governance, digital classrooms etc. In Kerala, services such as SPARK (Service and Payroll Administrative Repository for Kerala), online PSC application and exams, virtual courts, e-Health Kerala, THANAL, FRIENDS and K-SMART depend heavily on secure digital systems. Cyber security is crucial as it significantly impacts both individuals and organizations across various sectors. A single cyber attack can affect thousands of citizens. Hence, cyber security is a collective or shared responsibility. It is very important because it is necessary to:

1. Protect sensitive data (like Aadhar, PAN, bank account details etc.).
2. Prevents online fraud and scams.
3. Protects government databases.
4. Builds trust and reputation in digital services.

7.5 Different types of Cyber Security

Cyber Security encompasses a variety of specialized fields, each focusing on protecting different aspects of digital systems and data. Here are some of the main types of Cyber Security: (Figure 7.3)

- a. **Network Security:** It involves securing computer networks from intruders like hackers and malware. It uses tools such as intrusion detection systems, firewalls and Virtual Private Networks (VPNs). These tools protect the network's perimeter. For example, a firewall blocks unauthorized access while allowing outward communication.
- b. **Application Security:** Application Security focuses on protecting software and devices from cyber threats. It includes writing secure programs, checking applications for weaknesses and using firewalls to protect them from attacks. Regular updating and patching of software is a common and important application security practice.
- c. **Information Security:** It is also known as data security and it is dedicated to protecting data's confidentiality, integrity and availability. It uses techniques such as encryption, access control and prevents data loss to safeguard the data stored and in transit. Data encryption is the process of converting data into a coded form to protect it from unauthorized access. Only a person with the correct key or password can convert it back to its original form. For instance, if you encrypt sensitive data files on a corporate network, it will prevent data leakage in case of a breach.
- d. **Endpoint Security:** Endpoint security is a security approach used to protect user endpoints such as PCs, smart phones and tablets from viruses, malware, and unauthorized access. The tools used for endpoint security range from basic antivirus software to advanced solutions like EDR (Endpoint Detection and Response). When implemented properly, endpoint security ensures that each device is individually protected, preventing attackers from using a compromised endpoint to move to other systems within the organization.
- e. **Operational security:** It handles the processes and decisions for managing and protecting data assets, it covers managing permissions and access. The aim is to ensure that confidential information is only granted to authorized users. It involves options such as using multi-factor authentication to access corporate databases.



Figure 7.3 - Types of Cyber Security

- f. **Cloud Security:** This type of security protects data, applications and infrastructures involved in cloud computing. It often involves a shared responsibility between cloud service providers and clients to secure the data hosted in the cloud.



1. Protecting smart phones and laptops comes under
 - a) Network security b) Application security
 - c) Endpoint security d) Cloud security
2. Which type of security focuses on protecting data confidentiality and integrity?

7.6 Common Cyber Security Threats

Understanding the nature of common Cyber Security threats is essential to safeguard your digital assets. Here are some of the most prevalent types of security threats that you might encounter:

- a. **Phishing Attacks:** In such attacks, the attackers pretend to be the trusted sources and deceive individuals into sharing confidential and sensitive information (*Figure 7.4*). The kind of information they can be phishing for could include a wide range – such as credit card numbers, social security numbers, passwords etc. Phishing often involves sending deceptive emails that appear to be from reputable sources. These emails often prompt users to enter personal information at a fake website.



Figure. 7.4 - Phishing

- b. **Malware:** Malware is a term for malicious software. It is any program designed to damage or exploit any programmable device, service or network. Various forms of malware exist, such as Virus, Trojan, Worms, Spyware etc.

- c. **Ransomware:** This type of malware takes control of a victim's files or systems and asks for a ransom to regain access. Ransomware can prevent you from accessing the system or encrypt your data. Once your system is infiltrated, you will not be able to access it without a decryption key. In most cases the attackers promise to provide this key only upon payment.
- d. **Man-in-the-Middle attacks (MitM):** In these attacks, the perpetrator covertly intercepts (Figure 7.5). They may modify the communications between two parties who think they are directly connecting with each other. This can happen on unsecured Wi-Fi networks, where attackers intercept data being transferred from a device to the network.



Figure. 7.5 - Man in the Middle Attack

- e. **Denial-of-Service attacks (DoS):** These attacks block users from accessing a machine or network by temporarily or permanently stopping the internet-based services of the host (Figure 7.6). Attackers typically target web servers of high-profile organizations such as banks, media, government sites etc.

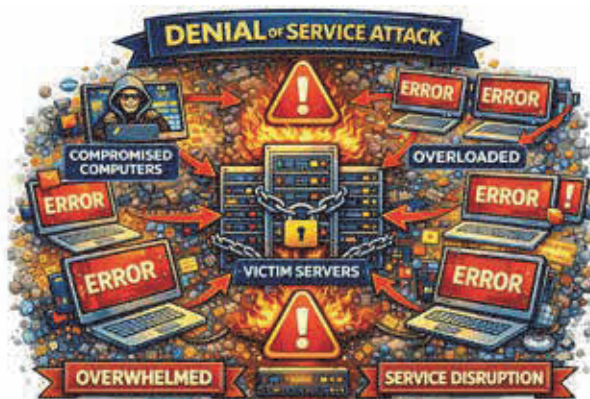


Figure 7.6 - Denial of Service Attack

- f. **Zero-Day Exploits:** These occur when hackers take advantage of a security vulnerability before developers become aware of it or before a patch is available. Since there is little or no time to fix the flaw, zero-day exploits are extremely dangerous and can cause serious damage to systems and data.
- g. **Insider threats:** Sometimes, the risk comes from within. Disgruntled employees, contractors, or anyone with inside access can misuse their credentials to steal data or disrupt the systems.

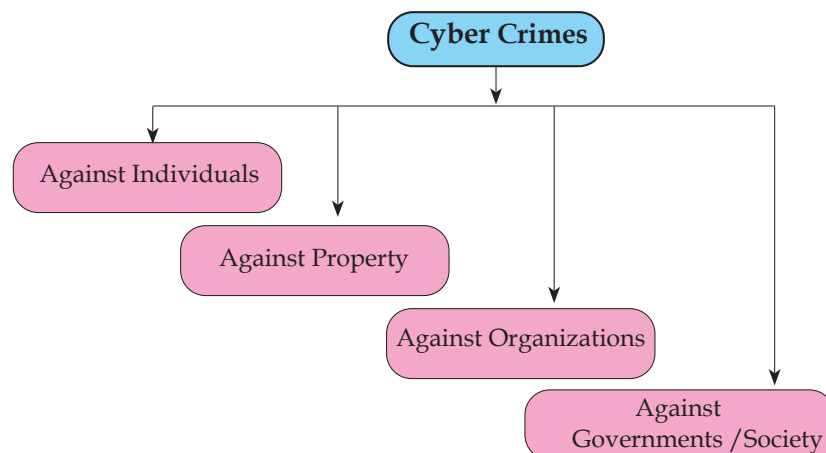
7.7 Cyber Crimes

Cyber crime refers to illegal activities carried out using computers, mobile devices or the internet. In cyber crimes, digital technologies are the target, the tool or both. These crimes affect individuals, organizations or even governments.

Computers can play different roles in cyber crimes depending on how they are used

- **Computer as a target:** The computer or network itself is attacked. Example: Hacking a server, spreading viruses, damaging data.
- **Computer as a tool:** The computer is used to commit the crime directly. Example: Sending phishing emails, online fraud, spreading malware.
- **Computer as a storage device:** Computers are used to store illegal data. Example: Storing stolen data, pirated software, and illegal content.
- **Computer as a communication medium:** Computer acts as a medium to trap victims or it is used to communicate or coordinate criminal activities. Example: Planning crimes through emails, chats, or social media.

Cyber crimes can be classified into four major categories:



7.7.1 Against Individuals

These types of cyber crimes target a person's privacy, identity or mental wellbeing. It includes

- **Identity theft:** Identity theft is a serious cyber crime that involves stealing and misusing personal information. Awareness, safe online behavior and security measures are essential to protect oneself from identity theft. Using stolen or fake identity information to commit fraud is referred to as Identity fraud or theft.
- **Email spoofing:** A spoofed email is one in which the e-mail header is forged so that the mail appears to originate from one source but actually it has been sent from another source.
- **Phishing:** Phishing is a type of identity theft that occurs online. It is an attempt to acquire information such as usernames, passwords, credit card details etc by pretending as the original website, mostly that of banks and other financial institutions. Phishing websites have URLs and home pages similar to their original ones. The act of creating such a misleading website is called *spoofing*. These frauds use this information to steal money. Phishing is currently the most widespread financial threat on the Internet.
- **Pharming:** In this case the users are redirected to fake websites often by manipulating DNS servers or exploiting website vulnerabilities. While phishing is like a direct attack, pharming is more like a trap door.
- **Spamming:** Spamming means sending multiple copies of unsolicited mails or mass e-mails such as chain letters.
- **Cyber defamation:** This occurs when defamation takes place with the help of computers and/or the internet. For instance, someone publishes defamatory matter about another person on a website or sends e-mails containing defamatory information.
- **Cyber stalking:** Cyber stalking is the activity of repeatedly harassing, threatening or following an individual's activity over internet.
- **Cyber bullying:** Cyber bullying uses digital platforms to insult, or embarrass someone, especially repeatedly.

7.7.2 Against Property

These crimes involve damage or theft of digital or physical property.

- **Credit card fraud:** As the name suggests, this is a fraud activity by using a credit card. This generally happens if someone gets to know the card number or other details or if the card gets stolen.
- **Intellectual Property crimes:** These include
 - i. **Software piracy:** Illegal copying of programs, distribution of copies of software.
 - ii. **Copyright infringement:** Using copyrighted material without proper permission.
 - iii. **Trademarks violations:** Using trademarks and associated rights without the permission of the actual holder.
 - iv. **Theft of computer source code:** Stealing, destroying or misusing the source code of a computer.
- **Internet time theft:** This occurs when internet hours paid by one person are used by an unauthorized individual.

7.7.3 Against Organizations

Cyber crime against organization include unauthorized accessing of computer, denial of service, computer contamination virus attack, e-mail bombing, salami attack, logic bomb, Trojan horse and data diddling.

- **Unauthorized accessing of computer:** Accessing the computer/network without permission of the owner. The intention behind this illegal access may be for:
 - i. **Changing/deleting data:** Unauthorized modification of data.
 - ii. **Computer voyeur:** The criminal reads or copies confidential or proprietary information, but the data is neither deleted nor changed.
- **Denial of service:** When the Internet server is flooded with continuous bogus requests so as to denying legitimate users to use the server or to crash the server.
- **Malware attacks:** Malware means malicious software designed to harm or exploit computer network or device.

The following are the different types of malware:

- i. **Virus:** A computer virus is a computer program that can replicate itself by attaching it to other files or programs. It needs a host to spread and can cause all sort of trouble like data loss or system crashes.
- ii. **Worm:** A worm is like a virus, but it can spread itself

without the need of a host. It is self-contained and can propagate through networks, causing chaos and consuming resources.

- iii. **Trojan:** An unauthorized program which functions from within that seem to be an authorized program, thereby concealing what it is actually doing. Once inside, it lets attackers access your system, steal data, or cause chaos.
- iv. **Ransomware:** It is like a digital kidnapper. It encrypts your files and demands payment (ransom) for the decryption key. Payment does not guarantee to getting the data back.
- v. **Spyware:** It is a software that secretly monitors and collects your data, like browsing habits or keystrokes. It's like having a digital spy in your system.
- **Email bombing:** Sending large number of mails to the individual, company or mail servers and thereby ultimately crash the system.
- **Salami attack:** Salami Attack is a type of cyber crime in which a large amount of money or data is stolen by taking very small amounts repeatedly, so that each individual theft goes unnoticed – just like slicing thin pieces of salami.. These attacks are used for the commission of financial crimes.
- **Logic bomb:** It is an event dependent program. As soon as the designated event occurs, it crashes the computer, release a virus or any other harmful possibilities.
- **Data Diddling:** This kind of attack involves altering raw data just before it is processed by a computer and then changing it back after the processing is over.

7.7.4 Against Governments / Society

- **Forgery:** Currency notes, revenue stamps, mark sheets etc. can be forged using computers, high quality scanners and printers.
- **Cyber terrorism:** Use of computer resources to intimidate or coerce people and carry out the activities of terrorism.
- **Web jacking:** Hackers gain access and control the website of another, and they may change the content of website for fulfilling political objective.



1. Altering data before processing and restoring it later is called
 - a) Logic bomb
 - b) Data diddling
 - c) Email bombing
 - d) Salami attack
2. Flooding a server with fake requests to crash it is known as _____ attack.
3. Malware that needs a host file to spread is called _____.

7.8 Cyber Security Controls

Cyber security controls are tools or steps that help to keep computers and data safe. There are three main types of controls.

1. **Preventive controls:** In this step we try to stop security problems *before they happen*.
 - **Use strong passwords:** Instead of common words or personal information, use unique, complex passwords and change passwords regularly.
 - **Keep software updated:** Regularly update operating system, applications and patch vulnerabilities.
 - **Multilevel authentication:** It is also called Two-Factor Authentication (2FA). It requires more than one method to verify the identity. Add extra security along with your password. Example: password and OTP or password and biometric verification.
 - **Firewalls:** A firewall monitors and controls incoming and outgoing network traffic to block unauthorised access. It acts like a protective wall between a computer or network and the internet. The firewall allows the safe data to pass through and blocks unauthorized or harmful access, helping to prevent hacking and cyber attacks.
2. **Detective Controls:** They are designed to identify and alert when something suspicious is happening or already occurred.
 - **Antivirus software:** Antivirus software is a security program used to protect computers and mobile devices from viruses and other harmful software. It detects, prevents, and removes malware such as viruses, worms, trojans, and spyware. Antivirus software scans files, emails, and websites to find threats and blocks them before they

can damage the system or steal data. Keeping antivirus software installed and regularly updated helps to ensure better protection against new cyber threats.

- An alert or email when someone logs into your social media account from a new device.

3. **Corrective Controls:** corrective controls help fix problems *after they happen*

- Reinstalling a clean operating system or app after a virus infection. Resetting a hacked password and removing unauthorized users from your account.

However, Cyber security focuses on prevention rather than cure. Cryptography, hashing and digital signatures are three important concepts that work together to ensure secure communication, data integrity, authentication and trust in digital systems.

Cryptography is a technique of protecting information by converting it into a secret code. In cryptography, readable data (plain text) is encrypted into an unreadable form (cipher text) so that unauthorized people cannot understand it. Only authorized users with the correct key can decrypt the data and read the original information. Cryptography is widely used to ensure data security, privacy and safe communication on the internet, such as in online banking and secure messaging.

Hashing is a process that uses a mathematical function to convert data into a fixed-length value called a *hash*. This hash acts as a unique digital fingerprint, which helps verify that the data has not been changed.

Digital signatures use cryptography and hashing to prove the identity of the sender and to ensure that a message or document cannot be denied later. Cryptography, hashing and digital signatures together form the foundation of modern cyber security and are widely used in applications such as secure websites, online transactions, e-governance services and electronic documents etc.



1. Antivirus software is an example for -----.
a) Preventive control b) Detective control
c) Corrective control d) Legal control
2. What is multilevel authentication?
3. Mention any two characteristics of a strong password.

7.9 Cyber Security Do's and Don'ts

Do's	Don'ts
Use strong passwords with letters, numbers and symbols.	Do not share passwords or OTPs with anyone. Do not save passwords on public or shared computers. Do not use the same password for multiple accounts. Do not over share personal information online.
Verify links and emails before clicking or downloading anything.	Do not click on unknown or suspicious links in emails or messages.
Download apps and files only from trusted sources and keep them updated.	Do not download apps or files from untrusted websites and donot ignore software and app updates.
Lock your devices using PIN, password or biometrics.	Do not leave devices unlocked or unattended.
Use antivirus software and scan your device regularly.	Do not ignore antivirus warnings or alerts.
Log out after using email or social media on shared computers.	Do not stay logged in on shared or public computers.
Use secure networks for online transactions.	Do not use public Wi-Fi for banking or sensitive activities.
Report suspicious messages to a trusted adult or authority.	Do not respond to online threats or fake prize messages.
Backup important data like assignments and projects.	Do not rely on a single copy of important data.
Use privacy settings on social media accounts.	Do not accept friend requests from unknown people.

7.10 Legal and ethical behaviour in cyber space

Cyberspace has become an important part of our daily life. We use the internet for learning, communication, shopping, banking and entertainment. While using cyberspace, it is very important to follow legal and ethical behavior to ensure safety, trust and respect for everyone.

- **Legal behavior in cyberspace:** It means to follow the laws and rules related to the use of computers and the internet. This includes not hacking systems, not spreading viruses, not stealing personal or financial information and not sharing illegal or copyrighted content. Cyber laws are made to protect users, data

and digital systems. When people follow these laws, it helps to prevent cyber crimes and ensures justice in the digital world. The Information Technology Act, 2000 was passed by the Indian Parliament to provide legal recognition to electronic records and digital signatures. This act was amended by The Information Technology Amendment Bill, 2008 to strengthen cyber security laws and address emerging cyber threats.

- **Ethical behavior in cyberspace:** It refers to do what are morally right activities while using technology, even when there is no direct law. Ethical users respect others' privacy, do not bully or harass people online, do not spread fake news, and use digital resources honestly. Ethical behavior also includes giving credit to original creators and using the internet responsibly. Together, legal and ethical behavior helps to create a safe, respectful and reliable cyberspace. When users act responsibly, cyberspace becomes a positive environment that supports learning, innovation and healthy communication.

◆———— Summary ———◆

The chapter explains the growing importance of protecting digital systems in today's technology-driven world. It introduces cyberspace and highlights how daily activities such as education, banking, governance and communication depend on computers and the internet. The chapter discusses vulnerabilities in cyberspace, explaining how software flaws, human errors, weak passwords, misconfigured networks and insecure hardware create opportunities for cyber attacks.

Students learn the meaning and scope of cyber security, including the roles of computers in cyber crimes and the importance of the CIA Triad – Confidentiality, Integrity and Availability, along with supporting concepts like authentication, authorization, access control and non-repudiation. The chapter also explains the importance of cyber security for individuals, organizations and governments, especially in protecting sensitive data and building trust in digital services.

Different types of cyber security such as network, application, information, endpoint, operational and cloud security are introduced. Common cyber threats and cyber crimes are explained with examples, covering crimes against individuals, property, organizations and society. The chapter also emphasizes cyber security controls, safe practices through Do's and Don'ts, and the need for legal and ethical behavior to ensure a safe and responsible cyberspace.



1. Define cyberspace and explain its components.
2. Explain why cyber security has become an important subject for students.
3. What is meant by vulnerability in cyberspace? Explain vulnerability in cyberspace with suitable examples.
4. Explain insecure BIOS settings as hardware vulnerability.
5. Describe the role of digital footprints in increasing cyber risks.
6. How does outdated software increase security risks?
7. Give one real-life situation from daily life where lack of cyber security can cause serious problems. Explain.
8. How can vulnerability in a system be misused by cyber attackers? Explain with one example.
9. How does human error lead to cyber attacks? Explain with examples.
10. "Using untrusted networks can expose personal data." Justify this statement.
11. Define non-repudiation. Why is it important?
12. Describe the CIA Triad and explain its importance in cyber security.
13. Explain authentication and authorization. List any two differences between them.
14. Explain how confidentiality, integrity and availability are ensured when students access online education portals.
15. What is malware? Name any two types of malware.
16. How do phishing emails deceive users?
17. What happens to data during a ransomware attack?
18. Explain Man-in-the-Middle attacks.
19. Describe Denial-of-Service attacks and their effects.
20. A hacker exploits a software flaw before a patch is released. What is this attack called? Why is it dangerous?
21. What are intellectual property crimes? Mention any two examples.
22. Explain cyber crimes against individuals with suitable examples.
23. Explain email bombing, salami attack, logic bomb and data diddling.
24. Describe the working and importance of firewalls in preventing cyber attacks.
25. What is multilevel authentication? Explain with suitable examples.
26. Explain antivirus software, its functions and the need for regular updates.
27. Explain cryptography, hashing and digital signatures and their roles.
28. Explain the three types of cyber security controls with suitable examples.
29. Explain how cryptography helps in online banking and secure messaging.
30. Mention any two characteristics of a strong password.
31. List and explain any five cyber security Do's and Don'ts
32. Why should students avoid public Wi-Fi for online transactions? Explain
33. Explain legal and ethical behavior in cyberspace.



8

AI and its Applications



"The science of today is the technology of tomorrow."

Edward Teller

8.1 Introduction to Artificial Intelligence (AI)

Artificial Intelligence (AI) is a branch of computer science that enables machines to perform tasks which typically require human intelligence. In other words, AI is the simulation of human intelligence in machines that are designed to think, make decisions, predict the future based on patterns in data, learn and improve through training, feedback or updated data. AI system can perform tasks that typically require human intelligence. AI is used to build agents like robots or virtual assistants that can mimic or simulate human behavior and make decisions on our behalf. A machine can be called artificially intelligent when it exhibits the following features.

- Thinks and learns like a human
- Mimics human intelligence
- Improves on its own
- Detects patterns and solves real world problems
- Comprehends language and images
- Predicts and takes decisions on its own



Figure 8.1 - Features of AI and Simulation of Human Intelligence

8.1.1 History and Evolution of AI

Artificial Intelligence or AI, is a part of computer science that tries to make machines smart enough to do tasks that normally need human intelligence—like understanding speech, recognizing pictures or making decisions.

The story of AI started in the 1950s when scientists first dreamed of building thinking machines. In 1956, a famous meeting called the Dartmouth Conference officially created the name “Artificial Intelligence.”

In the initial period, AI worked by following strict rules written by programmers. These early systems could only solve very simple problems. During 1960s and 1970s, progress was slow because the computers were weak and there was not much data to use.

Turing Test (1950)

In 1950, a British scientist named Alan Turing asked the big question - Can machines think?

He proposed the “Turing Test”, where a human interacts with both a machine and a person via text. If the human cannot differentiate between the machine and the other person, the machine passes the test, which suggests its human-like conversational behavior. The context is quite similar to chatting with a robot, like Siri or Alexa. Can the chatbot fool you into thinking it’s a person?

Dartmouth Conference (1956)

The term “Artificial Intelligence” was coined at the Dartmouth Conference, by John McCarthy, one of the organizers, marking the birth of AI as a field of study.

Early pioneers discussed autonomous automata. The concept of intelligent robots was later popularized by fictional characters like R2-D2 (1977).

Expert Systems (1980)

Experts created systems that could mimic human decision-making. These “expert systems” used rules and knowledge from humans to help make decisions.

Early systems helped doctors diagnose diseases by asking questions.

In the current context think of how apps like “WebMD” ask you about your symptoms to help figure out what might be wrong with you. Earlier systems, like MYCIN (1970s), were rule-based. Modern tools, like current symptom checkers, utilize Machine Learning.

Modern AI (2000s- present)

Starting in the 2000s, AI got smarter with the rise of Machine Learning (ML) and deep learning. AI systems began learning from huge amounts of data, becoming more capable. Neural networks, loosely inspired by the structure of the human brain, power these advancements. AI can now recognize faces in photos (like Facebook tagging your friends) or help self-driving cars understand the world around them.

Another example is that of Netflix recommending movies based on what you’ve watched. AI is learning from patterns in the data!

8.1.2 Applications of AI in Daily Life

AI surrounds us more than we realize:

Smartphones: AI is integrated into many smart-phone features like improving image quality in photos, filtering spam messages and powering personal assistants.

Health care: AI can help with medical diagnosis, personalize treatment plans and empower health trackers.

Transportation: From navigation apps optimizing routes to advanced driver-assistance systems in cars and eventually self-driving vehicles, AI is changing how we travel.

Finance: AI helps with fraud detection, provides personalized financial advice and aids in automated trading.

Education: AI can provide personalized learning experiences, automate grading and enhance education tools.

8.1.3 AI Timelines

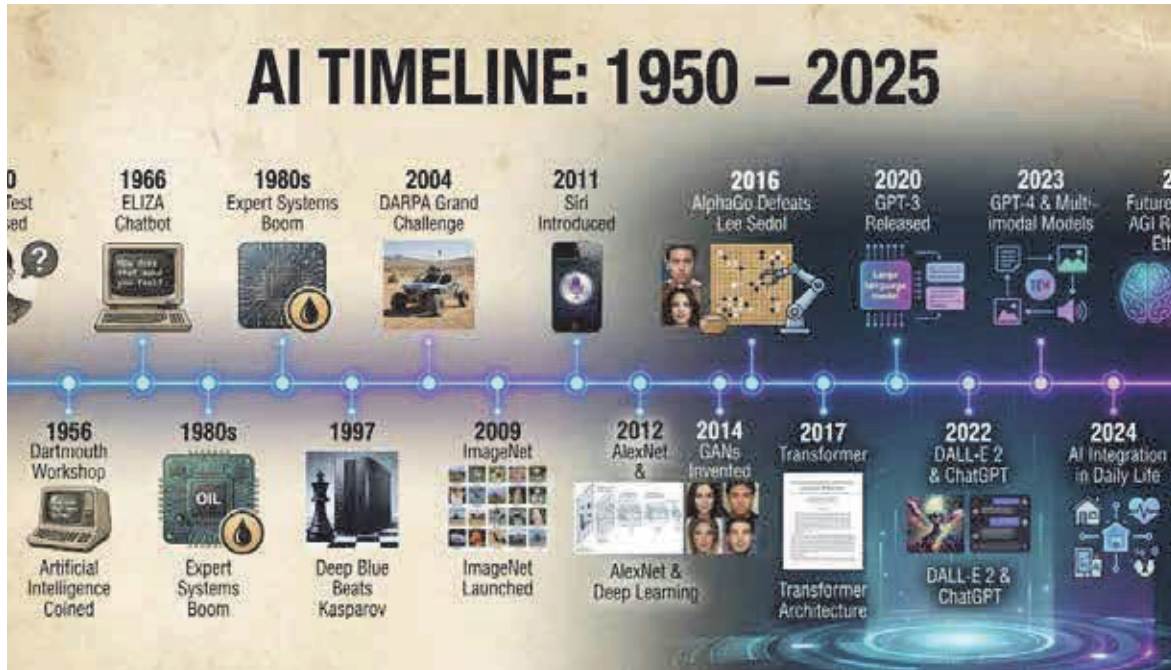


Figure 8.2 - AI History Timeline

Real-Life Issues of Artificial Intelligence

Artificial Intelligence is helpful in many real-life situations, but it also has certain limitations. For example, when an AI system is asked to generate a recipe for an eggless vegetable omelet, it may suggest ingredients that are not easily available or give steps that are not practical in real kitchens. Since AI does not have human senses like taste or smell, it cannot judge flavor or texture. Sometimes, the information provided may be incomplete or unclear due to limited or biased data. This shows that AI can support humans, but it cannot completely replace human experience and judgment.

Ethical Considerations

Although Artificial Intelligence (AI) offers many benefits and makes work easier, it is very important to use it in a responsible and ethical way. Ethical considerations help ensure that AI is used for the good of society and does not cause harm to individuals or communities.

Privacy

AI systems often collect and process large amounts of personal data, such as names, photos, locations or online activities. This information must be protected carefully. People's personal data should not be misused, shared without permission or exposed to security risks.

Bias

AI systems make decisions based on the data they are trained on. If the data contains bias or unfair patterns, the AI may also produce unfair or discriminatory results. Therefore, AI systems should be designed and tested to ensure that their decisions are fair and do not discriminate against any individual or group.

Accountability

When an AI system makes a decision, it should be clear who is responsible for that decision. Developers, companies or organizations using AI must take responsibility for its actions and outcomes, especially when mistakes or harmful decisions occur.

Employment

AI and automation can replace certain jobs that are repetitive or monotonous. While this can increase efficiency, it may also lead to loss of work for some workers. Society must adapt by creating new job opportunities and helping people learn new skills to work alongside AI technologies.



1. Describe Artificial Intelligence in your own words.
2. Who is regarded as the father of AI?
3. List any two areas where AI is applied today.
4. Why is ethical AI use important?
5. Give two examples of AI applications in daily life.
6. Explain why responsible use of AI is necessary.
7. Identify three examples of AI you have used today and describe their functions.
8. Discuss in groups how AI can improve life in your city or community.

8.2 Machine Learning – Teaching Machines to Learn

Machine Learning is a subset of Artificial Intelligence that enables computers to learn from data without being explicitly programmed. Instead of giving step-by-step instructions, we provide examples, and the machine recognizes patterns, draws insights, and makes predictions.



Figure 8.3 - Machine Learning

Consider it like teaching a child, you give numerous examples of apples and oranges rather than going into the details of each fruit. With time, the child will be able to recognize a new orange or apple on their own. In a similar vein, machine learning enables computers to “learn” from examples.

Importance of Machine Learning

Through Machine learning computers learn from data without being explicitly programmed. This has many applications like making search engines more accurate, powering personalized recommendations etc.

8.2.1 Types of Machine Learning

Based on how the learning process works, machine learning is mainly classified into three types: Supervised Learning, Unsupervised Learning and Reinforcement Learning.

1. Supervised Learning

In supervised learning, the computer is trained using labeled data. This means that each input is already associated with a correct output or answer. During training, the model learns the relationship between the input data and the expected output. Once trained, the model can use this knowledge to make predictions on new and unseen data.

Some examples of this are:

- Identifying ripe and unripe mangoes: The model is shown many images of mangoes, where each image is labeled as “ripe” or “unripe.” By studying these examples, the computer learns the differences and can correctly classify new mango images.

- Email spam detection: Emails are labeled as “spam” or “not spam.” The system learns from these labeled emails and can later identify whether a new email is spam or not.

2. Unsupervised Learning

In unsupervised learning, the computer works with unlabelled data, meaning there are no predefined answers. The model explores the data on its own to discover hidden patterns, similarities, or structures. It often groups or clusters data items that are similar to each other. Some instances of unsupervised learning are:

- Grouping students based on learning styles: The system analyses student behavior and performance to form groups with similar learning patterns.
- Clustering customers in e-commerce: Online shopping platforms group customers based on their shopping behavior, preferences and purchase history to understand trends and improve recommendations.

3. Reinforcement Learning

In reinforcement learning, the machine learns through trial and error. It performs actions and receives rewards for correct decisions and penalties for wrong ones. Over time, the system improves its strategy to maximize rewards. This type of learning is commonly used in game development, robotics and control systems.

Here are some examples:

- Training a robot to navigate a maze: The robot receives a reward when it reaches the endpoint. By repeating the task, it learns to choose better paths in future attempts.
- AI playing chess or Go: The AI learns by playing many games against itself. By winning or losing games, it improves its strategy and becomes more skilled over time.

8.2.2 Key Concepts in Machine Learning

Data Quality: The data should be accurate, complete and relevant. The machine picks up knowledge from its observations. Predictions will be unreliable if the training data is inaccurate, biased or lacking. For example, a facial recognition system trained mostly on images of people from one ethnicity may fail to recognize people from other ethnicities.

Training and Testing: Training data is the initial dataset used to train an AI model. Training data helps an AI model to learn and identify the patterns or performs particular tasks.

Testing data is used to evaluate the performance of the AI model.

Feature Selection: It is about choosing the most relevant features from a dataset to build a model. This helps to improve the model's performance, reduces complexity and makes it easier to understand. Predicting student performance based on attendance, study hours and previous grades, is an example.

8.2.3 Applications of Machine Learning

Healthcare: Predicting disease outbreaks, diagnosing illnesses from medical images.

Finance: Detecting fraud in credit card transactions.

E-commerce: Product recommendation systems like Amazon or Flipkart.

Education: Adaptive learning platforms that adjust lessons based on student performance.

Agriculture: Monitoring crop health and predicting yield.



1. How is machine learning different from traditional programming?
2. What are the three main types of machine learning?
3. Why is high-quality data important for machine learning?
4. Can you think of one example where reinforcement learning is used?
5. Define Machine Learning in your own words.

8.3 Artificial Neural Networks – The Machine Brain

Artificial Neural Networks (ANNs) are computational models inspired by the human brain. Just as the brain have billions of interconnected neurons that process information, ANNs use artificial neurons to analyze data, detect patterns and make predictions. These Neurons are sometimes called nodes.

A fundamental component of contemporary AI, ANNs are particularly useful in speech processing, image recognition and natural language comprehension applications. Like humans, they allow machines to learn from experience.

8.3.1 Structure of a Neural Network

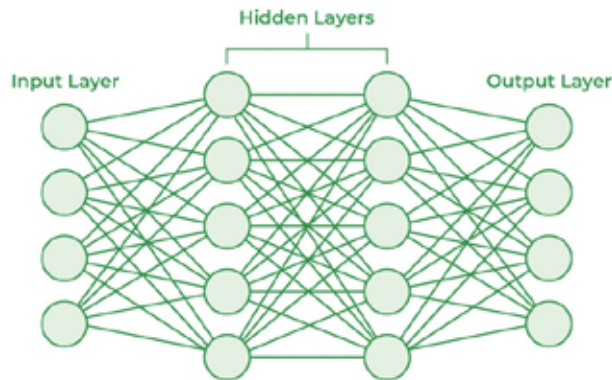


Figure 8.4 – Neural Network Structure

Input Layer: Input layer serves as the entry point for the data passing it to the subsequent layers. Each neuron in this layer represents specific feature. In image recognition, each pixel's intensity may be an input neuron.

The Hidden Layers: The Hidden layers perform the crucial computations, transforming the input data through various functions. There can be multiple hidden layers depending on the complexity of the task.

Output Layer: Output layer generates the final prediction or output. This layer communicates the outcome based on the processed information from the hidden layers.

Weights: Each connection has a strength (called a weight) that influences how signals propagate; these weights are what the network learns and tweaks during training.

For example, a neural network trained on handwritten digits can recognize the number '5' even if written in different styles or slanted positions. The network learns features such as curves, edges and orientation in the hidden layers.

8.3.2 How Neural Networks Learn

Forward Propagation: The input layer sends data to the output layer via the hidden layers.

Error Calculation: The difference between the predicted output and the actual output is measured.

Backpropagation: The network adjusts the weights of connections to minimize error.

Iteration: This process repeats until the network achieves maximum accuracy. The network improves with experience, similar to how humans learn by trial and error.

8.3.3 Applications of Neural Networks

Image Recognition: Facebook tagging, Google Photos, autonomous vehicles.

Speech Recognition: Converting spoken language into text, e.g., Siri, Google Assistant.

Medical Diagnosis: Detecting tumors or abnormalities in X-rays, MRI scans, or CT scans.

Natural Language Processing: Chat bots, translation tools, sentiment analysis.

Gaming: AI learns to play complex games like chess or Go.



1. How is an artificial neural network inspired by the human brain?
2. What are the main layers of a neural network?
3. How does a neural network improve its predictions?
4. Give two practical applications of neural networks.
5. Define an Artificial Neural Network.

8.4 Natural Language Processing (NLP) - Understanding Human Language

Humans communicate with each other using natural languages such as Malayalam, Hindi and English. For computers to interact effectively with people, they must be able to understand, interpret and generate human language. Natural Language Processing (NLP) is a branch of Artificial Intelligence that makes this possible. It helps machines bridge the gap between human communication and computer understanding.

In simple terms, NLP enables computers to:

- Read and understand text, such as emails, messages or documents
- Listen to and interpret speech, allowing voice-based interaction
- Translate between languages, helping people communicate across different regions
- Generate meaningful responses, making conversations with machines more natural and useful

NLP plays an important role in many everyday applications like chatbots, voice assistants, language translation tools and spam detection systems.



Figure 8.5 - NLP Functions and Applications

8.4.1 Natural Language Processing

It is about how computers can understand, interpret and generate human language. It involves steps like breaking down text and understanding the meaning of words and sentences.

NLP combines computer science, linguistics and machine learning. The process involves:

- **Tokenization:** Breaking text into smaller units like words or sentences.
- **Parsing:** Analyzing the grammatical structure of sentences.
- **Semantic Analysis:** Understanding the meaning of words and sentences.
- **Response Generation:** Producing meaningful and context-aware outputs.

8.4.2 Applications of NLP

- **Speech Recognition:** It turns spoken words into text, which allows devices to understand. Example: Voice-to-text typing, voice commands in smart phones.
- **Machine Translation:** It is instant translation of language, bridging communication gaps. It translates text from one language to another. Example: Google Translate.
- **Chatbots and virtual assistants:** It uses NLP to simulate human conversations. It answers user queries automatically. Example: Banking chatbots, customer support bots.
- **Sentiment Analysis:** Detects positive, negative or neutral sentiment. For example, social media posts, reviews or feedback.

- **Content generation:** Involves AI creating new text. For example, write articles, poetry etc.

8.4.3 Large Language Models (LLMs)

Modern NLP systems use Large Language Models (LLMs), such as GPT, which are trained using massive datasets to understand context and generate meaningful responses.

NLP is used in spam detection to analyze email content.

Siri, Alexa and Google Assistant use NLP to understand spoken commands and respond intelligently.



1. Why is NLP important in human-computer interaction?
2. Name two applications of NLP you use in daily life.
3. What is the role of Large Language Models in NLP?
4. Why is it important to address biases in NLP systems?
5. Define Natural Language Processing.

8.5 Prompt Engineering – Talking to Smart Machines

As AI language models become more common, it is important to know how to communicate effectively with them. Prompt Engineering is the practice of designing clear, precise and effective instructions (called prompts) to get the best responses from AI systems. Think of a prompt as a question or instruction you give to an AI. The quality of the output largely depends on how well the prompt is formulated.

8.5.1 Prompt Engineering

Prompt engineering is important as it directly affects the quality and relevance of the output. AI helps in saving time, improving accuracy and achieving desired results.

Poorly worded prompts can produce irrelevant or incorrect outputs.

Example:

Prompt Type	Response Quality
"Write about pollution."	"Vague, may produce a generic response"
"Write five sentences explaining how air pollution affects human health with examples."	"Specific, detailed, relevant response"

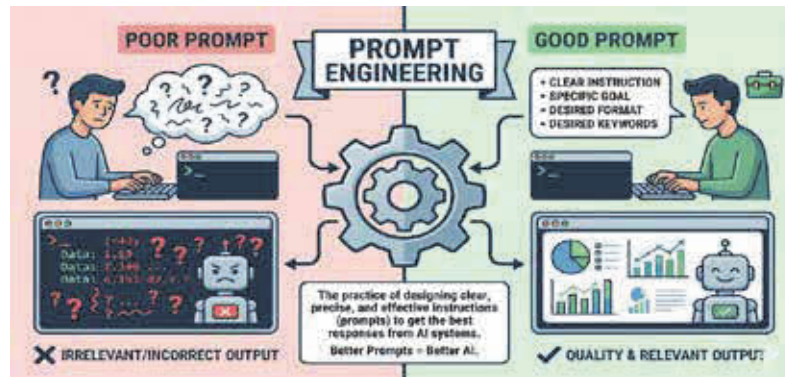


Figure 8.6 - Prompt Engineering Flow

Techniques for Effective Prompting

Be specific: goes beyond clarity, this involves precisely defining the scope and direction of the response you seek. Example: “List three advantages and two disadvantages of AI in education.”

Providing context: can include the background of the task, the target audience, or any prior information the AI needs. Example: “As a teacher, explain how AI can help students improve reading”

Defining output requirements: ensures the AI conforms to specific criteria like the number of words, whether to use bullet points etc. This helps in achieving a polished result.

Example: “Write a 100-word story suitable for class 11 students about robots in space exploration.”

Using examples: is particularly useful for complex tasks or creative content. It demonstrates the desired tone, structure, and detail.

Ethical Guidelines for Prompt Engineering

- Do not ask AI to generate personal or sensitive information.
- Avoid spreading misinformation.
- Do not use AI for harmful or dishonest purposes.
- Promote learning, creativity and responsible use of AI.

8.5.2 Applications for Prompt Engineering

- **Education:** Prompt engineering transforms education by facilitating personalized learning. Designing prompts to explain concepts, summarize content or create quizzes.
- **Content Creation:** Prompt engineering amplifies creative potential by enabling efficient and versatile generation of diverse content- writing articles, blogs, stories or poetry.

- **Programming:** Prompt engineering acts as a powerful tool for code development and comprehension. It generates code snippets or debugging solutions.
- **Data Analysis:** Prompt Engineering is a beneficial tool for optimizing data interpretation and reporting. AI tools can be used to summarize trends or generate visualization from data.



1. What is a prompt?
2. Why is the quality of a prompt important in AI interaction?
3. Mention two ethical principles to follow while using AI tools.
4. How can prompt engineering save time and improve accuracy in tasks?
5. Define Prompt Engineering in your own words.
6. Give an example of an effective prompt.
7. Explain one application of prompt engineering in education.

8.6 Robotics – Machines That Can Act

Robotics is a branch of science and engineering that combines mechanical design, electronics and Artificial Intelligence (AI) to build machines that can sense, decide and act in the real world. Robots can perform repetitive, dangerous or precise tasks, reducing human effort and risk.

To put it simply, a robot is a programmable machine that interacts with its surroundings and acts either fully or partially on its own. Robotics has various applications. In industry, robots are used for manufacturing, assembly and packaging. In medicine, they assist with surgeries, rehabilitation and patient care



Figure 8.7 - Robotic System Components

Robots are also found in exploration, from outer space to deep seas, and even in our homes for cleaning and companionship. They help students to learn coding, AI and automation.

8.6.1 Components of a Robot

A typical robot has three main components:

Sensors: Helps to perceive the environment. They can detect light, heat, sound, motion or pressure.

Eg: Obstacle detection sensors in autonomous vehicle.

Controllers: They can process information and make decisions. Controllers acts as the “brain” of the robot and make use of AI or pre-programmed instructions to decide the next action.

Actuators: Are used to perform movements or actions.

Eg: Motor, wheels, arms or grippers that carry out the robot’s tasks

Optional Components: They include Power supply (battery), communication modules, cameras or additional computing units.

8.6.2 How Robots Work

Sensing: Robots gathers data from sensors.

Processing: Controller interprets data and decides on actions.

Acting: Actuators execute movements or tasks.

Feedback: Sensors continuously provide updated information for adjustments.

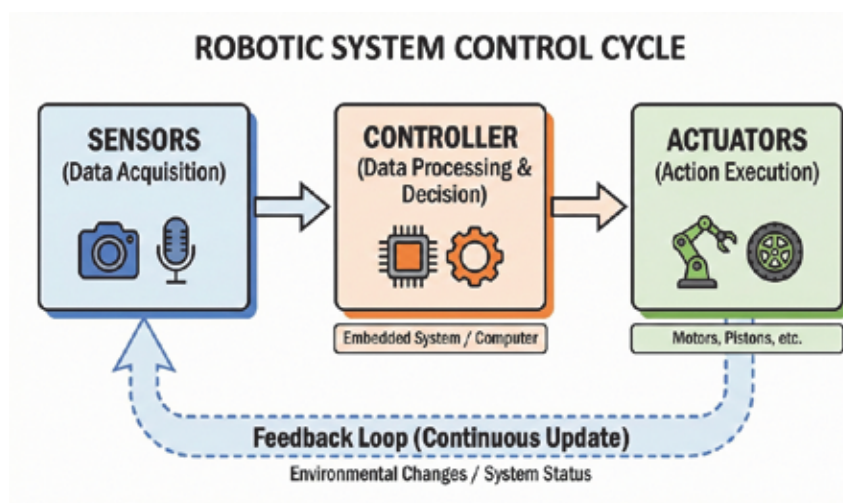


Figure 8.8 - Working cycle of Robots

8.6.3 Applications of Robotics

Manufacturing: Assembly line robots increase efficiency and precision.

Healthcare: Surgical robots assist doctors in delicate operations.

Space Exploration: Robots like NASA's rovers explore distant planets.

Disaster Rescue: Robots enter dangerous zones, e.g., collapsed buildings or nuclear sites.

Education: Robots help students to learn coding, AI and automation.

8.6.4 Ethical Considerations in Robotics

- **Employment Impact:** Automation may replace certain manual jobs.
- **Safety:** Robots must be programmed to avoid harming humans.
- **Responsibility:** Clear accountability for robot actions must be established.
- **Privacy:** Robots equipped with cameras or sensors must respect data privacy.



1. Define Robotics in your own words.
2. What are the three main components of a robot?
3. Name two real-world applications of robots.
4. Mention one ethical concern related to robotics.
5. What is a robot?
6. Identify two robots used in hospitals or factories. Describe their functions and how they improve efficiency.
7. Draw a simple block diagram of a robot showing sensors, controller, and actuators.

8.7 Computer Vision – Helping Machines See

A subfield of artificial intelligence called computer vision (CV) gives computers and other devices the ability to recognize, evaluate, and comprehend visual data from the outside world, including pictures and videos. Computer vision enables machines to “see” and comprehend their environment, much like humans do with their eyes.

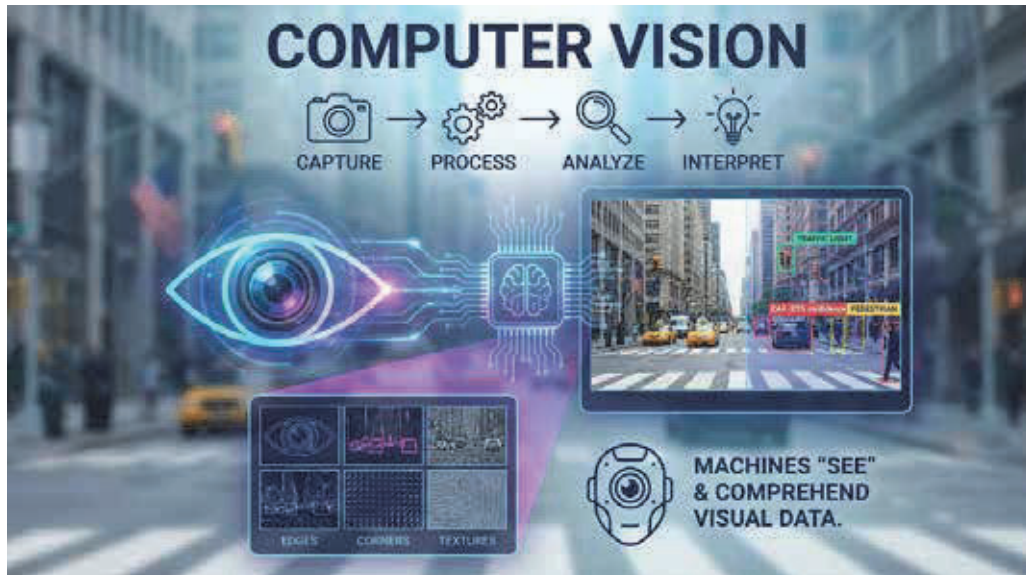


Figure 8.9- Computer Vision Process Diagram

8.7.1 Computer Vision

Computer vision enables computers to interpret and understand the visual world. It involves capturing, processing and analyzing images and videos. This works by extracting features from visual data like edges, corners or textures and then using these to identify objects, recognize patterns or track movement.

Image Acquisition: Computer captures images using cameras or sensors.

Pre-processing: Cleans the image removing noise or adjusting brightness.

Feature extraction: Identifies important characteristics like edges, corners or textures.

Recognition and classification: Use these features to identify objects or patterns.

Decision making: Uses the results to perform tasks like guiding a robot or flagging something.

Example:

A smart phone camera detecting a face to unlock the device. The camera captures the image, identifies facial features, compares it with stored data and unlocks the phone if it matches.

8.7.2 Applications of Computer Vision

Medical Imaging: Detecting tumors, fractures or anomalies in X-rays, MRI and CT scans.

Autonomous Vehicles: Self-driving cars detect obstacles, lane markings and traffic signs.

Security Systems: Face recognition for surveillance and access control.

Industrial Quality Control: Inspecting products on production lines for defects.

Agriculture: Monitoring crops, identifying diseased plants and estimating fruit ripeness.

8.7.3 Challenges and Ethical Considerations

Privacy: Surveillance cameras and facial recognition may intrude on personal privacy.

Bias: Misidentification may occur if the system is trained on non-diverse datasets.

Safety: Autonomous vehicles rely on accurate computer vision to avoid accidents.



1. What is computer vision and why is it important?
2. Name two practical applications of computer vision.
3. Why should privacy and fairness be considered when using computer vision systems?
4. How does computer vision impact healthcare and agriculture?
5. Describe a mini example of computer vision in action.
6. Find two applications of computer vision in daily life. Describe how they help humans.
7. What precautions should be taken to use computer vision ethically? Discuss in groups.

8.8 AI in Education – Personalized Learning

Artificial Intelligence is changing the way education is delivered by providing tools that adapt to the needs of each learner. Traditional teaching methods often follow a one-size-fits-all approach, but AI enables personalized learning, making education more effective and engaging. Personalized learning means that educational tools analyze a student's strengths, weaknesses, learning pace and style and then suggest exercises lessons or resources tailored specifically for that student.

8.8.1 Applications of AI in Education

1. **Adaptive Learning Platforms:** AI systems analyze student performance, adjusting pace and difficulty to maximize engagement and learning outcomes. Example: Khan Academy suggests lessons based on prior quiz performance.



Figure 8.10 - AI in Education

2. **Digital tutors and chatbots:** It offers instant feedback and personalized assistance, acting as constant learning companions. Can guide students through complex topics without waiting for teacher intervention.
3. **Learning Analytics:** It utilizes data to assess student performance. AI tracks student progress over time, highlighting areas needing improvement. Teachers receive insights to design customized teaching strategies.
4. **Automated Grading:** It streamlines the evaluation process, saving time for educators. AI can grade assignments and quizzes, reducing teacher workload.
5. **Inclusive education:** It is enhanced through AI tools that provide accessibility support, ensure student participation, and assist students with disabilities through voice recognition, text to speech or visual aids.

8.8.2 Role of Teachers in AI-Enabled Classrooms

In AI-enabled classrooms, teachers play a multifaceted role. They become facilitators, guiding students through personalized learning paths that AI suggests. They also focus on critical thinking, creativity and social skills, areas where human interaction is invaluable.

Teachers remain central to the learning process.

AI supports them by providing insights, saving time on repetitive tasks, and offering personalized content.

Teachers guide, mentor and motivate students while monitoring ethical use of AI tools.

Ethical Considerations

- **Data Privacy:** Student information must be protected.
- **Transparency:** Students should understand how AI tools make recommendations.
- **Equity:** Ensure AI systems do not favor certain groups over others.



1. How does AI support personalized learning?
2. What is the role of teachers in an AI-powered classroom?
3. Name one ethical concern while using AI in education.
4. Give an example of AI helping students with disabilities.
5. Explain how AI supports personalized learning.
6. Give one example of an AI application that helps students learn at their own pace.
7. List three ways AI can help you learn better in school.
8. Discuss how AI could help a teacher identify students who need extra help in a particular subject.
9. Suggest one AI tool or app that can make learning personalized.

8.9 Voice Assistants – Talking to AI

AI programs known as voice assistants are able to comprehend and react to spoken instructions. They do things like play music, set reminders, answer questions and control smart devices by combining speech recognition, Natural Language Processing (NLP) and cloud-based AI services.

Particularly for those with disabilities or those who would rather speak than type, voice assistants increase the accessibility, interactivity and convenience of technology.



Figure 8.11 - Voice Assistant Technology

8.9.1 How voice assistants work

Speech Recognition: It is the process where voice is converted into text. This allows the assistant to understand what you are saying.

Natural Language Processing: Helps the AI to interpret the meaning of text, considering context. AI understands the meaning and intent behind the words.

Decision Making: Happens where the assistant determines the most appropriate response or action to take on request.

Response generation: Takes place, creating a clear and personalized response. It provides spoken or written feedback.

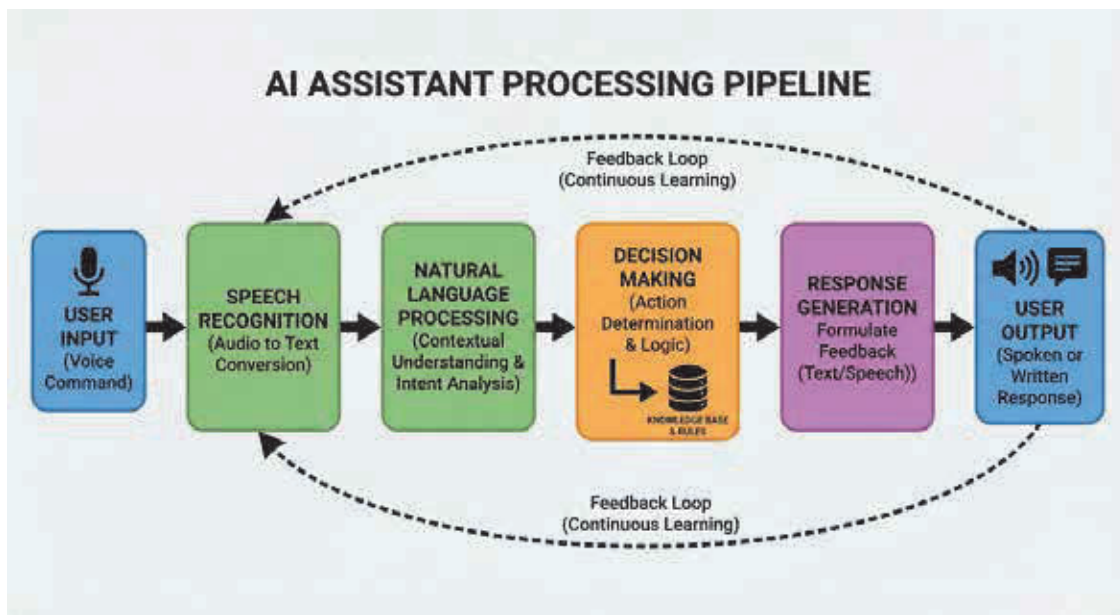


Figure 8.12 - Voice Assistant Processing

Example:

When you say, “Hey Siri, what’s the weather today?”, Siri converts your speech into text. Then it Analyses the query to understand that you want the weather report. Next it fetches weather data from the internet and finally responds aloud with, “Today’s temperature is 28°C”.

8.9.2 Applications of Voice Assistants

Voice assistants are commonly used on smart phones to perform tasks like setting alarms, sending messages or checking the weather. In smart homes, they enable controlling devices like lights, thermostats and entertainment systems hands-free. In healthcare, they can provide medication reminders, offer basic health information or assist with patient monitoring.

For navigation, voice assistants help with direction, real time traffic updates and finding points of interest. Productivity enables task management scheduling and quick information retrieval, setting alarms, reminders, creating calendar events or sending messages.

1. **Smart phones and Smart Homes:** Controlling lights, fans, thermostats or security systems using voice.
2. **Healthcare:** Assisting elderly or differently-abled individuals to set reminders or call for help
3. **Navigation:** Voice commands in GPS systems to find routes and traffic updates.
4. **Productivity:** Setting alarms, reminders, creating calendar events or sending messages.

Case Study:

Alexa, Amazon’s voice assistant, allows users to shop online, play music, and control smart home devices simply through voice commands. Over time, Alexa learns user preferences, making interactions faster and more personalized.

Advantages of Voice Assistants

- **Accessibility:** Helps people with disabilities interact with technology.
- **Convenience:** Tasks can be completed hands-free.
- **Learning Support:** Assists students with research and information retrieval.

Limitations and Ethical Considerations

- **Understanding Accents:** Voice assistants may misinterpret unfamiliar accents or dialects.
- **Privacy:** Recording and storing voice data may lead to privacy concerns.
- **Dependency:** Over-reliance may reduce critical thinking and problem-solving skills.



1. What is a voice assistant?
2. Name two examples of voice assistants.
3. Mention one advantage and one limitation of using voice assistants.
4. How do voice assistants combine AI technologies to function?
5. Define a voice assistant in your own words.
6. Describe one limitation or challenge of voice assistants.
7. List three voice assistants you have used or heard of and describe one feature of each.
8. Discuss in pairs: How can voice assistants be used safely without compromising privacy?
9. Design a simple task that a voice assistant could help you complete at home or school.

◆ Summary ◆

Machines can now carry out tasks that previously required human intelligence thanks to the groundbreaking technology known as Artificial Intelligence (AI). AI is changing how we live, work and learn, from using machine learning to identify patterns in data to use neural networks to simulate the human brain.

Computers can now comprehend and produce human language thanks to Natural Language Processing (NLP) and Prompt Engineering, which improves accessibility and ease of use in interactions. While voice assistants offer hands-free assistance in daily life, robotics and computer vision bring artificial intelligence (AI) into the physical world by enabling machines to sense, analyze and act.

AI in education promotes individualized instruction, giving students more agency and supporting educators with insights and tailored materials. But in addition to all of its advantages, AI also presents moral and societal issues like privacy, equity, and responsible use.

As we move forward, understanding AI, using it ethically, and leveraging its potential for creativity, innovation and problem-solving will be essential for students and society.

AI is no longer just a concept of science fiction—it is a practical, transformative force shaping the present and future of human life.



1. The ability of machines to perform tasks requiring human intelligence is called:
a) Programming b) Artificial Intelligence
c) Automation d) Machine Learning
2. Who is considered the father of Artificial Intelligence?
a) John McCarthy b) Alan Turing c) Isaac Newton d) Ada Lovelace
3. In supervised learning, the training data is:
a) Unlabeled b) Labeled c) Random d) Structured
4. Which layer of a neural network processes the input and performs computations?
a) Input layer b) Hidden layer c) Output layer d) Sensor layer
5. NLP helps computers to:
a) Recognize images b) Process human language
c) Control motors d) Store files
6. Which of the following is a key component of a robot?
a) Sensor b) Controller c) Actuator d) All of the above
7. Computer Vision is used in
a) Medical imaging b) Self-driving cars
c) Security surveillance d) All of the above
8. Large Language Models (LLMs) are used to
a) Generate meaningful text b) Train robots
c) Recognize images d) Detect motion
9. Voice assistants combine
a) NLP and speech recognition b) Robotics and actuators
c) Neural networks and sensors d) Machine translation and parsing

10. AI in education can provide
 - a) One-size-fits-all learning
 - b) Personalized learning
 - c) Only printed textbooks
 - d) Physical classrooms only
11. Define Artificial Intelligence with two examples from daily life.
12. Explain the difference between supervised and unsupervised learning.
13. What is an Artificial Neural Network? Name its three main layers.
14. Describe Natural Language Processing (NLP) and give two applications.
15. Explain what a prompt is and give an example of effective prompt engineering.
16. What are the three main components of a robot?
17. Define Computer Vision and give two examples of its applications.
18. How does AI support personalized learning in classrooms?
19. What is a voice assistant? Give two examples and one advantage of using it.
20. Name two ethical considerations while using AI systems.
21. Discuss how AI can create opportunities as well as challenges in society. Provide examples.
22. Design a simple AI-based idea for your school or community. Mention the data it would need and how it would help.
23. Do you think AI will ever replace teachers? Justify your answer with reasons.
24. Imagine you are designing a voice assistant for a multilingual region. How would you ensure it understands different languages and dialects effectively?
25. A factory wants to implement robotics for quality control. What ethical, safety, and efficiency factors should be considered?
26. Explain how Large Language Models could be misused. Suggest measures to prevent misuse.
27. How can AI help in agriculture using Computer Vision? Describe a practical scenario.
28. Compare and contrast Machine Learning and traditional programming in terms of adaptability and decision-making.
29. Discuss one scenario where improper prompt engineering could lead to wrong AI outputs. Suggest how to correct it.
30. Evaluate the impact of AI on accessibility for people with disabilities in education and daily life.
31. Interact with a chatbot or voice assistant. Write three questions and evaluate the responses.

32. Draw a simple block diagram of a robot showing sensors, controllers, and actuators.
33. Identify one AI application in healthcare, education, or agriculture and explain how it works.
34. Write two prompts for an AI model one vague and one specific. Compare the outputs and analyze the differences.
35. Discuss in groups how AI can be used ethically to improve environmental monitoring.