Chapter 1

Computer System

11120CH01

"A computer would deserve to be called intelligent if it could deceive a human into believing that it was human."

–Alan Turing

1.1 Introduction to Computer System

A computer is an electronic device that can be programmed to accept data (input), process it and generate result (output). A computer along with additional hardware and software together is called a computer system.

A computer system primarily comprises a central processing unit (CPU), memory, input/output devices and storage devices. All these components function together as a single unit to deliver the desired output. A computer system comes in various forms and sizes. It can vary from a high-end server to personal desktop, laptop, tablet computer, or a smartphone.

Figure 1.1 shows the block diagram of a computer system. The directed lines represent the flow of data and signal between the components.

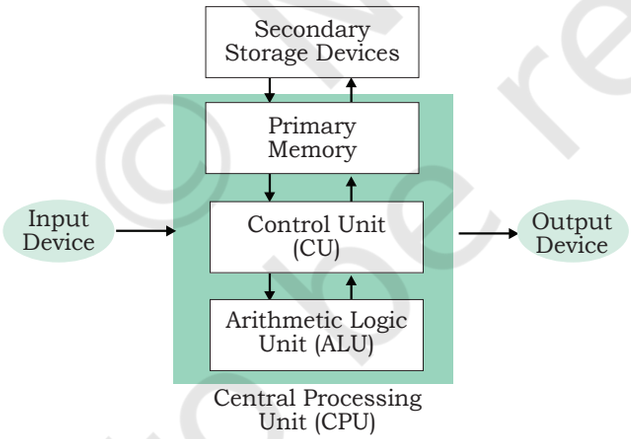


Figure 1.1: Components of a computer system

1.1.1 Central Processing Unit (CPU)

It is the electronic circuitry of a computer that carries out the actual processing and usually referred as the brain of the computer. It is commonly called processor also. Physically, a CPU can be placed on one or more microchips called integrated circuits (IC). The ICs comprise semiconductor materials.

In this chapter

- *Introduction to Computer System*
- *Evolution of Computer*
- *Computer Memory*
- *Data Transfer between Memory and CPU*
- *Data and Information*
- *Microprocessors*
- *Software*
- *Operating System*

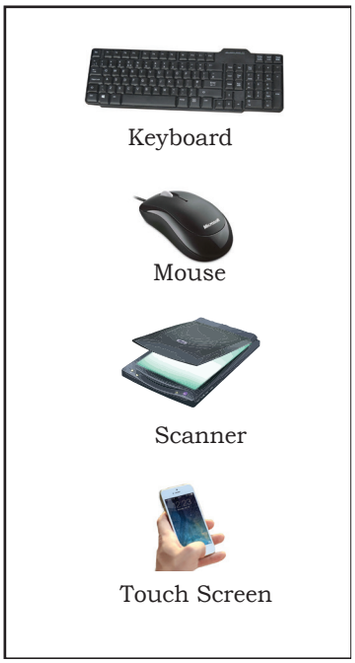


Figure 1.2: Input devices

The CPU is given instructions and data through programs. The CPU then fetches the program and data from the memory and performs arithmetic and logic operations as per the given instructions and stores the result back to memory.

While processing, the CPU stores the data as well as instructions in its local memory called registers. Registers are part of the CPU chip and they are limited in size and number. Different registers are used for storing data, instructions or intermediate results.

Other than the registers, the CPU has two main components — Arithmetic Logic Unit (ALU) and Control Unit (CU). ALU performs all the arithmetic and logic operations that need to be done as per the instruction in a program. CU controls sequential instruction execution, interprets instructions and guides data flow through the computer's memory, ALU and input or output devices. CPU is also popularly known as microprocessor. We will study more about it in section 1.5.

1.1.2 Input Devices

The devices through which control signals are sent to a computer are termed as input devices. These devices convert the input data into a digital form that is acceptable by the computer system. Some examples of input devices include keyboard, mouse, scanner, touch screen, etc., as shown in Figure 1.2. Specially designed braille keyboards are also available to help the visually impaired for entering data into a computer. Besides, we can now enter data through voice, for example, we can use Google voice search to search the web where we can input the search string through our voice.

Data entered through input device is temporarily stored in the main memory (also called RAM) of the computer system. For permanent storage and future use, the data as well as instructions are stored permanently in additional storage locations called secondary memory.

Figure 1.3: Output devices

1.1.3 Output Devices

The device that receives data from a computer system for display, physical production, etc., is called output device. It converts digital information into human-understandable form. For example, monitor, projector, headphone, speaker, printer, etc. Some output devices

are shown in Figure 1.3. A braille display monitor is useful for a visually challenged person to understand the textual output generated by computers.

A printer is the most commonly used device to get output in physical (hardcopy) form. Three types of commonly used printers are inkjet, laserjet and dot matrix. Now-a-days, there is a new type of printer called 3D-printer, which is used to build physical replica of a digital 3D design. These printers are being used in manufacturing industries to create prototypes of products. Their usage is also being explored in the medical field, particularly for developing body organs.

A punched card is a piece of stiff paper that stores digital data in the form of holes at predefined positions.

1.2 Evolution of Computer

From the simple calculator to a modern day powerful data processor, computing devices have evolved in a relatively short span of time. The evolution of computing devices in shown through a timeline in Figure 1.4

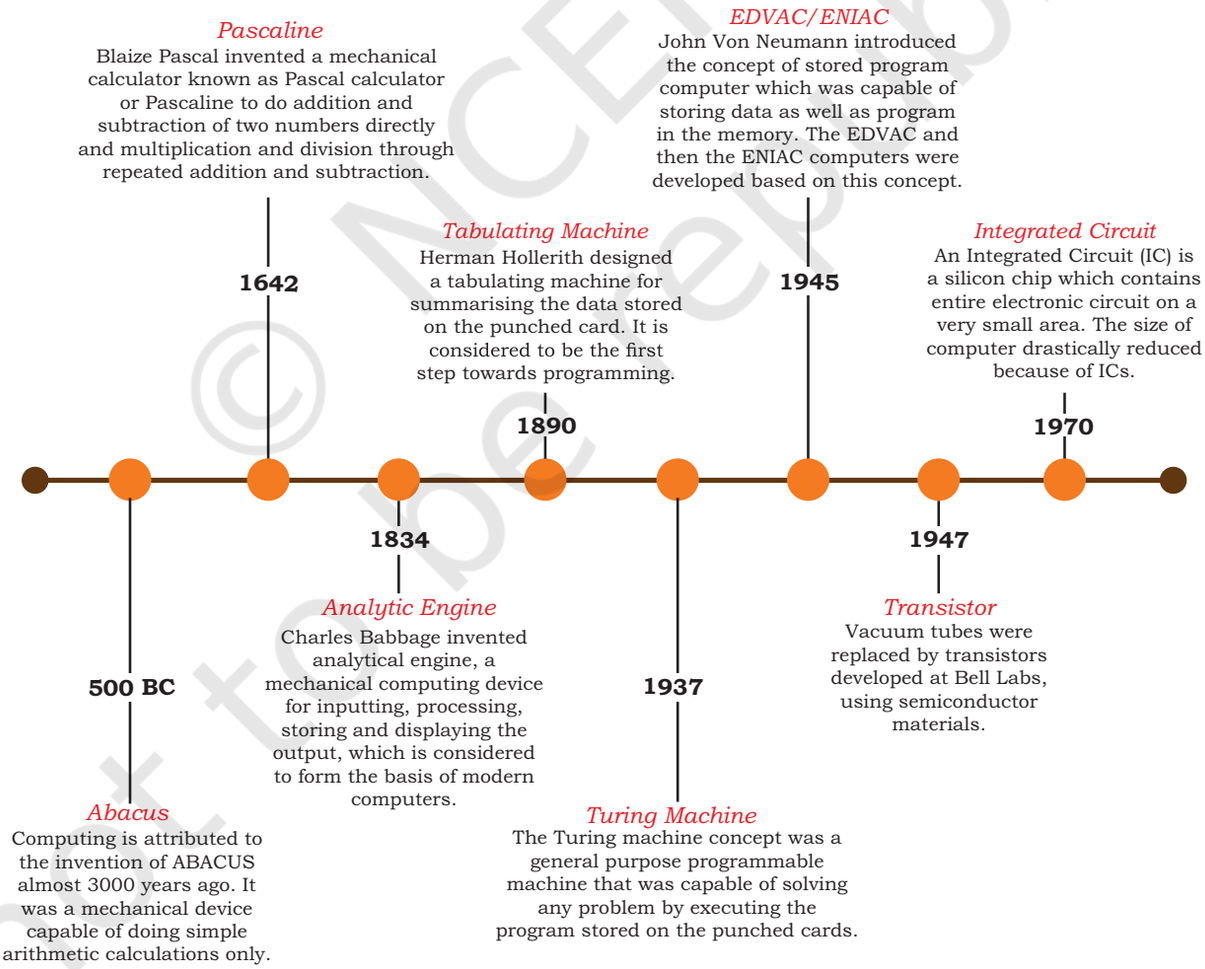


Figure 1.4: Timeline showing key inventions in computing technology

The Von Neumann architecture is shown in Figure 1.5. It consists of a Central Processing Unit (CPU) for processing arithmetic and logical instructions, a memory to store data and programs, input and output devices and communication channels to send or receive the output data. Electronic Numerical Integrator and Computer (ENIAC) is the first binary programmable computer based on Von Neumann architecture.

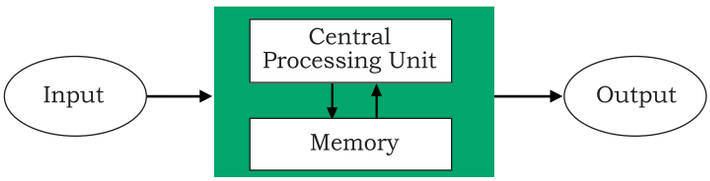


Figure 1.5: Von Neumann architecture for the computer

During the 1970s, Large Scale Integration (LSI) of electronic circuits allowed integration of complete CPU on a single chip, called microprocessor. Moore's Law predicted exponential growth in the number of transistors that could be assembled in a single microchip. In 1980s, the processing power of computers increased exponentially by integrating around 3 million components on a small-sized chip termed as Very Large Scale Integration (VLSI). Further advancement in technology has made it feasible to fabricate high density of transistors and other components (approx 106 components) on a single IC called Super Large Scale Integration (SLSI) as shown in Figure 1.6.

In 1965, Intel co-founder Gordon Moore introduced Moore's Law which predicted that the number of transistors on a chip would double every two years while the costs would be halved.

IBM introduced its first personal computer (PC) for the home user in 1981 and Apple introduced Macintosh

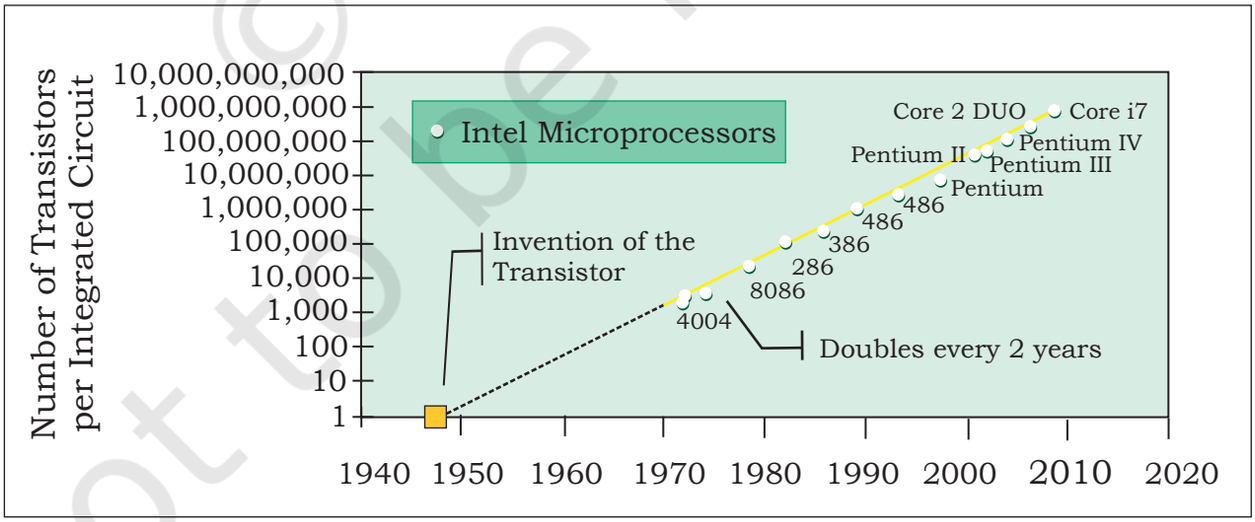


Figure 1.6: Exponential increase in number of transistors used in ICs over time

machines in 1984. The popularity of the PC surged by the introduction of Graphical User Interface (GUI) based operating systems by Microsoft and others in place of computers with only command line interface, like UNIX or DOS. Around 1990s, the growth of World Wide Web (WWW) further accelerated mass usage of computers and thereafter computers have become an indispensable part of everyday life.

Further, with the introduction of laptops, personal computing was made portable to a great extent. This was followed by smartphones, tablets and other personal digital assistants. These devices have leveraged the technological advancements in processor miniaturisation, faster memory, high speed data and connectivity mechanisms.

The next wave of computing devices includes the wearable gadgets, such as smart watch, lenses, headbands, headphones, etc. Further, smart appliances are becoming a part of the Internet of Things (IoT), by leveraging the power of Artificial Intelligence (AI).

1.3 Computer Memory

A computer system needs memory to store the data and instructions for processing. Whenever we talk about the 'memory' of a computer system, we usually talk about the main or primary memory. The secondary memory (also called storage device) is used to store data, instructions and results permanently for future use.

1.3.1 Units of Memory

A computer system uses binary numbers to store and process data. The binary digits 0 and 1, which are the basic units of memory, are called bits. Further, these bits are grouped together to form words. A 4-bit word is called a Nibble. Examples of nibble are 1001, 1010, 0010, etc. A two nibble word, i.e., 8-bit word is called a byte, for example, 01000110, 01111100, 10000001, etc.

Like any other standard unit, bytes are grouped together to make bigger chunks or units of memory. Table 1.1 shows different measurement units for digital data stored in storage devices.

Table 1.1 Measurement units for digital data

Unit	Description	Unit	Description
KB (Kilobyte)	1 KB = 1024 Bytes	PB (Petabyte)	1 PB = 1024 TB
MB (Megabyte)	1 MB = 1024 KB	EB (Exabyte)	1 EB = 1024 PB
GB (Gigabyte)	1 GB = 1024 MB	ZB (Zettabyte)	1 ZB = 1024 EB
TB (Terabyte)	1 TB = 1024 GB	YB (Yottabyte)	1 YB = 1024 ZB

1.3.2 Types of Memory

Human beings memorise many things over a lifetime, and recall from memory to make a decision or some action. However, we do not rely on our memory completely, and we make notes and store important data and information using other media, such as notebook, manual, journal, document, etc. Similarly, computers have two types of memory — primary and secondary.

(A) Primary Memory

Primary memory is an essential component of a computer system. Program and data are loaded into the primary memory before processing. The CPU interacts directly with the primary memory to perform read or write operation. It is of two types viz. (i) Random Access Memory (RAM) and (ii) Read Only Memory (ROM).

RAM is volatile, i.e., as long as the power is supplied to the computer, it retains the data in it. But as soon as the power supply is turned off, all the contents of RAM are wiped out. It is used to store data temporarily while the computer is working. Whenever the computer is started or a software application is launched, the required program and data are loaded into RAM for processing. RAM is usually referred to as main memory and it is faster than the secondary memory or storage devices.

On the other hand, ROM is non-volatile, which means its contents are not lost even when the power is turned off. It is used as a small but faster permanent storage for the contents which are rarely changed. For example, the startup program (boot loader) that loads the operating system into primary memory, is stored in ROM.

Think and Reflect

Suppose there is a computer with RAM but no secondary storage. Can we install a software on that computer?

(B) Cache Memory

RAM is faster than secondary storage, but not as fast as a computer processor. So, because of RAM, a CPU

may have to slow down. To speed up the operations of the CPU, a very high speed memory is placed between the CPU and the primary memory known as *cache*. It stores the copies of the data from frequently accessed primary memory locations, thus, reducing the average time required to access data from primary memory. When the CPU needs some data, it first examines the cache. In case the requirement is met, it is read from the cache, otherwise the primary memory is accessed.

(C) Secondary Memory

Primary memory has limited storage capacity and is either volatile (RAM) or read-only (ROM). Thus, a computer system needs auxiliary or secondary memory to permanently store the data or instructions for future use. The secondary memory is non-volatile and has larger storage capacity than primary memory. It is slower and cheaper than the main memory. But, it cannot be accessed directly by the CPU. Contents of secondary storage need to be first brought into the main memory for the CPU to access. Examples of secondary memory devices include Hard Disk Drive (HDD), CD/DVD, Memory Card, etc., as shown in Figure 1.7.

However, these days, there are secondary storage devices like SSD which support very fast data transfer speed as compared to earlier HDDs. Also, data transfer between computers have become easier and simple due to the availability of small-sized and portable flash or pen drives.

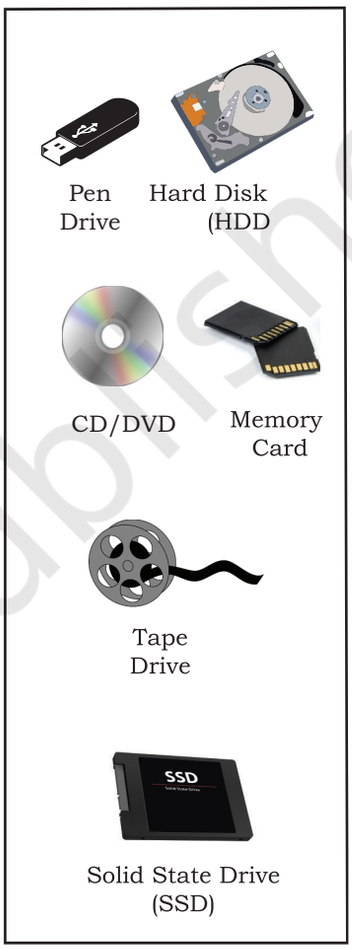


Figure 1.7: Storage devices

1.4 Data Transfer between Memory and CPU

Data need to be transferred between the CPU and primary memory as well as between the primary and secondary memory.

Data are transferred between different components of a computer system using physical wires called *bus*. For example, bus is used for data transfer between a USB port and hard disk or between a hard disk and main memory. Bus is of three types— (i) Data bus to transfer data between different components, (ii) Address bus to transfer addresses between CPU and main memory. The address of the memory location that the CPU wants to read or write from is specified in the address bus,

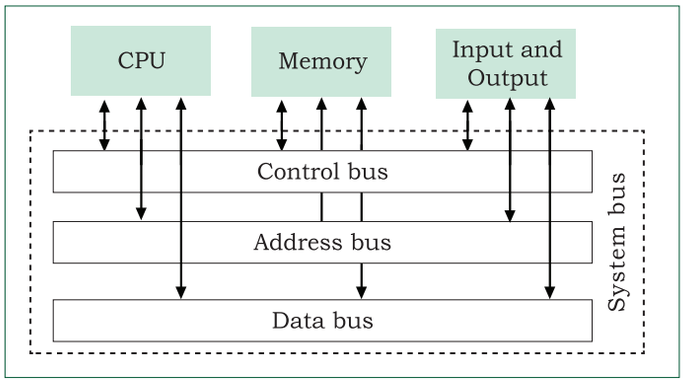


Figure 1.8: Data transfer between components through system bus

and (iii) Control bus to communicate control signals between different components of a computer. All these three buses collectively make the system bus, as shown in Figure 1.8.

As the CPU interacts directly with main memory, any data entered from input device or the data to be accessed from hard disk needs to be placed in the main memory for further processing. The data is then transferred between CPU and main memory using bus.

The CPU places on the address bus, the address of the main memory location from which it wants to read data or to write data. While executing the instructions, the CPU specifies the read or write control signal through the control bus.

As the CPU may require to read data from main memory or write data to main memory, a data bus is bidirectional. But the control bus and address bus are unidirectional. To write data into memory, the CPU places the data on the data bus, which is then written to the specific address provided through the address bus. In case of read operation, the CPU specifies the address, and the data is placed on the data bus by a dedicated hardware, called memory controller. The memory controller manages the flow of data into and out of the computer's main memory.

1.5 MICROPROCESSORS

In earlier days, a computer's CPU used to occupy a large room or multiple cabinets. However, with advancement in technology, the physical size of CPU has reduced and it is now possible to place a CPU on a single microchip only. A processor (CPU) which is implemented on a single microchip is called microprocessor. Nowadays, almost all the CPUs are microprocessors. Hence, the terms are used synonymously for practical purpose.

Microprocessor is a small-sized electronic component inside a computer that carries out various tasks involved in data processing as well as arithmetic and logical operations. These days, a microprocessor is built over an integrated circuit comprising millions of small components like resistors, transistors and diodes.

Microprocessors have evolved over time in terms of their increased processing capability, decreasing physical size and reduced cost. Currently available microprocessors are capable of processing millions of instructions per millisecond. Table 1.2 lists different types of microprocessors along with their generation, time period, and underlying technology since their inception in early 1970s.

Table 1.2 Generations of Microprocessor

Generation	Era	Chip type	Word size	Maximum memory size	Clock speed	Cores	Example*
First	1971-73	LSI	4 / 8 bit	1 KB	108 KHz-200 KHz	Single	Intel 8080
Second	1974-78	LSI	8 bit	1 MB	Upto 2 MHz	Single	Motorola 6800 Intel 8085
Third	1979-80	VLSI	16 bit	16 MB	4 MHz - 6 MHz	Single	Intel 8086
Fourth	1981-95	VLSI	32 bit	4 GB	Upto 133 MHz	Single	Intel 80386 Motorola 68030
Fifth	1995 till date	SLSI	64 bit	64 GB	533 MHz - 34 GHz	Multicore	Pentium, Celeron, Xeon

*few prominent examples are included.

1.5.1 Microprocessor Specifications

Microprocessors are classified on the basis of different features which include chip type, word size, memory size, clock speed, etc. These features are briefly explained below:

(A) Word Size

Word size is the maximum number of bits that a microprocessor can process at a time. Earlier, a word was of 8 bits, as it was the maximum limit at that time. At present, the minimum word size is 16 bits and maximum word size is 64 bits.

(B) Memory Size

Depending upon the word size, the size of RAM varies. Initially, RAM was very small (4MB) due to 4/8 bits word size. As word size increased to 64 bits, it has become feasible to use RAM of size upto 16 Exabytes (EB).

(C) Clock Speed

Computers have an internal clock that generates pulses (signals) at regular intervals of time. Clock speed simply means the number of pulses generated per second by the

Activity 1.1

The maximum memory size of microprocessors of different generations are given at Table 1.2. Represent each of the memory size in terms of power of 2.

Activity 1.2

Find out the clock speed of the microprocessor of your computer and compare with that of your peers?

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.

(D) Cores

Core is a basic computation unit of the CPU. Earlier processors had only one computation unit, thereby capable of performing only one task at a time. With the advent of multicore processor, it has become possible for the computer to execute multiple tasks, thereby increasing the system's performance. CPU with two, four, and eight cores is called dual-core, quad-core and octa-core processor, respectively.

1.5.2 Microcontrollers

The microcontroller is a small computing device which has a CPU, a fixed amount of RAM, ROM and other peripherals all embedded on a single chip as compared to microprocessor that has only a CPU on the chip. The structure of a microcontroller is shown in Figure 1.9. Keyboard, mouse, washing machine, digital camera, pendrive, remote controller, microwave are few examples of microcontrollers. As these are designed for specific tasks only, hence their size as well as cost is reduced.

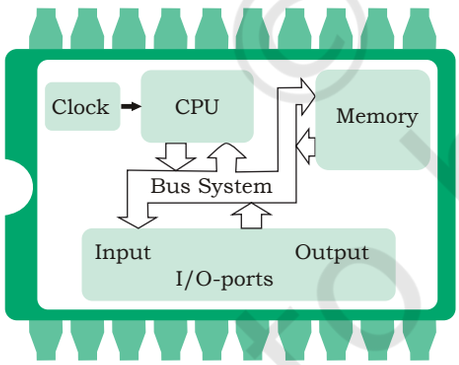


Figure 1.9: Structure of microcontroller

Because of the very small size of the microcontroller, it is embedded in another device or system to perform a specific functionality. For example, the microcontroller in a fully automatic washing machine is used to control the washing cycle without any human intervention. The cycle starts with the filling of water, after which the clothes are soaked and washed; thereafter the water is drained and the clothes are spin dry. The simple use of microcontroller has permitted repetitive execution of tedious tasks automatically without any human intervention, thereby saving precious time.

1.6 DATA AND INFORMATION

A computer is primarily for processing data. A computer system considers everything as data, be it instructions, pictures, songs, videos, documents, etc. Data can also be

raw and unorganised facts that are processed to get meaningful information.

So understanding the concept of data along with its different types is crucial to understand the overall functioning of a computer. Sometimes people use the terms data, information and knowledge interchangeably, which is incorrect.

1.6.1 Data and Its Types

A computer system has many input devices, which provide it with raw data in the form of facts, concepts, instructions, etc., Internally everything is stored in binary form (0 and 1), but externally, data can be input to a computer in the text form consisting of English alphabets A–Z, a–z, numerals 0–9, and special symbols like @, #, etc. Data can be input in other languages too or it can be read from the files. The input data may be from different sources, hence it may be in different formats. For example, an image is a collection of Red, Green, Blue (RGB) pixels, a video is made up of frames, and a fee receipt is made of numeric and non-numeric characters. Primarily, there are three types of data.

(A) Structured Data

Data which follows a strict record structure and is easy to comprehend is called structured data. Such data with pre-specified tabular format may be stored in a data file to access in the future. Table 1.3 shows structured data related to monthly attendance of students maintained by the school.

Table 1.3 Structured data: Monthly attendance records of students

Roll No	Name	Month	Attendance (in %)
R1	Mohan	May	95
R2	Sohan	May	75
R3	Sheen	May	92
R4	Geet	May	82
R5	Anita	May	97
R1	Mohan	July	98
R2	Sohan	July	65
R3	Sheen	July	85
R4	Geet	July	94
R5	Anita	July	85

Think and Reflect

Can you give some more examples of unstructured data?

It is clear that such data is organised in row/column format and is easily understandable. Structured data may be sorted in ascending or descending order. In the example, attendance data is sorted in increasing order on the column 'month'. Other examples of structured data include sales transactions, online railway ticket bookings, ATM transactions, etc.

(B) Unstructured Data

Data which are not organised in a pre-defined record format is called unstructured data. Examples include audio and video files, graphics, text documents, social media posts, satellite images, etc. Figure 1.10 shows a report card with monthly attendance record details sent to parents. Such data are unstructured as they consist of textual contents as well as graphics, which do not follow a specific format.

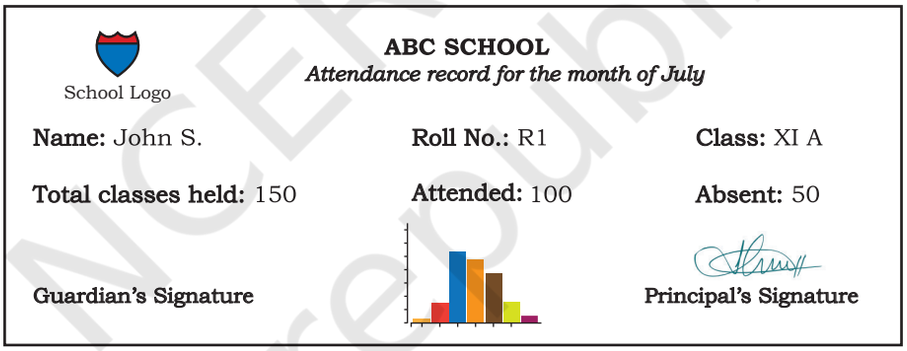


Figure 1.10: Unstructured data: Monthly attendance record

(C) Semi-structured Data

Data which have no well-defined structure but maintains internal tags or markings to separate data elements are called semi-structured data. Examples include email document, HTML page, comma separated values (csv file), etc. Figure 1.11 shows an example of semi-structured data containing student's month-wise attendance details. In this example, there is no specific format for each attendance record. Here, each data value is preceded by a tag (Name, Month, Class, Attendance) for the interpretation of the data value while processing.

Name: *Mohan*	Month: *July*	Class: *XI*	Attendance: *98*
Name: *Sohan*	Month: *July*	Class: *XI*	Attendance: *65*
Name: *Sheen*	Month: *July*	Class: *XI*	Attendance: *85*
Name: *Geet*	Month: *May*	Class: *XI*	Attendance: *82*
Name: *Geet*	Month: *July*	Class: *XI*	Attendance: *94*

Figure 1.11: Semi-structured data: Month-wise total attendance record maintained by the school

1.6.2 Data Capturing, Storage and Retrieval

To process data, we need to first input or capture the data. This is followed by its storage in a file or a database so that it can be used in the future. Whenever data is to be processed, it is first retrieved from the file or database so that we can perform further actions on it.

(A) Data Capturing

It involves the process of gathering data from different sources in the digital form. This capturing may vary from simple instruments like keyboard, barcode readers used at shopping outlets (Figure 1.12), comments or posts over social media, remote sensors on an earth orbiting satellite, etc. Sometimes, heterogeneity among data sources makes data capturing a complex task.

(B) Data Storage

It is the process of storing the captured data for processing later. Now-a-days data is being produced at a very high rate, and therefore data storage has become a challenging task. However, the decrease in the cost of digital storage devices has helped in simplifying this task. There are numerous digital storage devices available in the market like as shown in Figure 1.7.

Data keeps on increasing with time. Hence, the storage devices also require to be upgraded periodically. In large organisations, computers with larger and faster storage called data servers are deployed to store vast amount of data. Such dedicated computers help in processing data efficiently. However, the cost (both hardware and software) of setting up a data server as well as its maintenance is high, especially for small organisations and startups.

(C) Data Retrieval

It involves fetching data from the storage devices, for its processing as per the user requirement. As databases grow, the challenges involved in search and retrieval of the data in acceptable time, also increase. Minimising data access time is crucial for faster data processing.

1.6.3 Data Deletion and Recovery

One of the biggest threats associated with digital data is its deletion. The storage devices can malfunction or crash down resulting in the deletion of data stored. Users can

Activity 1.3

Visit some of the places like bank, automobile showroom, shopping mall, tehsil office, etc., and find out 2–3 names of tools or instruments used to capture data in digital format.

Figure 1.12: Capturing data using barcode reader

Activity 1.4

Explore possible ways of recovering deleted data or data from a corrupted device.

accidentally erase data from storage devices, or a hacker or malware can delete the digital data intentionally.

Deleting digitally stored data means changing the details of data at bit level, which can be very time-consuming. Therefore, when any data is simply deleted, its address entry is marked as free, and that much space is shown as empty to the user, without actually deleting the data.

In case data gets deleted accidentally or corrupted, there arises a need to recover the data. Recovery of the data is possible only if the contents or memory space marked as deleted have not been overwritten by some other data. Data recovery is a process of retrieving deleted, corrupted and lost data from secondary storage devices.

There are usually two security concerns associated with data. One is its deletion by some unauthorised person or software. These concerns can be avoided by limiting access to the computer system and using passwords for user accounts and files, wherever possible. There is also an option of encrypting files to protect them from unwanted modification.

Activity 1.5

Create a test file and then delete it using Shift+Delete from the keyboard. Now recover the file using the methods you have explored in Activity 1.4.

The other concern is related to unwanted recovery of data by unauthorised user or software. Many a times, we discard our old, broken or malfunctioning storage devices without taking care to delete data. We assume that the contents of deleted files are permanently removed. However, if these storage devices fall into the hands of mischief-mongers, they can easily recover data from such devices; this poses a threat to data confidentiality. This concern can be mitigated by using proper tools to delete or shred data before disposing off any old or faulty storage device.

1.7 SOFTWARE

Till now, we have studied about the physical components or the hardware of the computer system. But the hardware is of no use on its own. Hardware needs to be operated by a set of instructions. These sets of instructions are referred to as software. It is that component of a computer system, which we cannot

touch or view physically. It comprises the instructions and data to be processed using the computer hardware. The computer software and hardware complete any task together.

The software comprises a set of instructions which on execution deliver the desired outcome. In other words, each software is written for some computational purpose. Some examples of software include operating systems like Ubuntu or Windows 7/10, word processing tool like LibreOffice or Microsoft Word, video player like VLC Player, photo editors like GIMP and LibreOffice draw. A document or image stored on the hard disk or pen drive is referred to as a soft-copy. Once printed, the document or an image is called a hard-copy.

Hardware refers to the physical components of the computer system which can be seen and touched. For example, RAM, keyboard, printer, monitor, CPU, etc. On the other hand, software is a set of instructions and data that makes hardware functional to complete the desired task.

1.7.1 Need of Software

The sole purpose of a software is to make the computer hardware useful and operational. A software knows how to make different hardware components of a computer work and communicate with each other as well as with the end-user. We cannot instruct the hardware of a computer directly. Software acts as an interface between human users and the hardware.

Depending on the mode of interaction with hardware and functions to be performed, the software can be broadly classified into three categories *viz.* (i) System software, (ii) Programming tools and (iii) Application software.

1.7.2 System Software

The software that provides the basic functionality to operate a computer by interacting directly with its constituent hardware is termed as system software. A system software knows how to operate and use different hardware components of a computer. It provides services directly to the end user, or to some other software. Examples of system software include operating systems, system utilities, device drivers, etc.

(A) Operating System

As the name implies, the operating system is a system software that operates the computer. An operating system is the most basic system software, without which other software cannot work. The operating system manages other application programs and provides

access and security to the users of the system. Some of the popular operating systems are Windows, Linux, Macintosh, Ubuntu, Fedora, Android, iOS, etc.

(B) System Utilities

Software used for maintenance and configuration of the computer system is called system utility. Some system utilities are shipped with the operating system for example disk defragmentation tool, formatting utility, system restore utility, etc. Another set of utilities are those which are not shipped with the operating system but are required to improve the performance of the system, for example, anti-virus software, disk cleaner tool, disk compression software, etc.

(C) Device Drivers

Activity 1.6

Ask your teacher to help you locate any two device drivers installed on your computer.

As the name signifies, the purpose of a device driver is to ensure proper functioning of a particular device. When it comes to the overall working of a computer system, the operating system does the work. But everyday new devices and components are being added to a computer system. It is not possible for the operating system alone to operate all of the existing and new devices, where each device has diverse characteristics. The responsibility for overall control, operation and management of a particular device at the hardware level is delegated to its device driver.

The device driver acts as an interface between the device and the operating system. It provides required services by hiding the details of operations performed at the hardware level of the device. Just like a language translator, a device driver acts as a mediator between the operating system and the attached device. The categorisation of software is shown in Figure 1.13.

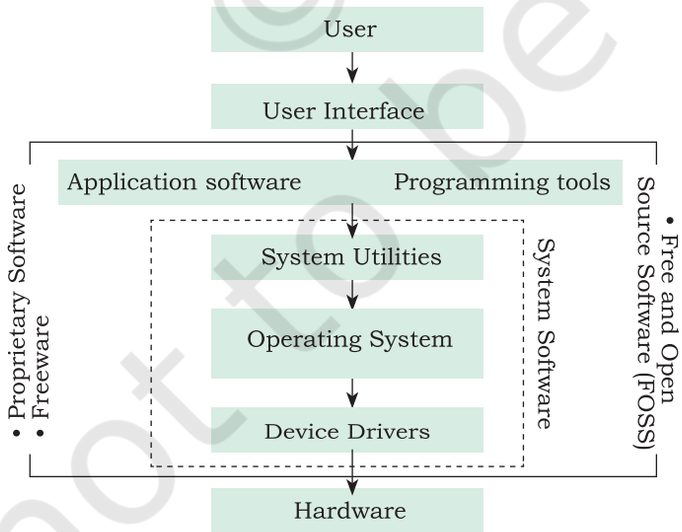


Figure 1.13: Categorisation of software

1.7.3 Programming Tools

In order to get some work done by the computer, we need to give instructions which are applied on the input data to get the desired outcome. Computer languages are developed for writing these instructions.

Notes

It is important to understand here that computers and humans understand completely different languages. While humans are able to write programs in high-level language, computers understand machine language. There is a continuous need for conversion from high level to machine level language, for which translators are needed. Also, to write the instruction, code editors (e.g., IDLE in Python) are needed. We will briefly describe here the programming languages, language translators and program development tools.

(A) Classification of Programming Languages

It is very difficult for a human being to write instructions in the form of 1s and 0s. So different types of computer programming languages are developed to simplify the coding. Two major categories of computer programming languages are low-level languages and high-level languages.

Low-level languages are machine dependent languages and include machine language and assembly language. Machine language uses 1s and 0s to write instructions which are directly understood and executed by the computer. But writing a code in machine language is difficult as one has to remember all operation codes and machine addresses. Also finding errors in the code written in machine language is difficult.

To simplify the writing of code, assembly language was developed that allowed usage of English-like words and symbols instead of 1s and 0s. But one major drawback of writing a code in this language is that the code is computer specific, i.e., the code written for one type of CPU cannot be used for another type of CPU.

High level languages are machine independent and are simpler to write code into. Instructions are using English like sentences and each high level language follows a set of rules, similar to natural languages. However, these languages are not directly understood by the computer. Hence, translators are needed to translate high-level language codes into machine language. Examples of high level language include C++, Java, Python, etc.

(B) Language Translators

As the computer can understand only machine language, a translator is needed to convert program written in assembly or high level language to machine

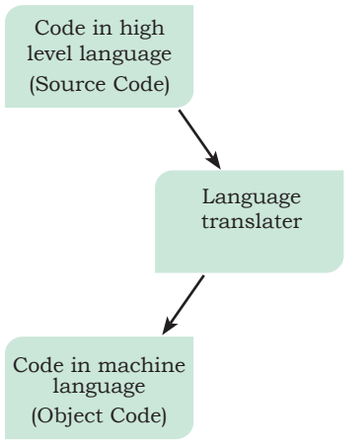


Figure 1.14: Translator to convert source code into object code

language. The program code written in assembly or high-level language is called source code. The source code is converted by a translator into the machine understandable form called object (machine) code as depicted in Figure 1.14.

As we have different types of computer languages, different translators are needed to convert the source code to machine code. The three types of translators used in computing systems are assembler, compiler and interpreter.

The translator used to convert the code written in assembly language to machine language is called *assembler*. Each assembler can understand a specific microprocessor instruction set only and hence, the machine code is not portable.

We also need translators to convert codes written in high level language (source code) to machine understandable form (machine code) for execution by the computer. *Compiler* converts the source code into machine code. If the code follows all syntactic rules of the language, then it is executed by the computer. Once translated, the compiler is not needed.

An interpreter translates one line at a time instead of the whole program at one go. Interpreter takes one line, converts it into executable code if the line is syntactically correct, and then it repeats these steps for all lines in the source code. Hence, interpreter is always needed whenever a source code is to be executed.

(C) Program Development Tools

Whenever we decide to write a program, we need a text editor. An editor is a software that allows us to create a text file where we type instructions and store the file as the source code. Then an appropriate translator is used to get the object code for execution. In order to simplify the program development, there are software called Integrated Development Environment (IDE) consisting of text editor, building tools and debugger. A program can be typed, compiled and debugged from the IDE directly. Besides Python IDLE, Netbeans, Eclipse, Atom, Lazarus are few other examples of IDEs. Debugger, as the name implies, is the software to detect and correct errors in the source code.

1.7.4 Application Software

The system software provides the core functionality of the computer system. However, different users need the computer system for different purposes depending upon their requirements. Hence, a new category of software is needed to cater to different requirements of the end-users. This specific software that works on top of the system software is termed as application software. There are again two broad categories of application software—general purpose and customised application software.

(A) General Purpose Software

The application software developed for generic applications, to cater to a bigger audience in general are called general purpose software. Such ready-made application software can be used by end users as per their requirements. For example, spreadsheet tool Calc of LibreOffice can be used by any computer user to do calculation or to create account sheet. Adobe Photoshop, GIMP, Mozilla web browser, iTunes, etc., fall in the category of general purpose software.

(B) Customised Software

These are custom or tailor-made application software, that are developed to meet the requirements of a specific organisation or an individual. They are better suited to the needs of an individual or an organisation, considering that they are designed as per special requirements. Some examples of user-defined software include websites, school management software, accounting software, etc. It is similar to buying a piece of cloth and getting a tailor-made garment with the fitting, colour, and fabric of our choice.

1.7.5 Proprietary or Free and Open Source Software

The developers of some application software provide their source code as well as the software freely to the public, with an aim to develop and improve further with each other's help. Such software is known as Free and Open Source Software (FOSS). For example, the source code of operating system Ubuntu is freely accessible for anyone with the required knowledge to improve or add new functionality. More examples of FOSS include Python, Libreoffice, Openoffice, Mozilla Firefox, etc. Sometimes, software are freely available for use but

Activity 1.7

With the help of your teacher, install one application software in your computer.

A computer system can work without application software, but it cannot work without system software. For example, we can use a computer even if no word processing software is installed, but if no operating system is installed, we cannot work on the computer. In other words, the use of computer is possible in the absence of application software.

Activity 1.8

With the help of your teacher, install one free and open source application software on your computer.

Think and Reflect

When a computer is turned on, who brings the OS into RAM from the secondary storage?

source code may not be available. Such software are called freeware. Examples of freeware are Skype, Adobe Reader, etc. When the software to be used has to be purchased from the vendor who has the copyright of the software, then it is a proprietary software. Examples of proprietary software include Microsoft Windows, Tally, Quickheal, etc. A software can be freeware or open source or proprietary software depending upon the terms and conditions of the person or group who has developed and released that software.

1.8 OPERATING SYSTEM

An operating system (OS) can be considered to be a resource manager which manages all the resources of a computer, i.e., its hardware including CPU, RAM, Disk, Network and other input-output devices. It also controls various application software and device drivers, manages system security and handles access by different users. It is the most important system software. Examples of popular OS are Windows, Linux, Android, Macintosh and so on.

The primary objectives of an operating system are two-fold. The first is to provide services for building and running application programs. When an application program needs to be run, it is the operating system which loads that program into memory and allocates it to the CPU for execution. When multiple application programs need to be run, the operating system decides the order of the execution.

The second objective of an operating system is to provide an interface to the user through which the user can interact with the computer. A user interface is a software component which is a part of the operating system and whose job is to take commands or inputs from a user for the operating system to process.

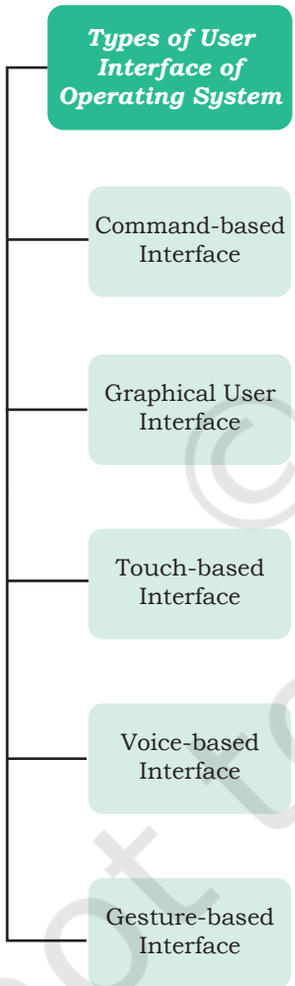


Figure 1.15: Types of user interface of OS

1.8.1 OS User Interface

There are different types of user interfaces each of which provides a different functionality. Some commonly used interfaces are shown in Figure 1.15.

(A) Command-based Interface

Command-based interface requires a user to enter the commands to perform different tasks like creating,

opening, editing or deleting a file, etc. The user has to remember the names of all such programs or specific commands which the operating system supports.

The primary input device used by the user for command based interface is the keyboard. Command based interface is often less interactive and usually allows a user to run a single program at a time.

Examples of operating systems with command-based interface include MS-DOS and Unix.

(B) Graphical User Interface

Graphical User Interface (GUI) lets users run programs or give instructions to the computer in the form of icons, menus and other visual options. Icons usually represent files and programs stored on the computer and windows represent running programs that the user has launched through the operating system.

The input devices used to interact with the GUI commonly include the mouse and the keyboard. Examples of operating systems with GUI interfaces include Microsoft Windows, Ubuntu, Fedora and Macintosh, among others.

(C) Touch-based Interface

Today smartphones, tablets and PCs allow users to interact with the system simply using the touch input. Using the touchscreen, a user provides inputs to the operating system, which are interpreted by the OS as commands like opening an app, closing an app, dialing a number, scrolling across apps, etc.

Examples of popular operating systems with touch-based interfaces are Android and iOS. Windows 8.1 and 10 also support touch-based interfaces on touchscreen devices.

(D) Voice-based Interface

Modern computers have been designed to address the needs of all types of users including people with special needs and people who want to interact with computers or smartphones while doing some other task. For users who cannot use the input devices like the mouse, keyboard, and touchscreens, modern operating systems provide other means of human-computer interaction. Users today can use voice-based commands to make a computer work in the desired way. Some operating

systems which provide voice-based control to users include iOS (Siri), Android (Google Now or "OK Google"), Microsoft Windows 10 (Cortana) and so on.

(E) Gesture-based Interface

Some smartphones based on Android and iOS as well as laptops let users interact with the devices using gestures like waving, tilting, eye motion and shaking. This technology is evolving faster and it has promising potential for application in gaming, medicine and other areas.

1.8.2 Functions of Operating System

Now let us explore the important services and tasks that an operating system provides for managing the computer system.

(A) Process Management

Operating system is called resource manager as it manages different resources like main memory, CPU, I/O devices, so that each resource is used optimally and system performance does not deteriorate.

While a computer system is operational, different tasks are running simultaneously. A program is intended to carry out various tasks. A task in execution is known as process. We can activate a system monitor program that provides information about the processes being executed on a computer. In some systems it can be activated using Ctrl+Alt+Delete. It is the responsibility of operating system to manage these processes and get multiple tasks completed in minimum time. As CPU is the main resource of computer system, its allocation among processes is the most important service of the operating system. Hence process management concerns the management of multiple processes, allocation of required resources, and exchange of information among processes.

(B) Memory Management

Primary or main memory of a computer system is usually limited. The main task of memory management is to give (allocate) and take (free) memory from running processes. Since there are multiple processes running at a time, there arises a need to dynamically (on-the-go) allocate and free memory to the processes. Operating system should do it without affecting other processes that are already residing in the memory and once the process is finished, it is again the responsibility of the operating system to take the memory space back for re-

utilisation. Hence, memory management concerns with management of main memory so that maximum memory is occupied or utilised by large number of processes while keeping track of each and every location within the memory as free or occupied.

(C) File Management

Data and programs are stored as files in the secondary storage of a computer system. File management involves the creation, updation, deletion and protection of these files in the secondary memory. Protection is a crucial function of an operating system, as multiple users can access and use a computer system. There must be a mechanism in place that will stop users from accessing files that belong to some other user and have not been shared with them. File management system manages secondary memory, while memory management system handles the main memory of a computer system.

(D) Device Management

A computer system has many I/O devices and hardware connected to it. Operating system manages these heterogeneous devices that are interdependent. The operating system interacts with the device driver and the related software for a particular device. The operating system must also provide the options for configuring a particular device, so that it may be used by an end user or some other device. Just like files, devices also need security measures and their access to different devices must be restricted by the operating system to the authorised users, software and other hardware only.

Summary

- A computing device, also referred as computer, processes the input data as per given instructions to generate desired output.
- Computer system has four physical components *viz.* (i) CPU, (ii) Primary Memory, (iii) Input Device and (iv) Output Devices. They are referred to as hardware of computer.
- Computer system has two types of primary memories *viz.* (i) RAM, the volatile memory and (ii) ROM, the non-volatile memory.

NOTES

- System bus is used to transfer data, addresses and control signals between components of the computer system.
- A microprocessor is a small-sized electronic component inside a computer that performs basic arithmetic and logical operations on data.
- Microcontroller is a small computing device which has a CPU, a fixed amount of RAM, ROM and other peripherals embedded on a single chip.
- Software is a set of instructions written to achieve the desired tasks and are mainly categorised as system software, programming tools and application software.
- Hardware of a computer cannot function on its own. It needs software to be operational or functional.
- Operating system is an interface between the user and the computer and supervises the working of computer system, i.e., it monitors and controls the hardware and software of the computer system.

EXERCISE

1. Name the software required to make a computer functional. Write down its two primary services.
2. How does the computer understand a program written in high level language?
3. Why is the execution time of the machine code less than that of source code?
4. What is the need of RAM? How does it differ from ROM?
5. What is the need for secondary memory?
6. How do different components of the computer communicate with each other?
7. Draw the block diagram of a computer system. Briefly write about the functionality of each component.
8. What is the primary role of system bus? Why is data bus is bidirectional while address bus is unidirectional?
9. Differentiate between proprietary software and freeware software. Name two software for each type.

10. Write the main difference between microcontroller and microprocessor. Why do smart home appliances have a microcontroller instead of microprocessor embedded in them?
11. Mention the different types of data that you deal with while browsing the Internet.
12. Categorise the following data as structured, semi-structured and unstructured:
    - Newspaper
    - Cricket Match Score
    - HTML Page
    - Patient records in a hospital
13. Name the input or output device used to do the following:
    a) To output audio
    b) To enter textual data
    c) To make hard copy of a text file
    d) To display the data or information
    e) To enter audio-based command
    f) To build 3D models
    g) To assist a visually-impaired individual in entering data
14. Identify the category (system, application, programming tool) of the following software:
    a) Compiler
    b) Assembler
    c) Ubuntu
    d) Text editor

Explore Yourself

1. Ask your teacher to help you locate any two device drivers installed on your computer.
2. Write any two system software and two application software installed on your computer.
3. Which microprocessor does your personal computer have? Which generation does it belong to?
4. What is the clock speed of your microprocessor?
5. Name any two devices in your school or home that have a microcontroller.

NOTES

6. Check the size of RAM and HDD of a computer in your school. Make a table and write their size in Bytes, Kilobytes, Megabytes and Gigabytes.
7. List all secondary storage devices available at your school or home.
8. Which operating system is installed on your computer at home or school?

Chapter 2

Encoding Schemes and Number System

11120CH02

2.1 Introduction

"We owe a lot to the Indians, who taught us how to count, without which no worthwhile scientific discovery could have been made."

–Albert Einstein

Have you ever thought how the keys on the computer keyboard that are in human recognisable form are interpreted by the computer system? This section briefly discusses text interpretation by the computer.

We have learnt in the previous chapter that computer understands only binary language of 0s and 1s. Therefore, when a key on the keyboard is pressed, it is internally mapped to a unique code, which is further converted to binary.

Example 2.1 When the key 'A' is pressed (Figure 2.1), it is internally mapped to a decimal value 65 (code value), which is then converted to its equivalent binary value for the computer to understand.

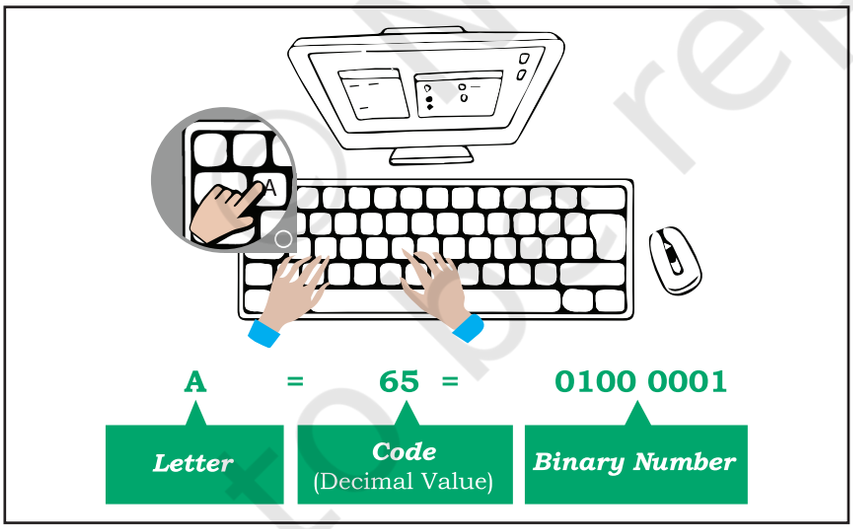


Figure 2.1: Encoding of data entered using keyboard

Similarly, when we press alphabet 'अ' on hindi keyboard, internally it is mapped to a hexadecimal value 0905, whose binary equivalent is 0000100100000101.

So what is encoding? The mechanism of converting data into an equivalent cipher using specific code is

In this chapter

- *Introduction to Encoding*
- *UNICODE*
- *Number System*
- *Conversion Between Number Systems*

Cipher means something converted to a coded form to hide/conceal it from others. It is also called encryption (converted to cipher) and sent to the receiver who in turn can decrypt it to get back the actual content.

called encoding. It is important to understand why code value 65 is used for the key "A" and not any other value? Is it same for all the keyboards irrespective of their make?

Yes, it is same for all the keyboards. This has been possible because of standard encoding schemes where each letter, numeral and symbol is encoded or assigned a unique code. Some of the well-known encoding schemes are described in the following sections.

2.1.1 American Standard Code for Information Interchange (ASCII)

In the early 1960s, computers had no way of communicating with each other due to different ways of representing keys of the keyboard. Hence, the need for a common standard was realised to overcome this shortcoming. Thus, encoding scheme ASCII was developed for standardising the character representation. ASCII is still the most commonly used coding scheme.

Initially ASCII used 7 bits to represent characters. Recall that there are only 2 binary digits (0 or 1). Therefore, total number of different characters on the English keyboard that can be encoded by 7-bit ASCII code is 2^7 = 128. Table 2.1 shows some printable characters for ASCII code. But ASCII is able to encode character set of English language only.

Table 2.1 ASCII code for some printable characters

Character	Decimal Value	Character	Decimal Value	Character	Decimal Value
Space	32	@	64	`	96
!	33	A	65	a	97
”	34	B	66	b	98
#	35	C	67	c	99
$	36	D	68	d	100
%	37	E	69	e	101
&	38	F	70	f	102
‘	39	G	71	g	103
(	40	H	72	h	104
)	41	I	73	i	105

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

- ASCII value of D is 68 and its equivalent 7-bit binary code = 1000100
- ASCII value of A is 65 and its equivalent 7-bit binary code = 1000001
- ASCII value of T is 84 and its equivalent 7-bit binary code = 1010100
- ASCII value of A is 65 and its equivalent 7-bit binary code = 1000001

Replace each alphabet in DATA with its ASCII code value to get its equivalent ASCII code and with 7-bit binary code to get its equivalent binary number as shown in Table 2.2.

Table 2.2 ASCII and Binary values for word DATA

	D	A	T	A
ASCII Code	68	65	84	65
Binary Code	1000100	1000001	1010100	1000001

2.1.2 Indian Script Code for Information Interchange (ISCII)

In order to facilitate the use of Indian languages on computers, a common standard for coding Indian scripts called ISCII was developed in India during mid 1980s. It is an 8-bit code representation for Indian languages which means it can represent 2^8=256 characters. It retains all 128 ASCII codes and uses rest of the codes (128) for additional Indian language character set. Additional codes have been assigned in the upper region (160–255) for the '*aksharas*' of the language.

2.1.3 UNICODE

There were many encoding schemes, for character sets of different languages. But they were not able to communicate with each other, as each of them represented characters in their own ways. Hence, text created using one encoding scheme was not recognised by another machine using different encoding scheme.

Therefore, a standard called UNICODE has been developed to incorporate all the characters of every written language of the world. UNICODE provides a unique number for every character, irrespective of device (server, desktop, mobile), operating system (Linux, Windows, iOS) or software application (different

Think and Reflect

Do we need to install some additional tool or font to type in an Indian language using UNICODE?

Activity 2.1

Explore and list down two font names for typing in any three Indian languages in UNICODE.

Think and Reflect

Why a character in UTF 32 takes more space than in UTF 16 or UTF 8?

browsers, text editors, etc.). Commonly used UNICODE encodings are UTF-8, UTF-16 and UTF-32. It is a superset of ASCII, and the values 0–128 have the same character as in ASCII. Unicode characters for Devanagari script is shown in Table 2.3. Each cell of the table contains a character along with its equivalent hexadecimal value.

Table 2.3 Unicode table for the Devanagari script

ऀ 0900	ँ 0901	ं 0902	ः 0903	ऄ 0904	अ 0905	आ 0906	इ 0907	ई 0908	उ 0909	ऊ 090A	ऋ 090B	ऌ 090C	ऍ 090D	ऎ 090E	ए 090F
ऐ 0910	ऑ 0911	ऒ 0912	ओ 0913	औ 0914	क 0915	ख 0916	ग 0917	घ 0918	ङ 0919	च 091A	छ 091B	ज 091C	झ 091D	ञ 091E	ट 091F
ठ 0920	ड 0921	ढ 0922	ण 0923	त 0924	थ 0925	द 0926	ध 0927	न 0928	ऩ 0929	प 092A	फ 092B	ब 092C	भ 092D	म 092E	य 092F
र 0930	ऱ 0931	ल 0932	ळ 0933	ऴ 0934	व 0935	श 0936	ष 0937	स 0938	ह 0939	ऺ 093A	ऻ 093B	़ 093C	ऽ 093D	ा 093E	ि 093F
ी 0940	ु 0941	ू 0942	ृ 0943	ॄ 0944	ॅ 0945	ॆ 0946	े 0947	ै 0948	ॉ 0949	ॊ 094A	ो 094B	ौ 094C	् 094D	ॎ 094E	ॏ 094F
ॐ 0950	॑ 0951	॒ 0952	॓ 0953	॔ 0954	ॕ 0955	ॖ 0956	ॗ 0957	क़ 0958	ख़ 0959	ग़ 095A	ज़ 095B	ड़ 095C	ढ़ 095D	फ़ 095E	य़ 095F
ॠ 0960	ॡ 0961	ॢ 0962	ॣ 0963	। 0964	॥ 0965	० 0966	१ 0967	२ 0968	३ 0969	४ 096A	५ 096B	६ 096C	७ 096D	८ 096E	९ 096F
॰ 0970	ॱ 0971	ॲ 0972	ऻ 0973	ॴ 0974	ॵ 0975	ॶ 0976	ॷ 0977	ॸ 0978	ॹ 0979	ॺ 097A	ॻ 097B	ॼ 097C	ॽ 097D	ॾ 097E	ॿ 097F

2.2 NUMBER SYSTEM

Till now, we have learnt that each key (representing character, special symbol, function keys, etc.) of the keyboard is internally mapped to an ASCII code following an encoding scheme. This encoded value is further converted to its equivalent binary representation so that the computer can understand it. In Figure 2.1, the code for character "A" belongs to the decimal number system and its equivalent binary value belongs to the binary number system. A number system is a method to represent (write) numbers.

Every number system has a set of unique characters or literals. The count of these literals is called the radix or base of the number system. The four different number systems used in the context of computer are shown in Figure 2.2. These number systems are explained in subsequent sections.

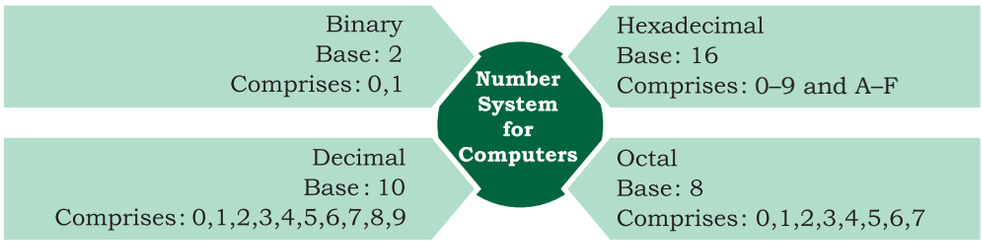


Figure 2.2: Four different number systems

Number systems are also called positional number system because the value of each symbol (i.e., digit and alphabet) in a number depends upon its position within the number. Number may also have a fractional part similar to decimal numbers used by us. The symbol at the right most position in the integer part in a given number has position 0. The value of position (also called position value) in the integer part increases from right to left by 1. On the other hand, the first symbol in the fraction part of the number has position number –1, which decreases by 1 while reading fraction part from left to right. Each symbol in a number has a positional value, which is computed using its position value and the base value of the number system. The symbol at position number 3 in a decimal system with base 10 has positional value 10^3. Adding the product of positional value and the symbol value results in the given number. Figure 2.3 shows the computation of decimal number 123.45 using its positional value.

Digit	1	2	3	.	4	5
Position Number	2	1	0		–1	–2
Positional Value	$(10)^2$	$(10)^1$	$(10)^0$		$(10)^{-1}$	$(10)^{-2}$

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

$$1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 + 4 \times 10^{-1} + 5 \times 10^{-2} = (123.45)_{10}$$

Figure 2.3: Computation of decimal number using its positional value

2.2.1 Decimal Number System

The decimal number system is used in our day-to-day life. It is known as base-10 system since 10 digits (0 to 9) are used. A number is presented by its two values — symbol value (any digit from 0 to 9) and positional value (in terms of base value). Figure 2.4 shows the integer and fractional part of decimal number 237.25 alongwith computation of the decimal number using positional values.

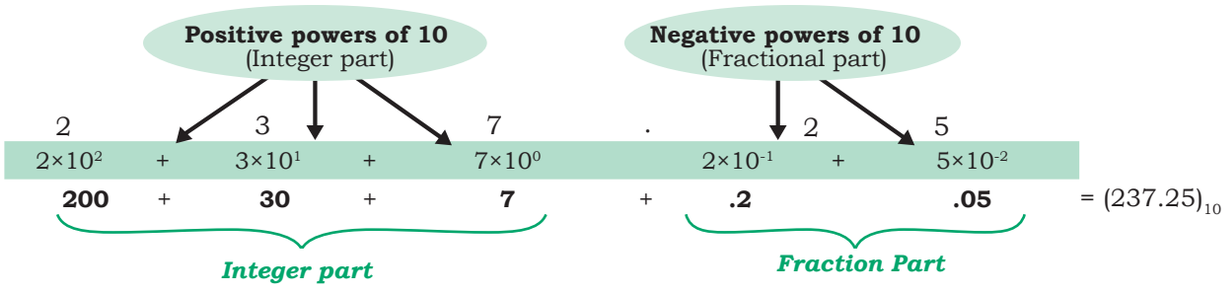


Figure 2.4: Positional value for digits of decimal number represented as power of base 10

Base value of a number system is used to distinguish a number in one number system from another number system. Base value is written as the subscript of the given number. For example, $(70)_8$ represents 70 as octal number and $(70)_{10}$ denotes 70 as decimal number.

2.2.2 Binary Number System

The ICs (Integrated Circuits) in a computer are made up of a large number of transistors which are activated by the electronic signals (low/high) they receive. The ON/high and OFF/low state of a transistor is represented using the two digits 1 and 0, respectively. These two digits 1 and 0 form the binary number system. This system is also referred as base-2 system as it has two digits only. Some examples of binary numbers are 1001011, 1011.101, 111111.01. A binary number can be mapped to an equivalent decimal number that can be easily understood by the human.

Table 2.4 Binary value for (0–9) digits of decimal number system

Decimal	Binary
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001

2.2.3 Octal Number System

With increase in the value of a decimal number, the number of bits (0/1) in its binary representation also increases. Sometimes, a binary number is so large that it becomes difficult to manage. Octal number system was devised for compact representation of the binary numbers. Octal number system is called base-8 system

as it has total eight digits (0-7), and positional value is expressed in powers of 8. Three binary digits (8=23) are sufficient to represent any octal digit. Table 2.5 shows the decimal and binary equivalent of 8 octal digits. Examples of octal numbers are $(237.05)_8$, $(13)_8$, and $(617.24)_8$.

Table 2.5 Decimal and binary equivalent of octal numbers 0–7

Octal Digit	Decimal Value	3 -bit Binary Number
0	0	000
1	1	001
2	2	010
3	3	011
4	4	100
5	5	101
6	6	110
7	7	111

2.2.4 Hexadecimal Number System

Hexadecimal numbers are also used for compact representation of binary numbers. It consists of 16 unique symbols (0–9, A–F), and is called base-16 system. In hexadecimal system, each alphanumeric digit is represented as a group of 4 binary digits because 4 bits (2^4=16) are sufficient to represent 16 alphanumeric symbols. Note here that the decimal numbers 10 through 15 are represented by the letters A through F. Examples of Hexadecimal numbers are $(23A.05)_{16}$, $(1C3)_{16}$, $(619B.A)_{16}$. Table 2.6 shows decimal and binary equivalent of 16 alphanumeric symbols used in hexadecimal number system.

Table 2.6 Decimal and binary equivalent of hexadecimal numbers 0–9, A–F

Hexadecimal Symbol	Decimal Value	4-bit Binary Number
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

2.2.5 Applications of Hexadecimal Number System

- Main memory is made up of memory locations where each location has a unique address. Usually, size of a memory address is 16-bit or 32-bit. To access 16-bit memory address, a programmer has to use 16 binary bits, which is difficult to deal with. To simplify the address representation, hexadecimal and octal numbers are used. Let us consider a 16-bit memory address 1100000011110001. Using the hexadecimal notation, this address is mapped to C0F1 which is more easy to remember. The equivalent octal representation for this 16-bit value is 140361.
- Hexadecimal numbers are also used for describing the colours on the webpage. Each colour is made up of three primary colours red, green and blue, popularly called RGB (in short). In most colour maps, each colour is usually chosen from a palette of 16 million colours. Therefore, 24 bits are required for representing each colour having three components (8 bits for Red, 8 bits for Green, 8 bits for Blue component). It is difficult to remember 24-bit binary colour code. Therefore, colour codes are written in hexadecimal form for compact representation. For example, 24-bit code for RED colour is 11111111,00000000,00000000. The equivalent hexadecimal notation is (FF,00,00), which can be easily remembered and used. Table 2.7 shows examples of some colours represented with decimal, binary and hexadecimal numbers.

Table 2.7 Colour codes in decimal, binary and hexadecimal numbers

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

2.3 CONVERSION BETWEEN NUMBER SYSTEMS

In the previous section, we learnt about different number systems used in computers. Now, let us learn how to convert a number from one number system to another number system for better understanding of the number representation in computers. Decimal number

system is most commonly used by humans, but digital systems understand binary numbers; whereas Octal and hexadecimal number systems are used to simplify the binary representation for us to understand.

2.3.1 Conversion from Decimal to other Number Systems

To convert a decimal number to any other number system (binary, octal or hexadecimal), use the steps given below.

Step 1: Divide the given number by the base value (b) of the number system in which it is to be converted

Step 2: Note the remainder

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

Step 4: Write the noted remainders in the reverse order (from bottom to top)

(A) Decimal to Binary Conversion

Since the base value of binary system is 2, the decimal number is repeatedly divided by 2 following the steps given in above till the quotient is 0. Record the remainder after each division and finally write the remainders in reverse order in which they are computed.

In Figure 2.1 you saw that the binary equivalent of 65 is $(1000001)_2$. Let us now convert a decimal value to its binary representation and verify that the binary equivalent of $(65)_{10}$ is $(1000001)_2$.

Activity 2.2

Convert the following decimal numbers in the form understood by computer.

(i) $(593)_{10}$ (ii) $(326)_{10}$
(iii) $(79)_{10}$

Activity 2.3

Express the following decimal numbers into octal numbers.

(i) $(913)_{10}$
(ii) $(845)_{10}$
(iii) $(66)_{10}$

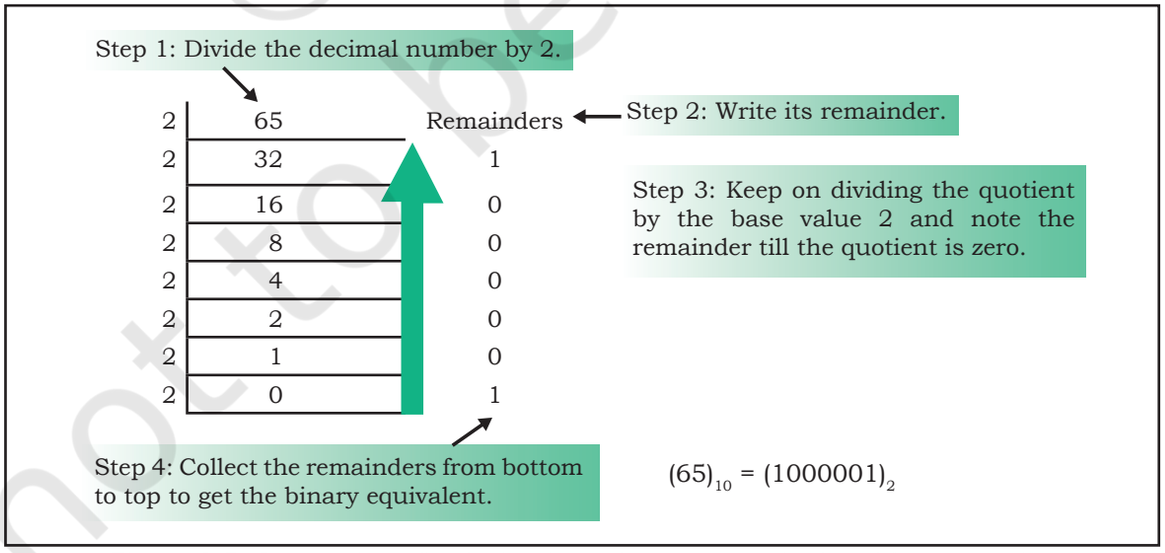


Figure 2.5: Conversion of a decimal number to its equivalent binary number

Example 2.3 Convert $(122)_{10}$ to binary number.

2	122	Remainders
2	61	0
2	30	1
2	15	0
2	7	1
2	3	1
2	1	1
	0	1

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

(B) Decimal to Octal Conversion

Since the base value of octal is 8, the decimal number is repeatedly divided by 8 to obtain its equivalent octal number.

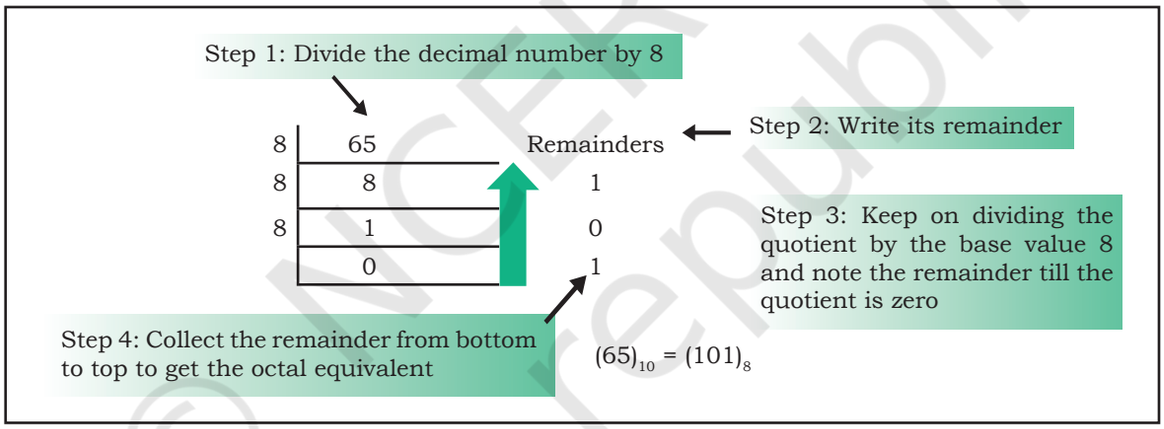


Figure 2.6: Conversion of a decimal number to its equivalent octal number

The octal equivalent of letter "A" by using its ASCII code value $(65)_{10}$ is calculated as shown in Figure 2.6.

Example 2.4 Convert $(122)_{10}$ to octal number.

8	122	Remainders
8	15	2
8	1	7
	0	1

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

(C) Decimal to Hexadecimal Conversion

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

equivalent of letter 'A' using its ASCII code $(65)_{10}$ is calculated as shown in Figure 2.7.

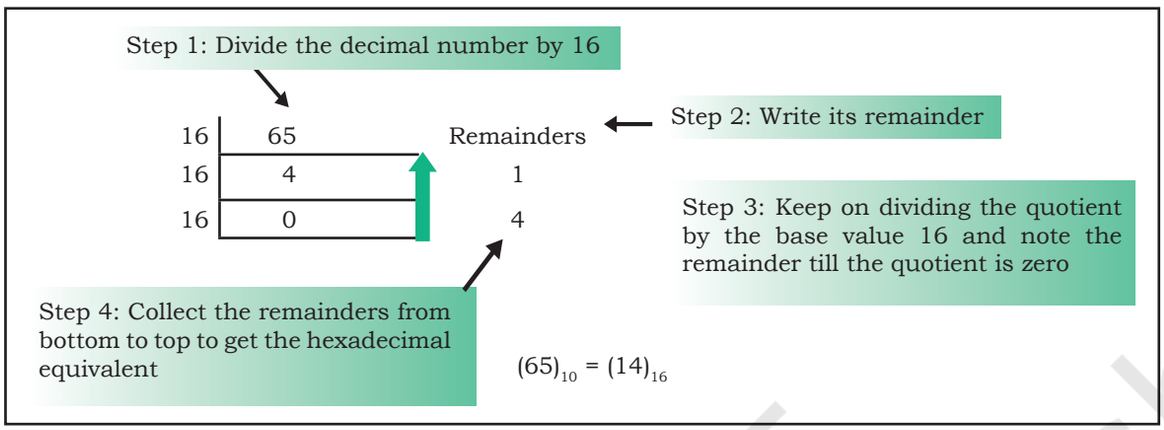


Figure 2.7: Conversion of a decimal number to its equivalent hexadecimal number

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

16 | 122 — Remainders
16 | 7 — A (Hexadecimal symbol equivalent to decimal number 10)
0 — 7

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

2.3.2 Conversion from other Number Systems to Decimal Number System

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

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

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

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

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

Activity 2.4

Convert the following numbers into decimal numbers.

(i) $(110101)_2$
(ii) $(1703)_8$
(iii) $(COF5)_{16}$

(A) Binary Number to Decimal Number

Since binary number system has base 2, the positional values are computed in terms of powers of 2. Using the above mentioned steps we can convert

a binary number to its equivalent decimal value as shown below:

Example 2.6 Convert $(1101)_2$ into decimal number.

Digit	1	1	0	1
Position Number	3	2	1	0
Positional Value	2^3	2^2	2^1	2^0
Decimal Number	1×2^3 +	1×2^2 +	0×2^1 +	1×2^0 = 8 + 4 + 0 + 1 = $(13)_{10}$

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

(B) Octal Number to Decimal Number

The following example shows how to compute the decimal equivalent of an octal number using base value 8.

Example 2.7 Convert $(257)_8$ into decimal number.

Digit	2	5	7
Position Number	2	1	0
Positional Value	8^2	8^1	8^0
Decimal Number	2×8^2	5×8^1 +	7×8^0 =128 + 40 + 7 = $(175)_{10}$

Why 3 bits in a binary number are grouped together to get octal number?
The base value of octal number system is 8. Convert value 8 in terms of exponent of 2, i.e., $8=2^3$. Hence, three binary digits are sufficient to represent all 8 octal digits. Simply stated, count all possible combinations of three binary digits, which are 2x2x2=8. Therefore, 3 bits are sufficient to represent any octal digit. Hence, 3-bit groups in a binary number are formed to get equivalent octal number.

(C) Hexadecimal Number to Decimal Number

For converting a hexadecimal number into decimal number, use steps given in this section with base value 16 of hexadecimal number system. Use decimal value equivalent to alphabet symbol of hexadecimal number in the calculation, as shown in Table 2.6.

Example 2.8 Convert $(3A5)_{16}$ into decimal number.

Digit	3	A	5
Position Number	2	1	0
Positional Value	16^2	16^1	16^0
Decimal Number	3×16^2	10×16^1 +	5×16^0 =768 + 160 + 5 = $(933)_{10}$

Note: *Use Table 2.5 for decimal value of alphabets*

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

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

(A) Binary Number to Octal Number

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

Example 2.9 Convert $(10101100)_2$ to octal number.

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

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

(B) Octal Number to Binary Number

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

Example 2.10 Convert $(705)_8$ to binary number.

Octal digits	7	0	5
Write 3-bits binary value for each digit	111	000	101

Therefore, $(705)_8 = (111000101)_2$

(C) Binary Number to Hexadecimal Number

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

Example 2.11 Convert $(0110101100)_2$ to hexadecimal number.

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

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

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

Think and Reflect

While converting the fractional part of a decimal number to another number system, why do we write the integer part from top to bottom and not other way?

for each group 1 A C

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

Activity 2.5

Write binary representation of the following numbers.
(i) $(F018)_{16}$
(ii) $(172)_{16}$
(iii) $(613)_8$

(D) Hexadecimal Number to Binary Number

Each hexadecimal symbol is an encoding for a 4-digit binary number. Hence, the binary equivalent of a hexadecimal number is obtained by substituting 4-bit binary equivalent of each hexadecimal digit and combining them together (see Table 2.5).

Example 2.12 Convert $(23D)_{16}$ to binary number.

Hexadecimal digits 2 3 D

Write 4-bit binary value for each digit 0010 0011 1101

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

2.3.4 Conversion of a Number with Fractional Part

Till now, we largely dealt with different conversions for whole number. In this section, we will learn about conversion of numbers with a fractional part.

(A) Decimal Number with Fractional Part to another Number System

To convert the fractional part of a decimal number to another number system with base value b, repeatedly multiply the fractional part by the base value b till the fractional part becomes 0. Use integer part from top to bottom to get equivalent number in that number system. If the fractional part does not become 0 in successive multiplication, then stop after, say 10 multiplications. In some cases, fractional part may start repeating, then stop further calculation.

Example 2.13 Convert $(0.25)_{10}$ to binary.

	Integer part
$0.25 \times 2 = 0.50$	0
$0.50 \times 2 = 1.00$	1

Since the fractional part is 0, the multiplication is stopped. Write the integer part from top to bottom to get binary number for the fractional part.

Therefore, $(0.25)_{10} = (0.01)_2$

Example 2.14 Convert $(0.675)_{10}$ to binary.

	Integer part
0.675 × 2 = 1.350	1
0.350 × 2 = 0.700	0
0.700 × 2 = 1.400	1
0.400 × 2 = 0.800	0
0.800 × 2 = 1.600	1
0.600 × 2 = 1.200	1
0.200 × 2 = 0.400	0

Since the fractional part (.400) is the repeating value in the calculation, the multiplication is stopped. Write the integer part from top to bottom to get binary number for the fractional part.

Therefore, (0.675)10= (0.1010110)2

Example 2.15 Convert $(0.675)_{10}$ to octal.

	Integer part
0.675 × 8 = 5.400	5
0.400 × 8 = 3.200	3
0.200 × 8 = 1.600	1
0.600 × 8 = 4.800	4
0.800 × 8 = 6.400	6

Since the fractional part (.400) is repeating, the multiplication is stopped. Write the integer part from top to bottom to get octal number for the fractional part.

Therefore, $(0.675)_{10} = (0.53146)_8$

Example 2.16 Convert $(0.675)_{10}$ to hexadecimal form.

	Integer part	
0.675 × 16 = 10.800	A	(Hexadecimal symbol for 10)
0.800 × 16 = 12.800	C	(Hexadecimal symbol for 12)

Since the fractional part (.800) is repeating, the multiplication is stopped. Write the integer part from top to bottom to get hexadecimal equivalent for the fractional part.

Therefore, $(0.675)_{10} = (0.\ AC)_{16}$

(B) Non-decimal Number with Fractional Part to Decimal Number System

Compute positional value of each digit in the given number using its base value. Add the product of

positional value and the digit to get the equivalent decimal number with fractional part.

Example 2.17 Convert $(100101.101)_2$ into decimal.

Digit	1	0	0	1	0	1	.	1	0	1
Fractional Value	2^5	2^4	2^3	2^2	2^1	2^0		2^{-1}	2^{-2}	2^{-3}
Decimal Value	1×2^5	$+0\times2^4$	$+0\times2^3$	$+1\times2^2$	$+0\times2^1$	$+1\times2^0$	+	1×2^{-1}	+ 0×2^{-2}	$+1\times2^{-3}$
	= 32	+ 0	+ 0	+ 4	+ 0	+ 1	+	0.5	+ 0	+ 0.125
				37					0.625	

= 37 + 0.625

Therefore, $(100101.101)_2 = (37.625)_{10}$

Example 2.18 Convert $(605.12)_8$ into decimal number.

Octal Digits	6	0	5	.	1	2
Positional Value	8^2	8^1	8^0		8^{-1}	8^{-2}
Decimal number	6×8^2	$+ 0\times8^1$	$+ 5\times8^0$	+	1×8^{-1}	$+ 2\times8^{-2}$
=	1024	+ 0	+ 5	+	.125	+ .03125
		1029			0.15625	

= 1029 + 0.15625

Therefore, $(605.12)_8 = (1029.15625)_{10}$

(C) Fractional Binary Number to Octal or Hexadecimal Number

To convert the fractional binary number into octal or hexadecimal value, substitute groups of 3-bit or 4-bit in integer part by the corresponding digit. Similarly, make groups of 3-bit or 4-bit for fractional part starting from left to right, and substitute each group by its equivalent digit or symbol in Octal or Hexadecimal number system. Add 0s at the end of the fractional part to make a perfect group of 3 or 4 bits.

Example 2.19 Convert $(10101100.01011)_2$ to octal number.

Make perfect group of 3 bits $\underline{010}\ \underline{101}\ \underline{100}\ .\ \underline{010}\ \underline{110}$
Write octal symbol for each group 2 5 4 . 2 6

Therefore, $(10101100.01011)_2 = (254.26)_8$

Note: Make 3-bit groups from right to left for the integer part and left to right for the fractional part.

Example 2.20 Convert $(10101100.010111)_2$ to hexadecimal number

Make perfect group of 4 bits 1010 1100 . 0101 1100
Write hexadecimal symbol for each group A C . 5 C

Therefore, $(10101100.010111)_2 = (AC.5C)_{16}$

Summary

- Encoding scheme maps text into the codes that facilitate communication among computers.
- Textual data is encoded using ASCII, ISCII or Unicode.
- Unicode scheme is a character encoding standard which can encode all the characters of almost all languages of the world.
- Computer being a digital system understands only binary numbers which are 0 and 1.
- Encoded text is converted to binary form for processing by the computer.
- Octal and hexadecimal number systems are used to simplify the binary coded representation as they allow grouping of 3 or 4 bits of binary numbers each, respectively.

Exercise

1. Write base values of binary, octal and hexadecimal number system.
2. Give full form of ASCII and ISCII.
3. Try the following conversions.
   (i) $(514)_8 = (?)_{10}$ (iv) $(4D9)_{16} = (?)_{10}$
   (ii) $(220)_8 = (?)_2$ (v) $(11001010)_2 = (?)_{10}$
   (iii) $(76F)_{16} = (?)_{10}$ (vi) $(1010111)_2 = (?)_{10}$
4. Do the following conversions from decimal number to other number systems.
   (i) $(54)_{10} = (?)_2$ (iv) $(889)_{10} = (?)_8$
   (ii) $(120)_{10} = (?)_2$ (v) $(789)_{10} = (?)_{16}$
   (iii) $(76)_{10} = (?)_8$ (vi) $(108)_{10} = (?)_{16}$
5. Express the following octal numbers into their equivalent decimal numbers.
   (i) 145 (ii) 6760 (iii) 455 (iv) 10.75

NOTES

6. Express the following decimal numbers into hexadecimal numbers.
   (i) 548 (ii) 4052 (iii) 58 (iv) 100.25
7. Express the following hexadecimal numbers into equivalent decimal numbers.
   (i) 4A2 (ii) 9E1A (iii) 6BD (iv) 6C.34
8. Convert the following binary numbers into octal and hexadecimal numbers.
   (i) 1110001000 (ii) 110110101 (iii) 1010100
   (iv) 1010.1001
9. Write binary equivalent of the following octal numbers.
   (i) 2306 (ii) 5610 (iii) 742 (iv) 65.203
10. Write binary representation of the following hexadecimal numbers.
    (i) 4026 (ii) BCA1 (iii) 98E (iv) 132.45
11. How does computer understand the following text? (hint: 7 bit ASCII code).
    (i) HOTS (ii) Main (iii) CaSe
12. The hexadecimal number system uses 16 literals (0–9, A–F). Write down its base value.
13. Let X be a number system having B symbols only. Write down the base value of this number system.
14. Write the equivalent hexadecimal and binary values for each character of the phrase given below.
    " हम सब एक"
15. What is the advantage of preparing a digital content in Indian language using UNICODE font?
16. Explore and list the steps required to type in an Indian language using UNICODE.
17. Encode the word 'COMPUTER' using ASCII and convert the encode value into binary values.

Chapter 3

Emerging Trends

11120CH03

"Computer science is no more about computers than astronomy is about telescopes"

–Edsger Dijkstra

3.1 Introduction

Computers have been around for quite some time now. New technologies and initiatives emerge with each passing day. In order to understand the existing technologies and have a better view of the developments around us, we must keep an eye on the emerging trends. Many new technologies are introduced almost every day. Some of these do not succeed and fade away over time. Some of these new technologies prosper and persist over time, gaining attention from users. Emerging trends are the state-of-the-art technologies, which gain popularity and set a new trend among users. In this chapter, we will learn about some emerging trends that will make a huge impact (in the future) on digital economy and interaction in digital societies.

3.2 Artificial Intelligence (AI)

Have you ever wondered how maps in your smartphone are able to guide you to take the fastest route to your destination by analysing real time data, such as traffic congestion? On uploading a photo on a social networking site, has it ever happened that your friends in the photograph were recognised and tagged automatically? These are some of the examples of application of Artificial Intelligence. The intelligent digital personal assistants like Siri, Google Now, Cortana, Alexa are all powered by AI. Artificial Intelligence endeavours to simulate the natural intelligence of human beings into machines, thus making them behave intelligently. An intelligent machine is supposed to imitate some of the cognitive functions of humans like learning, decision-making and problem solving. In order to make machines perform tasks with minimum human intervention, they are programmed to create a knowledge base and make

In this chapter

A knowledge base is a store of information consisting of facts, assumptions and rules which an AI system can use for decision making.

decisions based on it. AI system can also learn from past experiences or outcomes to make new decisions.

3.2.1 Machine Learning

Machine Learning is a subsystem of Artificial Intelligence, wherein computers have the ability to learn from data using statistical techniques, without being explicitly programmed by a human being. It comprises algorithms that use data to learn on their own and make predictions. These algorithms called models, are first trained and tested using a training data and testing data, respectively. After successive trainings, once these models are able to give results to an acceptable level of accuracy, they are used to make predictions about new and unknown data.

3.2.2 Natural Language Processing (NLP)

Activity 3.1

Find out how NLP is helping differently-abled persons?

The predictive typing feature of search engine that helps us by suggesting the next word in the sentence while typing keywords and the spell checking features are examples of Natural Language Processing (NLP). It deals with the interaction between human and computers using human spoken languages, such as Hindi, English, etc.

In fact it is possible to search the web or operate or control our devices using our voice. All this has been possible by NLP. An NLP system can perform text-to-speech and speech-to-text conversion as depicted in Figure 3.1.

Figure 3.1: Use of natural language processing

Machine translation is a rapidly emerging field where machines are already able to translate texts from one language to another with fair amount of correctness. Another emerging application area is automated customer service where a computer software can interact with customers to serve their queries or complaints.

3.2.3 Immersive Experiences

With the three-dimensional (3D) videography, the joy of watching movies in theatres has reached to a new level. Video games are also being developed to

provide immersive experiences to the player. Immersive experiences allow us to visualise, feel and react by stimulating our senses. It enhances our interaction and involvement, making them more realistic and engaging. Immersive experiences have been used in the field of training, such as driving simulators (Figure 3.2), flight simulator and so on. Immersive experience can be achieved using virtual reality and augmented reality.

Figure 3.2: Driving simulator

(A) Virtual Reality

Everything that we experience in our reality is perceived through our senses. From this came the idea that if we can present our senses with made-up or non-real information, our perception of reality would also alter in response to that. Virtual Reality (VR) is a three-dimensional, computer-generated situation that simulates the real world. The user can interact with and explore that environment by getting immersed in it while interacting with the objects and other actions of the user. At present, it is achieved with the help of VR Headsets. In order to make the experience of VR more realistic, it promotes other sensory information like sound, smell, motion, temperature, etc. It is a comparatively new field and has found its applications in gaming (Figure 3.3), military training, medical procedures, entertainment, social science and psychology, engineering and other areas where simulation is needed for a better understanding and learning.

Figure 3.3: Virtual Reality Headset

(B) Augmented Reality

The superimposition of computer generated perceptual information over the existing physical surroundings is called as Augmented Reality (AR). It adds components of the digital world to the physical world, along with the associated tactile and other sensory requirements, thereby making the environment interactive and digitally manipulable. Users can access information about the nearest places with reference to their current location. They can get information about places and choose on the basis of user reviews. With help of location-based AR App, travellers can access real-time information of historical places just by pointing their camera viewfinder to subjects as depicted in Figure 3.4. Location-based AR apps are major forms of AR apps.

Figure 3.4: Location-based Augmented Reality

Unlike Virtual Reality, the Augmented Reality does not create something new, it just alters or augments the perception of the underlying physical world through additional information.

3.2.4 Robotics

Robotics is an interdisciplinary branch of technology requiring applications of mechanical engineering, electronics, and computer science, among others. Robotics is primarily concerned with the design, fabrication, operation, and application of robots.

A robot is basically a machine capable of carrying out one or more tasks automatically with accuracy and precision. Unlike other machines, a robot is programmable by a computer, which means it can follow the instructions given through computer programs. Robots were initially conceptualised for doing repetitive industrial tasks that are boring or stressful for humans or were labour-intensive. Sensors are one of the prime components of a robot. Robot can be of many types, such as wheeled robots, legged robots, manipulators and humanoids. Robots that resemble humans are known as humanoids. Robots are being used in industries, medical science, bionics, scientific research, military, etc. Some examples are:

- NASA's Mars Exploration Rover (MER) mission is a robotic space mission to study about the planet Mars (Figure 3.5).

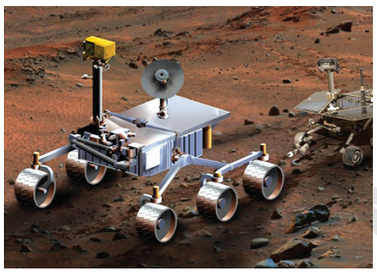


Figure 3.5: NASA's Mars Exploration Rover (MER)

Figure 3.6: Sophia is a humanoid

Figure 3.7: Drone

- Sophia is a humanoid that uses artificial intelligence, visual data processing, facial recognition and also imitates human gestures and facial expressions, as shown in Figure 3.6.
- A drone is an unmanned aircraft which can be remotely controlled or can fly autonomously through software-controlled flight plans in their embedded systems, working in conjunction with onboard sensors and GPS (Figure 3.7). They are being used in many fields, such as journalism, filming and aerial photography, shipping or delivery at short distances, disaster management, search and rescue operations, healthcare, geographic mapping and structural safety inspections, agriculture, wildlife monitoring or pooching, besides law-enforcement and border patrolling.

Think and Reflect

Can a drone be helpful in the event of a natural calamity?

Activity 3.2

Find out what role are robots playing in the medical field?

3.3 BIG DATA

With technology making an inroad into almost every sphere of our lives, data is being produced at a colossal rate. Today, there are over a billion Internet users, and a majority of the world's web traffic is coming from smartphones. Figure 3.8 shows that at the current pace, around 2.5 quintillion bytes of data are created each day, and the pace is increasing with the continuous evolution of the Internet of Things (IoT).

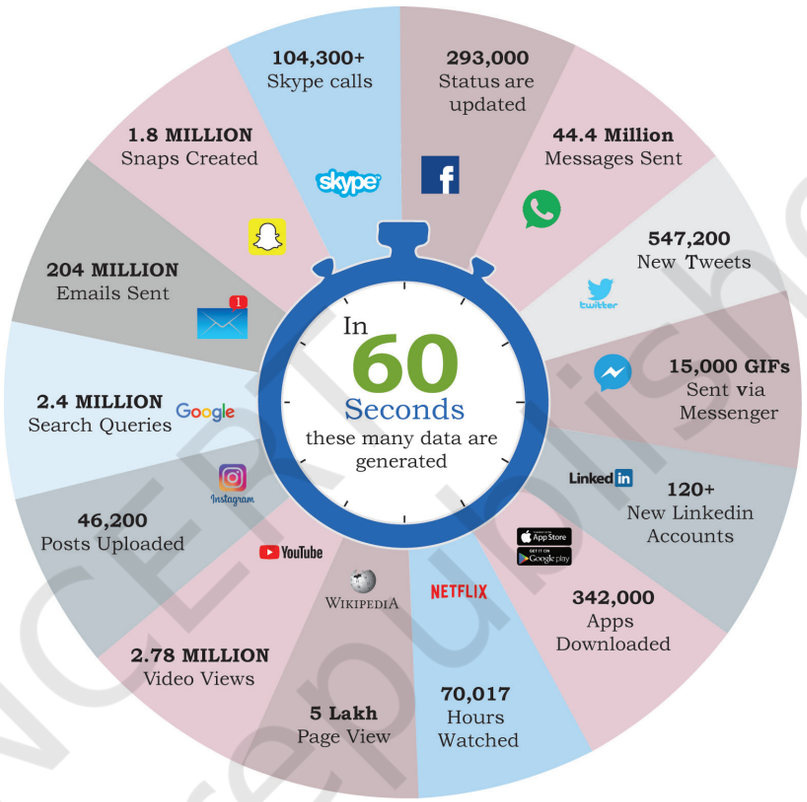


Figure 3.8: Sources of big data (numbers are approximate)

This results in the generation of data sets of enormous volume and complexity called *Big Data*. Such data cannot be processed and analysed using traditional data processing tools as the data is not only voluminous, but also unstructured like our posts, instant messages and chats, photographs that we share through various sites, our tweets, blog articles, news items, opinion polls and their comments, audio/video chats, etc. Big Data not only represents voluminous data, it also involves various challenges like integration, storage, analysis, searching, processing, transfer, querying and visualisation of such data. Big data sometimes hold rich information and knowledge which is of high business value, and therefore there is a keen effort in developing software and methods to process and analyse big data.

Think and Reflect

How are your digital activities contributing to generation of Big data?

3.3.1 Characteristics of Big Data

Big data exhibits following five characteristics shown in Figure 3.9, that distinguish it from traditional data.

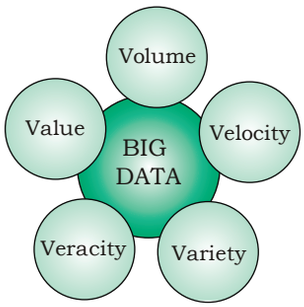


Figure 3.9: Characteristics of big data

(A) Volume

The most prominent characteristic of big data is its enormous size. If a particular dataset is of such large size that it is difficult to process it with traditional DBMS tools, it can be termed as big data.

(B) Velocity

It represents the rate at which the data under consideration are being generated and stored. Big data has an exponentially higher rate of generation than traditional data sets.

(C) Variety

It asserts that a dataset has varied data, such as structured, semi-structured and unstructured data. Some examples are text, images, videos, web-pages and so on.

(D) Veracity

Big data can be sometimes inconsistent, biased, noisy or there can be abnormality in the data or issues with the data collection methods. Veracity refers to the trustworthiness of the data because processing such incorrect data can give wrong results or mislead the interpretations.

(E) Value

Big data is not only just a big pile of data, but also possess to have hidden patterns and useful knowledge which can be of high business value. But as there is cost of investment of resources in processing big data, we should make a preliminary enquiry to see the potential of the big data in terms of value discovery or else our efforts could be in vain.

3.3.2 Data Analytics

"Data analytics is the process of examining data sets in order to draw conclusions about the information they contain, with the aid of specialised systems and software."

Data analytics technologies and techniques are becoming popular day-by-day. They are used in commercial industries to enable organisations to make more informed business decisions. In the field of science and technology, it can be useful for researchers to verify or disprove scientific models, theories and hypotheses. Pandas is a library of the programming language

Python that can be used as a tool to make data analysis much simpler.

3.4 INTERNET OF THINGS (IoT)

The term computer network that we commonly use is the network of computers. Such a network consists of a laptop, desktop, server, or a portable device like tablet, smartphone, smart watch, etc., connected through wire or wireless. We can communicate between these devices using Internet or LAN. Now imagine what if our bulbs, fans and refrigerator also became a part of this network. How will they communicate with each other, and what will they communicate? Think about the advantages and tasks that can be accomplished if all these devices with smart connectivity features are able to communicate amongst themselves and we are also able to communicate with them using computers or smartphones.

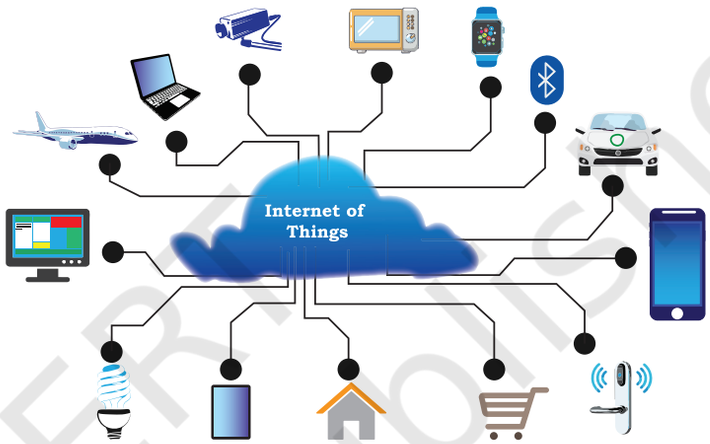


Figure 3.10: Internet of Things (IoT)

The 'Internet of Things' is a network of devices that have an embedded hardware and software to communicate (connect and exchange data) with other devices on the same network as shown in Figure 3.10. At present, in a typical household, many devices have advanced hardware (microcontrollers) and software. These devices are used in isolation from each other, with maximum human intervention needed for operational directions and input data. IoT tends to bring together these devices to work in collaboration and assist each other in creating an intelligent network of things. For example, if a microwave oven, an air conditioner, door lock, CCTV camera or other such devices are enabled to connect to the Internet, we can access and remotely control them on-the-go using our smartphone.

Activity 3.3

Explore and list a few IoT devices available in the market.

3.4.1 Web of Things (WoT)

Internet of Things allows us to interact with different devices through Internet with the help of smartphones

Activity 3.4

We use GPS to navigate outdoors. VPS is another emerging trend that uses Augmented Reality. Explore and find its other utilities.

or computers, thus creating a personal network. But to interact with 'n' number of different devices, we need to install 'n' different apps. Wouldn't it be convenient to have one interface to connect all the devices? The web is already being used as a system to communicate with each other. So, will it be possible to use the web in such a way that all things can communicate with each other in the most efficient manner by integrating them together? Web of Things (WoT) allows use of web services to connect anything in the physical world, besides human identities on web. It will pave way for creating smart homes, smart offices, smart cities and so on.

3.4.2 Sensors

What happens when you hold your mobile vertically or horizontally? The display also changes to vertical or horizontal with respect to the way we hold our mobile. This is possible with the help of two sensors, namely accelerometer and gyroscope (gyro). The accelerometer sensor in the mobile phones detects the orientation of the phone. The Gyroscope sensors, tracks rotation or twist of your hand and add to the information supplied by the accelerometer.

Sensors are very commonly used as monitoring and observing elements in real world applications. The evolution of smart electronic sensors is contributing in a large way to the evolution of IoT. It will lead to creation of new sensor-based, intelligent systems.

A smart sensor is a device that takes input from the physical environment and uses built-in computing resources to perform predefined functions upon detection of specific input and then process data before passing it on.

3.4.3 Smart Cities

With rapid urbanisation, the load on our cities are increasing day-by-day, and there are challenges in management of resources like land water, waste, air pollution, health and sanitation, traffic congestions, public safety and security, besides the overall city infrastructures including road, rail, bridge, electricity, subways, disaster management, sports facilities, etc. These challenges are forcing many city planners around the world to look for smarter ways to manage them and make cities sustainable and livable.

Think and Reflect

What are your ideas of transforming your city into a smart city?

The idea of a smart city as shown in Figure 3.11 makes use of computer and communication technology along with IoT to manage and distribute resources efficiently. The smart building shown here uses sensors to detect earthquake tremors and then warn nearby buildings so that they can prepare themselves accordingly. The smart bridge uses wireless sensors to detect any loose bolt, cable or crack. It alerts concerned authorities through SMS. The smart tunnel also uses wireless sensors to detect any leakage or congestion in the tunnel. This information can be sent as wireless signals across the network of sensor nodes to a centralised computer for further analysis.

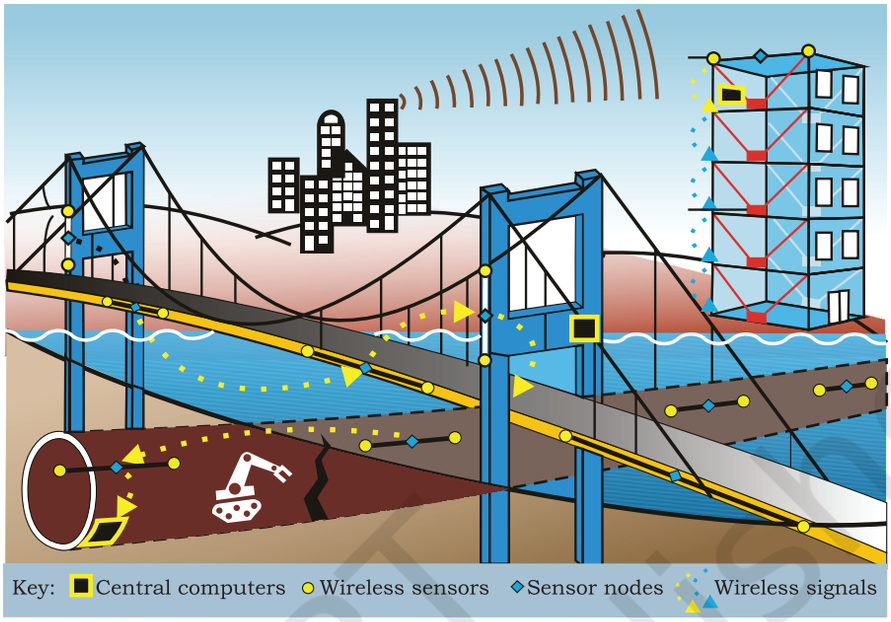


Figure 3.11: Smart city

Every sphere of life in a city like transportation systems, power plants, water supply networks, waste management, law enforcement, information systems, schools, libraries, hospitals and other community services work in unison to optimise the efficiency of city operations and services.

3.5 CLOUD COMPUTING

Cloud computing is an emerging trend in the field of information technology, where computer-based services are delivered over the Internet or the cloud, and it is accessible to the user from anywhere using any device. The services comprise software, hardware (servers), databases, storage, etc. These resources are provided by companies called cloud service providers and usually charge on a pay per use basis, like the way we pay for electricity usage. We already use cloud services while storing our pictures and files as backup on Internet, or host a website on the Internet. Through

cloud computing, a user can run a bigger application or process a large amount of data without having the required storage or processing power on their personal computer as long as they are connected to the Internet. Besides other numerous features, cloud computing offers cost-effective, on-demand resources. A user can avail need-based resources from the cloud at a very reasonable cost.

3.5.1 Cloud Services

A better way to understand the cloud is to interpret everything as a service. A "service" corresponds to any facility provided by the cloud. There are three standard models to categorise different computing services delivered through cloud as shown in Figure 3.12. These are Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS).

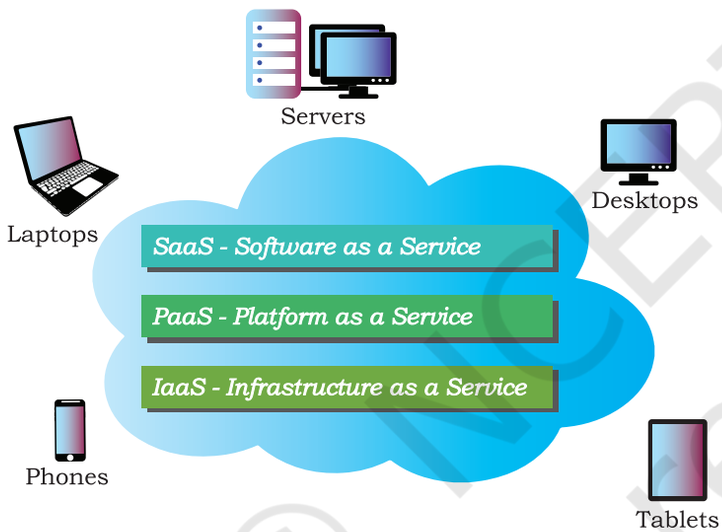


Figure 3.12: Cloud computing

(A) Infrastructure as a Service (IaaS)

The IaaS providers can offer different kinds of computing infrastructure, such as servers, virtual machines (VM), storage and backup facility, network components, operating systems or any other hardware or software. Using IaaS from the cloud, a user can use the hardware infrastructure located at a remote location to configure, deploy and execute any software application on that cloud infrastructure. They can outsource the hardware and software on a demand basis and pay as per the usage, thereby they can save the cost of software, hardware and other infrastructures as well as the cost of setting up, maintenance and security.

(B) Platform as a Service (PaaS)

The facility provided by the cloud, where a user can install and execute an application without worrying about the underlying infrastructure and their setup. That is, PaaS provides a platform or environment to develop, test, and deliver software applications.

Suppose we have developed a web application using MySQL and Python. To run this application online, we can avail a pre-configured Apache server from cloud having MySQL and Python pre-installed. Thus, we are not required to install MySQL and Python on the cloud, nor do we need to configure the web server (Apache, nginx). In PaaS, the user has complete control over the deployed application and its configuration. It provides a deployment environment for developers at a much reduced cost lessening the complexity of buying and managing the underlying hardware and software.

Activity 3.5

Name a few data centers in India along with the major services that they provide.

(C) Software as a Service (SaaS)

SaaS provides on-demand access to application software, usually requiring a licensing or subscription by the user. While using Google Doc, Microsoft Office 365, Drop Box, etc., to edit a document online, we use SaaS from cloud. A user is not concerned about installation or configuration of the software application as long as the required software is accessible. Like PaaS, a user is provided access to the required configuration settings of the application software, that they are using at present.

In all of the above standard service models, a user can use on-demand infrastructure or platform or software and is usually charged as per usage, thereby eliminating the need of a huge investment upfront for a new or evolving organisation. In order to utilise and harness the benefits of cloud computing, Government of India has embarked upon an ambitious initiative — "GI Cloud" which has been named as 'MeghRaj' (https://cloud.gov.in).

3.6 GRID COMPUTING

A grid is a computer network of geographically dispersed and heterogeneous computational resources as shown in Figure 3.13. Unlike cloud, whose primary focus is to provide services, a grid is more application specific and creates a sense of a virtual supercomputer with an enormous processing power and storage. The constituent resources are called nodes. These different nodes temporarily come together to solve a single large task and to reach a common goal.

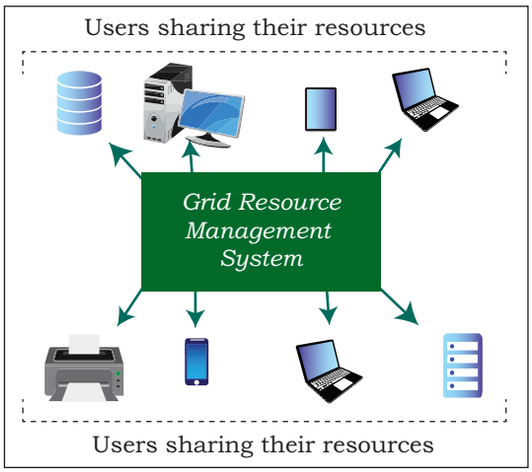


Figure 3.13: Grid computing

Think and Reflect

How can some of the emerging trends discussed in this chapter be used as assistive tools for people with disabilities?

Nowadays, countless computational nodes ranging from hand-held mobile devices to personal computers and workstations are connected to LAN or Internet. Therefore, it is economically feasible to reuse or utilise their resources like memory as well as processing power. The grid provides an opportunity to solve computationally intense scientific and research problems without actually procuring a costly hardware.

Grid can be of two types— (i) Data grid, used to manage large and distributed data having required multi-user access, and (ii) CPU or Processor grid, where processing is moved from one PC to another as needed or a large task is divided into subtasks and divided to various nodes for parallel processing.

Grid computing is different from IaaS cloud service. In case of IaaS cloud service, there is a service provider who rents the required infrastructure to the users. Whereas in grid computing, multiple computing nodes join together to solve a common computational problem.

To set up a grid, by connecting numerous nodes in terms of data as well as CPU, a middleware is required to implement the distributed processor architecture. The Globus toolkit (http://toolkit.globus.org/toolkit) is one such software toolkit used for building grids, and it is open source. It includes software for security, resource management, data management, communication, fault detection, etc.

3.7 BLOCKCHAINS

Traditionally, we perform digital transactions by storing data in a centralised database and the transactions performed are updated one by one on the database. That is how the ticket booking websites or banks operate. However, since all the data are stored on a central location, there are chances of data being hacked or lost.

The blockchain technology works on the concept of decentralised and shared database where each computer has a copy of the database. A block can be thought as a secured chunk of data or valid transaction. Each block has some data called its header, which is visible to every other node, while only the owner has access to the private data of the block. Such blocks form a chain

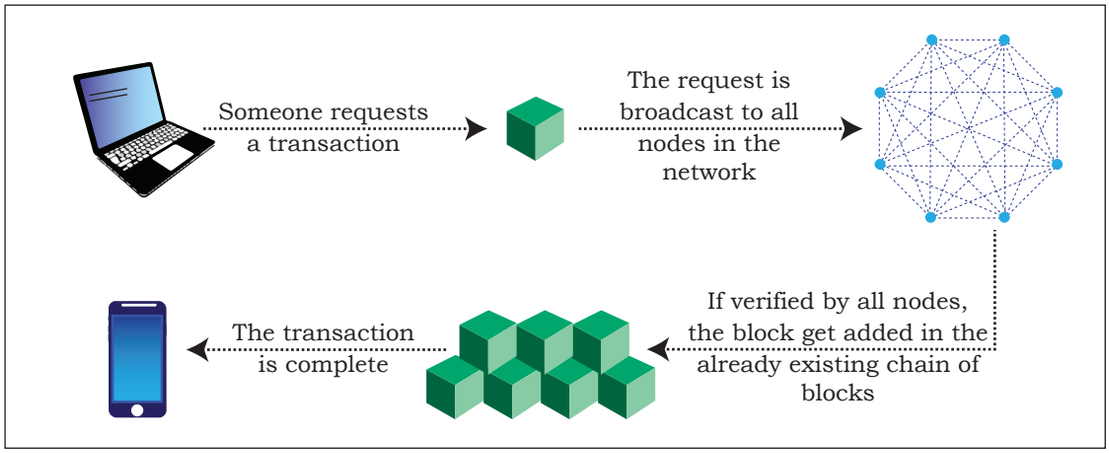


Figure 3.14: Block chain technology

called blockchain as shown in Figure 3.14. We can define blockchain as a system that allows a group of connected computers to maintain a single updated and secure ledger. Each computer or node that participates in the blockchain receives a full copy of the database. It maintains an 'append only' open ledger which is updated only after all the nodes within the network authenticate the transaction. Safety and security of the transactions are ensured because all the members in the network keep a copy of the blockchain and so it is not possible for a single member of the network to make changes or alter data.

The most popular application of blockchains technology is in digital currency. However, due to its decentralised nature with openness and security, blockchains are being seen as one of the ways to ensure transparency, accountability and efficiency in business and as well as governance systems.

For example, in healthcare, better data sharing between healthcare providers would result in a higher probability of accurate diagnosis, more effective treatments, and the overall increased ability of healthcare organisations to deliver cost-effective care. Another potential application can be for land registration records, to avoid various disputes arising out of land ownership claims and encroachments. A blockchain-based voting system can solve the problem of vote alterations and other issues. Since everything gets stored in the ledger, voting can become more transparent and authentic. The blockchain technology can be used in diverse sectors,

Think and Reflect

Name any two areas other than those given where the concept of blockchain technology can be useful.

NOTES

such as banking, media, telecom, travel and hospitality and other areas.

SUMMARY

- Artificial Intelligence (AI) endeavours to simulate the natural intelligence of human beings into machines thus making them intelligent.
- Machine learning comprises algorithms that use data to learn on their own and make predictions.
- Natural Language Processing (NLP) facilitates communicating with intelligent systems using a natural language.
- Virtual Reality (VR) allows a user to look at, explore and interact with the virtual surroundings, just like one can do in the real world.
- The superimposition of computer-generated perceptual information over the existing physical surroundings is called Augmented Reality.
- Robotics can be defined as the science or study of the technology primarily associated with the design, fabrication, theory and application of robots.
- Big data holds rich information and knowledge which can be of high business value. Five characteristics of big data are: Volume, Velocity, Variety, Veracity and Value.
- Data analytics is the process of examining data sets in order to draw conclusions about the information they contain.
- The Internet of Things (IoT) is a network of devices that have an embedded hardware and software to communicate (connect and exchange data) with other devices on the same network.
- A sensor is a device that takes input from the physical environment and uses built-in computing resources to perform predefined functions upon detection of specific input and then processes data before passing it on.
- Cloud computing allows resources located at remote locations to be made available to anyone anywhere. Cloud services can be Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS).

NOTES

- A grid is a computer network of geographically dispersed and heterogeneous computational resources.
- Blockchain is a system that allows a group of connected computers to maintain a single updated and secure ledger which is updated only after all the nodes in the network authenticate the transaction.

EXERCISE

1. List some of the cloud-based services that you are using at present.
2. What do you understand by the Internet of Things? List some of its potential applications.
3. Write short notes on the following:
   a) Cloud Computing
   b) Big data and its Characteristics
4. Explain the following along with their applications.
   a) Artificial Intelligence
   b) Machine Learning
5. Differentiate between cloud computing and grid computing with suitable examples.
6. Justify the following statement:
   "Storage of data is cost-effective and time saving in cloud computing."
7. What is on-demand service? How it is provided in cloud computing?
8. Write examples of the following:
   a) Government provided cloud computing platform
   b) Large scale private cloud service providers and the services they provide
9. A company interested in cloud computing is looking for a provider who offers a set of basic services, such as virtual server provisioning and on demand storage that can be combined into a platform for deploying and running customised applications. What type of cloud computing model fits these requirements?
   a) Platform as a Service
   b) Software as a Service
   c) Application as a Service
   d) Infrastructure as a Service

NOTES

10. If the government plans to make a smart school by applying IoT concepts, how can each of the following be implemented in order to transform a school into IoT-enabled smart school?
    a) e-textbooks
    b) Smart boards
    c) Online Tests
    d) Wifi sensors on classrooms doors
    e) Sensors in buses to monitor their location
    f) Wearables (watches or smart belts) for attendance monitoring.
11. Five friends plan to try a startup. However, they have a limited budget and limited computer infrastructure. How can they avail the benefits of cloud services to launch their startup?
12. Governments provide various scholarships to students of different classes. Prepare a report on how blockchain technology can be used to promote accountability, transparency and efficiency in distribution of scholarships?
13. How are IoT and WoT related?
14. Match the columns:

Column A	**Column B**
1. You got a reminder to take medication	A. Smart Parking
2. You got an SMS alert that you forgot to lock the door	B. Smart Wearable
3. You got an SMS alert that parking space is available near your block	C. Home Automation
4. You turned off your LED TV from your wrist watch	D. Smart Health

Chapter 4

Introduction to Problem Solving

11120CH04

4.1 Introduction

"Computer Science is a science of abstraction -creating the right model for a problem and devising the appropriate mechanizable techniques to solve it."

–A. Aho and J. Ullman

Today, computers are all around us. We use them for doing various tasks in a faster and more accurate manner. For example, using a computer or smartphone, we can book train tickets online.

India is a big country and we have an enormous railway network. Thus, railway reservation is a complex task. Making reservation involves information about many aspects, such as details of trains (train type, types of berth and compartments in each train, their schedule, etc.), simultaneous booking of tickets by multiple users and many other related factors.

It is only due to the use of computers that today, the booking of the train tickets has become easy. Online booking of train tickets has added to our comfort by enabling us to book tickets from anywhere, anytime.

We usually use the term computerisation to indicate the use of computer to develop software in order to automate any routine human task efficiently. Computers are used for solving various day-to-day problems and thus problem solving is an essential skill that a computer science student should know. It is pertinent to mention that computers themselves cannot solve a problem. Precise step-by-step instructions should be given by us to solve the problem. Thus, the success of a computer in solving a problem depends on how correctly and precisely we define the problem, design a solution (algorithm) and implement the solution (program) using a programming language. Thus, problem solving is the process of identifying a problem, developing an algorithm for the identified problem and finally implementing the algorithm to develop a computer program.

In this chapter

GIGO (Garbage In Garbage Out)
The correctness of the output that a computer gives depends upon the correctness of input provided.

4.2 STEPS FOR PROBLEM SOLVING

Suppose while driving, a vehicle starts making a strange noise. We might not know how to solve the problem right away. First, we need to identify from where the noise is coming? In case the problem cannot be solved by us, then we need to take the vehicle to a mechanic. The mechanic will analyse the problem to identify the source of the noise, make a plan about the work to be done and finally repair the vehicle in order to remove the noise. From the above example, it is explicit that, finding the solution to a problem might consist of multiple steps.

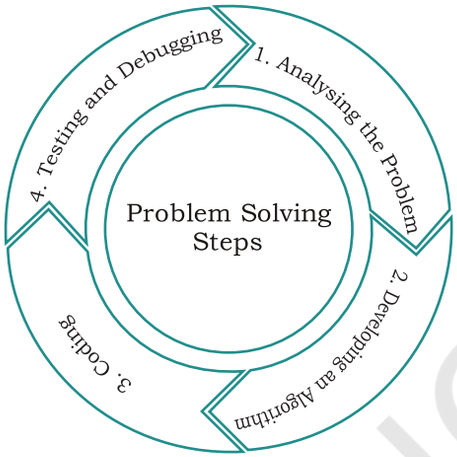


Figure 4.1: Steps for problem solving

When problems are straightforward and easy, we can easily find the solution. But a complex problem requires a methodical approach to find the right solution. In other words, we have to apply problem solving techniques. Problem solving begins with the precise identification of the problem and ends with a complete working solution in terms of a program or software. Key steps required for solving a problem using a computer are shown in Figure 4.1 and are discussed in following subsections.

4.2.1 Analysing the problem

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

Algorithm
A set of exact steps which when followed, solve the problem or accomplish the required task.

4.2.2 Developing an Algorithm

It is essential to device a solution before writing a program code for a given problem. The solution is represented in natural language and is called an algorithm. We can imagine an algorithm like a very well-written recipe for

a dish, with clearly defined steps that, if followed, one will end up preparing the dish.

We start with a tentative solution plan and keep on refining the algorithm until the algorithm is able to capture all the aspects of the desired solution. For a given problem, more than one algorithm is possible and we have to select the most suitable solution. The algorithm is discussed in section 4.3.

4.2.3 Coding

After finalising the algorithm, we need to convert the algorithm into the format which can be understood by the computer to generate the desired solution. Different high level programming languages can be used for writing a program.

It is equally important to record the details of the coding procedures followed and document the solution. This is helpful when revisiting the programs at a later stage. Coding is explained in detail in section 4.8.

4.2.4 Testing and Debugging

The program created should be tested on various parameters. The program should meet the requirements of the user. It must respond within the expected time. It should generate correct output for all possible inputs. In the presence of syntactical errors, no output will be obtained. In case the output generated is incorrect, then the program should be checked for logical errors, if any.

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

Once the software application has been developed, tested and delivered to the user, still problems in terms of functioning can come up and need to be resolved from time to time. The maintenance of the solution, thus, involves fixing the problems faced by the user,

Activity 4.1

What sequence of steps will you follow to compute the LCM of two numbers?

answering the queries of the user and even serving the request for addition or modification of features.

4.3 ALGORITHM

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

1) remove the bicycle from the stand,
2) sit on the seat of the bicycle,
3) start peddling,
4) use breaks whenever needed and
5) stop on reaching the destination.

Let us now find Greatest Common Divisor (GCD) of two numbers 45 and 56.

Note: GCD is the largest number that divides both the given numbers.

Step 1: Find the numbers (divisors) which can divide the given numbers

```
Divisors of 45 are: 1, 3, 5, 9, 15, and 45
Divisors of 54 are: 1, 2, 3, 6, 9, 18, 27, and 54
```

Step 2: Then find the largest common number from these two lists.

```
Therefore, GCD of 45 and 54 is 9
```

Hence, it is clear that we need to follow a sequence of steps to accomplish the task. Such a finite sequence of steps required to get the desired output is called an algorithm. It will lead to the desired result in a finite amount of time, if followed correctly. Algorithm has a definite beginning and a definite end, and consists of a finite number of steps.

The origin of the term Algorithm is traced to Persian astronomer and mathematician, Abu Abdullah Muhammad ibn Musa Al-Khwarizmi (c. 850 AD) as the Latin translation of Al-Khwarizmi was called 'Algorithmi'.

4.3.1 Why do we need an Algorithm?

A programmer writes a program to instruct the computer to do certain tasks as desired. The computer then follows the steps written in the program code. Therefore, the programmer first prepares a roadmap of the program to be written, before actually writing the code. Without

a roadmap, the programmer may not be able to clearly visualise the instructions to be written and may end up developing a program which may not work as expected.

Such a roadmap is nothing but the algorithm which is the building block of a computer program. For example, searching using a search engine, sending a message, finding a word in a document, booking a taxi through an app, performing online banking, playing computer games, all are based on algorithms.

Writing an algorithm is mostly considered as a first step to programming. Once we have an algorithm to solve a problem, we can write the computer program for giving instructions to the computer in high level language. If the algorithm is correct, computer will run the program correctly, every time. So, the purpose of using an algorithm is to increase the reliability, accuracy and efficiency of obtaining solutions.

(A) Characteristics of a good algorithm

- Precision — the steps are precisely stated or defined.
- Uniqueness — results of each step are uniquely defined and only depend on the input and the result of the preceding steps.
- Finiteness — the algorithm always stops after a finite number of steps.
- Input — the algorithm receives some input.
- Output — the algorithm produces some output.

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

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

4.4 Representation of Algorithms

Using their algorithmic thinking skills, the software designers or programmers analyse the problem and identify the logical steps that need to be followed to reach a solution. Once the steps are identified, the need is to

write down these steps along with the required input and desired output. There are two common methods of representing an algorithm —flowchart and pseudocode. Either of the methods can be used to represent an algorithm while keeping in mind the following:

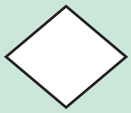
- it showcases the logic of the problem solution, excluding any implementational details
- it clearly reveals the flow of control during execution of the program

4.4.1 Flowchart — Visual Representation of Algorithms

A flowchart is a visual representation of an algorithm. A flowchart is a diagram made up of boxes, diamonds and other shapes, connected by arrows. Each shape represents a step of the solution process and the arrow represents the order or link among the steps.

There are standardised symbols to draw flowcharts. Some are given in Table 4.1.

Table 4.1 Shapes or symbols to draw flow charts

Flowchart symbol	Function	Description
	Start/End	Also called "Terminator" symbol. It indicates where the flow starts and ends.
	Process	Also called "Action Symbol," it represents a process, action, or a single step.
	Decision	A decision or branching point, usually a yes/no or true/false question is asked, and based on the answer, the path gets split into two branches.
	Input/Output	Also called data symbol, this parallelogram shape is used to input or output data
	Arrow	Connector to show order of flow between shapes.

Example 4.1: Write an algorithm to find the square of a number.

Before developing the algorithm, let us first identify the input, process and output:

- Input: Number whose square is required
- Process: Multiply the number by itself to get its square
- Output: Square of the number

Algorithm to find square of a number.

Step 1: Input a number and store it to num
Step 2: Compute num * num and store it in square
Step 3: Print square

The algorithm to find square of a number can be represented pictorially using flowchart as shown in Figure 4.2.

Think and Reflect

What will happen if an algorithm does not stop after a finite number of steps?

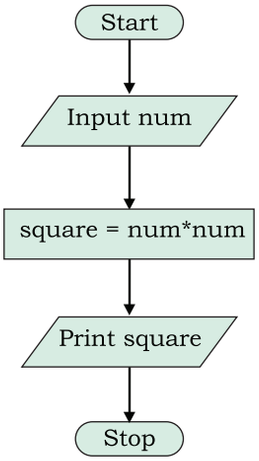


Figure 4.2: Flowchart to calculate square of a number

Activity 4.2

Draw a flowchart that represents the attainment of your career goal.

Example 4.2: Draw a flowchart to solve the problem of a non-functioning light bulb

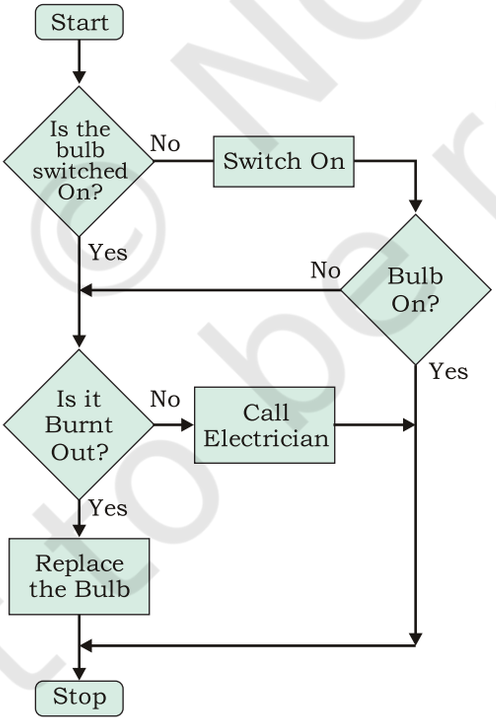


Figure 4.3: Flowchart to solve the problem of a non-functioning light bulb

4.4.2 Pseudocode

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

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

Activity 4.3

Write a pseudocode for creating a scoreboard for a hockey match.

Example 4.3: Write an algorithm to display the sum of two numbers entered by user, using both pseudocode and flowchart.

Pseudocode for the sum of two numbers will be:

```
INPUT num1
INPUT num2
COMPUTE Result = num1 + num2
PRINT Result
```

The flowchart for this algorithms is given in Figure 4.4.

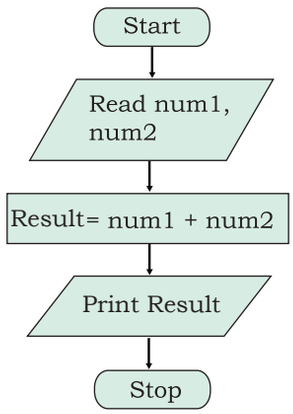


Figure 4.4: Flowchart to display sum of two numbers

NOTES

Example 4.4: Write an algorithm to calculate area and perimeter of a rectangle, using both pseudocode and flowchart.

Pseudocode for calculating area and perimeter of a rectangle.

```
INPUT length
INPUT breadth
COMPUTE Area = length * breadth
PRINT Area
COMPUTE Perim = 2 * (length + breadth)
PRINT Perim
```

The flowchart for this algorithm is given in Figure 4.5.

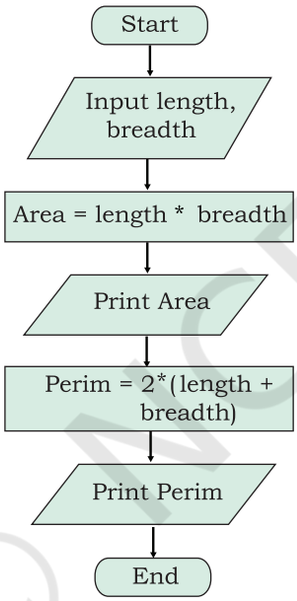


Figure 4.5: Flowchart to calculate area and perimeter of a rectangle

(A) *Benefits of Pseudocode*

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

4.5 FLOW OF CONTROL

The flow of control depicts the flow of events as represented in the flow chart. The events can flow in a sequence, or on branch based on a decision or even repeat some part for a finite number of times.

4.5.1 Sequence

If we look at the examples 4.3 and 4.4, the statements are executed one after another, i.e., in a sequence. Such algorithms where all the steps are executed one after the other are said to execute in sequence. However, statements in an algorithm may not always execute in a sequence. We may sometimes require the algorithm to either do some routine tasks in a repeated manner or behave differently depending on the outcomes of previous steps. In this section, we are going to learn how to write algorithms for such situations.

4.5.2 Selection

Consider the map of a neighbourhood as shown in Figure 4.6. Let us assume that the pink building with the red roof is the school; the yellow painted house at the far end of the map is a house.

Think and Reflect

Can you list some of the routine activities in your daily life where decision making is involved?

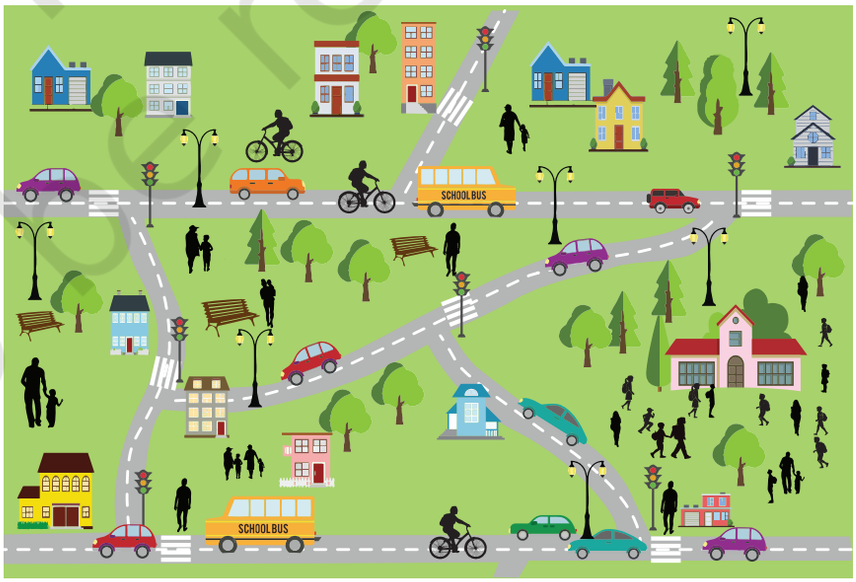


Figure 4.6: Decision making in real life

Notes

With reference to Figure 4.6, let us answer the following questions :

- Is there a predefined route for walking from home to school?
- Can we have a different route while coming back?

As seen from the map, there can be multiple routes between home and school. We might take the shortest route in the morning. But while coming back home in the afternoon, the shortest route might have heavy traffic. Therefore, we could take another route with less traffic. Hence, the above problem involves some decision-making based on certain conditions.

Let us look at some other examples where decision making is dependent on certain conditions. For example,

(i) Checking eligibility for voting.

Depending on their age, a person will either be allowed to vote or not allowed to vote:

- If age is greater than or equal to 18, the person is eligible to vote
- If age is less than 18, the person is not eligible to vote

(ii) Let us consider another example

If a student is 8 years old and the student likes Maths

put the student in Group A

Otherwise

Put the student in Group B

In which group will these students go as per the above condition?

Outcome

- 8-year-old Ravi who does not like Maths: Group B
- 8-year-old Priti who likes Maths: Group A
- 7-year-old Anish who likes Maths: Group B

In these examples, any one of the alternatives is selected based on the outcome of a condition. Conditionals are used to check possibilities. The program checks one or more conditions and perform operations (sequence of actions) depending on true or false value of the condition. These true or false values are called binary values.

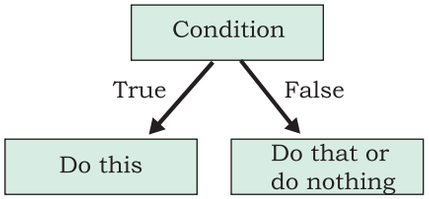


Figure 4.7: Actions depending on true or false of a condition

Conditionals are written in the algorithm as follows:

```
If <condition> then
     steps to be taken when the
condition is true/fulfilled
```

There are situations where we also need to take action when the condition is not fulfilled (Figure 4.7). To represent that, we can write:

```
If <condition> is true then
steps to be taken when the condition is
true/fulfilled
otherwise
     steps to be taken when the condition is
false/not fulfilled
```

In programming languages, 'otherwise' is represented using Else keyword. Hence, a true/false conditional is written using if-else block in actual programs.

Example 4.5: Let us write an algorithm to check whether a number is odd or even.

- Input: Any number
- Process: Check whether the number is even or not
- Output: Message “Even” or “Odd”

Pseudocode of the algorithm can be written as follows:

```
PRINT "Enter the Number"
INPUT number
IF number MOD 2 == 0 THEN
     PRINT "Number is Even"
ELSE
     PRINT "Number is Odd"
```

The flowchart representation of the algorithm in shown in Figure 4.8.

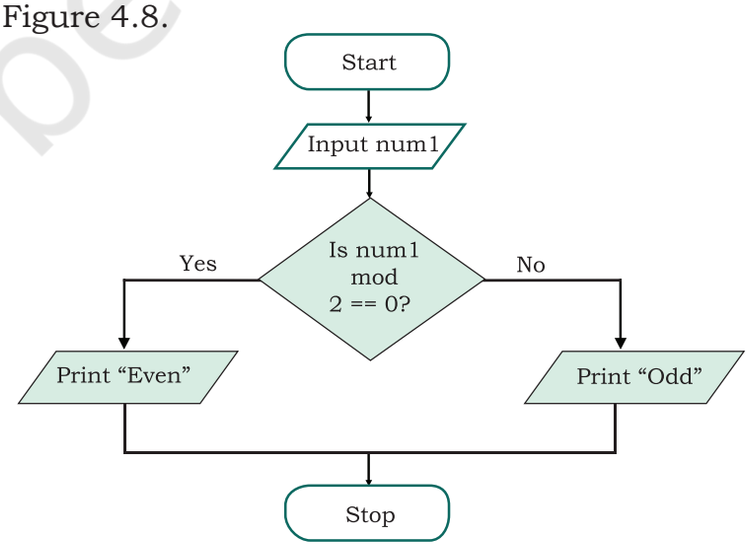


Figure 4.8: Flowchart to check whether a number is even or odd

NOTES

Example 4.6: Let us write a pseudocode and draw a flowchart where multiple conditions are checked to categorise a person as either child (<13), teenager (>=13 but <20) or adult (>=20),based on age specified:

- Input: Age
- Process: Check Age as per the given criteria
- Output: Print either “Child”, “Teenager”, “Adult”

Pseudocode is as follows:

```
INPUT Age
IF Age < 13 THEN
        PRINT "Child"
  ELSE IF Age < 20 THEN
        PRINT "Teenager"
  ELSE
        PRINT "Adult"
```

The flowchart representation of the algorithm in shown in Figure 4.9

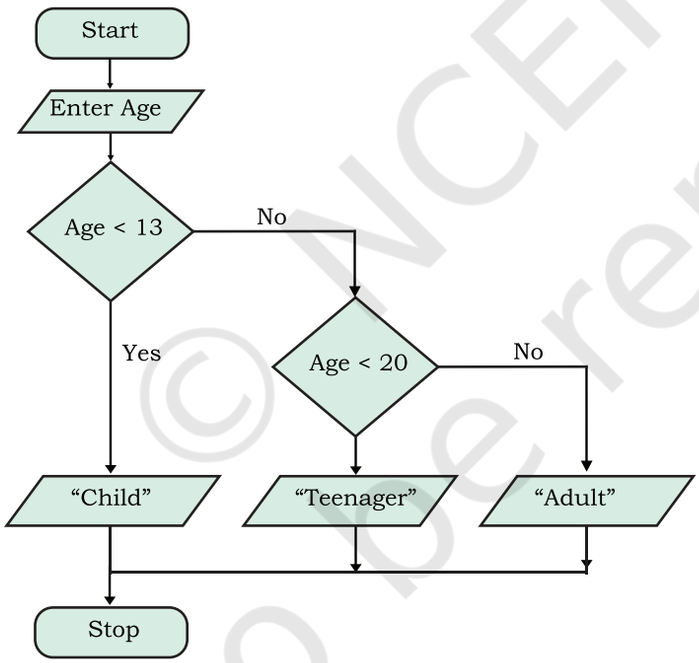


Figure 4.9: Flowchart to check multiple conditions

Example 4.7: Algorithm for a card game called “Dragons and Wizards”.

Make two teams DRAGONS and WIZARDS

The rules for the game are as follows:

- If the card drawn is a diamond or a club, Team DRAGONS gets a point
- If the card drawn is a heart which is a number, Team WIZARDS gets a point

NOTES

- If the card drawn is a heart that is not a number, Team DRAGONS gets a point
- For any other card, Team WIZARDS gets a point
- The team with highest point is the winner

Let us identify the following for a card:
Input: shape, value
Process: Increment in respective team scores by one based on the outcome of the card drawn, as defined in the rules.
Output: Winning team
Now let us write the conditionals for this game:

```
IF (shape is diamond) OR (shape is club)
     Team DRAGONS gets a point
ELSE IF (shape is heart) AND (value is
number)
     Team WIZARDS gets a point
ELSE IF (shape is heart) AND (value is not a
number)
     Team DRAGONS gets a point
ELSE
     Team WIZARDS gets a point
```

The pseudocode for the program can be as follows:
Note: Dpoint (for Dragon) and Wpoint (for Wizard) store points scored by the respective teams.

```
INPUT shape
INPUT value
SET Dpoint = 0, Wpoint = 0
IF (shape is diamond) OR (shape is club) THEN
     INCREMENT Dpoint
ELSE IF (shape is heart) AND (value is
number)THEN
     INCREMENT Wpoint
ELSE IF (shape is heart) AND (value is not a
number)THEN
     INCREMENT Dpoint
ELSE
     INCREMENT Wpoint
END IF
IF Dpoint > Wpoint THEN
  PRINT "Dragon team is the winner"
ELSE
  PRINT "Wizard team is the winner"
```

4.5.3 Repetition

When giving directions to go someplace, we say something like, “walk 50 steps then turn right”. Or “Walk till the next

crossing then take a right turn". Consider some other examples like:

- Clap your hands five times
- Walk 10 steps ahead
- Jump on the spot till you get tired

These are the kind of statements we use, when we want something to be done repeatedly, for a given number of times. Likewise, suppose 10 cards need to be withdrawn in the previous card game (example 4.7), then the pseudocode needs to be repeated 10 times to decide the winner. All these are examples of repetitions. In programming, repetition is also known as iteration or loop. A loop in an algorithm means execution of some program statements repeatedly till some specified condition is satisfied.

Think and Reflect

Can you list some of the routine activities in your daily life where repetition or iteration is involved?

Example 4.8: Write pseudocode and draw a flowchart to accept 5 numbers and find their average.

The flowchart representation is shown in Figure 4.10.

Pseudocode will be as follows:

Step 1: SET count = 0, sum = 0
Step 2: WHILE count < 5 , REPEAT steps 3 to 5
Step 3: INPUT a number to num
Step 4: sum = sum + num
Step 5: count = count + 1
Step 6: COMPUTE average = sum/5
Step 7: PRINT average

In example 4.8, a counter called "count" keeps track of number of times the loop has been repeated. After every iterationof the loop, the value of count is incremented by 1 until it performs the set number of repetitions, given in the iteration condition.

There are situations when we are not aware beforehand about the number of times a set

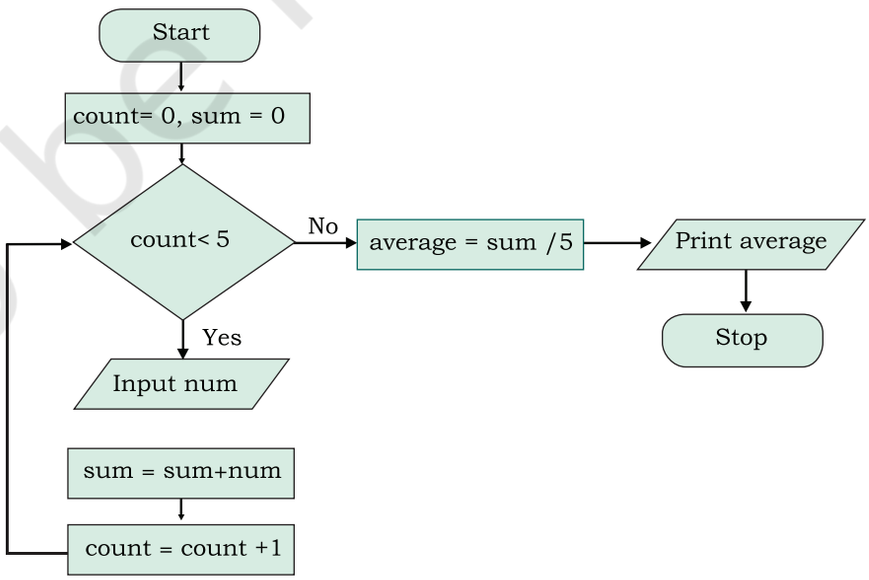


Figure 4.10: Flowchart to Calculate the Average of 5 Numbers

of statements need to be repeated. Such requirements of unknown number of repetitions are handled using WHILE construct.

Example 4.9: Write pseudocode and draw flowchart to accept numbers till the user enters 0 and then find their average.

Pseudocode is as follows:

Step 1: SET count = 0, sum = 0
Step 2: INPUT num
Step 3: WHILE num is not equal to 0, REPEAT Steps 4 to 6
Step 4: sum = sum + num
Step 5: count = count + 1
Step 6: INPUT num
Step 7: COMPUTE average = sum/count
Step 8: PRINT average
The flowchart representation is shown in Figure 4.11.

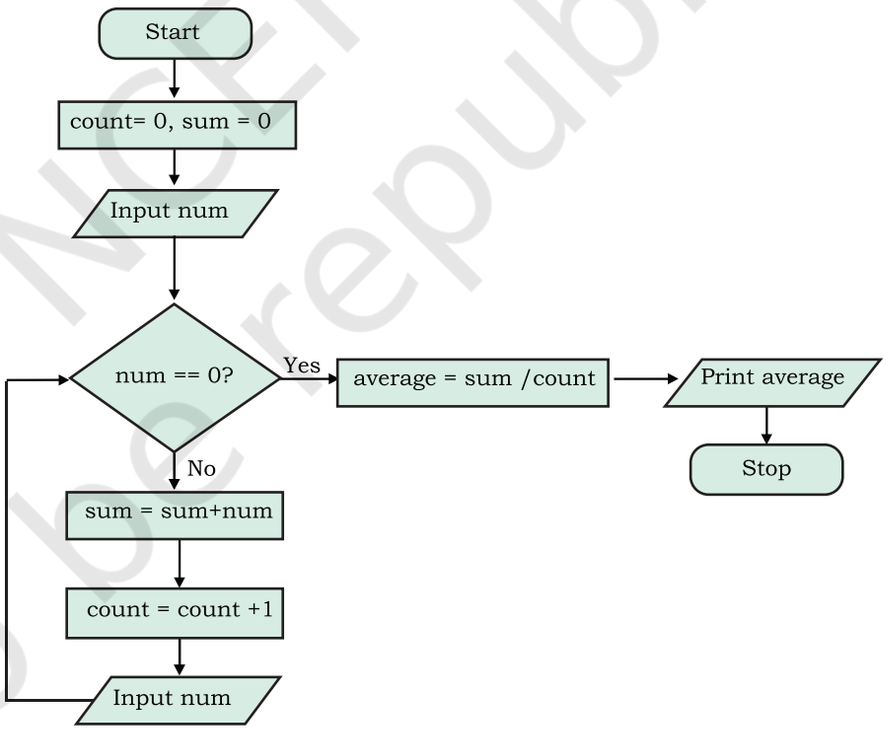


Figure 4.11: Flowchart to accept numbers till the user enters 0

In this example, we do not know how many numbers a user is going to enter before entering 0. This is handled by checking the condition repeatedly till the condition becomes false.

Activity 4.4

Let us answer the following questions using the pesudocode given in example 4.9:

1) What will the sum when the input are 6, 7, 4, 8, 2, 5, 0, 3, 1.
2) What will be the value of count?
3) Why did we use the input statement to enter num twice?
4) Why did we divide sum by count?
5) Can there be any other approach?

4.6 Verifying Algorithms

Can you imagine what would happen if a banking software does not work correctly? Suppose functioning of the online money transfer module is not programmed correctly, and it credits into the account only half the amount transacted! What happens if the account is debited instead of being credited. Such a faulty software will mess up the working of the complete system and cause havoc! Today software are used in even more critical services — like in the medical field or in space shuttles. Such software needs to work correctly in every situation. Therefore, the software designer should make sure that the functioning of all the components are defined correctly, checked and verified in every possible way.

Think and Reflect

Why is verification of algorithm an important step in problem solving?

When we were told that the formula for the sum of first N natural numbers is $\frac{N(N+1)}{2}$, how did we verify it? Well, we can check this for small numbers, for which we can manually calculate the sum. Let N = 6, then the sum is 1 + 2 + 3 + 4 + 5 + 6 = 21

Using formula we get sum = $\frac{6x(6+1)}{2}$

We can try with some more numbers this way to ensure that the formula works correctly.

Activity 4.5

Write an algorithm to take as input the measurement of length and breadth in feet and inches (e.g., 5 ft 6 inch) of a rectangular shape and calculate its area and perimeter.

In the same way, when we have written an algorithm, we want to verify that it is working as expected. To verify, we have to take different input values and go through all the steps of the algorithm to yield the desired output for each input value and may modify or improve as per the need. The method of taking an input and running through the steps of the algorithm is sometimes called *dry run*. Such a dry run will help us to:

1. Identify any incorrect steps in the algorithm
2. Figure out missing details or specifics in the algorithm

It is important to decide the type of input value to be used for the simulation. In case all possible input values are not tested, then the program will fail. What if there is some other case for which it does not work? Let us look at some examples.

Write an algorithm to calculate the time taken to go from place A to C (T_total) via B where time taken to

NOTES

go from A to B (T1) and B to C (T2) are given. That is, we want the algorithm to add time given in hours and minutes. One way to write the algorithm is:

```
PRINT value for T1
INPUT hh1
INPUT mm1
PRINT value for T2
INPUT hh2
INPUT mm2
hh_total = hh1 + hh2 (Add hours)
mm_total = mm1 + mm2 (Add mins)
Print T_total as hh_total, mm_total
```

Now let us verify. Suppose the first example we take is T1 = 5 hrs 20 mins and T2 = 7 hrs 30 mins. On dry run, we get the result 12 hrs and 50 mins. This looks fine.

Now let us take another example where T1 = 4 hrs 50 mins and T2 = 2 hrs 20 mins, and we end up getting the result as 6 hrs 70 mins which is not how we measure time. The result should have been 7 hrs 10 mins.

With this second example we realise that our algorithm will work only when mm1 + mm2 (mm_total) < 60. For all other cases, it will give us output not the way we want. When mm_total >= 60, the algorithm should increase the sum of hours (hh_total) by 1 and reduce mm_total by 60, i.e., (mm_total - 60). So the modified algorithm will be:

```
PRINT value for T1
INPUT hh1
INPUT mm1
PRINT value for T2
INPUT hh2
INPUT mm2
hh_total = hh1 + hh2 (Add hours)
mm_total = mm1 + mm2 (Add mins)
hh_total = hh1 + hh2 (Add hours)
mm_total = mm1 + mm2 (Add mins)
IF (mm_total >= 60) THEN
     hh_total = hh_total + 1
     mm_total = mm_total - 60
PRINT T_total as hh_total, mm_total
```

Now we can simulate through algorithm for T1 = 4 hrs 50 mins and T2 = 2 hrs 20 mins, and get T_total = 7 hrs and 10 mins, which means the algorithm is working correctly.

Notes

Suppose we develop some software without verifying the underlying algorithm and if there are errors in the algorithm, then the software developed will not run. Hence, it is important to verify an algorithm since the effort required to catch and fix a mistake is minimal.

4.7 Comparison of Algorithm

There can be more than one approach to solve a problem using computer and hence we can have more than one algorithm. Then one may ask which algorithm should be used?

Consider the problem of finding whether a given number is prime or not. Prime numbers are of great importance in computer science as they find application in databases, security, file compression or decompression, modulation or demodulation, etc. There can be four different ways to write algorithms to check whether a given number is prime or not as shown below:

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

(ii) In (i), instead of testing all the numbers till the dividend, only test up to half of the given value (dividend) because the divisor can not be more than half of the dividend

(iii) In method (i), only test up to the square root of the dividend (numbers)

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

All these four methods can check if a given number is prime or not. Now the question is which of these methods is better or efficient?

Algorithm (i) requires large number of calculations (means more processing time) as it checks for all the numbers as long as the divisor is less than the number. If the given number is large, this method will take more time to give the output.

The spirit of problem solving by decomposition is to 'divide and conquer'. In words of *Howard Raffa*, a famous mathematician:

"Decompose a complex problem into simpler problems get one's thinking straight in these simpler problems, put these analyses together with logical glue"

Algorithm (ii) is more efficient than (i) as it checks for divisibility till half the number, and thus it reduces the time for computation of the prime number. Algorithm (iii) is even more efficient as it checks for divisibility till square root of the number, thereby further reducing the time taken.

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

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

4.8 CODING

Once an algorithm is finalised, it should be coded in a high-level programming language as selected by the programmer. The ordered set of instructions are written in that programming language by following its syntax. Syntax is the set of rules or grammar that governs the formulation of the statements in the language, such as spellings, order of words, punctuation, etc.

The machine language or low level language consisting of 0s and 1s only is the ideal way to write a computer program. Programs written using binary digits are directly understood by the computer hardware, but they are difficult to deal with and comprehend by humans. This led to the invention of high-level languages which are close to natural languages and are easier to read, write, and maintain, but are not directly understood by the computer hardware. An advantage of using high-level languages is that they are portable, i.e., they can run on different types of computers with little

Notes

or no modifications. Low-level programs can run on only one kind of computer and have to be rewritten in order to run on another type of system. A wide variety of high-level languages, such as FORTRAN, C, C++, Java, Python, etc., exist.

A program written in a high-level language is called source code. We need to translate the source code into machine language using a compiler or an interpreter, so that it can be understood by the computer. We have learnt about the compiler and interpreter in Chapter 1.

There are multiple programming languages available and choosing the one suitable for our requirements requires us to consider many factors. It depends on the platform (OS) where the program will run. We need to decide whether the application would be a desktop application, a mobile application or a web application. Desktop and mobile applications are generally developed for a particular operating system and for certain hardware whereas the web applications are accessed in different devices using web browsers and may use resources available over cloud.

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

4.9 Decomposition

Sometimes a problem may be complex, that is, its solution is not directly derivable. In such cases, we need to decompose it into simpler parts. Let us look at the Railway reservation system we talked about earlier. The complex task of designing a good railway reservation system is seen as designing the different components of the system and then making them work with each other effectively.

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

in Figure 4.12. These sub problems are relatively easier to solve than the original problem. Finally, the sub-problems are combined in a logical way to obtain the solution for the bigger, main problem.

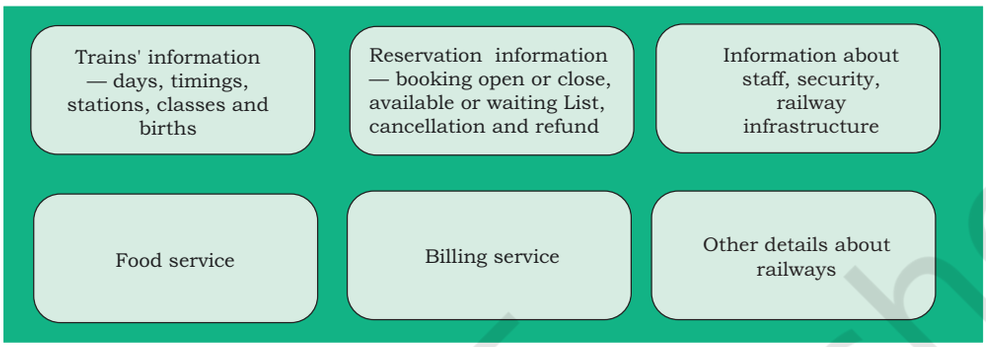


Figure 4.12: Railway reservation system

Breaking down a complex problem into sub problems also means that each sub problem can be examined in detail. Each sub problem can be solved independently and by different persons (or teams). Having different teams working on different sub problems can also be advantageous because specific sub problems can be assigned to teams who are experts in solving such problems.

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

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

SUMMARY

- An algorithm is defined as a step-by-step procedure designed to perform an operation which will lead to the desired result, if followed correctly.
- Algorithms have a definite beginning and a definite end, and a finite number of steps.
- A good algorithm, which is precise, unique and finite, receives input and produces an output.

- In order to write effective algorithms we need to identify the input, the process to be followed and the desired output.
- A flowchart is a type of diagram that represents the algorithm graphically using boxes of various kinds, in an order connected by arrows.
- An algorithm where all the steps are executed one after the other is said to execute in sequence.
- Decision making involves selection of one of the alternatives based on outcome of a condition.
- An algorithm may have a certain set of steps, which are repeating for a finite number of times, such an algorithm is said to be iterative.
- There can be more than one approach to solve a problem and hence we can have more than one algorithm for a particular problem.
- The choice of algorithm should be made on the basis of time and space complexity.

Exercise

1. Write pseudocode that reads two numbers and divide one by another and display the quotient.
2. Two friends decide who gets the last slice of a cake by flipping a coin five times. The first person to win three flips wins the cake. An input of 1 means player 1 wins a flip, and a 2 means player 2 wins a flip. Design an algorithm to determine who takes the cake?
3. Write the pseudocode to print all multiples of 5 between 10 and 25 (including both 10 and 25).
4. Give an example of a loop that is to be executed a certain number of times.
5. Suppose you are collecting money for something. You need ₹ 200 in all. You ask your parents, uncles and aunts as well as grandparents. Different people may give either ₹ 10, ₹ 20 or even ₹ 50. You will collect till the total becomes 200. Write the algorithm.
6. Write the pseudocode to print the bill depending upon the price and quantity of an item. Also print

NOTES

Bill GST, which is the bill after adding 5% of tax in the total bill.

7. Write pseudocode that will perform the following:
   a) Read the marks of three subjects: Computer Science, Mathematics and Physics, out of 100
   b) Calculate the aggregate marks
   c) Calculate the percentage of marks
8. Write an algorithm to find the greatest among two different numbers entered by the user.
9. Write an algorithm that performs the following: Ask a user to enter a number. If the number is between 5 and 15, write the word GREEN. If the number is between 15 and 25, write the word BLUE. if the number is between 25 and 35, write the word ORANGE. If it is any other number, write that ALL COLOURS ARE BEAUTIFUL.
10. Write an algorithm that accepts four numbers as input and find the largest and smallest of them.
11. Write an algorithm to display the total water bill charges of the month depending upon the number of units consumed by the customer as per the following criteria:
    - for the first 100 units @ 5 per unit
    - for next 150 units @ 10 per unit
    - more than 250 units @ 20 per unit

    Also add meter charges of 75 per month to calculate the total water bill .
12. What are conditionals? When they are required in a program?
13. Match the pairs

Flowchart Symbol	Functions
	Flow of Control
	Process Step
	Start/Stop of the Process

Data

Decision Making

14. Following is an algorithm for going to school or college. Can you suggest improvements in this to include other options?

    Reach_School_Algorithm

    a) Wake up

    b) Get ready

    c) Take lunch box

    d) Take bus

    e) Get off the bus

    f) Reach school or college

15. Write a pseudocode to calculate the factorial of a number (Hint: Factorial of 5, written as $5! = 5 \times 4 \times 3 \times 2 \times 1$) .

16. Draw a flowchart to check whether a given number is an Armstrong number. An Armstrong number of three digits is an integer such that the sum of the cubes of its digits is equal to the number itself. For example, 371 is an Armstrong number since 3**3 + 7**3 + 1**3 = 371.

17. Following is an algorithm to classify numbers as "Single Digit", "Double Digit" or "Big".

    Classify_Numbers_Algo

```
INPUT Number
IF Number < 9
     "Single Digit"
Else If Number < 99
     "Double Digit"
Else
     "Big"
```

    Verify for (5, 9, 47, 99, 100 200) and correct the algorithm if required

18. For some calculations, we want an algorithm that accepts only positive integers upto 100.

NOTES

Accept_1to100_Algo

```
INPUT Number
IF (0<= Number) AND (Number <= 100)
     ACCEPT
Else
     REJECT
```

a) On what values will this algorithm fail?

b) Can you improve the algorithm?

Chapter 5

Getting Started with Python

11120CH05

5.1 Introduction to Python

We have written algorithms for different problems in Chapter 4. Let us now move a step further and create programs using any version of Python 3. But before learning about Python programming language, let us understand what is a programming language and how it works.

An ordered set of instructions to be executed by a computer to carry out a specific task is called a program, and the language used to specify this set of instructions to the computer is called a programming language.

As we know that computers understand the language of 0s and 1s which is called machine language or low level language. However, it is difficult for humans to write or comprehend instructions using 0s and 1s. This led to the advent of high-level programming languages like Python, C++, Visual Basic, PHP, Java that are easier to manage by humans but are not directly understood by the computer.

A program written in a high-level language is called source code. Recall from Chapter 1 that language translators like compilers and interpreters are needed to translate the source code into machine language. Python uses an interpreter to convert its instructions into machine language, so that it can be understood by the computer. An interpreter processes the program statements one by one, first translating and then executing. This process is continued until an error is encountered or the whole program is executed successfully. In both the cases, program execution will stop. On the contrary, a compiler translates the entire source code, as a whole, into the object code. After scanning the whole program, it generates error messages, if any.

"Computer programming is an art, because it applies accumulated knowledge to the world, because it requires skill and ingenuity, and especially because it produces objects of beauty. A programmer who subconsciously views himself as an artist will enjoy what he does and will do it better."

– Donald Knuth

In this chapter

- *Introduction to Python*
- *Python Keywords*
- *Identifiers*
- *Comments*
- *Data Types*
- *Operators*
- *Expressions*
- *Statement*
- *Input and Output*
- *Type Conversion*
- *Debugging*

5.1.1 Features of Python

- Python is a high level language. It is a free and open source language.
- It is an interpreted language, as Python programs are executed by an interpreter.
- Python programs are easy to understand as they have a clearly defined syntax and relatively simple structure.
- Python is case-sensitive. For example, NUMBER and number are not same in Python.
- Python is portable and platform independent, means it can run on various operating systems and hardware platforms.
- Python has a rich library of predefined functions.
- Python is also helpful in web development. Many popular web services and applications are built using Python.
- Python uses indentation for blocks and nested blocks.

Downloading Python

The latest version of Python 3 is available on the official website:

https://www.python.org/

5.1.2 Working with Python

To write and run (execute) a Python program, we need to have a Python interpreter installed on our computer or we can use any online Python interpreter. The interpreter is also called Python shell. A sample screen of Python interpreter is shown in Figure 5.1:

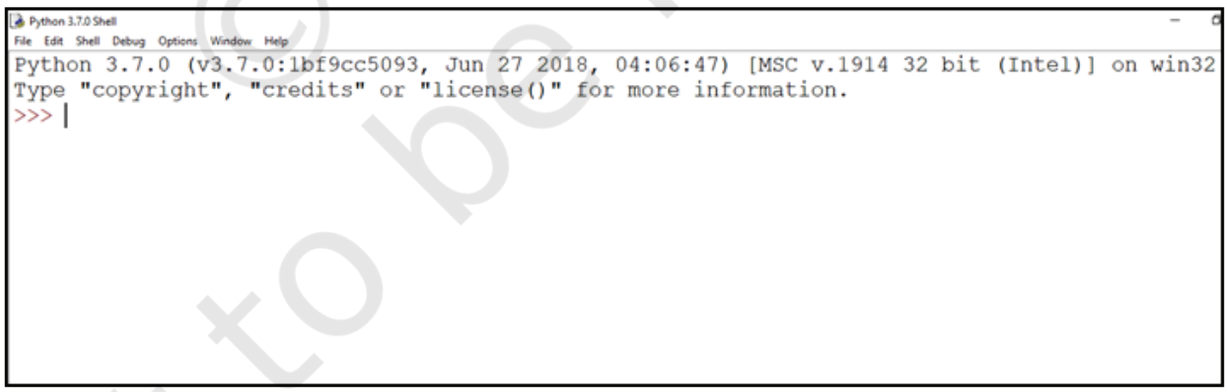


Figure 5.1: Python interpreter or shell

In the above screen, the symbol >>> is the Python prompt, which indicates that the interpreter is ready to take instructions. We can type commands or statements on this prompt to execute them using a Python interpreter.

5.1.3 Execution Modes

There are two ways to use the Python interpreter:

a) Interactive mode
b) Script mode

Interactive mode allows execution of individual statement instantaneously. Whereas, Script mode allows us to write more than one instruction in a file called Python source code file that can be executed.

(A) Interactive Mode

To work in the interactive mode, we can simply type a Python statement on the >>> prompt directly. As soon as we press enter, the interpreter executes the statement and displays the result(s), as shown in Figure 5.2.

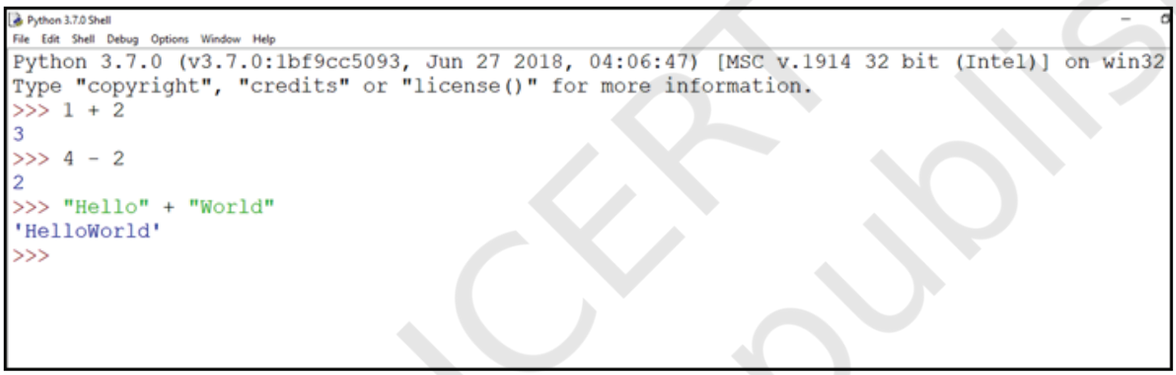


Figure 5.2: Python interpreter in interactive mode

Working in the interactive mode is convenient for testing a single line code for instant execution. But in the interactive mode, we cannot save the statements for future use and we have to retype the statements to run them again.

(B) Script Mode

In the script mode, we can write a Python program in a file, save it and then use the interpreter to execute it. Python scripts are saved as files where file name has extension ".py". By default, the Python scripts are saved in the Python installation folder. To execute a script, we can either:

a) Type the file name along with the path at the prompt. For example, if the name of the file is prog5-1.py, we type prog5-1.py. We can otherwise open the program directly from IDLE as shown in Figure 5.3.
b) While working in the script mode, after saving the file, click [Run]->[Run Module] from the menu as shown in Figure 5.4.

c) The output appears on shell as shown in Figure 5.5.

Program 5-1 Write a program to show print statement in script mode.

Figure 5.3: Python source code file (prog5-1.py)

Figure 5.4: Execution of Python in Script mode using IDLE

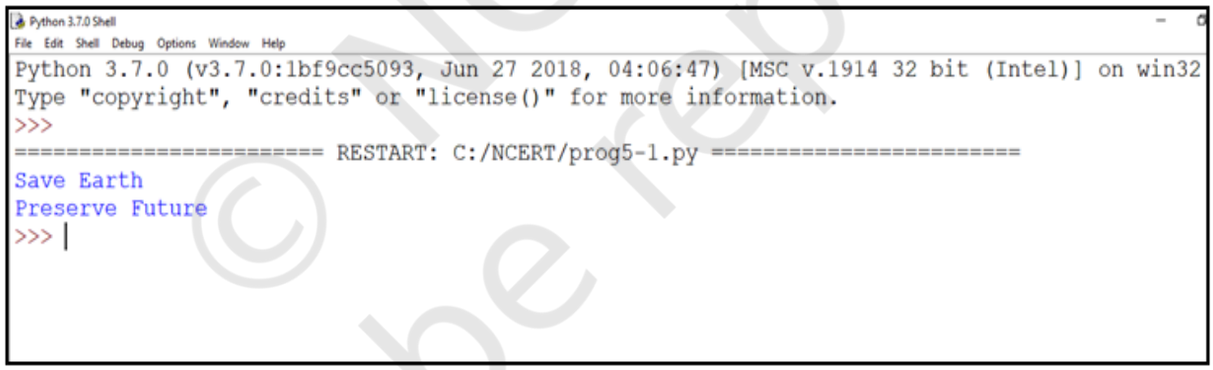


Figure 5.5: Output of a program executed in script mode

5.2 PYTHON KEYWORDS

Keywords are reserved words. Each keyword has a specific meaning to the Python interpreter, and we can use a keyword in our program only for the purpose for which it has been defined. As Python is case sensitive, keywords must be written exactly as given in Table 5.1.

Table 5.1 Python keywords

False	class	finally	is	return
None	continue	for	lambda	try

NOTES

True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

5.3 IDENTIFIERS

In programming languages, identifiers are names used to identify a variable, function, or other entities in a program. The rules for naming an identifier in Python are as follows:

- The name should begin with an uppercase or a lowercase alphabet or an underscore sign (_). This may be followed by any combination of characters a–z, A–Z, 0–9 or underscore (_). Thus, an identifier cannot start with a digit.
- It can be of any length. (However, it is preferred to keep it short and meaningful).
- It should not be a keyword or reserved word given in Table 5.1.
- We cannot use special symbols like !, @, #, $, %, etc., in identifiers.

For example, to find the average of marks obtained by a student in three subjects, we can choose the identifiers as `marks1, marks2, marks3` and `avg` rather than a, b, c, or A, B, C.

```
avg = (marks1 + marks2 + marks3)/3
```

Similarly, to calculate the area of a rectangle, we can use identifier names, such as `area, length, breadth` instead of single alphabets as identifiers for clarity and more readability.

```
area = length * breadth
```

5.4 VARIABLES

A variable in a program is uniquely identified by a name (identifier). Variable in Python refers to an object — an item or element that is stored in the memory. Value of a variable can be a string (e.g., 'b', 'Global Citizen'), numeric (e.g., 345) or any combination of alphanumeric characters (CD67). In Python we can use an assignment statement to create new variables and assign specific values to them.

```
gender   = 'M'
message  = "Keep Smiling"
price    = 987.9
```

Program 5-2 Write a program to display values of variables in Python.

```
#Program 5-2
#To display values of variables
message = "Keep Smiling"
print(message)
userNo = 101
print('User Number is', userNo)
```

Output:

```
Keep Smiling
User Number is 101
```

In the program 5-2, the variable message holds string type value and so its content is assigned within double quotes " " (can also be within single quotes ' '), whereas the value of variable userNo is not enclosed in quotes as it is a numeric value.

Variable declaration is implicit in Python, means variables are automatically declared and defined when they are assigned a value the first time. Variables must always be assigned values before they are used in expressions as otherwise it will lead to an error in the program. Wherever a variable name occurs in an expression, the interpreter replaces it with the value of that particular variable.

Program 5-3 Write a Python program to find the area of a rectangle given that its length is 10 units and breadth is 20 units.

```
#Program 5-3
#To find the area of a rectangle
length = 10
breadth = 20
area = length * breadth
print(area)
```

Output:

```
200
```

5.5 COMMENTS

Comments are used to add a remark or a note in the source code. Comments are not executed by interpreter.

They are added with the purpose of making the source code easier for humans to understand. They are used primarily to document the meaning and purpose of source code and its input and output requirements, so that we can remember later how it functions and how to use it. For large and complex software, it may require programmers to work in teams and sometimes, a program written by one programmer is required to be used or maintained by another programmer. In such situations, documentations in the form of comments are needed to understand the working of the program.

In Python, a comment starts with # (hash sign). Everything following the # till the end of that line is treated as a comment and the interpreter simply ignores it while executing the statement.

Example 5.1

```
#Variable amount is the total spending on
#grocery
amount = 3400
#totalMarks is sum of marks in all the tests
#of Mathematics
totalMarks = test1 + test2 + finalTest
```

Program 5-4 Write a Python program to find the sum of two numbers.

```
#Program 5-4
#To find the sum of two numbers
num1 = 10
num2 = 20
result = num1 + num2
print(result)
```

Output:

```
30
```

5.6 EVERYTHING IS AN OBJECT

Python treats every value or data item whether numeric, string, or other type (discussed in the next section) as an object in the sense that it can be assigned to some variable or can be passed to a function as an argument.

Every object in Python is assigned a unique identity (ID) which remains the same for the lifetime of that object. This ID is akin to the memory address of the object. The function `id()` returns the identity of an object.

In the context of Object Oriented Programming (OOP), objects are a representation of the real world, such as employee, student, vehicle, box, book, etc. In any object oriented programming language like C++, JAVA, etc., each object has two things associated with it: (i) data or attributes and (ii) behaviour or methods. Further there are concepts of class and class hierarchies from which objects can be instantiated. However, OOP concepts are not in the scope of our present discussions.

Python also comes under the category of object oriented programming. However, in Python, the definition of object is loosely casted as some objects may not have attributes or others may not have methods.

Example 5.2

```
>>> num1 = 20
>>> id(num1)
1433920576          #identity of num1
>>> num2 = 30 - 10
>>> id(num2)
1433920576          #identity of num2 and num1
                         #are same as both
refers to                              #object 20
```

5.7 DATA TYPES

Every value belongs to a specific data type in Python. Data type identifies the type of data values a variable can hold and the operations that can be performed on that data. Figure 5.6 enlists the data types available in Python.

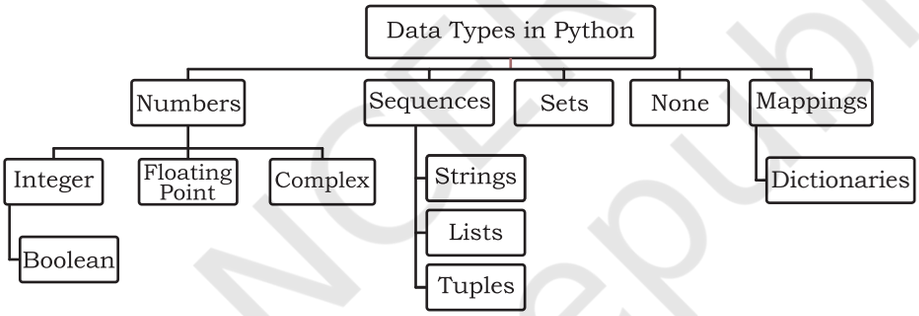


Figure 5.6: Different data types in Python

5.7.1 Number

Number data type stores numerical values only. It is further classified into three different types: `int`, `float` and `complex`.

Table 5.2 Numeric data types

Type/ Class	Description	Examples
`int`	integer numbers	–12, –3, 0, 125, 2
`float`	real or floating point numbers	–2.04, 4.0, 14.23
`complex`	complex numbers	3 + 4j, 2 – 2j

Boolean data type `(bool)` is a subtype of integer. It is a unique data type, consisting of two constants, `True` and `False`. Boolean `True` value is non-zero, non-null and non-empty. Boolean `False` is the value zero.

NOTES

Let us now try to execute few statements in interactive mode to determine the data type of the variable using built-in function `type()`.

Example 5.3

```
>>> num1 = 10
>>> type(num1)
<class 'int'>

>>> num2 = -1210
>>> type(num2)
<class 'int'>

>>> var1 = True
>>> type(var1)
<class 'bool'>

>>> float1 = -1921.9
>>> type(float1)
<class 'float'>
>>> float2 = -9.8*10**2
>>> print(float2, type(float2))
-980.0000000000001 <class 'float'>

>>> var2 = -3+7.2j
>>> print(var2, type(var2))
(-3+7.2j) <class 'complex'>
```

Variables of simple data types like integers, float, boolean, etc., hold single values. But such variables are not useful to hold a long list of information, for example, names of the months in a year, names of students in a class, names and numbers in a phone book or the list of artefacts in a museum. For this, Python provides data types like tuples, lists, dictionaries and sets.

5.7.2 Sequence

A Python sequence is an ordered collection of items, where each item is indexed by an integer. The three types of sequence data types available in Python are Strings, Lists and Tuples. We will learn about each of them in detail in later chapters. A brief introduction to these data types is as follows:

(A) String

String is a group of characters. These characters may be alphabets, digits or special characters including spaces. String values are enclosed either in single quotation

NOTES

marks (e.g., 'Hello') or in double quotation marks (e.g., "Hello"). The quotes are not a part of the string, they are used to mark the beginning and end of the string for the interpreter. For example,

```
>>> str1 = 'Hello Friend'
>>> str2 = "452"
```

We cannot perform numerical operations on strings, even when the string contains a numeric value, as in str2.

(B) List

List is a sequence of items separated by commas and the items are enclosed in square brackets [ ].

Example 5.4

```
#To create a list
>>> list1 = [5, 3.4, "New Delhi", "20C", 45]
#print the elements of the list list1
>>> print(list1)
[5, 3.4, 'New Delhi', '20C', 45]
```

(C) Tuple

Tuple is a sequence of items separated by commas and items are enclosed in parenthesis ( ). This is unlike list, where values are enclosed in brackets [ ]. Once created, we cannot change the tuple.

Example 5.5

```
#create a tuple tuple1
>>> tuple1 = (10, 20, "Apple", 3.4, 'a')
#print the elements of the tuple tuple1
>>> print(tuple1)
(10, 20, "Apple", 3.4, 'a')
```

5.7.3 Set

Set is an unordered collection of items separated by commas and the items are enclosed in curly brackets { }. A set is similar to list, except that it cannot have duplicate entries. Once created, elements of a set cannot be changed.

Example 5.6

```
#create a set
>>> set1 = {10,20,3.14,"New Delhi"}
>>> print(type(set1))
<class 'set'>
>>> print(set1)
{10, 20, 3.14, "New Delhi"}
#duplicate elements are not included in set
```

```
>>> set2 = {1,2,1,3}
>>> print(set2)
{1, 2, 3}
```

5.7.4 None

None is a special data type with a single value. It is used to signify the absence of value in a situation. `None` supports no special operations, and it is neither `False` nor 0 (zero).

Example 5.7

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

5.7.5 Mapping

Mapping is an unordered data type in Python. Currently, there is only one standard mapping data type in Python called dictionary.

(A) Dictionary

Dictionary in Python holds data items in key-value pairs. Items in a dictionary are enclosed in curly brackets { }. Dictionaries permit faster access to data. Every key is separated from its value using a colon (:) sign. The key : value pairs of a dictionary can be accessed using the key. The keys are usually strings and their values can be any data type. In order to access any value in the dictionary, we have to specify its key in square brackets [ ].

Example 5.8

```
#create a dictionary
>>> dict1 = {'Fruit':'Apple',
'Climate':'Cold', 'Price(kg)':120}
>>> print(dict1)
{'Fruit': 'Apple', 'Climate': 'Cold',
'Price(kg)': 120}
>>> print(dict1['Price(kg)'])
120
```

5.7.6 Mutable and Immutable Data Types

Sometimes we may require to change or update the values of certain variables used in a program. However, for certain data types, Python does not allow us to

change the values once a variable of that type has been created and assigned values.

Variables whose values can be changed after they are created and assigned are called mutable. Variables whose values cannot be changed after they are created and assigned are called immutable. When an attempt is made to update the value of an immutable variable, the old variable is destroyed and a new variable is created by the same name in memory.

Python data types can be classified into mutable and immutable as shown in Figure 5.7.

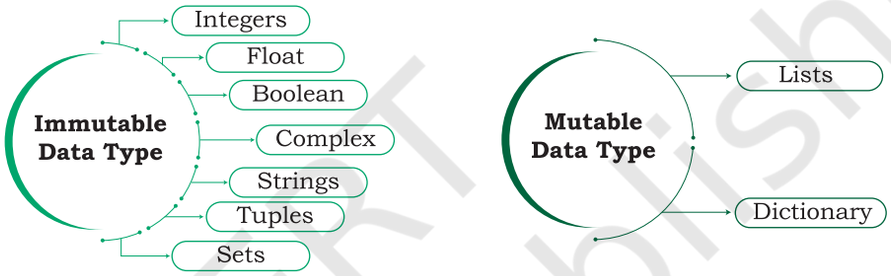


Figure 5.7: Classification of data types

Let us now see what happens when an attempt is made to update the value of a variable.

```
>>> num1 = 300
```

This statement will create an object with value 300 and the object is referenced by the identifier `num1` as shown in Figure 5.8.

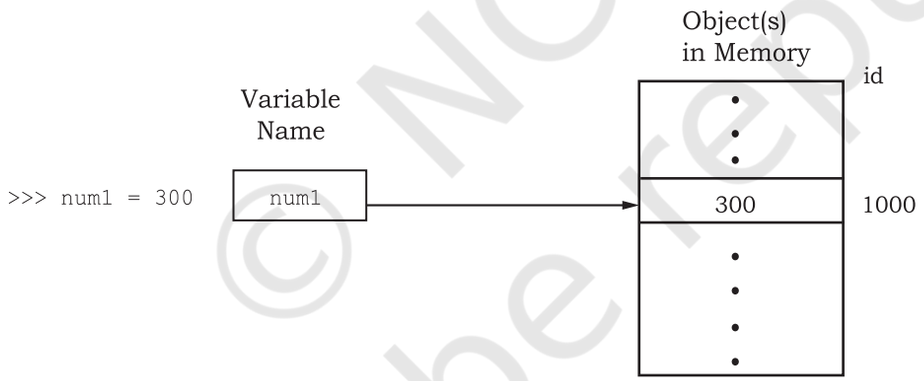


Figure 5.8: Object and its identifier

```
>>> num2 = num1
```

The statement `num2 = num1` will make `num2` refer to the value 300, also being referred by `num1`, and stored at memory location number, say 1000. So, `num1` shares the referenced location with `num2` as shown in Figure 5.9.

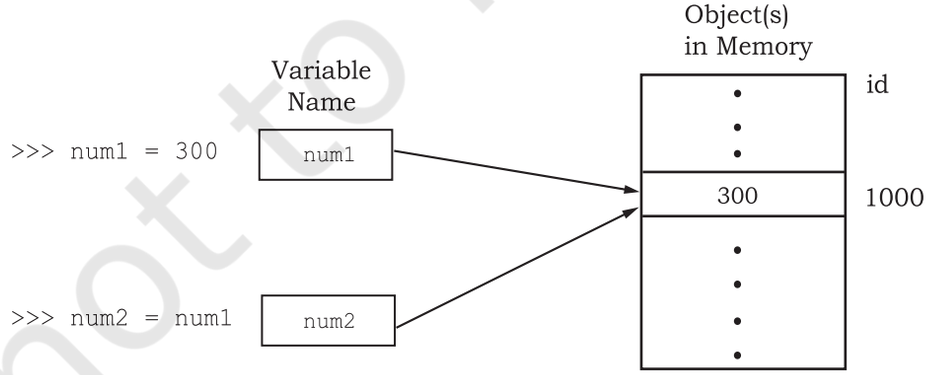


Figure 5.9: Variables with same value have same identifier

In this manner Python makes the assignment effective by copying only the reference, and not the data:

```
>>> num1 = num2 + 100
```

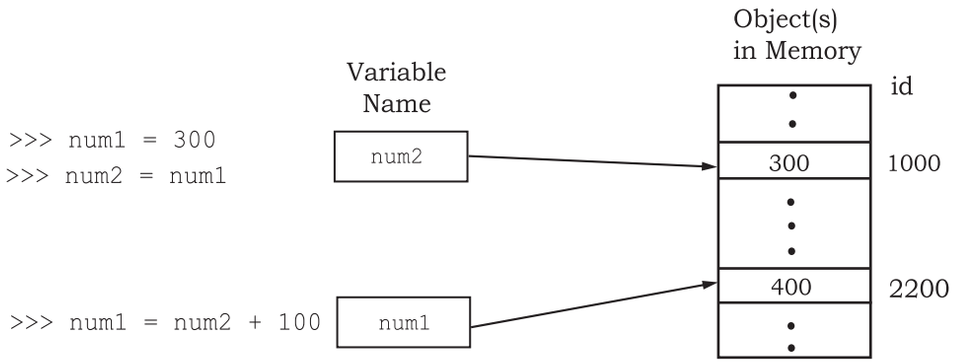


Figure 5.10: Variables with different values have different identifiers

This statement 1 `num1 = num2 + 100` links the variable `num1` to a new object stored at memory location number say 2200 having a value 400. As `num1` is an integer, which is an immutable type, it is rebuilt, as shown in Figure 5.10.

Python compares strings lexicographically, using ASCII value of the characters. If the first character of both the strings are same, the second character is compared, and so on.

5.7.7 Deciding Usage of Python Data Types

It is preferred to use lists when we need a simple iterable collection of data that may go for frequent modifications. For example, if we store the names of students of a class in a list, then it is easy to update the list when some new students join or some leave the course. Tuples are used when we do not need any change in the data. For example, names of months in a year. When we need uniqueness of elements and to avoid duplicacy it is preferable to use sets, for example, list of artefacts in a museum. If our data is being constantly modified or we need a fast lookup based on a custom key or we need a logical association between the key : value pair, it is advised to use dictionaries. A mobile phone book is a good application of dictionary.

5.8 OPERATORS

An operator is used to perform specific mathematical or logical operation on values. The values that the operators work on are called operands. For example, in the expression `10 + num`, the value `10`, and the variable `num` are operands and the + (plus) sign is an operator. Python supports several kinds of operators whose categorisation is briefly explained in this section.

5.8.1 Arithmetic Operators

Python supports arithmetic operators that are used to perform the four basic arithmetic operations as well as modular division, floor division and exponentiation.

Table 5.3 Arithmetic Operators in Python

Operator	Operation	Description	Example (Try in Lab)

+	Addition	Adds the two numeric values on either side of the operator
		
		This operator can also be used to concatenate two strings on either side of the operator
			`>>> num1 = 5`
			`>>> num2 = 6`
			`>>> num1 + num2`
			`11`
			`>>> str1 = "Hello"`
			`>>> str2 = "India"`
			`>>> str1 + str2`
			`'HelloIndia'`

-	Subtraction	Subtracts the operand on the right from the operand on the left	`>>> num1 = 5`
			`>>> num2 = 6`
			`>>> num1 - num2`
			`-1`

*	Multiplication	Multiplies the two values on both side of the operator
		
		Repeats the item on left of the operator if first operand is a string and second operand is an integer value
			`>>> num1 = 5`
			`>>> num2 = 6`
			`>>> num1 * num2`
			`30`
			`>>> str1 = 'India'`
			`>>> str1 * 2`
			`'IndiaIndia'`

/	Division	Divides the operand on the left by the operand on the right and returns the quotient	`>>> num1 = 8`
			`>>> num2 = 4`
			`>>> num2 / num1`
			`0.5`

%	Modulus	Divides the operand on the left by the operand on the right and returns the remainder	`>>> num1 = 13`
			`>>> num2 = 5`
			`>>> num1 % num2`
			`3`

//	Floor Division	Divides the operand on the left by the operand on the right and returns the quotient by removing the decimal part. It is sometimes also called integer division.	`>>> num1 = 13`
			`>>> num2 = 4`
			`>>> num1 // num2`
			`3`
			`>>> num2 // num1`
			`0`

**	Exponent	Performs exponential (power) calculation on operands. That is, raise the operand on the left to the power of the operand on the right	`>>> num1 = 3`
			`>>> num2 = 4`
			`>>> num1 ** num2`
			`81`

5.8.2 Relational Operators

Relational operator compares the values of the operands on its either side and determines the relationship among

them. Assume the Python variables `num1 = 10, num2 = 0, num3 = 10, str1 = "Good", str2 = "Afternoon"` for the following examples:

Table 5.4 Relational operators in Python

Operator	Operation	Description	Example (Try in Lab)

==	Equals to	If the values of two operands are equal, then the condition is True, otherwise it is False	`>>> num1 == num2`
			`False`
			`>> str1 == str2`
			`False`

!=	Not equal to	If values of two operands are not equal, then condition is True, otherwise it is False	`>>> num1 != num2`
			`True`
			`>>> str1 != str2`
			`True`
			`>>> num1 != num3`
			`False`

>	Greater than	If the value of the left-side operand is greater than the value of the right-side operand, then condition is True, otherwise it is False	`>>> num1 > num2`
			`True`
			`>>> str1 > str2`
			`True`

<	Less than	If the value of the left-side operand is less than the value of the right-side operand, then condition is True, otherwise it is False	`>>> num1 < num3`
			`False`
			`>>> str2 < str1`
			`True`

>=	Greater than or equal to	If the value of the left-side operand is greater than or equal to the value of the right-side operand, then condition is True, otherwise it is False	`>>> num1 >= num2`
			`True`
			`>>> num2 >= num3`
			`False`
			`>>> str1 >= str2`
			`True`

<=	Less than or equal to	If the value of the left operand is less than or equal to the value of the right operand, then is True otherwise it is False	`>>> num1 <= num2`
			`False`
			`>>> num2 <= num3`
			`True`
			`>>> str1 <= str2`
			`False`

5.8.3 Assignment Operators

Assignment operator assigns or changes the value of the variable on its left.

Table 5.5 Assignment operators in Python

Operator	Description	Example (Try in Lab)

=	Assigns value from right-side operand to left-side operand	`>>> num1 = 2`
		`>>> num2 = num1`
		`>>> num2`
		`2`
		`>>> country = 'India'`
		`>>> country`
		`'India'`

+=	It adds the value of right-side operand to the left-side operand and assigns the result to the left-side operand
	Note: x += y is same as x = x + y
		>>> num1 = 10
		>>> num2 = 2
		>>> num1 += num2
		>>> num1
		12
		>>> num2
		2
		>>> str1 = 'Hello'
		>>> str2 = 'India'
		>>> str1 += str2
		>>> str1
		'HelloIndia'

-=	It subtracts the value of right-side operand from the left-side operand and assigns the result to left-side operand
	Note: x -= y is same as x = x - y
		>>> num1 = 10
		>>> num2 = 2
		>>> num1 -= num2
		>>> num1
		8

*=	It multiplies the value of right-side operand with the value of left-side operand and assigns the result to left-side operand
	Note: x *= y is same as x = x * y
		>>> num1 = 2
		>>> num2 = 3
		>>> num1 *= 3
		
		>>> num1
		6
		>>> a = 'India'
		>>> a *= 3
		>>> a
		'IndiaIndiaIndia'

/=	It divides the value of left-side operand by the value of right-side operand and assigns the result to left-side operand
	Note: x /= y is same as x = x / y
		>>> num1 = 6
		>>> num2 = 3
		>>> num1 /= num2
		>>> num1
		2.0

%=	It performs modulus operation using two operands and assigns the result to left-side operand
	Note: x %= y is same as x = x % y
		>>> num1 = 7
		>>> num2 = 3
		>>> num1 %= num2
		>>> num1
		1

//=	It performs floor division using two operands and assigns the result to left-side operand
	Note: x //= y is same as x = x // y
		>>> num1 = 7
		>>> num2 = 3
		>>> num1 //= num2
		>>> num1
		2

**=	It performs exponential (power) calculation on operators and assigns value to the left-side operand
	Note: x **= y is same as x = x ** y
		>>> num1 = 2
		>>> num2 = 3
		>>> num1 **= num2
		>>> num1
		8

5.8.4 Logical Operators

There are three logical operators supported by Python. These operators (`and, or, not`) are to be written in lower case only. The logical operator evaluates to either `True` or `False` based on the logical operands on either side. Every value is logically either `True` or `False`. By default, all values are `True` except `None`, `False`, 0 (zero), empty collections "", (), [], {}, and few other special values. So if we say `num1 = 10, num2 = -20`, then both `num1` and `num2` are logically `True`.

Table 5.6 Logical operators in Python

Operator	Operation	Description	Example (Try in Lab)

`and`	Logical AND	If both the operands are `True`, then condition becomes `True`	`>>> True and True`
			`True`
			`>>> num1 = 10`
			`>>> num2 = -20`
			`>>> bool(num1 and num2)`
			`True`
			`>>> True and False`
			`False`
			`>>> num3 = 0`
			`>>> bool(num1 and num3)`
			`False`
			`>>> False and False`
			`False`

`or`	Logical OR	If any of the two operands are `True`, then condition becomes `True`	`>>> True or True`
			`True`
			`>>> True or False`
			`True`
			`>>> bool(num1 or num3)`
			`True`
			`>>> False or False`
			`False`

`not`	Logical NOT	Used to reverse the logical state of its operand	`>>> num1 = 10`
			`>>> bool(num1)`
			`True`
			`>>> not num1`
			`>>> bool(num1)`
			`False`

5.8.5 Identity Operators

Identity operators are used to determine whether the value of a variable is of a certain type or not. Identity operators can also be used to determine whether two

variables are referring to the same object or not. There are two identity operators.

Table 5.7 Identity operators in Python

Operator	Description	Example (Try in Lab)

is	Evaluates `True` if the variables on either side of the operator point towards the same memory location and `False` otherwise. `var1` is `var2` results to `True` if id(`var1`) is equal to id(`var2`)	`>>> num1 = 5`
		`>>> type(num1) is int`
		`True`
		`>>> num2 = num1`
		`>>> id(num1)`
		`1433920576`
		`>>> id(num2)`
		`1433920576`
		`>>> num1 is num2`
		`True`

is not	Evaluates to `False` if the variables on either side of the operator point to the same memory location and `True` otherwise. `var1` is not `var2` results to `True` if id(`var1`) is not equal to id(`var2`)	`>>> num1 is not num2` `False`

5.8.6 Membership Operators

Membership operators are used to check if a value is a member of the given sequence or not.

Table 5.8 Membership operators in Python

Operator	Description	Example (Try in Lab)

in	Returns `True` if the variable/value is found in the specified sequence and `False` otherwise	`>>> a = [1,2,3]`
		`>>> 2 in a`
		`True`
		`>>> '1' in a`
		`False`

not in	Returns `True` if the variable/value is not found in the specified sequence and `False` otherwise	`>>> a = [1,2,3]`
		`>>> 10 not in a`
		`True`
		`>>> 1 not in a`
		`False`

5.9 EXPRESSIONS

An expression is defined as a combination of constants, variables, and operators. An expression always evaluates to a value. A value or a standalone variable is also considered as an expression but a standalone operator is not an expression. Some examples of valid expressions are given below.

```
  (i) 100                (iv) 3.0 + 3.14
 (ii) num                 (v) 23/3 -5 * 7(14 -2)
(iii) num - 20.4         (vi) "Global" + "Citizen"
```

5.9.1 Precedence of Operators

Evaluation of the expression is based on precedence of operators. When an expression contains different kinds of operators, precedence determines which operator should be applied first. Higher precedence operator is evaluated before the lower precedence operator. Most of the operators studied till now are binary operators. Binary operators are operators with two operands. The unary operators need only one operand, and they have a higher precedence than the binary operators. The minus (-) as well as + (plus) operators can act as both unary and binary operators, but `not` is a unary logical operator.

```
#Depth is using - (minus) as unary operator
Value = -Depth
#not is a unary operator, negates True
print(not(True))
```

The following table lists precedence of all operators from highest to lowest.

Table 5.9 Precedence of all operators in Python

Order of Precedence	Operators	Description
1	**	Exponentiation (raise to the power)
2	~ ,+, -	Complement, unary plus and unary minus
3	* ,/, %, //	Multiply, divide, modulo and floor division
4	+, -	Addition and subtraction
5	<= , < , > , >=, == , !=	Relational and Comparison operators
6	=, %=, /=, //=, -=, +=, *=, **=	Assignment operators
7	is, is not	Identity operators
8	in, not in	Membership operators
9	not	Logical operators
10	and	
11	or	

Note:

a) Parenthesis can be used to override the precedence of operators. The expression within () is evaluated first.

b) For operators with equal precedence, the expression is evaluated from left to right.

Example 5.9 How will Python evaluate the following expression?

```
20 + 30 * 40
```

NOTES

Solution:

```
= 20 + (30 * 40)          #Step 1
```
#precedence of * is more than that of +
```
= 20 + 1200               #Step 2
= 1220                    #Step 3
```

Example 5.10 How will Python evaluate the following expression?

```
20 - 30 + 40
```

Solution:

The two operators (–) and (+) have equal precedence. Thus, the first operator, i.e., subtraction is applied before the second operator, i.e., addition (left to right).

```
= (20 - 30) + 40    #Step 1
= -10 + 40          #Step 2
= 30                #Step 3
```

Example 5.11 How will Python evaluate the following expression?

```
(20 + 30) * 40
```

Solution:

```
= (20 + 30) * 40          # Step 1
```
#using parenthesis(), we have forced precedence of + to be more than that of *
```
= 50 * 40                 # Step 2
= 2000                    # Step 3
```

Example 5.12 How will the following expression be evaluated in Python?

```
15.0 / 4 + (8 + 3.0)
```

Solution:

```
= 15.0 / 4 + (8.0 + 3.0)  #Step 1
= 15.0 / 4.0 + 11.0       #Step 2
= 3.75 + 11.0             #Step 3
= 14.75                   #Step 4
```

5.10 STATEMENT

In Python, a statement is a unit of code that the Python interpreter can execute.

Example 5.13

```
>>> x = 4              #assignment statement
>>> cube = x ** 3      #assignment statement
>>> print (x, cube)    #print statement
4 64
```

5.11 Input and Output

Sometimes, a program needs to interact with the user's to get some input data or information from the end user and process it to give the desired output. In Python, we have the `input()` function for taking the user input. The `input()` function prompts the user to enter data. It accepts all user input as string. The user may enter a number or a string but the `input()` function treats them as strings only. The syntax for `input()` is:

```
input ([Prompt])
```

`Prompt` is the string we may like to display on the screen prior to taking the input, and it is optional. When a prompt is specified, first it is displayed on the screen after which the user can enter data. The `input()` takes exactly what is typed from the keyboard, converts it into a string and assigns it to the variable on left-hand side of the assignment operator (=). Entering data for the input function is terminated by pressing the enter key.

Example 5.14

```
>>> fname = input("Enter your first name: ")
Enter your first name: Arnab
>>> age = input("Enter your age: ")
Enter your age: 19
>>> type(age)
<class 'str'>
```

The variable `fname` will get the string 'Arnab', entered by the user. Similarly, the variable `age` will get the string '19'. We can typecast or change the datatype of the string data accepted from user to an appropriate numeric value. For example, the following statement will convert the accepted string to an integer. If the user enters any non-numeric value, an error will be generated.

Example 5.15

```
#function int() to convert string to integer
>>> age = int( input("Enter your age:"))
Enter your age: 19
>>> type(age)
<class 'int'>
```

Python uses the `print()` function to output data to standard output device — the screen. We will learn about function in Chapter 7. The function `print()` evaluates the expression before displaying it on the screen. The `print()`

Observe that a plus sign does not add any space between the two strings while a comma inserts a space between two strings in a print statement.

outputs a complete line and then moves to the next line for subsequent output. The syntax for `print()` is:

```
print(value [, ..., sep = ' ', end = '\n'])
```

- `sep`: The optional parameter `sep` is a separator between the output values. We can use a character, integer or a string as a separator. The default separator is space.
- `end`: This is also optional and it allows us to specify any string to be appended after the last value. The default is a new line.

Example 5.16

Statement	Output
`print("Hello")`	`Hello`
`print(10*2.5)`	`25.0`
`print("I" + "love" + "my" + "country")`	`Ilovemycountry`
`print("I'm", 16, "years old")`	`I'm 16 years old`

The third print function in the above example is concatenating strings, and we use + (plus) between two strings to concatenate them. The fourth print function also appears to be concatenating strings but uses commas (,) between strings. Actually, here we are passing multiple arguments, separated by commas to the print function. As arguments can be of different types, hence the print function accepts integer (16) along with strings here. But in case the print statement has values of different types and '+' is used instead of comma, it will generate an error as discussed in the next section under explicit conversion.

5.12 TYPE CONVERSION

Consider the following program

```
num1 = input("Enter a number and I'll double
it: ")
num1 = num1 * 2
print(num1)
```

The program was expected to display double the value of the number received and store in variable `num1`. So if a user enters 2 and expects the program to display 4 as the output, the program displays the following result:

```
Enter a number and I'll double it: 2
22
```

This is because the value returned by the input function is a string ("2") by default. As a result, in statement num1 = num1 * 2, num1 has string value and * acts as repetition operator which results in output as "22". To get 4 as output, we need to convert the data type of the value entered by the user to integer. Thus, we modify the program as follows:

```
num1 = input("Enter a number and I'll double
it: ")
num1 = int(num1)   #convert string input to
                   #integer
num1 = num1 * 2
print(num1)
```

Now, the program will display the expected output as follows:

```
Enter a number and I'll double it: 2
4
```

Let us now understand what is type conversion and how it works. As and when required, we can change the data type of a variable in Python from one type to another. Such data type conversion can happen in two ways: either explicitly (forced) when the programmer specifies for the interpreter to convert a data type to another type; or implicitly, when the interpreter understands such a need by itself and does the type conversion automatically.

5.12.1 Explicit Conversion

Explicit conversion, also called type casting happens when data type conversion takes place because the programmer forced it in the program. The general form of an explicit data type conversion is:

```
(new_data_type) (expression)
```

With explicit type conversion, there is a risk of loss of information since we are forcing an expression to be of a specific type. For example, converting a floating value of x = 20.67 into an integer type, i.e., int(x) will discard the fractional part .67. Following are some of the functions in Python that are used for explicitly converting an expression or a variable to a different type.

Table 5.10 Explicit type conversion functions in Python

Function	Description
int(x)	Converts x to an integer
float(x)	Converts x to a floating-point number

`str(x)`	Converts x to a string representation
`chr(x)`	Converts ASCII value of x to character
`ord(x)`	returns the character associated with the ASCII code x

Program 5-5 Program of explicit type conversion from int to float.

```
#Program 5-5
#Explicit type conversion from int to float
num1 = 10
num2 = 20
num3 = num1 + num2
print(num3)
print(type(num3))
num4 = float(num1 + num2)
print(num4)
print(type(num4))
```

Output:

```
30
<class 'int'>
30.0
<class 'float'>
```

Program 5-6 Program of explicit type conversion from float to int.

```
#Program 5-6
#Explicit type conversion from float to int
num1 = 10.2
num2 = 20.6
num3 = (num1 + num2)
print(num3)
print(type(num3))
num4 = int(num1 + num2)
print(num4)
print(type(num4))
```

Output:

```
30.8
<class 'float'>
30
<class 'int'>
```

Program 5-7 Example of type conversion between numbers and strings.

```
#Program 5-7
#Type Conversion between Numbers and Strings
priceIcecream = 25
priceBrownie = 45
totalPrice = priceIcecream + priceBrownie
print("The total is Rs." + totalPrice )
```

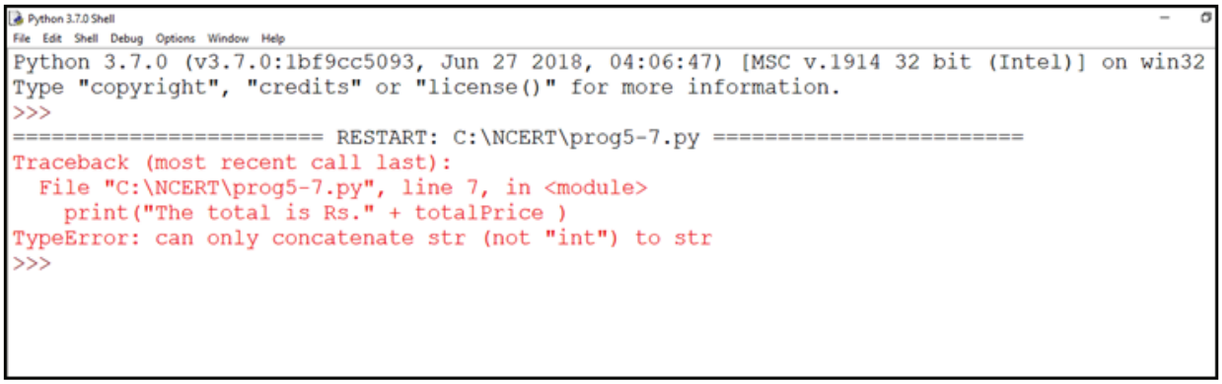


Figure 5.11: Output of program 5-7

On execution, program 5-7 gives an error as shown in Figure 5.11, informing that the interpreter cannot convert an integer value to string implicitly. It may appear quite intuitive that the program should convert the integer value to a string depending upon the usage. However, the interpreter may not decide on its own when to convert as there is a risk of loss of information. Python provides the mechanism of the explicit type conversion so that one can clearly state the desired outcome. Program 5-8 works perfectly using explicit type casting:

Program 5-8 Program to show explicit type casting.

```
#Program 5-8
#Explicit type casting
priceIcecream = 25
priceBrownie = 45
totalPrice = priceIcecream + priceBrownie
print("The total in Rs." + str(totalPrice))
```

Output:

```
The total in Rs.70
```

Similarly, type casting is needed to convert float to string. In Python, one can convert string to integer or float values whenever required.

Program 5-9 Program to show explicit type conversion.

```
#Program 5-9
#Explicit type conversion
icecream = '25'
brownie = '45'
#String concatenation
price = icecream + brownie
print("Total Price Rs." + price)
#Explicit type conversion - string to integer
```

```
price = int(icecream)+int(brownie)
print("Total Price Rs." + str(price))
```

Output:

```
Total Price Rs.2545
Total Price Rs.70
```

5.12.2 Implicit Conversion

Implicit conversion, also known as coercion, happens when data type conversion is done automatically by Python and is not instructed by the programmer.

Program 5-10 Program to show implicit conversion from int to float.

```
#Program 5-10
#Implicit type conversion from int to float

num1 = 10           #num1 is an integer
num2 = 20.0         #num2 is a float
sum1 = num1 + num2  #sum1 is sum of a float
and an integer
print(sum1)
print(type(sum1))
```

Output:

```
30.0
<class 'float'>
```

In the above example, an integer value stored in variable `num1` is added to a float value stored in variable `num2`, and the result was automatically converted to a float value stored in variable `sum1` without explicitly telling the interpreter. This is an example of implicit data conversion. One may wonder why was the float value not converted to an integer instead? This is due to type promotion that allows performing operations (whenever possible) by converting data into a wider-sized data type without any loss of information.

5.13 DEBUGGING

A programmer can make mistakes while writing a program, and hence, the program may not execute or may generate wrong output. The process of identifying and removing such mistakes, also known as bugs or errors, from a program is called debugging. Errors occurring in programs can be categorised as:

i) Syntax errors

ii) Logical errors

iii) Runtime errors

5.13.1 Syntax Errors

Like other programming languages, Python has its own rules that determine its syntax. The interpreter interprets the statements only if it is syntactically (as per the rules of Python) correct. If any syntax error is present, the interpreter shows error message(s) and stops the execution there. For example, parentheses must be in pairs, so the expression (10 + 12) is syntactically correct, whereas (7 + 11 is not due to absence of right parenthesis. Such errors need to be removed before the execution of the program

5.13.2 Logical Errors

A logical error is a bug in the program that causes it to behave incorrectly. A logical error produces an undesired output but without abrupt termination of the execution of the program. Since the program interprets successfully even when logical errors are present in it, it is sometimes difficult to identify these errors. The only evidence to the existence of logical errors is the wrong output. While working backwards from the output of the program, one can identify what went wrong.

For example, if we wish to find the average of two numbers 10 and 12 and we write the code as 10 + 12/2, it would run successfully and produce the result 16. Surely, 16 is not the average of 10 and 12. The correct code to find the average should have been (10 + 12)/2 to give the correct output as 11.

Logical errors are also called semantic errors as they occur when the meaning of the program (its semantics) is not correct.

5.13.3 Runtime Error

A runtime error causes abnormal termination of program while it is executing. Runtime error is when the statement is correct syntactically, but the interpreter cannot execute it. Runtime errors do not appear until after the program starts running or executing.

For example, we have a statement having division operation in the program. By mistake, if the denominator entered is zero then it will give a runtime error like "division by zero".

Let us look at the program 5-11 showing two types of runtime errors when a user enters non-integer value

or value '0'. The program generates correct output when the user inputs an integer value for num2.

Program 5-11 Example of a program which generates runtime error.

```
#Program 5-11
#Runtime Errors Example
num1 = 10.0
num2 = int(input("num2 = "))
#if user inputs a string or a zero, it leads
to runtime error
print(num1/num2)
```

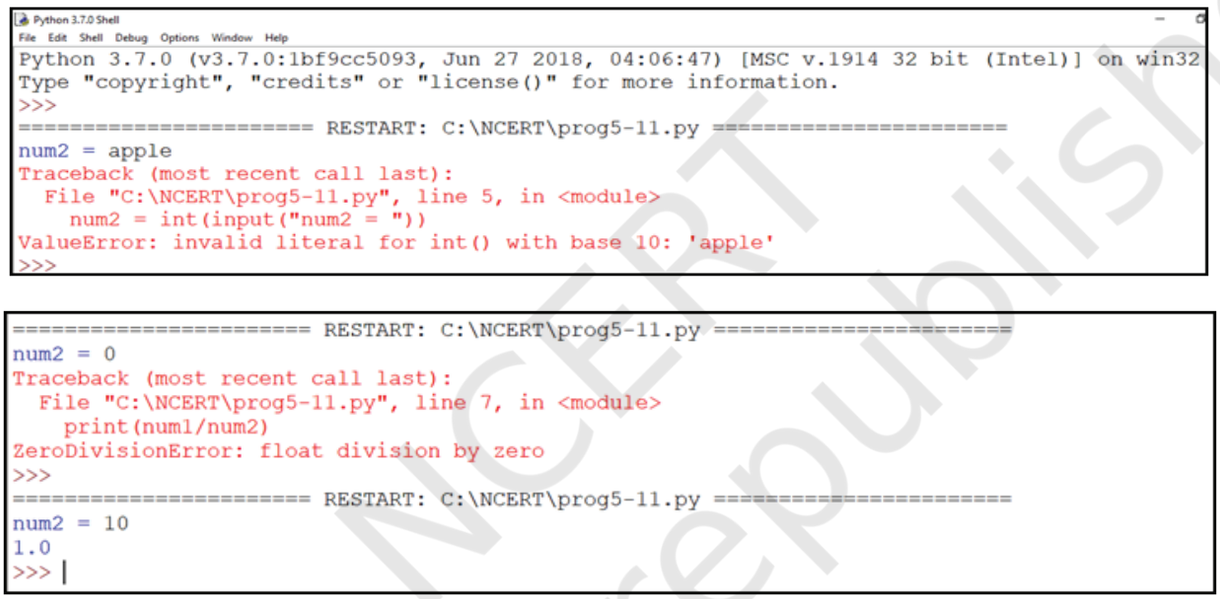


Figure 5.11: Output of program 5-11

SUMMARY

- Python is an open-source, high level, interpreter-based language that can be used for a multitude of scientific and non-scientific computing purposes.
- Comments are non-executable statements in a program.
- An identifier is a user defined name given to a variable or a constant in a program.
- The process of identifying and removing errors from a computer program is called debugging.
- Trying to use a variable that has not been assigned a value gives an error.
- There are several data types in Python — integer, boolean, float, complex, string, list, tuple, sets, None and dictionary.

- Datatype conversion can happen either explicitly or implicitly.
- Operators are constructs that manipulate the value of operands. Operators may be unary or binary.
- An expression is a combination of values, variables and operators.
- Python has `input()` function for taking user input.
- Python has `print()` function to output data to a standard output device.

EXERCISE

1. Which of the following identifier names are invalid and why?

i	`Serial_no.`	v	`Total_Marks`
ii	`1st_Room`	vi	`total-Marks`
iii	`Hundred$`	vii	`_Percentage`
iv	`Total Marks`	viii	`True`

2. Write the corresponding Python assignment statements:
   a) Assign 10 to variable `length` and 20 to variable `breadth`.
   b) Assign the average of values of variables `length` and `breadth` to a variable `sum`.
   c) Assign a list containing strings 'Paper', 'Gel Pen', and 'Eraser' to a variable `stationery`.
   d) Assign the strings 'Mohandas', 'Karamchand', and 'Gandhi' to variables `first`, `middle` and `last`.
   e) Assign the concatenated value of string variables `first`, `middle` and `last` to variable `fullname`. Make sure to incorporate blank spaces appropriately between different parts of names.
3. Write logical expressions corresponding to the following statements in Python and evaluate the expressions (assuming variables `num1`, `num2`, `num3`, `first`, `middle`, `last` are already having meaningful values):
   a) The sum of 20 and –10 is less than 12.
   b) `num3` is not more than 24.

NOTES

NOTES

c) 6.75 is between the values of integers num1 and num2.

d) The string 'middle' is larger than the string 'first' and smaller than the string 'last'.

e) List Stationery is empty.

4. Add a pair of parentheses to each expression so that it evaluates to True.

a) `0 == 1 == 2`

b) `2 + 3 == 4 + 5 == 7`

c) `1 < -1 == 3 > 4`

5. Write the output of the following:

a)
```
num1 = 4
num2 = num1 + 1
num1 = 2
print (num1, num2)
```

b)
```
num1, num2 = 2, 6
num1, num2 = num2, num1 + 2
print (num1, num2)
```

c)
```
num1, num2 = 2, 3
num3, num2 = num1, num3 + 1
print (num1, num2, num3)
```

6. Which data type will be used to represent the following data values and why?

a) Number of months in a year

b) Resident of Delhi or not

c) Mobile number

d) Pocket money

e) Volume of a sphere

f) Perimeter of a square

g) Name of the student

h) Address of the student

7. Give the output of the following when num1 = 4, num2 = 3, num3 = 2

a)
```
num1 += num2 + num3
print (num1)
```

b)
```
num1 = num1 ** (num2 + num3)
print (num1)
```

c)
```
num1 **= num2 + num3
```

d)
```
num1 = '5' + '5'
print(num1)
```

NOTES

e) `print(4.00/(2.0+2.0))`

f)
```
num1 = 2+9*((3*12)-8)/10
print(num1)
```

g)
```
num1 = 24 // 4 // 2
print(num1)
```

h)
```
num1 = float(10)
print (num1)
```

i)
```
num1 = int('3.14')
print (num1)
```

j) `print('Bye'  == 'BYE')`

k) `print(10 != 9 and 20 >= 20)`

l)
```
print(10 + 6 * 2 ** 2 != 9//4 -3 and 29
>= 29/9)
```

m) `print(5 % 10 + 10 < 50 and 29 <= 29)`

n)
```
print((0 < 6) or (not(10 == 6) and
(10<0)))
```

8. Categorise the following as syntax error, logical error or runtime error:

   a) `25 / 0`

   b) `num1 = 25; num2 = 0; num1/num2`

9. A dartboard of radius 10 units and the wall it is hanging on are represented using a two-dimensional coordinate system, with the board's center at coordinate (0,0). Variables x and y store the x-coordinate and the y-coordinate of a dart that hits the dartboard. Write a Python expression using variables x and y that evaluates to `True` if the dart hits (is within) the dartboard, and then evaluate the expression for these dart coordinates:

   a) `(0,0)`

   b) `(10,10)`

   c) `(6, 6)`

   d) `(7,8)`

10. Write a Python program to convert temperature in degree Celsius to degree Fahrenheit. If water boils at 100 degree C and freezes as 0 degree C, use the program to find out what is the boiling point and freezing point of water on the Fahrenheit scale.

    (Hint: T(°F) = T(°C) × 9/5 + 32)

11. Write a Python program to calculate the amount payable if money has been lent on simple interest.

NOTES

Principal or money lent = P, Rate of interest = R% per annum and Time = T years. Then Simple Interest (SI) = (P x R x T)/ 100.

Amount payable = Principal + SI.

P, R and T are given as input to the program.

12. Write a program to calculate in how many days a work will be completed by three persons A, B and C together. A, B, C take x days, y days and z days respectively to do the job alone. The formula to calculate the number of days if they work together is xyz/(xy + yz + xz) days where x, y, and z are given as input to the program.
13. Write a program to enter two integers and perform all arithmetic operations on them.
14. Write a program to swap two numbers using a third variable.
15. Write a program to swap two numbers without using a third variable.
16. Write a program to repeat the string "GOOD MORNING" n times. Here 'n' is an integer entered by the user.
17. Write a program to find average of three numbers.
18. The volume of a sphere with radius r is $4/3\pi r^3$. Write a Python program to find the volume of spheres with radius 7cm, 12cm, 16cm, respectively.
19. Write a program that asks the user to enter their name and age. Print a message addressed to the user that tells the user the year in which they will turn 100 years old.
20. The formula $E = mc^2$ states that the equivalent energy (E) can be calculated as the mass (m) multiplied by the speed of light (c = about 3×10^8 m/s) squared. Write a program that accepts the mass of an object and determines its energy.
21. Presume that a ladder is put upright against a wall. Let variables length and angle store the length of the ladder and the angle that it forms with the ground as it leans against the wall. Write a Python program to compute

the height reached by the ladder on the wall for the following values of length and angle:

a) 16 feet and 75 degrees
b) 20 feet and 0 degrees
c) 24 feet and 45 degrees
d) 24 feet and 80 degrees

Case Study-based Question

Schools use "Student Management Information System" (SMIS) to manage student-related data. This system provides facilities for:

- recording and maintaining personal details of students.
- maintaining marks scored in assessments and computing results of students.
- keeping track of student attendance.
- managing many other student-related data. Let us automate this process step by step.

Identify the personal details of students from your school identity card and write a program to accept these details for all students of your school and display them in the following format.

Name of School

Student Name: PQR Roll No: 99
Class: XI Section: A
Address : Address Line 1
Address Line 2

City: ABC Pin Code: 999999
Parent's/ Guardian's Contact No: 9999999999

Documentation Tips

It is a fact that a properly documented program is easy to read, understand and is flexible for future development. Therefore, it is important that one pays extra attention to documentation while coding. Let us assess the documentation done by us in our case study program and also find out whether our friends also pay similar attention to documentation or not.

NOTES

Following is a checklist of good documentation points:

- Objective of the program is clearly stated in the beginning.
- Objective of each function is clearly mentioned in the beginning of each function.
- Comments are inserted at the proper place so as to enhance the understandability and readability of the program.
  (***Note:*** Over commenting doesn't help)
- Variables and function names are meaningful and appropriate.
- Single letter variable names are not used.
- Program name is meaningful.
  (***Note:*** It is not proper to use your name as program name, for example, 'raman. py' or 'namya.py' to denote your program code. It is more appropriate to use the program name as 'bankingProject.py' for a banking related program or 'admProcess' for an admission related program.)
- Program code is properly indented.
- Same naming conventions are used throughout the program.
  (***Note:*** Some of the naming conventions are firstNum, first_num, to denote the variable having first number).

Let's do this exercise for our peer's case studies as well and provide a feedback to them.

A relevant peer feedback helps in improving the documentation of the projects. It also helps in identifying our mistakes and enriches us with better ideas used by others.

Chapter 6

Flow of Control

11120CH06

6.1 Introduction

"Don't you hate code that's not properly indented? Making it [indenting] part of the syntax guarantees that all code is properly indented."

– G. van Rossum

In Figure 6.1, we see a bus carrying the children to school. There is only one way to reach the school. The driver has no choice, but to follow the road one milestone after another to reach the school. We learnt in Chapter 5 that this is the concept of sequence, where Python executes one statement after another from beginning to the end of the program. These are the kind of programs we have been writing till now.

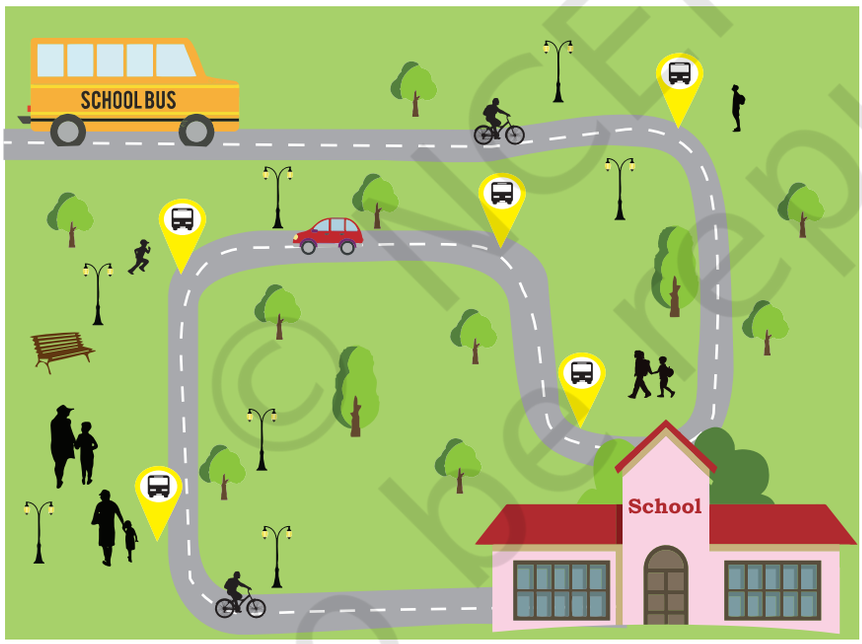


Figure 6.1: Bus carrying students to school

Let us consider a program 6-1 that executes in sequence, that is, statements are executed in an order in which they are written.

The order of execution of the statements in a program is known as flow of control. The flow of control can be implemented using control structures. Python supports two types of control structures—selection and repetition.

In this chapter

- » *Introduction to Flow of Control*
- » *Selection*
- » *Indentation*
- » *Repetition*
- » *Break and Continue Statements*
- » *Nested Loops*

Program 6-1 Program to print the difference of two numbers.

```
#Program 6-1
#Program to print the difference of two input numbers
num1 = int(input("Enter first number: "))
num2 = int(input("Enter second number: "))
diff = num1 - num2
print("The difference of",num1,"and",num2,"is",diff)
```

Output:

```
Enter first number 5
Enter second number 7
The difference of 5 and 7 is -2
```

6.2 SELECTION

Now suppose we have ₹10 to buy a pen. On visiting the stationery shop, there are a variety of pens priced at ₹10 each. Here, we have to decide which pen to buy. Similarly, when we use the direction services of a digital map, to reach from one place to another, we notice that sometimes it shows more than one path like the least crowded path, shortest distance path, etc. We decide the path as per our priority. A decision involves selecting from one of the two or more possible options. In programming, this concept of decision making or selection is implemented with the help of `if..else` statement.

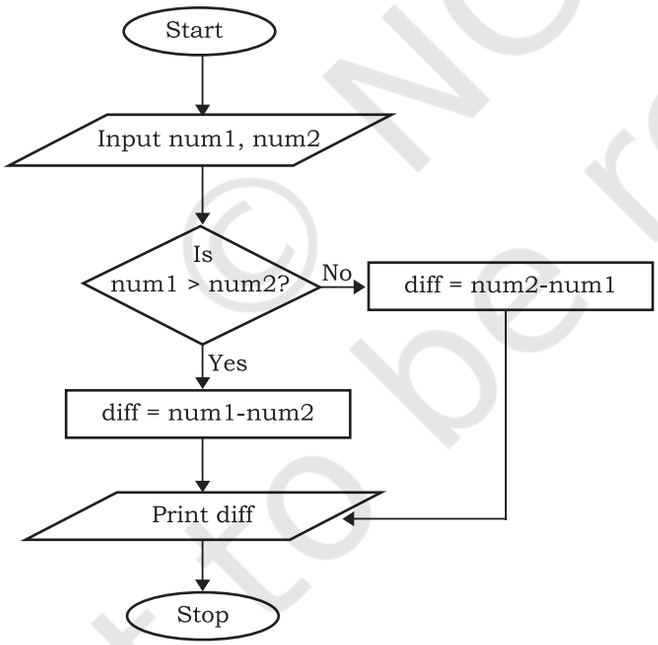


Figure 6.2: Flow chart depicting decision making

Now, suppose we want to display the positive difference of the two numbers num1 and num2 given at program 6-1. For that, we need to modify our approach. Look at the flowchart shown in Figure 6.2 that subtracts the smaller number from the bigger number so that we always get a positive difference. This selection is based upon the values that are input for the two numbers num1 and num2.

The syntax of `if` statement is:

```
if condition:
     statement(s)
```

In the following example, if the age entered by the user is greater than 18, then print that the user is eligible to vote. If the condition is true, then the indented statement(s) are executed. The indentation implies that its execution is dependent on the condition. There is no limit on the number of statements that can appear as a block under the `if` statement.

Example 6.1

```
age = int(input("Enter your age "))
if age >= 18:
    print("Eligible to vote")
```

A variant of `if` statement called `if..else` statement allows us to write two alternative paths and the control condition determines which path gets executed. The syntax for `if..else` statement is as follows.

```
if condition:
     statement(s)
else:
     statement(s)
```

Let us now modify the example on voting with the condition that if the age entered by the user is greater than 18, then to display that the user is eligible to vote. Otherwise display that the user is not eligible to vote.

```
age = int(input("Enter your age: "))
if age >= 18:
     print("Eligible to vote")
else:
     print("Not eligible to vote")
```

Now let us use the same concept to modify program 6-1, so that it always gives a positive difference as the output. From the flow chart in Figure 6.2, it is clear that we need to decide whether num1 > num2 or not and take action accordingly.

We have to specify two blocks of statements since num1 can be greater than num2 or vice-versa as shown in program 6-2.

Many a times there are situations that require multiple conditions to be checked and it may lead to many alternatives. In such cases we can chain the conditions using `if..elif` (`elif` means `else..if`).

Program 6-2 Program to print the positive difference of two numbers.

```
#Program 6-2
#Program to print the positive difference of two numbers
num1 = int(input("Enter first number: "))
num2 = int(input("Enter second number: "))
if num1 > num2:
     diff = num1 - num2
else:
     diff = num2 - num1
print("The difference of",num1,"and",num2,"is",diff)
```

Output:

```
Enter first number: 5
Enter second number: 6
The difference of 5 and 6 is 1
```

The syntax for a selection structure using `elif` is as shown below.

```
if condition:
     statement(s)
elif condition:
     statement(s)
elif condition:
     statement(s)
else:
     statement(s)
```

Example 6.2 Check whether a number is positive, negative, or zero.

```
number = int(input("Enter a number: ")
if number > 0:
     print("Number is positive")
elif number < 0:
     print("Number is negative")
else:
     print("Number is zero")
```

Example 6.3 Display the appropriate message as per the colour of signal at the road crossing.

```
signal = input("Enter the colour: ")
if signal == "red" or signal == "RED":
     print("STOP")
elif signal == "orange" or signal ==
"ORANGE":
```

```
        print("Be Slow")
elif signal == "green" or signal == "GREEN":
        print("Go!")
```

Number of `elif` is dependent on the number of conditions to be checked. If the first condition is false, then the next condition is checked, and so on. If one of the conditions is true, then the corresponding indented block executes, and the `if` statement terminates.

Let us write a program to create a simple calculator to perform basic arithmetic operations on two numbers. The program should do the following:

- Accept two numbers from the user.
- Ask user to input any of the operator (+, -, *, /). An error message is displayed if the user enters anything else.
- Display only positive difference in case of the operator "-".
- Display a message "Please enter a value other than 0" if the user enters the second number as 0 and operator '/' is entered.

Program 6-3 Write a program to create a simple calculator performing only four basic operations.

```
#Program to create a four function calculator
result = 0
val1 = float(input("Enter value 1: "))
val2 = float(input("Enter value 2: "))
op = input("Enter any one of the operator (+,-,*,/): ")
if op == "+":
    result = val1 + val2
elif op == "-":
    if val1 > val2:
        result = val1 - val2
    else:
        result = val2 - val1
elif op == "*":
        result = val1 * val2
elif op == "/":
    if val2 == 0:
        print("Error! Division by zero is not allowed. Program
terminated")
    else:
        result = val1/val2
else:
    print("Wrong input,program terminated")
print("The result is ",result)
```

Output:

```
Enter value 1: 84
Enter value 2: 4
Enter any one of the operator (+,-,*,/): /
The result is 21.0
```

In the program, for the operators "-" and "/", there exists an `if..else` condition within the `elif` block. This is called nested `if`. We can have many levels of nesting inside `if..else` statements.

6.3 INDENTATION

In most programming languages, the statements within a block are put inside curly brackets. However, Python uses indentation for block as well as for nested block structures. Leading whitespace (spaces and tabs) at the beginning of a statement is called indentation. In Python, the same level of indentation associates statements into a single block of code. The interpreter checks indentation levels very strictly and throws up syntax errors if indentation is not correct. It is a common practice to use a single tab for each level of indentation.

In the program 6-4, the `if-else` statement has two blocks of statements and the statements in each block are indented with the same amount of spaces or tabs.

Program 6-4 Program to find the larger of the two pre-specified numbers.

```
#Program 6-4
#Program to find larger of the two numbers
num1 = 5
num2 = 6
if num1 > num2:                          #Block1
        print("first number is larger")
        print("Bye")
else:                                    #Block2
        print("second number is larger")
        print("Bye Bye")
```

Output:

```
second number is larger
Bye Bye
```

6.4 Repetition

Often, we repeat a tasks, for example, payment of electricity bill, which is done every month. Figure 6.3 shows the life cycle of butterfly that involves four stages, i.e., a butterfly lays eggs, turns into a caterpillar, becomes a pupa, and finally matures as a butterfly. The cycle starts again with laying of eggs by the butterfly.

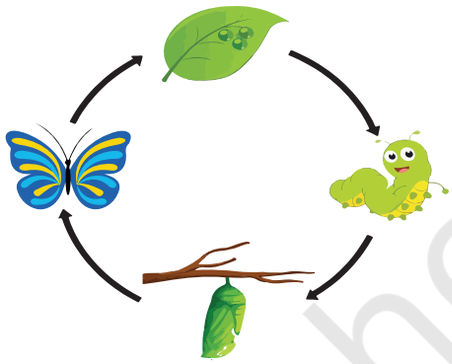


Figure 6.3: Iterative process occurring in nature

This kind of repetition is also called iteration. Repetition of a set of statements in a program is made possible using looping constructs. To understand further, let us look at the program 6-5.

Program 6-5 Write a program to print the first five natural numbers.

```
#Program 6-5
#Print first five natural numbers
print(1)
print(2)
print(3)
print(4)
print(5)
```

Output:

```
1
2
3
4
5
```

What should we do if we are asked to print the first 100,000 natural numbers? Writing 100,000 print statements would not be an efficient solution. It would be tedious and not the best way to do the task. Writing a program having a loop or repetition is a better solution. The program logic is given below:

1. Take a variable, say count, and set its value to 1.
2. Print the value of count.
3. Increment the variable (count += 1).

4. Repeat steps 2 and 3 as long as count has a value less than or equal to 100,000 (count <= 100,000).

Looping constructs provide the facility to execute a set of statements in a program repetitively, based on a condition. The statements in a loop are executed again and again as long as particular logical condition remains true. This condition is checked based on the value of a variable called the loop's control variable. When the condition becomes false, the loop terminates. It is the responsibility of the programmer to ensure that this condition eventually does become false so that there is an exit condition and it does not become an infinite loop. For example, if we did not set the condition count <= 100000, the program would have never stopped. There are two looping constructs in Python - `for` and `while`.

6.4.1 The 'For' Loop

The `for` statement is used to iterate over a range of values or a sequence. The `for` loop is executed for each of the items in the range. These values can be either numeric, or, as we shall see in later chapters, they can be elements of a data type like a string, list, or tuple.

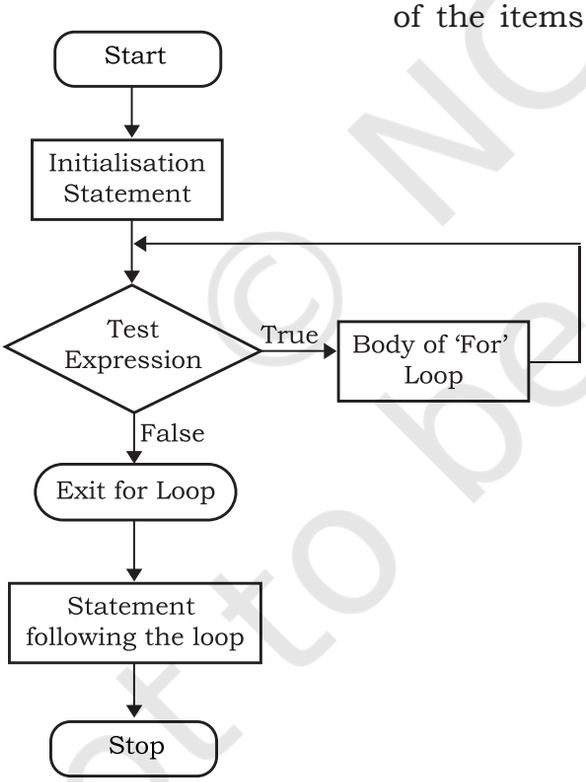


Figure 6.4: Flow chart of for loop

With every iteration of the loop, the control variable checks whether each of the values in the range have been traversed or not. When all the items in the range are exhausted, the statements within loop are not executed; the control is then transferred to the statement immediately following the `for` loop. While using `for` loop, it is known in advance the number of times the loop will execute. The flowchart depicting the execution of a `for` loop is given in Figure 6.4.

(A) Syntax of the For Loop

```
for <control-variable> in <sequence/
items in range>:
    <statements inside body of the
     loop>
```

Program 6-6 Program to print the characters in the string 'PYTHON' using `for` loop.

```
#Program 6-6
#Print the characters in word PYTHON using for loop
for letter in 'PYTHON':
    print(letter)
```

Output:

```
P
Y
T
H
O
N
```

Program 6-7 Program to print the numbers in a given sequence using `for` loop.

```
#Program 6-7
#Print the given sequence of numbers using for loop
count = [10,20,30,40,50]
for num in count:
    print(num)
```

Output:

```
10
20
30
40
50
```

Program 6-8 Program to print even numbers in a given sequence using `for` loop.

```
#Program 6-8
#Print even numbers in the given sequence
numbers = [1,2,3,4,5,6,7,8,9,10]
for num in numbers:
    if (num % 2) == 0:
        print(num,'is an even Number')
```

Output:

```
2 is an even Number
4 is an even Number
```

```
6  is an even Number
8  is an even Number
10 is an even Number
```

Note: Body of the loop is indented with respect to the `for` statement.

(B) The Range() Function

The range() is a built-in function in Python. Syntax of range() function is:

```
range([start], stop[, step])
```

It is used to create a list containing a sequence of integers from the given start value upto stop value (excluding stop value), with a difference of the given step value. We will learn about functions in the next chapter. To begin with, simply remember that function takes parameters to work on. In function range(), start, stop and step are parameters.

The start and step parameters are optional. If start value is not specified, by default the list starts from 0. If step is also not specified, by default the value increases by 1 in each iteration. All parameters of range() function must be integers. The step parameter can be a positive or a negative integer excluding zero.

Example 6.4

```
#start and step not specified
>>> list(range(10)-)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

#default step value is 1
>>> list(range(2, 10))
[2, 3, 4, 5, 6, 7, 8, 9]

#step value is 5
>>> list(range(0, 30, 5))
[0, 5, 10, 15, 20, 25]

#step value is -1. Hence, decreasing
#sequence is generated
>>> range(0, -9, -1)
[0, -1, -2, -3, -4, -5, -6, -7, -8]
```

The function range() is often used in for loops for generating a sequence of numbers.

Program 6-9 Program to print the multiples of 10 for numbers in a given range.

```
#Program 6-9
#Print multiples of 10 for numbers in a given range
for num in range(5):
    if num > 0:
        print(num * 10)
```

Output:

```
10
20
30
40
```

6.4.2 The 'While' Loop

The `while` statement executes a block of code repeatedly as long as the control condition of the loop is true. The control condition of the `while` loop is executed before any statement inside the loop is executed. After each iteration, the control condition is tested again and the loop continues as long as the condition remains true. When this condition becomes false, the statements in the body of loop are not executed and the control is transferred to the statement immediately following the body of `while` loop. If the condition of the `while` loop is initially false, the body is not executed even once.

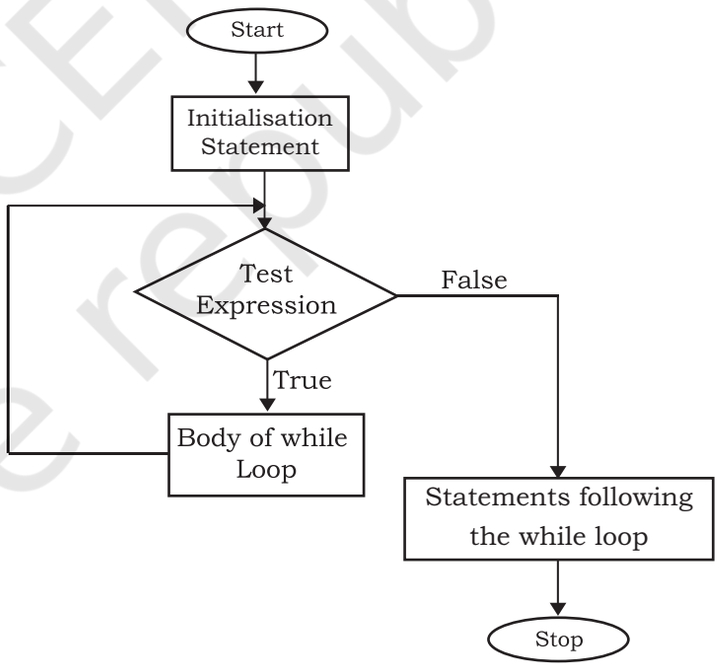


Figure 6.5: Flow chart of while Loop

The statements within the body of the `while` loop must ensure that the condition eventually becomes false; otherwise the loop will become an infinite loop, leading to a logical error in the program. The flowchart of `while` loop is shown in Figure 6.5.

Syntax of `while` Loop

```
while test_condition:
  body of while
```

Program 6-10 Program to print first 5 natural numbers using `while` loop.

```
#Program 6-10
#Print first 5 natural numbers using while loop
count = 1
while count <= 5:
     print(count)
     count += 1
```

Output:

```
1
2
3
4
5
```

Program 6-11 Program to find the factors of a whole number using `while` loop.

```
#Program 6-11
#Find the factors of a number using while loop
num = int(input("Enter a number to find its factor: "))
print (1, end=' ')   #1 is a factor of every number
factor = 2
while factor <= num/2 :
     if num % factor == 0:
#the optional parameter end of print function specifies the delimeter
#blank space(' ') to print next value on same line
          print(factor, end=' ')
     factor += 1
print (num, end=' ') #every number is a factor of itself
```

Output:

```
Enter a number to find its factors : 6
1 2 3 6
```

Note: Body of the loop is indented with respect to the `while` statement. Similarly, the statements within `if` are indented with respect to positioning of `if` statement.

6.5 BREAK AND CONTINUE STATEMENT

Looping constructs allow programmers to repeat tasks efficiently. In certain situations, when some particular condition occurs, we may want to exit from a loop (come

out of the loop forever) or skip some statements of the loop before continuing further in the loop. These requirements can be achieved by using `break` and `continue` statements, respectively. Python provides these statements as a tool to give more flexibility to the programmer to control the flow of execution of a program.

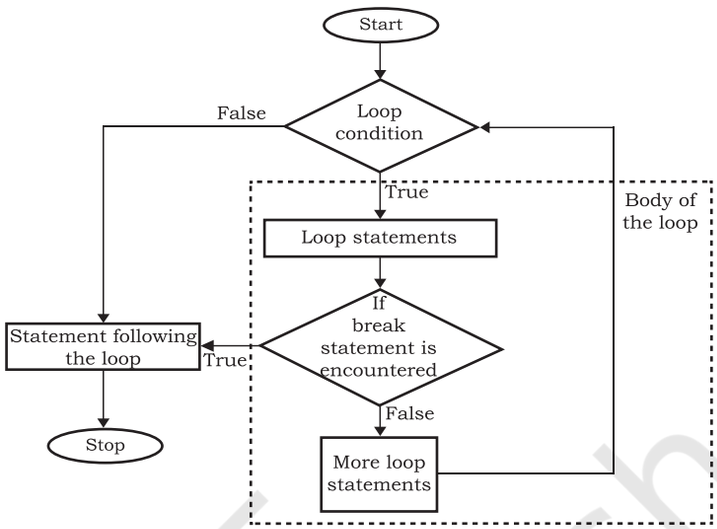


Figure 6.5: Flowchart for using break statement in loop

6.5.1 Break Statement

The `break` statement alters the normal flow of execution as it terminates the current loop and resumes execution of the statement following that loop.

Program 6-12 Program to demonstrate use of `break` statement.

```
#Program 6-12
#Program to demonstrate the use of break statement in loop
num = 0
for num in range(10):
     num = num + 1
     if num == 8:
          break
     print('Num has value ' + str(num))
print('Encountered break!! Out of loop')
```

Output:

```
Num has value 1
Num has value 2
Num has value 3
Num has value 4
Num has value 5
Num has value 6
Num has value 7
Encountered break!! Out of loop
```

Note: When value of `num` becomes 8, the `break` statement is executed and the `for` loop terminates.

Program 6-13 Find the sum of all the positive numbers entered by the user. As soon as the user enters a neagtive number, stop taking in any further input from the user and display the sum .

```
#Program 6-13
#Find the sum of all the positive numbers entered by the user
#till the user enters a negative number.
entry = 0
sum1 = 0
print("Enter numbers to find their sum, negative number ends the
loop:")
while True:
#int() typecasts string to integer
     entry = int(input())
     if (entry < 0):
          break
     sum1 += entry
print("Sum =", sum1)
```

Output:

```
Enter numbers to find their sum, negative number ends the loop:
3
4
5
-1
Sum = 12
```

Program 6-14 Program to check if the input number is prime or not.

```
#Program 6-14
#Write a Python program to check if a given number is prime or not.
num = int(input("Enter the number to be checked: "))
flag = 0                    #presume num is a prime number
if num > 1 :
     for i in range(2, int(num / 2)):
          if (num % i == 0):
               flag = 1      #num is a not prime number
               break         #no need to check any further

     if flag == 1:
          print(num , "is not a prime number")
     else:
          print(num , "is a prime number")
```

```
else :
        print("Entered number is <= 1, execute again!")
```

Output 1:

```
Enter the number to be checked: 20
20 is not a prime number
```

Output 2:

```
Enter the number to check: 19
19 is a prime number
```

Output 3:

```
Enter the number to check: 2
2 is a prime number
```

Output 4:

```
Enter the number to check: 1
Entered number is <= 1,  execute again!
```

6.5.2 Continue Statement

When a `continue` statement is encountered, the control skips the execution of remaining statements inside the body of the loop for the current iteration and jumps to the beginning of the loop for the next iteration. If the loop's condition is still true, the loop is entered again, else the control is transferred to the statement immediately following the loop. Figure 6.7 shows the flowchart of `continue` statement.

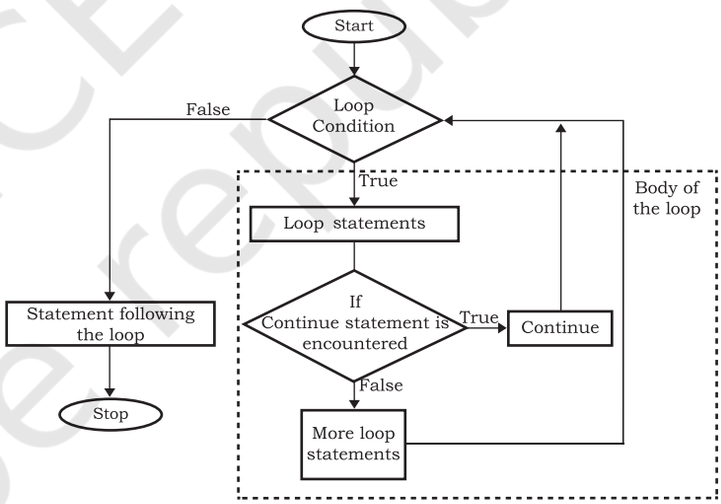


Figure 6.6: Flow chart of continue statement

Program 6-15 Program to demonstrate the use of continue statement.

```
#Program 6-15
#Prints values from 0 to 6 except 3
num = 0
for num in range(6):
    num = num + 1
    if num == 3:
```

```
            continue
        print('Num has value ' + str(num))
print('End of loop')
```

Output:

```
Num has value 1
Num has value 2
Num has value 4
Num has value 5
Num has value 6
End of loop
```

Observe that the value 3 is not printed in the output, but the loop continues after the `continue` statement to print other values till the `for` loop terminates.

6.6 NESTED LOOPS

A loop may contain another loop inside it. A loop inside another loop is called a nested loop.

Program 6-16 Program to demonstrate working of nested `for` loops.

```
#Program 6-16
#Demonstrate working of nested for loops
for var1 in range(3):
    print( "Iteration " + str(var1 + 1) + " of outer loop")
    for var2 in range(2):    #nested loop
        print(var2 + 1)
    print("Out of inner loop")
print("Out of outer loop")
```

Output:

```
Iteration 1 of outer loop
1
2
Out of inner loop
Iteration 2 of outer loop
1
2
Out of inner loop
Iteration 3 of outer loop
1
2
Out of inner loop
Out of outer loop
```

Python does not impose any restriction on how many loops can be nested inside a loop or on the levels of nesting. Any type of loop (`for/while`) may be nested within another loop (`for/while`).

Program 6-17 Program to print the pattern for a number input by the user.

```
#Program 6-17
#Program to print the pattern for a number input by the user
#The output pattern to be generated is
#1
#1 2
#1 2 3
#1 2 3 4
#1 2 3 4 5
num = int(input("Enter a number to generate its pattern = "))
for i in range(1,num + 1):
     for j in range(1,i + 1):
          print(j, end = " ")
     print()
```

Output:

```
Enter a number to generate its pattern = 5
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

Program 6-18 Program to find prime numbers between 2 to 50 using nested for loops.

```
#Program 6-18
#Use of nested loops to find the prime numbers between 2 to 50

num = 2
for i in range(2, 50):
     j= 2
     while ( j <= (i/2)):
          if (i % j == 0):      #factor found
               break            #break out of while loop
          j += 1
     if ( j > i/j) :           #no factor found
```

```
            print ( i, "is a prime number")
print ("Bye Bye!!")
```

Output:

```
2 is a prime number
3 is a prime number
5 is a prime number
7 is a prime number
11 is a prime number
13 is a prime number
17 is a prime number
19 is a prime number
23 is a prime number
29 is a prime number
31 is a prime number
37 is a prime number
41 is a prime number
43 is a prime number
47 is a prime number
Bye Bye!!
```

Program 6-19 Write a program to calculate the factorial of a given number.

```
#Program 6-19
#The following program uses a for loop nested inside an if..else
#block to calculate the factorial of a given number


num = int(input("Enter a number: "))
fact = 1
# check if the number is negative, positive or zero
if num < 0:
    print("Sorry, factorial does not exist for negative numbers")
elif num == 0:
    print("The factorial of 0 is 1")
else:
    for i in range(1, num + 1):
        fact = fact * i
    print("factorial of ", num, " is ", fact)
```

Output:

```
Enter a number: 5
Factorial of  5  is  120
```

NOTES

Summary

- The `if` statement is used for selection or decision making.
- The looping constructs `while` and `for` allow sections of code to be executed repeatedly under some condition.
- `for` statement iterates over a range of values or a sequence.
- The statements within the body of `for` loop are executed till the range of values is exhausted.
- The statements within the body of a `while` are executed over and over until the condition of the `while` is false.
- If the condition of the `while` loop is initially false, the body is not executed even once.
- The statements within the body of the `while` loop must ensure that the condition eventually becomes false; otherwise, the loop will become an infinite loop, leading to a logical error in the program.
- The `break` statement immediately exits a loop, skipping the rest of the loop's body. Execution continues with the statement immediately following the body of the loop. When a `continue` statement is encountered, the control jumps to the beginning of the loop for the next iteration.
- A loop contained within another loop is called a nested loop.

Exercise

1. What is the difference between `else` and `elif` construct of `if` statement?
2. What is the purpose of range() function? Give one example.
3. Differentiate between `break` and `continue` statements using examples.
4. What is an infinite loop? Give one example.
5. Find the output of the following program segments:

(i)
```
a = 110
while a > 100:
    print(a)
    a -= 2
```

NOTES

(ii)
```
for i in range(20,30,2):
    print(i)
```

(iii)
```
country = 'INDIA'
for  i in country:
    print (i)
```

(iv)
```
i = 0; sum = 0
while i < 9:
    if i % 4 == 0:
        sum = sum + i
    i = i + 2
print (sum)
```

(v)
```
for x in range(1,4):
    for y in range(2,5):
        if x * y > 10:
            break
        print (x * y)
```

(vi)
```
var = 7
while var > 0:
    print ('Current variable value: ', var)
    var = var -1
    if var == 3:
        break
    else:
        if var == 6:
            var = var -1
            continue
    print ("Good bye!")
```

PROGRAMMING EXERCISES

1. Write a program that takes the name and age of the user as input and displays a message whether the user is eligible to apply for a driving license or not. (the eligible age is 18 years).
2. Write a function to print the table of a given number. The number has to be entered by the user.
3. Write a program that prints minimum and maximum of five numbers entered by the user.
4. Write a program to check if the year entered by the user is a leap year or not.
5. Write a program to generate the sequence: –5, 10, –15, 20, –25..... upto `n`, where `n` is an integer input by the user.
6. Write a program to find the sum of 1+ 1/8 + 1/27.....$1/n^3$, where n is the number input by the user.

NOTES

7. Write a program to find the sum of digits of an integer number, input by the user.
8. Write a function that checks whether an input number is a palindrome or not.

   [***Note***: A number or a string is called palindrome if it appears same when written in reverse order also. For example, 12321 is a palindrome while 123421 is not a palindrome]
9. Write a program to print the following patterns:

i)	ii)
* * * * * * * * * * * * *	1 2 1 2 3 2 1 2 3 4 3 2 1 2 3 4 5 4 3 2 1 2 3 4 5
iii)	**iv)**
1 2 3 4 5 1 2 3 4 1 2 3 1 2 1	* * * * * * * *

10. Write a program to find the grade of a student when grades are allocated as given in the table below.

Percentage of Marks	Grade
Above 90%	A
80% to 90%	B
70% to 80%	C
60% to 70%	D
Below 60%	E

Percentage of the marks obtained by the student is input to the program.

CASE STUDY-BASED QUESTIONS

Let us add more functionality to our SMIS developed in Chapter 5.

6.1 Write a menu driven program that has options to

- accept the marks of the student in five major subjects in Class X and display the same.
- calculate the sum of the marks of all subjects. Divide the total marks by number of subjects (i.e. 5), calculate percentage = total marks/5 and display the percentage.

NOTES

- Find the grade of the student as per the following criteria:

Criteria	Grade
percentage > 85	A
percentage < 85 && percentage >= 75	B
percentage < 75 && percentage >= 50	C
percentage > 30 && percentage <= 50	D
percentage <30	Reappear

Let's peer review the case studies of others based on the parameters given under "DOCUMENTATION TIPS" at the end of Chapter 5 and provide a feedback to them.

Chapter 7

Functions

11120CH07

7.1 Introduction

"Once you succeed in writing the programs for [these] complicated algorithms, they usually run extremely fast. The computer doesn't need to understand the algorithm, its task is only to run the programs."

– R. Tarjan

Till now we have written some programs and might have realised that as the problem gets complex, the number of lines in a program increase, which makes the program look bulky and difficult to manage. Consider the following problem statement:

There is a company that manufactures tents as per user's requirements. The shape of the tent is cylindrical surmounted by a conical top.

Figure 7.1: Shape of a tent

The company performs the following tasks to fix the selling price of each tent.

1. Accept user requirements for the tent, such as
   a) height
   b) radius
   c) slant height of the conical part
2. Calculate the area of the canvas used
3. Calculate the cost of the canvas used for making the tent
4. Calculate the net payable amount by the customer that is inclusive of the 18% tax

The company has created a computer program for quick and accurate calculation for the payable amount as shown in program 7-1.

In this chapter

- *Introduction to Functions*
- *User Defined Functions*
- *Scope of a Variable*
- *Python Standard Library*

Program 7-1 Program to calculate the payable amount for the tent.

```
#Program 7-1
#Program to calculate the payable amount for the tent without
#functions

print( "Enter values for the cylindrical part of the tent in
meters\n")
h = float(input("Enter height of the cylindrical part: "))
r = float(input("Enter radius: "))

l = float(input("Enter the slant height of the conical part in
meters: "))
csa_conical = 3.14*r*l          #Area of conical part
csa_cylindrical = 2*3.14*r*h    #Area of cylindrical part

# Calculate area of the canvas used for making the tent
canvas_area = csa_conical + csa_cylindrical
print("The area of the canvas is",canvas_area,"m^2")

#Calculate cost of the canvas
unit_price = float(input("Enter the cost of 1 m^2 canvas: "))
total_cost= unit_price * canvas_area
print("The total cost of canvas = ",total_cost)

#Add tax to the total cost to calculate net amount payable by the
#customer
tax = 0.18 * total_cost;
net_price = total_cost + tax
print("Net amount payable = ",net_price)
```

Another approach to solve the above problem is to divide the program into different blocks of code as shown in Figure 7.2.

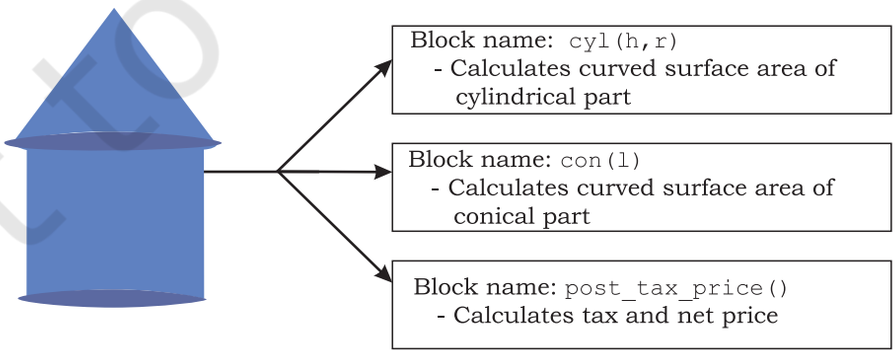


Figure 7.2: Calculation of the cost of the tent

The process of dividing a computer program into separate independent blocks of code or separate sub-problems with different names and specific functionalities is known as modular programming. In this chapter, we will learn about the benefits of this approach.

7.2 FUNCTIONS

In programming, the use of function is one of the means to achieve modularity and reusability. Function can be defined as a named group of instructions that accomplish a specific task when it is invoked. Once defined, a function can be called repeatedly from different places of the program without writing all the codes of that function everytime, or it can be called from inside another function, by simply writing the name of the function and passing the required parameters, if any (Section 7.3). The programmer can define as many functions as desired while writing the code. The program 7-1 is rewritten using user defined functions as shown in program 7-2.

Program 7-2 Program to calculate the payable amount for the tent using user defined functions.

```
#Program 7-2
#Program to calculate the cost of tent
#function definition
def cyl(h,r):
     area_cyl = 2*3.14*r*h     #Area of cylindrical part
     return(area_cyl)

#function definition
def con(l,r):
     area_con = 3.14*r*l       #Area of conical part
     return(area_con)

#function definition
def post_tax_price(cost):      #compute payable amount for the tent
     tax = 0.18 * cost;
     net_price = cost + tax
     return(net_price)

print("Enter values of cylindrical part of the tent in meters:")
h = float(input("Height: "))
r = float(input("Radius: "))
```

```
 csa_cyl = cyl(h,r)  #function call

l = float(input("Enter slant height of the conical area in meters: "))
csa_con = con(l,r)  #function call

#Calculate area of the canvas used for making the tent
canvas_area = csa_cyl + csa_con
print("Area of canvas = ",canvas_area," m^2")

#Calculate cost of canvas
unit_price = float(input("Enter cost of 1 m^2 canvas in rupees: "))
total_cost = unit_price * canvas_area
print("Total cost of canvas before tax = ",total_cost)
print("Net amount payable (including tax) = ",post_tax_price(total_
cost))
```

If we compare program 7-1 and 7-2, it is evident that program 7-2 looks more organised and easier to read.

7.2.1 The Advantages of Function

Suppose in further the company decides to design another type of tent whose base is rectangular, while the upper part remains the same. In such a scenario, some part of the existing code can be reused by calling the function `con(l,r)`. If the company develops other products or provides services, and where 18% tax rate is to be applied, the programmer can use the function `post_tax_price(cost)` directly.

Thus, following are the advantages of using functions in a program:

- Increases readability, particularly for longer code as by using functions, the program is better organised and easy to understand.
- Reduces code length as same code is not required to be written at multiple places in a program. This also makes debugging easier.
- Increases reusability, as function can be called from another function or another program. Thus, we can reuse or build upon already defined functions and avoid repetitions of writing the same piece of code.
- Work can be easily divided among team members and completed in parallel.

7.3 USER DEFINED FUNCTIONS

Taking advantage of reusability feature of functions, there is a large number of functions already available

in Python under standard library (section 7.5). We can directly call these functions in our program without defining them. However, in addition to the standard library functions, we can define our own functions while writing the program. Such functions are called user defined functions. Thus, a function defined to achieve some task as per the programmer's requirement is called a user defined function.

7.3.1 Creating User Defined Function

A function definition begins with `def` (short for define). The syntax for creating a user defined function is as follows:

```
def<Function name> ([parameter 1, parameter 2,....]):
    set of instructions to be executed
    [return <value>]
```

Function Header

Function Body (Should be indented within the function header)

- The items enclosed in "[ ]" are called parameters and they are optional. Hence, a function may or may not have parameters. Also, a function may or may not return a value.
- Function header always ends with a colon (:).
- Function name should be unique. Rules for naming identifiers also applies for function naming.
- The statements outside the function indentation are not considered as part of the function.

Program 7-3 Write a user defined function to add 2 numbers and display their sum.

```
#Program 7-3
#Function to add two numbers
#The requirements are listed below:
  #1. We need to accept 2 numbers from the user.
  #2. Calculate their sum
  #3. Display the sum.

#function definition
def addnum():
    fnum = int(input("Enter first number: "))
    snum = int(input("Enter second number: "))
    sum = fnum + snum
    print("The sum of ",fnum,"and ",snum,"is ",sum)

#function call
addnum()
```

In order to execute the function addnum(), we need to call it. The function can be called in the program by writing function name followed by () as shown in the last line of program 7-3.

Output:

```
Enter first number: 5
Enter second number: 6
The sum of 5 and 6 is 11
```

7.3.2 Arguments and Parameters

In the above example, the numbers were accepted from the user within the function itself, but it is also possible for a user defined function to receive values at the time of being called. An argument is a value passed to the function during the function call which is received in corresponding parameter defined in function header.

Program 7-4 Write a program using a user defined function that displays sum of first *n* natural numbers, where *n* is passed as an argument.

```
#Program 7-4
#Program to find the sum of first n natural numbers
#The requirements are:
  #1. n be passed as an argument
  #2. Calculate sum of first n natural numbers
  #3. Display the sum

#function header
def sumSquares(n):          #n is the parameter
     sum = 0
     for i in range(1,n+1):
          sum = sum + i
     print("The sum of first",n,"natural numbers is: ",sum)

num = int(input("Enter the value for n: "))
#num is an argument referring to the value input by the user
sumSquares(num)          #function call
```

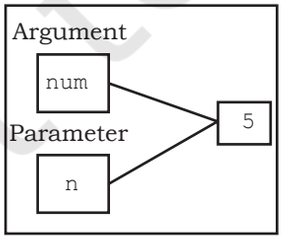


Figure 7.3: Both argument and parameter refers to the same value

Let us assume that the user has input 5 during the execution of the program 7-4. So, `num` refers to the value 5. It is then used as an argument in the function:

```
sumSquares(num)
```

Since the function is called, the control is transferred to execute the function

```
def sumSquares(n):
```

where parameter n also refers to the value 5 which num is referring to as shown in Figure 7.3.

Since both `num` and `n` are referring to the same value, they are bound to have the same identity. We can use the id() function to find the identity of the object that the argument and parameter are referring to. Let us understand this with the help of the following example.

Program 7-5 Write a program using user defined function that accepts an integer and increments the value by 5. Also display the `id` of argument (before function call), `id` of parameter before increment and after increment.

```
#Program 7-5
#Function to add 5 to a user input number
#The requirements are listed below:
  #1. Display the id()of argument before function call.
  #2. The function should have one parameter to accept the argument
  #3. Display the value and id() of the parameter.
  #4. Add 5 to the parameter
  #5. Display the new value and id()of the parameter to check
      #whether the parameter is assigned a new memory location or
      #not.
def incrValue(num):
    #id of Num before increment
    print("Parameter num has value:",num,"\nid =",id(num))
    num = num + 5
    #id of Num after increment
    print("num incremented by 5 is",num,"\nNow id is ",id(num))
number = int(input("Enter a number: "))
print("id of argument number is:",id(number))          #id of Number
incrValue(number)
```

Output:

```
Enter a number: 8
id of argument number is: 1712903328
Parameter num has value: 8
id = 1712903328
num incremented by 5 is 13
Now id is 1712903408
```

number and num have the same id

The id of Num has changed.

Let us understand the above output through illustration (see Figure 7.4):

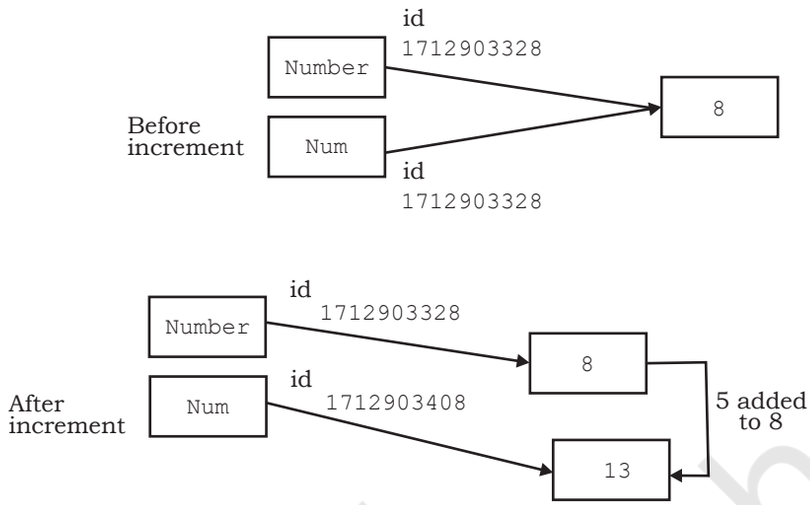


Figure 7.4: ID of argument and parameter before and after increment

Both argument and parameter can have the same name as shown in program 7-6.

Program 7-6 Write a program using a user defined function myMean() to calculate the mean of floating values stored in a list.

```
#Program 7-6
#Function to calculate mean
#The requirements are listed below:
  #1. The function should have 1 parameter (list containing floating
      #point values)
  #2. To calculate mean by adding all the numbers and dividing by
      #total number of elements

def myMean(myList): #function to compute means of values in list
     total = 0
     count = 0
     for i in myList:
          total = total + i          #Adds each element i to total
          count = count + 1          #Counts the number of elements
     mean = total/count              #mean is calculated
     print("The calculated mean is:",mean)
myList = [1.3,2.4,3.5,6.9]
#Function call with list "myList" as an argument
myMean(myList)
```

Output:

```
The calculated mean is: 3.5250000000000004
```

Program 7-7 Write a program using a user defined function calcFact() to calculate and display the factorial of a number num passed as an argument.

```
#Program 7-7
#Function to calculate factorial
#The requirements are listed below:
  #1. The function should accept one integer argument from user.
  #2. Calculate factorial. For example:
  #3. Display factorial

def calcFact(num):
     fact = 1
     for i in range(num,0,-1):
          fact = fact * i
     print("Factorial of",num,"is",fact)

num = int(input("Enter the number: "))
calcFact(num)
```

Output:

```
Enter the number: 5
Factorial of 5 is 120
```

Note: Since multiplication is commutative 5! = 5*4*3*2*1 = 1*2*3*4*5

(A) String as Parameters

In programs 7-5 to 7-7, the arguments passed are of numeric type only. However, in some programs, user may need to pass string values as an argument, as shown in program 7-8.

Program 7-8 Write a program using a user defined function that accepts the first name and lastname as arguments, concatenate them to get full name and displays the output as:

```
Hello full name
```

For example, if first name is Gyan and lastname is Vardhan, the output should be:

```
Hello Gyan Vardhan

#Program 7-8
#Function to display full name
#The requirements are listed below:
   #1. The function should have 2 parameters to accept first name and
       #last name.
   #2. Concatenate names using + operator with a space between first
       #name and last name.
   #3. Display full name.
```

```
def fullname(first,last):
#+ operator is used to concatenate strings
     fullname = first + " " + last
     print("Hello",fullname)
#function ends here
first = input("Enter first name: ")
last = input("Enter last name: ")
#function call
fullname(first,last)
```

Output:

```
Enter first name: Gyan
Enter last name: Vardhan
Hello Gyan Vardhan
```

(B) Default Parameter

Python allows assigning a default value to the parameter. A default value is a value that is predecided and assigned to the parameter when the function call does not have its corresponding argument.

Program 7-9 Write a program that accepts numerator and denominator of a fractional number and calls a user defined function mixedFraction() when the fraction formed is not a proper fraction. The default value of denominator is 1. The function displays a mixed fraction only if the fraction formed by the parameters does not evaluate to a whole number.

```
#Program 7-9
#Function to display mixed fraction for an improper fraction
#The requirements are listed below:
  #1. Input numerator and denominator from the user.
  #2. Check if the entered numerator and denominator form a proper
      #fraction.
  #3. If they do not form a proper fraction, then call
      #mixedFraction().
  #4. mixedFraction()display a mixed fraction only when the fraction
      #does not evaluate to a whole number.

def mixedFraction(num,deno = 1):
     remainder = num % deno
#check if the fraction does not evaluate to a whole number
     if remainder!= 0:
          quotient = int(num/deno)
          print("The mixed fraction=", quotient,"(",remainder, "/",
deno,")")
     else:
```

```
            print("The given fraction evaluates to a whole number")
#function ends here
num = int(input("Enter the numerator: "))
deno = int(input("Enter the denominator: "))
print("You entered:",num,"/",deno)
if num > deno:    #condition to check whether the fraction is
improper
     mixedFraction(num,deno)      #function call
else:
     print("It is a proper fraction")
```

Output:

```
Enter the numerator: 17
Enter the denominator: 2
You entered: 17 / 2
The mixed fraction =  8 ( 1 / 2 )
```

In the above program, the denominator entered is 2, which is passed to the parameter "deno" so the default value of the argument deno is overwritten.

Let us consider the following function call:

```
mixedFraction(9)
```

Here, num will be assigned 9 and deno will use the default value 1.

***Note*:**

- A function argument can also be an expression, such as

```
mixedFraction(num+5,deno+5)
```

In such a case, the argument is evaluated before calling the function so that a valid value can be assigned to the parameter.

- The parameters should be in the same order as that of the arguments.

The default parameters must be the trailing parameters in the function header that means if any parameter is having default value then all the other parameters to its right must also have default values. For example,

```
def mixedFraction(num,deno = 1)
def mixedFraction(num = 2,deno = 1)
```

Let us consider few more function definition headers:

```
#incorrect as default must be the last
#parameter
def calcInterest(principal = 1000, rate,
time = 5):
#correct
def calcInterest(rate,principal = 1000,
time = 5):
```

7.3.3 Functions Returning Value

A function may or may not return a value when called. The return statement returns the values from the function. In the examples given so far, the function performs calculations and display result(s).They do not return any value. Such functions are called void functions. But a situation may arise, wherein we need to send value(s) from the function to its calling function. This is done using `return` statement.

The `return` statement does the following:

- returns the control to the calling function.
- return value(s) or `None`.

Program 7-10 Write a program using user defined function calcPow() that accepts base and exponent as arguments and returns the value $Base^{exponent}$ where Base and exponent are integers.

```
#Program 7-10
#Function to calculate and display base raised to the power exponent
#The requirements are listed below:
   #1. Base and exponent are to be accepted as arguments.
   #2. Calculate Base^exponent
   #3. Return the result (use return statement )
   #4. Display the returned value.
def calcpow(number,power):          #function definition
     result = 1
     for i in range(1,power+1):
          result = result * number
     return result

base = int(input("Enter the value for the Base: "))
expo = int(input("Enter the value for the Exponent: "))
answer = calcpow(base,expo)         #function call
print(base,"raised to the power",expo,"is",answer)
```

Output:

```
Enter the value for the Base: 5
Enter the value for the Exponent: 4
5 raised to the power 4 is 625
```

So far we have learnt that a function may or may not have parameter(s) and a function may or may not return any value(s). In Python, as per our requirements, we can have the function in either of the following ways:

- Function with no argument and no return value
- Function with no argument and with return value(s)
- Function with argument(s) and no return value

- Function with argument(s) and return value(s)

7.3.4 Flow of Execution

Flow of execution can be defined as the order in which the statements in a program are executed. The Python interpreter starts executing the instructions in a program from the first statement. The statements are executed one by one, in the order of appearance from top to bottom.

When the interpreter encounters a function definition, the statements inside the function are not executed until the function is called. Later, when the interpreter encounters a function call, there is a little deviation in the flow of execution. In that case, instead of going to the next statement, the control jumps to the called function and executes the statement of that function. After that, the control comes back the point of function call so that the remaining statements in the program can be executed. Therefore, when we read a program, we should not simply read from top to bottom. Instead, we should follow the flow of control or execution. It is also important to note that a function must be defined before its call within a program.

Program 7-11 Program to understand the low of execution using functions.

```
#Program 7-11
#print using functions
helloPython()                          #Function Call

def helloPython():                     #Function definition
     print("I love Programming")
```

On executing the above code the following error is produced:

```
Traceback (most recent call last):
  File "C:\NCERT\Prog 7-11.py", line 3, in <module>
    helloPython()                          #Function Call
NameError: name 'helloPython' is not defined
```

The error 'function not defined' is produced even though the function has been defined. When a function call is encountered, the control has to jump to the function definition and execute it. In the above program, since the function call precedes the function definition, the interpreter does not find the function definition and hence an error is raised.

That is why, the function definition should be made before the function call as shown below:

```
def helloPython(): #Function definition
    print("I love Programming")

helloPython()      #Function Call
```

```
[2]     def Greetings(Name):          #Function Header
[3]         print("Hello "+Name)

[1]     Greetings("John")             #Function Call
[4]     print("Thanks")
```

```
[4]     def RectangleArea(l,b):       #Function Header
[5]     return l*b

[1]     l = input("Length: ")
[2]     b = input("Breadth: ")
[3][6]  Area = RectangleArea(l,b)     #Function Call
[7]     print(Area)
[8]     print("thanks")
```

Figure 7.5: Order of execution of statements

Figure 7.5 explains the flow of execution for two programs. The number in square brackets shows the order of execution of the statements.

Sometime, a function needs to return multiple values which may be returned using tuple. Program 7-12 shows a function which returns two values area and perimeter of rectangle using tuple.

Program 7-12 Write a program using user defined function that accepts length and breadth of a rectangle and returns the area and perimeter of the rectangle.

```
#Program 7-12
#Function to calculate area and perimeter of a rectangle
#The requirements are listed below:
  #1. The function should accept 2 parameters.
  #2. Calculate area and perimeter.
  #3. Return area and perimeter.

def calcAreaPeri(Length,Breadth):
     area = length * breadth
     perimeter = 2 * (length + breadth)
     #a tuple is returned consisting of 2 values area and perimeter
     return (area,perimeter)

l = float(input("Enter length of the rectangle: "))
b = float(input("Enter breadth of the rectangle: "))
#value of tuples assigned in order they are returned
area,perimeter = calcAreaPeri(l,b)
print("Area is:",area,"\nPerimeter is:",perimeter)
```

Output:

```
Enter Length of the rectangle: 45
Enter Breadth of the rectangle: 66
Area is: 2970.0
Perimeter is: 222.0
```

Multiple values in Python are returned through a tuple. (Ch. 10)

Program 7-13 Write a program that simulates a traffic light . The program should consist of the following:

1. A user defined function trafficLight( ) that accepts input from the user, displays an error message if the user enters anything other than RED, YELLOW, and GREEN. Function light() is called and following is displayed depending upon return value from light().
   a) "STOP, your life is precious" if the value returned by light() is 0.
   b) "Please WAIT, till the light is Green " if the value returned by light() is 1
   c) "GO! Thank you for being patient" if the value returned by light() is 2.
2. A user defined function light() that accepts a string as input and returns 0 when the input is RED, 1 when the input is YELLOW and 2 when the input is GREEN. The input should be passed as an argument.
3. Display " SPEED THRILLS BUT KILLS" after the function trafficLight( ) is executed.

```
#Program 7-13
#Function to simulate a traffic light
#It is required to make 2 user defined functions trafficLight() and
#light().

def trafficLight():
     signal = input("Enter the colour of the traffic light: ")
     if (signal not in ("RED","YELLOW","GREEN")):
          print("Please enter a valid Traffic Light colour in
CAPITALS")
     else:
          value = light(signal)           #function call to light()
          if (value == 0):
               print("STOP, Your Life is Precious.")
          elif (value == 1):
               print ("PLEASE GO SLOW.")
          else:
```

```
                print("GO!,Thank you for being patient.")
#function ends here

def light(colour):
     if (colour == "RED"):
          return(0);
     elif (colour == "YELLOW"):
          return (1)
     else:
          return(2)
#function ends here

trafficLight()
print("SPEED THRILLS BUT KILLS")
```

Output:

```
Enter the colour of the traffic light: YELLOW
PLEASE GO SLOW.
SPEED THRILLS BUT KILLS
```

7.4 SCOPE OF A VARIABLE

A variable defined inside a function cannot be accessed outside it. Every variable has a well-defined accessibility. The part of the program where a variable is accessible can be defined as the scope of that variable. A variable can have one of the following two scopes:

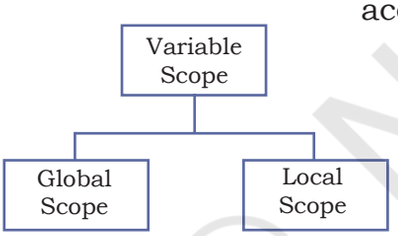


Figure 7.6: Scope of a variable

A variable that has global scope is known as a global variable and a variable that has a local scope is known as a local variable.

(A) Global Variable

In Python, a variable that is defined outside any function or any block is known as a global variable. It can be accessed in any functions defined onwards. Any change made to the global variable will impact all the functions in the program where that variable can be accessed.

(B) Local Variable

A variable that is defined inside any function or a block is known as a local variable. It can be accessed only in the function or a block where it is defined. It exists only till the function executes.

Program 7-14 Program to access any variable outside the function

```
#Program 7-14
#To access any variable outside the function
num = 5
def myFunc1( ):
    y = num + 5
    print("Accessing num -> (global) in myFunc1, value = ",num)
    print("Accessing y-> (local variable of myFunc1) accessible, value=",y)

myFunc1()
print("Accessing num outside myFunc1 ",num)
print("Accessing y outside myFunc1 ",y)
```

Output:

```
Accessing num -> (global) in myFunc1, value =  5
Accessing y-> (local variable of myFunc1) accessible, value =  10
Accessing num outside myFunc1  5
Traceback (most recent call last):
  File "C:\NCERT\Prog 7-14.py", line 9, in <module>
    print("Accessing y outside myFunc1 ",y)
NameError: name 'y' is not defined
```

y generates error when it is accessed outside myfunc1()

→ Global variable output, → Local variable output

Note:

- Any modification to global variable is permanent and affects all the functions where it is used.
- If a variable with the same name as the global variable is defined inside a function, then it is considered local to that function and hides the global variable.
- If the modified value of a global variable is to be used outside the function, then the keyword `global` should be prefixed to the variable name in the function.

Program 7-15 Write a program to access any variable outside the function.

```
#Program 7-15
#To access any variable outside the function

num = 5
def myfunc1():
     #Prefixing global informs Python to use the updated global
     #variable num outside the function
     global num
     print("Accessing num =",num)
     num = 10
     print("num reassigned =",num)
#function ends here

myfunc1()
print("Accessing num outside myfunc1",num)
```

Output:

```
Accessing num = 5
num reassigned = 10
Accessing num outside myfunc1 10
```

Global variable num is accessed as the ambiguity is resolved by prefixing global to it

7.5 PYTHON STANDARD LIBRARY

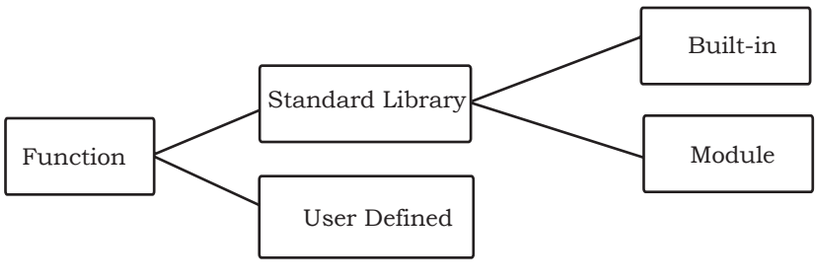


Figure 7.7: Types of functions

Python has a very extensive standard library. It is a collection of many built in functions that can be called in the program as and when required, thus saving programmer's time of creating those commonly used functions everytime.

7.5.1 Built-in functions

Built-in functions are the ready-made functions in Python that are frequently used in programs. Let us inspect the following Python program:

```
#Program to calculate square of a number
a = int(input("Enter a number: ")
b = a * a
print(" The square of ",a ,"is", b)
```

In the above program `input()`, `int()` and `print()` are the built-in functions. The set of instructions to be executed for these built-in functions are already defined in the python interpreter.

Let us consider the following Python statement consisting of a function call to a built in function and answer the given questions:

```
fname = input("Enter your name: ")
```

What is the name of the function being used?

- `input()`

Does the function accept a value or argument?

- Yes, because the parenthesis "()" consists of a string "Enter your name".

Does the function return a value?

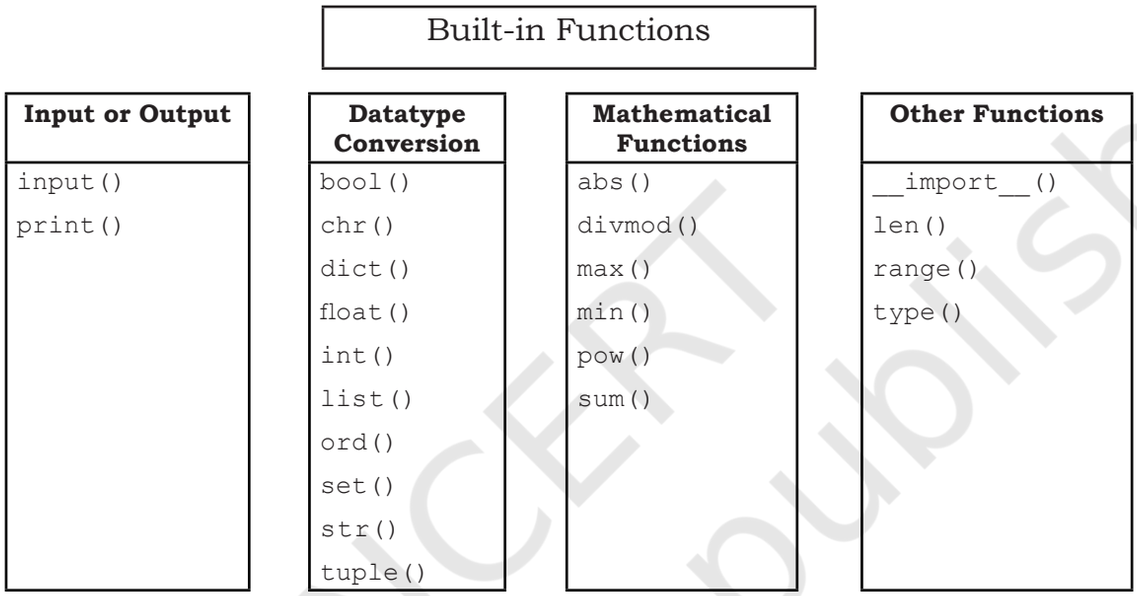
- Yes, since there is an assignment (=) operator preceding the function name, it means that the function returns a value which is stored in the variable fname.

Hence, the function input() accepts a value and returns a value.

Now consider the built-in functions int() and print(), and answer the questions below:

- Does the function accept a value or argument?
- Does the function return a value?

Following is a categorised list of some of the frequently used built-in functions in Python:



We have already used some of the built-in functions. Let us get familiar with some of them as explained in Table 7.1.

Table 7.1 Commonly used built-in functions

Function Syntax	Arguments	Returns	Example Output

abs(x)	x may be an integer or floating point number	Absolute value of x	`>>> abs(4)`
			`4`
			`>>> abs(-5.7)`
			`5.7`

divmod(x,y)	x and y are integers	A tuple: (quotient, remainder)	`>>> divmod(7,2)`
			`(3, 1)`
			`>>> divmod(7.5,2)`
			`(3.0, 1.5)`
			`>>> divmod(-7,2)`
			`(-4, 1)`

max(sequence) or max(x,y,z,...)	x,y,z,.. may be integer or floating point number	Largest number in the sequence/ largest of two or more arguments	`>>> max([1,2,3,4])`
			`4`
			`>>> max("Sincerity")`
			`'y' #Based on ASCII value`

			>>> max(23,4,56) 56

min(sequence)
or
min(x,y,z,...)
	x, y, z,.. may be integer or floating point number	Smallest number in the sequence/ smallest of two or more arguments	>>> min([1,2,3,4])
			1
			>>> min("Sincerity")
			'S'
			#Uppercase letters have lower ASCII values than lowercase letters.
			>>> min(23,4,56)
			4

pow(x,y[,z])	x, y, z may be integer or floating point number	x^y (x raised to the power y)
		if z is provided, then: (x^y) % z
			>>> pow(5,2)
			25.0
			>>> pow(5.3,2.2)
			39.2
			>>> pow(5,2,4)
			1

sum(x[,num])	x is a numeric sequence and num is an optional argument	Sum of all the elements in the sequence from left to right.
		if given parameter, num is added to the sum
			>>> sum([2,4,7,3])
			16
			>>> sum([2,4,7,3],3)
			19
			>>> sum((52,8,4,2))
			66

len(x)	x can be a sequence or a dictionary	Count of elements in x	>>> len("Patience")
			8
			>>> len([12,34,98])
			3
			>>> len((9,45))
			2 >>>len({1:"Anuj",2:"Razia",
			3:"Gurpreet",4:"Sandra"})
			4

7.5.2 Module

Other than the built-in functions, the Python standard library also consists of a number of modules. While a function is a grouping of instructions, a module is a grouping of functions. As we know that when a program grows, function is used to simplify the code and to avoid repetition. For a complex problem, it may not be feasible to manage the code in one single file. Then, the program is divided into different parts under different levels, called modules. Also, suppose we have created some functions in a program and we want to reuse them in another program. In that case, we can save those functions under a module and reuse them. A module is created as a python (.py) file containing a collection of function definitions.

To use a module, we need to import the module. Once we import a module, we can directly use all the functions of that module. The syntax of import statement is as follows:

```
import modulename1 [,modulename2, …]
```

This gives us access to all the functions in the module(s). To call a function of a module, the function name should be preceded with the name of the module with a dot(.) as a separator.
The syntax is as shown below:

```
modulename.functionname()
```

(A) Built-in Modules

Python library has many built-in modules that are really handy to programmers. Let us explore some commonly used modules and the frequently used functions that are found in those modules:

- math
- random
- statistics

Remember, Python is case sensitive. All the module names are in lowercase.

1. Module name : math

It contains different types of mathematical functions. Most of the functions in this module return a float value. Some of the commonly used functions in `math` module are given in Table 7.2. In order to use the math module we need to import it using the following statement:

```
import math
```

Table 7.2 Commonly used functions in math **module**

Function Syntax	Arguments	Returns	Example Output

math.ceil(x)	x may be an integer or floating point number	ceiling value of x	`>>> math.ceil(-9.7)`
			`-9`
			`>>> math.ceil (9.7)`
			`10`
			`>>> math.ceil(9)`
			`9`

math.floor(x)	x may be an integer or floating point number	floor value of x	`>>> math.floor(-4.5)`
			`-5`
			`>>> math.floor(4.5)`
			`4`
			`>>> math.floor(4)`
			`4`

math.fabs(x)	x may be an integer or floating point number	absolute value of x	`>>> math.fabs(6.7)`
			`6.7`
			`>>> math.fabs(-6.7)`
			`6.7`
			`>>> math.fabs(-4)`
			`4.0`

math.factorial(x)	x is a positive integer	factorial of x	`>>> math.factorial(5)` `120`

math.fmod(x,y)	x and y may be an integer or floating point number	x % y with sign of x	`>>> math.fmod(4,4.9)`
			`4.0`
			`>>> math.fmod(4.9,4.9)`
			`0.0`
			`>>> math.fmod(-4.9,2.5)`
			`-2.4`
			`>>> math.fmod(4.9,-4.9)`
			`0.0`

math.gcd(x,y)	x, y are positive integers	gcd (greatest common divisor) of x and y	`>>> math.gcd(10,2)` `2`

math.pow(x,y)	x, y may be an integer or floating point number	x^y (x raised to the power y)	`>>> math.pow(3,2)`
			`9.0`
			`>>> math.pow(4,2.5)`
			`32.0`
			`>>> math.pow(6.5,2)`
			`42.25`
			`>>> math.pow(5.5,3.2)`
			`233.97`

math.sqrt(x)	x may be a positive integer or floating point number	square root of x	`>>> math.sqrt(144)`
			`12.0`
			`>>> math.sqrt(.64)`
			`0.8`

math.sin(x)	x may be an integer or floating point number in radians	sine of x in radians	`>>> math.sin(0)`
			`0`
			`>>> math.sin(6)`
			`-0.279`

2. Module name : random

This module contains functions that are used for generating random numbers. Some of the commonly used functions in `random` module are given in Table 7.3. For using this module, we can import it using the following statement:

```
import random
```

Table 7.3 Commonly used functions in random module

Function Syntax	Argument	Return	Example Output
random.random()	No argument (void)	Random Real Number (float) in the range 0.0 to 1.0	`>>> random.random()` `0.65333522`

random.randint(x,y)	x, y are integers such that x <= y	Random integer between x and y	>>> random.randint(3,7)
			4
			>>> random.randint(-3,5)
			1
			>>> random.randint(-5,-3)
			-5.0

random.randrange(y)	y is a positive integer signifying the stop value	Random integer between 0 and y	>>> random.randrange(5) 4
random.randrange(x,y)	x and y are positive integers signifying the start and stop value	Random integer between x and y	>>> random.randrange(2,7) 2

3. Module name : statistics

This module provides functions for calculating statistics of numeric (Real-valued) data. Some of the commonly used functions in `statistics` module are given in Table 7.4. It can be included in the program by using the following statements:

```
import statistics
```

Table 7.4 Some of the function available through statistics module

Function Syntax	Argument	Return	Example Output
statistics.mean(x)	x is a numeric sequence	arithmetic mean	>>> statistics.mean([11,24,32,45,51]) 32.6
statistics.median(x)	x is a numeric sequence	median (middle value) of x	>>>statistics.median([11,24,32,45,51]) 32

statistics.mode(x)	x is a sequence	mode (the most repeated value)	>>> statistics.mode([11,24,11,45,11])
			11
			>>> statistics.mode(("red","blue","red"))
			'red'

Note:

- `import` statement can be written anywhere in the program
- Module must be imported only once
- In order to get a list of modules available in Python, we can use the following statement:
  `>>> help("module")`
- To view the content of a module say `math`, type the following:
  `>>> help("math")`

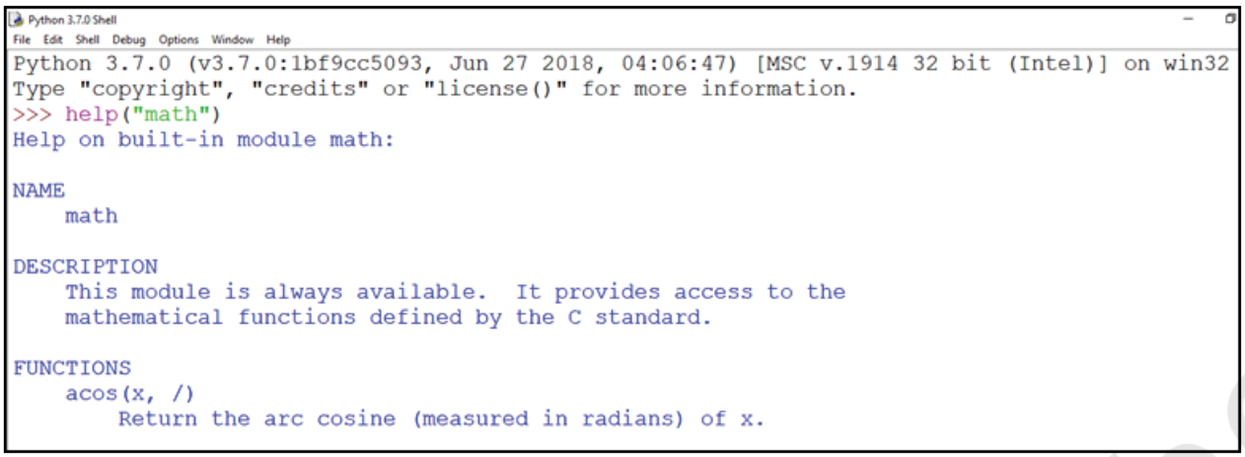


Figure 7.8: Content of module "math"

- The modules in the standard library can be found in the Lib folder of Python.

(B) From Statement

Instead of loading all the functions into memory by importing a module, `from` statement can be used to access only the required functions from a module. It loads only the specified function(s) instead of all the functions in a module.

Its syntax is

```
>>> from modulename import functionname [,
functionname,...]
```

To use the function when imported using "from statement" we do not need to precede it with the module name. Rather we can directly call the function as shown in the following examples:

Example 7.5

```
>>> from random import random
>>> random()       #Function called without
the module name
```

Output:

```
0.9796352504608387
```

Good Programming Practice: Only using the required function(s) rather than importing a module saves memory.

Example 7.6

```
>>> from math import ceil,sqrt
>>> value = ceil(624.7)
>>> sqrt(value)
```

Output:

```
25.0
```

In example 7.2, the ceil value of 624.7 is stored in the variable "value" and then `sqrt` function is applied on the

variable "value". The above example can be rewritten as:

```
>>> sqrt(ceil(624.7))
```

The execution of the function `sqrt()` is dependent on the output of `ceil()` function.

If we want to extract the integer part of 624.7 (we will use `trunc()` function from math module), we can use the following statements.

```
#ceil and sqrt already been imported above
>>> from math import trunc
>>> sqrt(trunc(625.7))
```

Output:

```
25.0
```

A programming statement wherein the functions or expressions are dependent on each other's execution for achieving an output is termed as composition, here are some other examples of composition:

- `a = int(input("First number: "))`
- `print("Square root of ",a ," = ",math.sqrt(a))`
- `print(floor(a+(b/c)))`
- `math.sin(float(h)/float(c))`

Besides the available modules in Python standard library, we can also create our own module consisting of our own functions.

Program 7-16 Create a user defined module `basic_math` that contains the following user defined functions:

1. To add two numbers and return their sum.
2. To subtract two numbers and return their difference.
3. To multiply two numbers and return their product.
4. To divide two numbers and return their quotient and print "Division by Zero" error if the denominator is zero.
5. Also add a docstring to describe the module. After creating module, import and execute functions.

"""Docstrings""" is also called Python documentation strings. It is a multiline comment that is added to describe the modules, functions, etc. They are typically added as the first line, using 3 double quotes.

```
#Program 7-16
#The requirement is:
 #1. Write a docstring describing the module.
 #2. Write user defined functions as per the specification.
 #3. Save the file.
 #4. Import at shell prompt and execute the functions.
```

```
"""
        basic_math Module
       ********************
This module contains basic arithmetic operations
that can be carried out on numbers

"""
#Beginning of module
def addnum(x,y):
    return(x + y)
def subnum(x,y):
    return(x - y)
def multnum(x,y):
    return(x * y)
def divnum(x,y):
    if y == 0:
        print ("Division by Zero Error")
    else:
        return (x/y)          #End of module
```

Output:

```
#Statements for using module basic_math
>>> import basic_math
#Display descriptions of the said module
>>> print(basic_math.__doc__)

        basic_math Module
       ********************
This module contains basic arithmetic operations
that can be carried out on numbers

>>> a = basic_math.addnum(2,5)  #Call addnum() function of the
>>> a                           #basic_math module
7
>>> a = basic_math.subnum(2,5)  #Call subnum() function of the
>>> a                           #basic_ math module
-3
>>> a = basic_math.multnum(2,5) #Call multnum() function of the
>>> a                           #basic_math module
10
>>> a = basic_math.divnum(2,5)  #Call divnum() function of the
>>> a                           #basic_math module
0.4
>>> a = basic_math.divnum(2,0)  #Call divnum() function of the
Zero Divide Error               #basic_math module
```

`__doc__` variable stores the docstring. To display docstring of a module we need to import the module and type the following:

```
print(<modulename>.__doc__)          #__ are 2 underscore without space
```

NOTES

SUMMARY

- In programming, functions are used to achieve modularity and reusability.
- Function can be defined as a named group of instructions that are executed when the function is invoked or called by its name. Programmers can write their own functions known as user defined functions.
- The Python interpreter has a number of functions built into it. These are the functions that are frequently used in a Python program. Such functions are known as built-in functions.
- An argument is a value passed to the function during function call which is received in a parameter defined in function header.
- Python allows assigning a default value to the parameter.
- A function returns value(s) to the calling function using return statement.
- Multiple values in Python are returned through a Tuple.
- Flow of execution can be defined as the order in which the statements in a program are executed.
- The part of the program where a variable is accessible is defined as the scope of the variable.
- A variable that is defined outside any particular function or block is known as a global variable. It can be accessed anywhere in the program.
- A variable that is defined inside any function or block is known as a local variable. It can be accessed only in the function or block where it is defined. It exists only till the function executes or remains active.
- The Python standard library is an extensive collection of functions and modules that help the programmer in the faster development of programs.
- A module is a Python file that contains definitions of multiple functions.
- A module can be imported in a program using import statement.
- Irrespective of the number of times a module is imported, it is loaded only once.
- To import specific functions in a program from a module, `from` statement can be used.

NOTES

EXERCISE

1. Observe the following programs carefully, and identify the error:

a)
```
def create (text, freq):
        for i in range (1, freq):
              print text
create(5)           #function call
```

b)
```
from math import sqrt,ceil
def calc():
        print cos(0)
calc()              #function call
```

c)
```
mynum = 9
def add9():
        mynum = mynum + 9
        print mynum
add9()               #function call
```

d)
```
def findValue( val1 = 1.1, val2, val3):
        final = (val2 + val3)/ val1
        print(final)
findvalue()          #function call
```

e)
```
def greet():
        return("Good morning")
greet() = message  #function call
```

2. How is math.ceil(89.7) different from math.floor (89.7)?
3. Out of random() and randint(), which function should we use to generate random numbers between 1 and 5. Justify.
4. How is built-in function pow() function different from function math.pow() ? Explain with an example.
5. Using an example show how a function in Python can return multiple values.
6. Differentiate between following with the help of an example:
   a) Argument and Parameter
   b) Global and Local variable
7. Does a function always return a value? Explain with an example.

NOTES

ACTIVITY-BASED QUESTIONS

Note: Writing a program implies:

- Adding comments as part of documentation
- Writing function definition
- Executing the function through a function call

1. To secure your account, whether it be an email, online bank account or any other account, it is important that we use authentication. Use your programming expertise to create a program using user defined function named login that accepts userid and password as parameters (login(uid,pwd)) that displays a message “account blocked” in case of three wrong attempts. The login is successful if the user enters user ID as "ADMIN" and password as "St0rE@1". On successful login, display a message “login successful”.
2. XYZ store plans to give festival discount to its customers. The store management has decided to give discount on the following criteria:

Shopping Amount	Discount Offered
>=500 and <1000	5%
>=1000 and <2000	8%
>=2000	10%

An additional discount of 5% is given to customers who are the members of the store. Create a program using user defined function that accepts the shopping amount as a parameter and calculates discount and net amount payable on the basis of the following conditions:

Net Payable Amount = Total Shopping Amount – Discount.

3. ‘Play and learn’ strategy helps toddlers understand concepts in a fun way. Being a senior student you have taken responsibility to develop a program using user defined functions to help children master two and three-letter words using English alphabets and addition of single digit numbers. Make sure that you perform a careful analysis of the type of questions that can be included as per the age and curriculum.

NOTES

4. Take a look at the series below:
   1, 1, 2, 3, 5, 8, 13, 21, 34, 55...
   To form the pattern, start by writing 1 and 1. Add them together to get 2. Add the last two numbers: 1+2 = 3.Continue adding the previous two numbers to find the next number in the series. These numbers make up the famed Fibonacci sequence: previous two numbers are added to get the immediate new number.
5. Create a menu driven program using user defined functions to implement a calculator that performs the following:
   a) Basic arithmetic operations(+,-,*,/)
   b) $\log_{10}$(x),sin(x),cos(x)

SUGGESTED LAB. EXERCISES

1. Write a program to check the divisibility of a number by 7 that is passed as a parameter to the user defined function.
2. Write a program that uses a user defined function that accepts name and gender (as M for Male, F for Female) and prefixes Mr/Ms on the basis of the gender.
3. Write a program that has a user defined function to accept the coefficients of a quadratic equation in variables and calculates its determinant. For example : if the coefficients are stored in the variables a,b,c then calculate determinant as b^2-4ac. Write the appropriate condition to check determinants on positive, zero and negative and output appropriate result.
4. ABC School has allotted unique token IDs from (1 to 600) to all the parents for facilitating a lucky draw on the day of their Annual day function. The winner would receive a special prize. Write a program using Python that helps to automate the task.(Hint: use random module)
5. Write a program that implements a user defined function that accepts Principal Amount, Rate, Time, Number of Times the interest is compounded to calculate and displays compound interest. (Hint: CI=P*(1+r/n)^{nt})

NOTES

6. Write a program that has a user defined function to accept 2 numbers as parameters, if number 1 is less than number 2 then numbers are swapped and returned, i.e., number 2 is returned in place of number1 and number 1 is reformed in place of number 2, otherwise the same order is returned.
7. Write a program that contains user defined functions to calculate area, perimeter or surface area whichever is applicable for various shapes like square, rectangle, triangle, circle and cylinder. The user defined functions should accept the values for calculation as parameters and the calculated value should be returned. Import the module and use the appropriate functions.
8. Write a program that creates a GK quiz consisting of any five questions of your choice. The questions should be displayed randomly. Create a user defined function score() to calculate the score of the quiz and another user defined function remark (scorevalue) that accepts the final score to display remarks as follows:

Marks	Remarks
5	Outstanding
4	Excellent
3	Good
2	Read more to score more
1	Needs to take interest
0	General knowledge will always help you. Take it seriously.

CASE STUDY-BASED QUESTION

For the SMIS system extended in Chapter 6 let us do the following:

1. 7.1 Convert all the functionality in Chapter 5 and 6 using user defined functions.
2. 7.2 Add another user defined function to the above menu to check if the student has short attendance or not. The function should accept total number of working days in a month and check if the student is a defaulter by calculating his or her attendance using the formula: Count of days the student was

NOTES

present or the total number of working days. In case the attendance calculated is less than 78%, the function should return 1 indicating short attendance otherwise the function should return 0 indicating attendance is not short.

Let's peer review the case studies of others based on the parameters given under "DOCUMENTATION TIPS" at the end of Chapter 5 and provide a feedback to them.

Chapter 8

Strings

11120CH08

"The great thing about a computer notebook is that no matter how much you stuff into it, it doesn't get bigger or heavier."

– Bill Gates

8.1 Introduction

We have studied in Chapter 5, that a sequence is an orderly collection of items and each item is indexed by an integer. Following sequence data types in Python were also briefly introduced in Chapter 5.

- Strings
- Lists
- Tuples

Another data type 'Dictionary' was also introduced in chapter 5 which falls under the category of mapping. In this chapter, we will go through strings in detail. List will be covered in Chapter 9 whereas tuple and dictionary will be discussed in Chapter 10.

8.2 Strings

String is a sequence which is made up of one or more UNICODE characters. Here the character can be a letter, digit, whitespace or any other symbol. A string can be created by enclosing one or more characters in single, double or triple quote.

Example 8.1

```
>>> str1 = 'Hello World!'
>>> str2 = "Hello World!"
>>> str3 = """Hello World!"""
>>> str4 = '''Hello World!'''
```

`str1`, `str2`, `str3`, `str4` are all string variables having the same value 'Hello World!'. Values stored in `str3` and `str4` can be extended to multiple lines using triple codes as can be seen in the following example:

```
>>> str3 = """Hello World!
           welcome to the world of Python"""
>>> str4 = '''Hello World!
           welcome to the world of Python'''
```

In this chapter

- » *Introduction to Strings*
- » *String Operations*
- » *Traversing a String*
- » *Strings Methods and Built-in Functions*
- » *Handling Strings*

Python does not have a character data type. String of length one is considered as character.

8.2.1 Accessing Characters in a String

Each individual character in a string can be accessed using a technique called indexing. The index specifies the character to be accessed in the string and is written in square brackets ([ ]). The index of the first character (from left) in the string is 0 and the last character is n-1 where n is the length of the string. If we give index value out of this range then we get an *IndexError*. The index must be an integer (positive, zero or negative).

```
#initializes a string str1
>>> str1 = 'Hello World!'
#gives the first character of str1
>>> str1[0]
 'H'
#gives seventh character of str1
>>> str1[6]
'W'
#gives last character of str1
>>> str1[11]
'!'
#gives error as index is out of range
>>> str1[15]
IndexError: string index out of range
```

The index can also be an expression including variables and operators but the expression must evaluate to an integer.

```
#an expression resulting in an integer index
#so gives 6th character of str1
>>> str1[2+4]
 'W'
#gives error as index must be an integer
>>> str1[1.5]
TypeError: string indices must be integers
```

Python allows an index value to be negative also. Negative indices are used when we want to access the characters of the string from right to left. Starting from right hand side, the first character has the index as -1 and the last character has the index –n where n is the length of the string. Table 8.1 shows the indexing of characters in the string 'Hello World!' in both the cases, i.e., positive and negative indices.

```
>>> str1[-1] #gives first character from right
'!'
>>> str1[-12]#gives last character from right
'H'
```

Table 8.1 Indexing of characters in string 'Hello World!'

Positive Indices	0	1	2	3	4	5	6	7	8	9	10	11
String	H	e	l	l	o		W	o	r	l	d	!
Negative Indices	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

An inbuilt function len() in Python returns the length of the string that is passed as parameter. For example, the length of string `str1 = 'Hello World!'` is 12.

```
#gives the length of the string str1
>>> len(str1)
12
#length of the string is assigned to n
>>> n = len(str1)
>>> print(n)
12
#gives the last character of the string
>>> str1[n-1]
'!'
#gives the first character of the string
>>> str1[-n]
'H'
```

8.2.2 String is Immutable

A string is an immutable data type. It means that the contents of the string cannot be changed after it has been created. An attempt to do this would lead to an error.

```
>>> str1 = "Hello World!"
#if we try to replace character 'e' with 'a'
>>> str1[1] = 'a'
TypeError: 'str' object does not support item
assignment
```

8.3 STRING OPERATIONS

As we know that string is a sequence of characters. Python allows certain operations on string data type, such as concatenation, repetition, membership and slicing. These operations are explained in the following subsections with suitable examples.

8.3.1 Concatenation

To concatenate means to join. Python allows us to join two strings using concatenation operator plus which is denoted by symbol +.

```
>>> str1 = 'Hello'      #First string
>>> str2 = 'World!'      #Second string
>>> str1 + str2          #Concatenated strings
'HelloWorld!'
                    #str1 and str2 remain same
>>> str1              #after this operation.
'Hello'
>>> str2
'World!'
```

8.3.2 Repetition

Python allows us to repeat the given string using repetition operator which is denoted by symbol *.

```
#assign string 'Hello' to str1
>>> str1 = 'Hello'
#repeat the value of str1 2 times
>>> str1 * 2
'HelloHello'
#repeat the value of str1 5 times
>>> str1 * 5
'HelloHelloHelloHelloHello'
```

Note: `str1` still remains the same after the use of repetition operator.

8.3.3 Membership

Python has two membership operators `'in'` and `'not in'`. The `'in'` operator takes two strings and returns `True` if the first string appears as a substring in the second string, otherwise it returns `False`.

```
>>> str1 = 'Hello World!'
>>> 'W' in str1
True
>>> 'Wor' in str1
True
>>> 'My' in str1
False
```

The `'not in'` operator also takes two strings and returns `True` if the first string does not appear as a substring in the second string, otherwise returns `False`.

```
>>> str1 = 'Hello World!'
>>> 'My' not in str1
True
>>> 'Hello' not in str1
False
```

8.3.4 Slicing

In Python, to access some part of a string or substring, we use a method called slicing. This can be done by

specifying an index range. Given a string `str1`, the slice operation `str1[n:m]` returns the part of the string str1 starting from index n (inclusive) and ending at m (exclusive). In other words, we can say that `str1[n:m]` returns all the characters starting from `str1[n]` till `str1[m-1]`. The numbers of characters in the substring will always be equal to difference of two indices m and n, i.e., (m-n).

```
>>> str1 = 'Hello World!'
#gives substring starting from index 1 to 4
>>> str1[1:5]
'ello'
#gives substring starting from 7 to 9
>>> str1[7:10]
'orl'
#index that is too big is truncated down to
#the end of the string
>>> str1[3:20]
'lo World!'
#first index > second index results in an
#empty '' string
 >>> str1[7:2]
```

If the first index is not mentioned, the slice starts from index.

```
#gives substring from index 0 to 4
>>> str1[:5]
'Hello'
```

If the second index is not mentioned, the slicing is done till the length of the string.

```
#gives substring from index 6 to end
>>> str1[6:]
'World!'
```

The slice operation can also take a third index that specifies the 'step size'. For example, `str1[n:m:k]`, means every k^{th} character has to be extracted from the string `str1` starting from n and ending at m-1. By default, the step size is one.

```
>>> str1[0:10:2]
'HloWr'
>>> str1[0:10:3]
'HlWl'
```

Negative indexes can also be used for slicing.

```
 #characters at index -6,-5,-4,-3 and -2 are
 #sliced
>>> str1[-6:-1]
```

```
'World'
```

If we ignore both the indexes and give step size as -1

```
#str1 string is obtained in the reverse order
>>> str1[::-1]
'!dlroW olleH'
```

8.4 TRAVERSING A STRING

We can access each character of a string or traverse a string using `for` loop and `while` loop.

(A) String Traversal Using `for` Loop:

```
>>> str1 = 'Hello World!'
>>> for ch in str1:
        print(ch,end = '')
Hello World!        #output of for loop
```

In the above code, the loop starts from the first character of the string `str1` and automatically ends when the last character is accessed.

(B) String Traversal Using `while` Loop:

```
>>> str1 = 'Hello World!'
>>> index = 0
#len(): a function to get length of string
>>> while  index < len(str1):
        print(str1[index],end = '')
        index += 1

Hello World!            #output of while loop
```

Here `while` loop runs till the condition index < len(str) is `True`, where index varies from 0 to len`(str1)` -1.

8.5 STRING METHODS AND BUILT-IN FUNCTIONS

Python has several built-in functions that allow us to work with strings. Table 8.2 describes some of the commonly used built-in functions for string manipulation.

Table 8.2 Built-in functions for string manipulations

Method	Description	Example

len()	Returns the length of the given string	`>>> str1 = 'Hello World!'`
		`>>> len(str1)`
		`12`

title()	Returns the string with first letter of every word in the string in uppercase and rest in lowercase	`>>> str1 = 'hello WORLD!'`
		`>>> str1.title()`
		`'Hello World!'`

lower()	Returns the string with all uppercase letters converted to lowercase	`>>> str1 = 'hello WORLD!'`
		`>>> str1.lower()`
		`'hello world!'`

upper()	Returns the string with all lowercase letters converted to uppercase	`>>> str1 = 'hello WORLD!'`
		`>>> str1.upper()`
		`'HELLO WORLD!'`

count(str, start, end)	Returns number of times substring `str` occurs in the given string. If we do not give start index and end index then searching starts from index 0 and ends at length of the string	`>>> str1 = 'Hello World! Hello Hello'`
		`>>> str1.count('Hello',12,25)`
		`2`
		`>>> str1.count('Hello')`
		`3`

find(str,start, end)	Returns the first occurrence of index of substring `str` occurring in the given string. If we do not give start and end then searching starts from index 0 and ends at length of the string. If the substring is not present in the given string, then the function returns -1	`>>> str1 = 'Hello World! Hello Hello'`
		`>>> str1.find('Hello',10,20)`
		`13`
		`>>> str1.find('Hello',15,25)`
		`19`
		`>>> str1.find('Hello')`
		`0`
		`>>> str1.find('Hee')`
		`-1`

index(str, start, end)	Same as find() but raises an exception if the substring is not present in the given string	`>>> str1 = 'Hello World! Hello Hello'`
		`>>> str1.index('Hello')`
		`0`
		`>>> str1.index('Hee')`
		`ValueError: substring not found`

endswith()	Returns True if the given string ends with the supplied substring otherwise returns False	`>>> str1 = 'Hello World!'`
		`>>> str1.endswith('World!')`
		`True`
		`>>> str1.endswith('!')`
		`True`
		`>>> str1.endswith('lde')`
		`False`

startswith()	Returns True if the given string starts with the supplied substring otherwise returns False	`>>> str1 = 'Hello World!'`
		`>>> str1.startswith('He')`
		`True`
		`>>> str1.startswith('Hee')`
		`False`

isalnum()	Returns True if characters of the given string are either alphabets or numeric. If whitespace or special symbols are part of the given string or the string is empty it returns False	>>> str1 = 'HelloWorld'
		>>> str1.isalnum()
		True
		>>> str1 = 'HelloWorld2'
		>>> str1.isalnum()
		True
		>>> str1 = 'HelloWorld!!'
		>>> str1.isalnum()
		False

islower()	Returns True if the string is non-empty and has all lowercase alphabets, or has at least one character as lowercase alphabet and rest are non-alphabet characters	>>> str1 = 'hello world!'
		>>> str1.islower()
		True
		>>> str1 = 'hello 1234'
		>>> str1.islower()
		True
		>>> str1 = 'hello ??'
		>>> str1.islower()
		True
		>>> str1 = '1234'
		>>> str1.islower()
		False
		>>> str1 = 'Hello World!'
		>>> str1.islower()
		False

isupper()	Returns True if the string is non-empty and has all uppercase alphabets, or has at least one character as uppercase character and rest are non-alphabet characters	>>> str1 = 'HELLO WORLD!'
		>>> str1.isupper()
		True
		>>> str1 = 'HELLO 1234'
		>>> str1.isupper()
		True
		>>> str1 = 'HELLO ??'
		>>> str1.isupper()
		True
		>>> str1 = '1234'
		>>> str1.isupper()
		False
		>>> str1 = 'Hello World!'
		>>> str1.isupper()
		False

isspace()	Returns True if the string is non-empty and all characters are white spaces (blank, tab, newline, carriage return)	>>> str1 = ' \n \t \r'
		>>> str1.isspace()
		True
		>>> str1 = 'Hello \n'
		>>> str1.isspace()
		False

istitle()	Returns True if the string is non-empty and title case, i.e., the first letter of every word in the string in uppercase and rest in lowercase	>>> str1 = 'Hello World!'
		>>> str1.istitle()
		True
		>>> str1 = 'hello World!'
		>>> str1.istitle()
		False

lstrip()	Returns the string after removing the spaces only on the left of the string	>>> str1 = ' Hello World! '
		>>> str1.lstrip()
		'Hello World! '

rstrip()	Returns the string after removing the spaces only on the right of the string	>>> str1 = ' Hello World!'
		>>> str1.rstrip()
		' Hello World!'

strip()	Returns the string after removing the spaces both on the left and the right of the string	>>> str1 = ' Hello World!'
		>>> str1.strip()
		'Hello World!'

replace(oldstr, newstr)	Replaces all occurrences of old string with the new string	>>> str1 = 'Hello World!'
		>>> str1.replace('o','*')
		'Hell* W*rld!'
		>>> str1 = 'Hello World!'
		>>> str1.replace('World','Country')
		'Hello Country!'
		>>> str1 = 'Hello World! Hello'
		>>> str1.replace('Hello','Bye')
		'Bye World! Bye'

join()	Returns a string in which the characters in the string have been joined by a separator	>>> str1 = ('HelloWorld!')
		>>> str2 = '-' #separator
		>>> str2.join(str1)
		'H-e-l-l-o-W-o-r-l-d-!'

partition())	Partitions the given string at the first occurrence of the substring (separator) and returns the string partitioned into three parts.
	1. Substring before the separator
	2. Separator
	3. Substring after the separator
	If the separator is not found in the string, it returns the whole string itself and two empty strings
		`>>> str1 = 'India is a Great Country'`
		`>>> str1.partition('is')`
		`('India ', 'is', ' a Great Country')`
		`>>> str1.partition('are')`
		`('India is a Great Country',' ',' ')`

split()	Returns a list of words delimited by the specified substring. If no delimiter is given then words are separated by space.	`>>> str1 = 'India is a Great Country'`
		`>>> str1.split()`
		`['India','is','a','Great', 'Country']`
		`>>> str1 = 'India is a Great Country'`
		`>>> str1.split('a')`
		`['Indi', ' is ', ' Gre', 't Country']`

8.6 HANDLING STRINGS

In this section, we will learn about user defined functions in Python to perform different operations on strings.

Program 8-1 Write a program with a user defined function to count the number of times a character (passed as argument) occurs in the given string.

```
#Program 8-1
#Function to count the number of times a character occurs in a
#string
def charCount(ch,st):
     count = 0
     for character in st:
          if character == ch:
               count += 1
     return count
#end of function

st = input("Enter a string: ")
ch = input("Enter the character to be searched: ")
count = charCount(ch,st)
print("Number of times character",ch,"occurs in the string
is:",count)
```

Output:

```
Enter a string: Today is a Holiday
Enter the character to be searched: a
Number of times character a occurs in the string is: 3
```

Program 8-2 Write a program with a user defined function with string as a parameter which replaces all vowels in the string with '*'.

```
#Program 8-2
#Function to replace all vowels in the string with  '*'
def replaceVowel(st):
     #create an empty string
     newstr = ''
     for character in st:
          #check if next character is a vowel
          if character in 'aeiouAEIOU':
               #Replace vowel with *
               newstr += '*'
          else:
               newstr += character
     return newstr
#end of function
st = input("Enter a String: ")
st1 = replaceVowel(st)
print("The original String is:",st)
print("The modified String is:",st1)
```

Output:

```
Enter a String: Hello World
The original String is: Hello World
The modified String is: H*ll* W*rld
```

Program 8-3 Write a program to input a string from the user and print it in the reverse order without creating a new string.

```
#Program 8-3
#Program to display string in reverse order
st = input("Enter a string: ")
for i in range(-1,-len(st)-1,-1):
    print(st[i],end='')
```

Output:

```
Enter a string: Hello World
dlroW olleH
```

Program 8-4 Write a program which reverses a string passed as parameter and stores the reversed string in a new string. Use a user defined function for reversing the string.

```
#Program 8-4
#Function to reverse a string
def reverseString(st):
     newstr = ''         #create a new string
     length = len(st)
     for i in range(-1,-length-1,-1):
         newstr += st[i]
     return newstr
#end of function
st = input("Enter a String: ")
st1 = reverseString(st)
print("The original String is:",st)
print("The reversed String is:",st1)
```

Output:

```
Enter a String: Hello World
The original String is: Hello World
The reversed String is: dlroW olleH
```

Program 8-5 Write a program using a user defined function to check if a string is a palindrome or not. (A string is called palindrome if it reads same backwards as forward. For example, Kanak is a palindrome.)

```
#Program 8-5
#Function to check if a string is palindrome or not
def checkPalin(st):
     i = 0
     j = len(st) - 1
     while(i <= j):
          if(st[i] != st[j]):
               return False
          i += 1
          j -= 1
     return True
#end of function
st = input("Enter a String: ")
result = checkPalin(st)
if result == True:
     print("The given string",st,"is a palindrome")
else:
     print("The given string",st,"is not a palindrome")
```

Output 1:

```
Enter a String: kanak
The given string kanak is a palindrome
```

Output 2:

```
Enter a String: computer
The given string computer is not a palindrome
```

NOTES

SUMMARY

- A string is a sequence of characters enclosed in single, double or triple quotes.
- Indexing is used for accessing individual characters within a string.
- The first character has the index 0 and the last character has the index n-1 where n is the length of the string. The negative indexing ranges from -n to -1.
- Strings in Python are immutable, i.e., a string cannot be changed after it is created.
- Membership operator `in` takes two strings and returns `True` if the first string appears as a substring in the second `else` returns `False`. Membership operator '`not in`' does the reverse.
- Retrieving a portion of a string is called slicing. This can be done by specifying an index range. The slice operation `str1[n:m]` returns the part of the string `str1` starting from index n (inclusive) and ending at m (exclusive).
- Each character of a string can be accessed either using a `for` loop or `while` loop.
- There are many built-in functions for working with strings in Python.

EXERCISE

1. Consider the following string mySubject:

```
mySubject = "Computer Science"
```

What will be the output of the following string operations :

```
i.   print(mySubject[0:len(mySubject)])
ii.  print(mySubject[-7:-1])
iii. print(mySubject[::2])
iv.  print(mySubject[len(mySubject)-1])
v.   print(2*mySubject)
vi.  print(mySubject[::-2])
vii. print(mySubject[:3] + mySubject[3:])
viii. print(mySubject.swapcase())
ix.  print(mySubject.startswith('Comp'))
x.   print(mySubject.isalpha())
```

2. Consider the following string myAddress:

```
myAddress = "WZ-1,New Ganga Nagar,New Delhi"
```

What will be the output of following string operations :

```
i. print(myAddress.lower())
```

NOTES

```
 ii.  print(myAddress.upper())
iii.  print(myAddress.count('New'))
 iv.  print(myAddress.find('New'))
  v.  print(myAddress.rfind('New'))
 vi.  print(myAddress.split(','))
vii.  print(myAddress.split(' '))
viii. print(myAddress.replace('New','Old'))
 ix.  print(myAddress.partition(','))
  x.  print(myAddress.index('Agra'))
```

PROGRAMMING PROBLEMS

1. Write a program to input line(s) of text from the user until enter is pressed. Count the total number of characters in the text (including white spaces),total number of alphabets, total number of digits, total number of special symbols and total number of words in the given text. (Assume that each word is separated by one space).
2. Write a user defined function to convert a string with more than one word into title case string where string is passed as parameter. (Title case means that the first letter of each word is capitalised)
3. Write a function deleteChar() which takes two parameters one is a string and other is a character. The function should create a new string after deleting all occurrences of the character from the string and return the new string.
4. Input a string having some digits. Write a function to return the sum of digits present in this string.
5. Write a function that takes a sentence as an input parameter where each word in the sentence is separated by a space. The function should replace each blank with a hyphen and then return the modified sentence.

Chapter 9

Lists

11120CH09

"Measuring programming progress by lines of code is like measuring aircraft building progress by weight."

–Bill Gates

9.1 Introduction to List

The data type list is an ordered sequence which is mutable and made up of one or more elements. Unlike a string which consists of only characters, a list can have elements of different data types, such as integer, float, string, tuple or even another list. A list is very useful to group together elements of mixed data types. Elements of a list are enclosed in square brackets and are separated by comma. Like string indices, list indices also start from 0.

Example 9.1

```
#list1 is the list of six even numbers
>>> list1 = [2,4,6,8,10,12]
>>> print(list1)
[2, 4, 6, 8, 10, 12]

#list2 is the list of vowels
>>> list2 = ['a','e','i','o','u']
>>> print(list2)
['a', 'e', 'i', 'o', 'u']

#list3 is the list of mixed data types
>>> list3 = [100,23.5,'Hello']
>>> print(list3)
[100, 23.5, 'Hello']

#list4 is the list of lists called nested
#list
>>> list4 =[['Physics',101],['Chemistry',202],
           ['Maths',303]]
>>> print(list4)
[['Physics', 101], ['Chemistry', 202],
 ['Maths', 303]]
```

9.1.1 Accessing Elements in a List

The elements of a list are accessed in the same way as characters are accessed in a string.

In this chapter

- *Introduction to List*
- *List Operations*
- *Traversing a List*
- *List Methods and Built-in Functions*
- *Nested Lists*
- *Copying Lists*
- *List as Arguments to Function*
- *List Manipulation*

NOTES

```
#initializes a list list1
>>> list1 = [2,4,6,8,10,12]
>>> list1[0] #return first element of list1
2
>>> list1[3] #return fourth element of list1
8
#return error as index is out of range
>>> list1[15]
IndexError: list index out of range
#an expression resulting in an integer index
>>> list1[1+4]
12
>>> list1[-1] #return first element from right
12
#length of the list list1 is assigned to n
>>> n = len(list1)
>>> print(n)
6
#return the last element of the list1
>>> list1[n-1]
12
#return the first element of list1
>>> list1[-n]
2
```

9.1.2 Lists are Mutable

In Python, lists are mutable. It means that the contents of the list can be changed after it has been created.

```
#List list1 of colors
>>> list1 = ['Red','Green','Blue','Orange']
#change/override the fourth element of list1
>>> list1[3] = 'Black'
>>> list1      #print the modified list list1
['Red', 'Green', 'Blue', 'Black']
```

9.2 LIST OPERATIONS

The data type list allows manipulation of its contents through various operations as shown below.

9.2.1 Concatenation

Python allows us to join two or more lists using concatenation operator depicted by the symbol +.

```
#list1 is list of first five odd integers
>>> list1 = [1,3,5,7,9]
#list2 is list of first five even integers
>>> list2 = [2,4,6,8,10]
#elements of list1 followed by list2
```

NOTES

```
>>> list1 + list2
[1, 3, 5, 7, 9, 2, 4, 6, 8, 10]
>>> list3 = ['Red','Green','Blue']
>>> list4 = ['Cyan', 'Magenta', 'Yellow'
,'Black']
>>> list3 + list4
['Red','Green','Blue','Cyan','Magenta',
   'Yellow','Black']
```

Note that, there is no change in ongoing lists, i.e., `list1, list2, list3, list4` remain the same after concatenation operation. If we want to merge two lists, then we should use an assignment statement to assign the merged list to another list. The concatenation operator '+' requires that the operands should be of list type only. If we try to concatenate a list with elements of some other data type, `TypeError` occurs.

```
>>> list1 = [1,2,3]
>>> str1 = "abc"
>>> list1 + str1
TypeError: can only concatenate list (not
"str") to list
```

9.2.2 Repetition

Python allows us to replicate a list using repetition operator depicted by symbol *.

```
>>> list1 = ['Hello']
#elements of list1 repeated 4 times
>>> list1 * 4
['Hello', 'Hello', 'Hello', 'Hello']
```

9.2.3 Membership

Like strings, the membership operators `in` checks if the element is present in the list and returns `True`, else returns `False`.

```
>>> list1 = ['Red','Green','Blue']
>>> 'Green' in list1
True
>>> 'Cyan' in list1
False
```

The `not in` operator returns `True` if the element is not present in the list, else it returns `False`.

```
>>> list1 = ['Red','Green','Blue']
>>> 'Cyan' not in list1
True
>>> 'Green' not in list1
False
```

NOTES

9.2.4 Slicing

Like strings, the slicing operation can also be applied to lists.

```
>>> list1 =['Red','Green','Blue','Cyan',
'Magenta','Yellow','Black']
>>> list1[2:6]
['Blue', 'Cyan', 'Magenta', 'Yellow']

 #list1 is truncated to the end of the list
>>> list1[2:20] #second index is out of range
['Blue', 'Cyan', 'Magenta', 'Yellow',
'Black']

>>> list1[7:2]     #first index > second index
[]                 #results in an empty list

#return sublist from index 0 to 4
>>> list1[:5]      #first index missing
['Red','Green','Blue','Cyan','Magenta']


#slicing with a given step size
>>> list1[0:6:2]
['Red','Blue','Magenta']

#negative indexes
#elements at index -6,-5,-4,-3 are sliced
>>> list1[-6:-2]
['Green','Blue','Cyan','Magenta']


#both first and last index missing
>>> list1[::2]     #step size 2 on entire list
['Red','Blue','Magenta','Black']

#negative step size
#whole list in the reverse order
>>> list1[::-1]
['Black','Yellow','Magenta','Cyan','Blue',
'Green','Red']
```

9.3 TRAVERSING A LIST

We can access each element of the list or traverse a list using a `for` loop or a `while` loop.

(A) List Traversal Using `for` Loop:

```
>>> list1 = ['Red','Green','Blue','Yellow',
             'Black']
```

```
>>> for item in list1:
        print(item)
```

Output:

```
Red
Green
Blue
Yellow
Black
```

Another way of accessing the elements of the list is using range() and len() functions:

```
>>> for i in range(len(list1)):
        print(list1[i])
```

Output:

```
Red
Green
Blue
Yellow
Black
```

`len(list1)` returns the length or total number of elements of list1.

(B) List Traversal Using `while` Loop:

```
>>> list1 = ['Red','Green','Blue','Yellow',
             'Black']
>>> i = 0
>>> while i < len(list1):
        print(list1[i])
        i += 1
```

Output:

```
Red
Green
Blue
Yellow
Black
```

9.4 LIST METHODS AND BUILT-IN FUNCTIONS

The data type list has several built-in methods that are useful in programming. Some of them are listed in Table 9.1.

Table 9.1 Built-in functions for list manipulations

Method	Description	Example

len()	Returns the length of the list passed as the argument	`>>> list1 = [10,20,30,40,50]`
		`>>> len(list1)`
		`5`

list()	Creates an empty list if no argument is passed	`>>> list1 = list()` `>>> list1`

	Creates a list if a sequence is passed as an argument	`[ ]`
		`>>> str1 = 'aeiou'`
		`>>> list1 = list(str1)`
		`>>> list1`
		`['a', 'e', 'i', 'o', 'u']`

append()	Appends a single element passed as an argument at the end of the list
	
	The single element can also be a list
		`>>> list1 = [10,20,30,40]`
		`>>> list1.append(50)`
		`>>> list1`
		`[10, 20, 30, 40, 50]`
		`>>> list1 = [10,20,30,40]`
		`>>> list1.append([50,60])`
		`>>> list1`
		`[10, 20, 30, 40, [50, 60]]`

extend()	Appends each element of the list passed as argument to the end of the given list	`>>> list1 = [10,20,30]`
		`>>> list2 = [40,50]`
		`>>> list1.extend(list2)`
		`>>> list1`
		`[10, 20, 30, 40, 50]`

insert()	Inserts an element at a particular index in the list	`>>> list1 = [10,20,30,40,50]`
		`>>> list1.insert(2,25)`
		`>>> list1`
		`[10, 20, 25, 30, 40, 50]`
		`>>> list1.insert(0,5)`
		`>>> list1`
		`[5, 10, 20, 25, 30, 40, 50]`

count()	Returns the number of times a given element appears in the list	`>>> list1 = [10,20,30,10,40,10]`
		`>>> list1.count(10)`
		`3`
		`>>> list1.count(90)`
		`0`

index()	Returns index of the first occurrence of the element in the list. If the element is not present, ValueError is generated	`>>> list1 = [10,20,30,20,40,10]`
		`>>> list1.index(20)`
		`1`
		`>>> list1.index(90)`
		`ValueError: 90 is not in list`

remove()	Removes the given element from the list. If the element is present multiple times, only the first occurrence is removed. If the element is not present, then ValueError is generated	`>>> list1 = [10,20,30,40,50,30]`
		`>>> list1.remove(30)`
		`>>> list1`
		`[10, 20, 40, 50, 30]`
		
		`>>> list1.remove(90)`
		`ValueError:list.remove(x):x not in list`

pop()	Returns the element whose index is passed as parameter to this function and also removes it from the list. If no parameter is given, then it returns and removes the last element of the list	`>>> list1 = [10,20,30,40,50,60]`
		`>>> list1.pop(3)`
		`40`
		`>>> list1`
		`[10, 20, 30, 50, 60]`
		`>>> list1 = [10,20,30,40,50,60]`
		`>>> list1.pop()`
		`60`

		>>> list1 [10, 20, 30, 40, 50]

reverse()	Reverses the order of elements in the given list	>>> list1 = [34,66,12,89,28,99]
		>>> list1.reverse()
		>>> list1
		[99, 28, 89, 12, 66, 34]
		
		>>> list1 = ['Tiger' ,'Zebra' , 'Lion' , 'Cat' ,'Elephant' ,'Dog']
		>>> list1.reverse()
		>>> list1
		['Dog', 'Elephant', 'Cat', 'Lion', 'Zebra', 'Tiger']

sort()	Sorts the elements of the given list in-place	>>>list1=['Tiger','Zebra','Lion', 'Cat', 'Elephant' ,'Dog']
		>>> list1.sort()
		>>> list1
		['Cat', 'Dog', 'Elephant', 'Lion', 'Tiger', 'Zebra']
		
		>>> list1 = [34,66,12,89,28,99]
		>>> list1.sort(reverse = True)
		>>> list1
		[99,89,66,34,28,12]

sorted()	It takes a list as parameter and creates a new list consisting of the same elements arranged in sorted order	>>> list1 = [23,45,11,67,85,56]
		>>> list2 = sorted(list1)
		>>> list1
		[23, 45, 11, 67, 85, 56]
		>>> list2
		[11, 23, 45, 56, 67, 85]

min()
 	
max()
 	
sum()
	Returns minimum or smallest element of the list
	
	Returns maximum or largest element of the list
	Returns sum of the elements of the list
		>>> list1 = [34,12,63,39,92,44]
		>>> min(list1)
		12
		>>> max(list1)
		92
		>>> sum(list1)
		284

9.5 NESTED LISTS

When a list appears as an element of another list, it is called a nested list.

Example 9.2

```
>>> list1 = [1,2,'a','c',[6,7,8],4,9]
#fifth element of list is also a list
>>> list1[4]
[6, 7, 8]
```

To access the element of the nested list of `list1`, we have to specify two indices `list1[i][j]`. The first index `i` will take us to the desired nested list and second index `j` will take us to the desired element in that nested list.

```
>>> list1[4][1]
7
#index i gives the fifth element of list1
#which is a list
#index j gives the second element in the
#nested list
```

9.6 COPYING LISTS

Given a list, the simplest way to make a copy of the list is to assign it to another list.

```
>>> list1 = [1,2,3]
>>> list2 = list1
>>> list1
[1, 2, 3]
>>> list2
[1, 2, 3]
```

The statement `list2 = list1` does not create a new list. Rather, it just makes `list1` and `list2` refer to the same list object. Here `list2` actually becomes an alias of `list1`. Therefore, any changes made to either of them will be reflected in the other list.

```
>>> list1.append(10)
>>> list1
[1, 2, 3, 10]
>>> list2
[1, 2, 3, 10]
```

We can also create a copy or clone of the list as a distinct object by three methods. The first method uses slicing, the second method uses built-in function list() and the third method uses copy() function of python library `copy`.

Method 1

We can slice our original list and store it into a new variable as follows:

```
newList = oldList[:]
```

Example 9.3

```
>>> list1 = [1,2,3,4,5]
>>> list2 = list1[:]
>>> list2
[1, 2, 3, 4, 5]
```

Method 2

We can use the built-in function list() as follows:

```
newList = list(oldList)
```

Example 9.4

```
>>> list1 = [10,20,30,40]
>>> list2 = list(list1)
>>> list2
[10, 20, 30, 40]
```

Method 3

We can use the copy () function as follows:

```
import copy        #import the library copy
#use copy()function of library copy
newList = copy.copy(oldList)
```

Example 9.5

```
>>> import copy
>>> list1 = [1,2,3,4,5]
>>> list2 = copy.copy(list1)
>>> list2
[1, 2, 3, 4, 5]
```

9.7 LIST AS ARGUMENT TO A FUNCTION

Whenever a list is passed as an argument to a function, we have to consider two scenarios:

(A) Elements of the original list may be changed, i.e. changes made to the list in the function are reflected back in the calling function.

For example in the following program list `list1` of numbers is passed as an argument to function `increment()`. This function increases every element of the list by 5.

Program 9-1 Program to increment the elements of a list. The list is passed as an argument to a function.

```
#Program 9-1
#Function to increment the elements of the list passed as argument
def increment(list2):
     for i in range(0,len(list2)):
          #5 is added to individual elements in the list
          list2[i] += 5
     print('Reference of list Inside Function',id(list2))
#end of function
list1 = [10,20,30,40,50]   #Create a list
print("Reference of list in Main",id(list1))
print("The list before the function call")
print(list1)
increment(list1)           #list1 is passed as parameter to function
print("The list after the function call")
print(list1)
```

Output:

```
Reference of list in Main 70615968
The list before the function call
[10, 20, 30, 40, 50]
Reference of list Inside Function 70615968 #The id remains same
The list after the function call
[15, 25, 35, 45, 55]
```

Observe that, when we pass a list as an argument, we actually pass a reference to the list. Hence any change made to `list2` inside the function is reflected in the actual list `list1`.

(B) If the list is assigned a new value inside the function then a new list object is created and it becomes the local copy of the function. Any changes made inside the local copy of the function are not reflected back to the calling function.

Program 9-2 Program to increment the elements of the list passed as parameter.

```
#Program 9-2
#Function to increment the elements of the list passed as argument
def increment(list2):
     print("\nID of list inside function before assignment:",
id(list2))
     list2 = [15,25,35,45,55] #List2 assigned a new list
     print("ID of list changes inside function after assignment:",
id(list2))
     print("The list inside the function after assignment is:")
     print(list2)
#end of function

list1 = [10,20,30,40,50]          #Create a list
print("ID of list before function call:",id(list1))
print("The list before function call:")
print(list1)
increment(list1)  #list1 passed as parameter to function
print('\nID of list after function call:',id(list1))
print("The list after the function call:")
print(list1)
```

Output:

```
ID of list before function call: 65565640
The list before function call:
[10, 20, 30, 40, 50]

ID of list inside function before assignment:65565640
ID of list changes inside function  after assignment:65565600
The list inside the function after assignment is:
```

```
[15, 25, 35, 45, 55]

ID of list after function call: 65565640
The list after the function call:
[10, 20, 30, 40, 50]
```

9.8 LIST MANIPULATION

In this chapter, we have learnt to create a list and the different ways to manipulate lists. In the following programs, we will apply the various list manipulation methods.

Program 9-3 Write a menu driven program to perform various list operations, such as:

- Append an element
- Insert an element
- Append a list to the given list
- Modify an existing element
- Delete an existing element from its position
- Delete an existing element with a given value
- Sort the list in ascending order
- Sort the list in descending order
- Display the list.

```
#Program 9-3
#Menu driven program to do various list operations
myList = [22,4,16,38,13] #myList already has 5 elements
choice = 0
while True:
    print("The list 'myList' has the following elements", myList)
    print("\nL I S T   O P E R A T I O N S")
    print(" 1. Append an element")
    print(" 2. Insert an element at the desired position")
    print(" 3. Append a list to the given list")
    print(" 4. Modify an existing element")
    print(" 5. Delete an existing element by its position")
    print(" 6. Delete an existing element by its value")
    print(" 7. Sort the list in ascending order")
    print(" 8. Sort the list in descending order")
    print(" 9. Display the list")
    print(" 10. Exit")
    choice = int(input("ENTER YOUR CHOICE (1-10): "))

    #append element
    if choice == 1:
        element = int(input("Enter the element to be appended: "))
        myList.append(element)
```

```
            print("The element has been appended\n")

        #insert an element at desired position
        elif choice == 2:
            element = int(input("Enter the element to be inserted: "))
            pos = int(input("Enter the position:"))
            myList.insert(pos,element)
            print("The element has been inserted\n")

        #append a list to the given list
        elif choice == 3:
            newList = eval(input( "Enter the elements separated by commas"))
            myList.extend(list(newList))
            print("The list has been appended\n")

        #modify an existing element
        elif choice == 4:
            i = int(input("Enter the position of the element to be
modified: "))
            if i < len(myList):
                newElement = int(input("Enter the new element: "))
                oldElement = myList[i]
                myList[i] = newElement
                print("The element",oldElement,"has been modified\n")
            else:
                print("Position of the element is more than the length
of list")

        #delete an existing element by position
        elif choice == 5:
            i = int(input("Enter the position of the element to be
deleted: "))
            if i < len(myList):
                element = myList.pop(i)
                print("The element",element,"has been deleted\n")
            else:
                print("\nPosition of the element is more than the length
of list")

        #delete an existing element by value
        elif choice == 6:
            element = int(input("\nEnter the element to be deleted: "))
            if element in myList:
                myList.remove(element)
                print("\nThe element",element,"has been deleted\n")
            else:
                print("\nElement",element,"is not present in the list")

        #list in sorted order
```

```
    elif choice == 7:
        myList.sort()
        print("\nThe list has been sorted")

    #list in reverse sorted order
    elif choice == 8:
        myList.sort(reverse = True)
        print("\nThe list has been sorted in reverse order")

    #display the list
    elif choice == 9:
        print("\nThe list is:", myList)

    #exit from the menu
    elif choice == 10:
        break
    else:
        print("Choice is not valid")
        print("\n\nPress any key to continue.............")
        ch = input()
```

Output:

```
The list 'myList' has the following elements [22, 4, 16, 38, 13]

L I S T   O P E R A T I O N S
 1. Append an element
 2. Insert an element at the desired position
 3. Append a list to the given list
 4. Modify an existing element
 5. Delete an existing element by its position
 6. Delete an existing element by its value
 7. Sort the list in ascending order
 8. Sort the list in descending order
 9. Display the list
10. Exit
ENTER YOUR CHOICE (1-10): 8

The list has been sorted in reverse order
The list 'myList' has the following elements [38, 22, 16, 13, 4]

L I S T   O P E R A T I O N S
 1. Append an element
 2. Insert an element at the desired position
 3. Append a list to the given list
 4. Modify an existing element
 5. Delete an existing element by its position
 6. Delete an existing element by its value
 7. Sort the list in ascending order
 8. Sort the list in descending order
 9. Display the list
10. Exit
```

```
ENTER YOUR CHOICE (1-10): 5
Enter the position of the element to be deleted: 2
The element 16 has been deleted

The list 'myList' has the following elements [38, 22, 13, 4]

L I S T   O P E R A T I O N S
 1. Append an element
 2. Insert an element at the desired position
 3. Append a list to the given list
 4. Modify an existing element
 5. Delete an existing element by its position
 6. Delete an existing element by its value
 7. Sort the list in ascending order
 8. Sort the list in descending order
 9. Display the list
10. Exit
```

Program 9-4 A program to calculate average marks of n students using a function where n is entered by the user.

```
#Program 9-4
#Function to calculate average marks of n students
def computeAverage(list1,n):
     #initialize total
     total = 0
     for marks in list1:
          #add marks to total
          total = total + marks
     average = total / n
     return average

#create an empty list
list1 = []
print("How many students marks you want to enter: ")
n = int(input())
for i in range(0,n):
     print("Enter marks of student",(i+1),":")
     marks = int(input())
     #append marks in the list
     list1.append(marks)
average = computeAverage(list1,n)
print("Average marks of",n,"students is:",average)
```

Output:

```
How many students marks you want to enter:
5
Enter marks of student 1:
45
```

```
Enter marks of student 2:
89
Enter marks of student 3:
79
Enter marks of student 4:
76
Enter marks of student 5:
55
Average marks of 5 students is: 68.8
```

Program 9-5 Write a user-defined function to check if a number is present in the list or not. If the number is present, return the position of the number. Print an appropriate message if the number is not present in the list.

```
#Program 9-5
#Function to check if a number is present in the list or not
def linearSearch(num,list1):
     for i in range(0,len(list1)):
          if list1[i] == num:          #num is present
               return i                #return the position
     return None                       #num is not present in the list
#end of function

list1 = []                             #Create an empty list
print("How many numbers do you want to enter in the list: ")
maximum = int(input())
print("Enter a list of numbers: ")
for i in range(0,maximum):
     n = int(input())
     list1.append(n)                   #append numbers to the list
num = int(input("Enter the number to be searched: "))
result = linearSearch(num,list1)
if result is None:
     print("Number",num,"is not present in the list")
else:
     print("Number",num,"is present at",result + 1, "position")
```

Output:

```
How many numbers do you want to enter in the list:
5
Enter a list of numbers:
23
567
12
89
324
Enter the number to be searched:12
Number 12 is present at 3 position
```

NOTES

SUMMARY

- Lists are mutable sequences in Python, i.e., we can change the elements of the list.
- Elements of a list are put in square brackets separated by comma.
- A list within a list is called a nested list. List indexing is same as that of strings and starts at 0. Two way indexing allows traversing the list in the forward as well as in the backward direction.
- Operator `+` concatenates one list to the end of other list.
- Operator `*` repeats a list by specified number of times.
- Membership operator `in` tells if an element is present in the list or not and `not in` does the opposite.
- Slicing is used to extract a part of the list.
- There are many list manipulation functions including: len(), list(), append(), extend(), insert(), count(), find(), remove(), pop(), reverse(), sort(), sorted(), min(), max(), sum().

EXERCISE

1. What will be the output of the following statements?

i.
```
list1 = [12,32,65,26,80,10]
list1.sort()
print(list1)
```

ii.
```
list1 = [12,32,65,26,80,10]
sorted(list1)
print(list1)
```

iii.
```
list1 = [1,2,3,4,5,6,7,8,9,10]
list1[::-2]
list1[:3] + list1[3:]
```

iv.
```
list1 = [1,2,3,4,5]
list1[len(list1)-1]
```

2. Consider the following list `myList`. What will be the elements of `myList` after the following two operations:

```
myList = [10,20,30,40]
 i. myList.append([50,60])
ii. myList.extend([80,90])
```

NOTES

3. What will be the output of the following code segment:

```
myList = [1,2,3,4,5,6,7,8,9,10]
for i in range(0,len(myList)):
    if i%2 == 0:
        print(myList[i])
```

4. What will be the output of the following code segment:

a.
```
myList = [1,2,3,4,5,6,7,8,9,10]
del myList[3:]
print(myList)
```

b.
```
myList = [1,2,3,4,5,6,7,8,9,10]
del myList[:5]
print(myList)
```

c.
```
myList = [1,2,3,4,5,6,7,8,9,10]
del myList[::2]
print(myList)
```

5. Differentiate between append() and extend() functions of list.

6. Consider a list:

```
list1 = [6,7,8,9]
```

What is the difference between the following operations on list1:

```
a. list1 * 2
b. list1 *= 2
c. list1 =  list1 * 2
```

7. The record of a student (Name, Roll No., Marks in five subjects and percentage of marks) is stored in the following list:

```
stRecord = ['Raman','A-36',[56,98,99,72,69],
            78.8]
```

Write Python statements to retrieve the following information from the list `stRecord`.

a) Percentage of the student
b) Marks in the fifth subject
c) Maximum marks of the student
d) Roll no. of the student
e) Change the name of the student from 'Raman' to 'Raghav'

PROGRAMMING PROBLEMS

1. Write a program to find the number of times an element occurs in the list.

2. Write a program to read a list of n integers (positive

NOTES

as well as negative). Create two new lists, one having all positive numbers and the other having all negative numbers from the given list. Print all three lists.

3. Write a function that returns the largest element of the list passed as parameter.
4. Write a function to return the second largest number from a list of numbers.
5. Write a program to read a list of n integers and find their median.

   Note: The median value of a list of values is the middle one when they are arranged in order. If there are two middle values then take their average.

   Hint: You can use an built-in function to sort the list
6. Write a program to read a list of elements. Modify this list so that it does not contain any duplicate elements, i.e., all elements occurring multiple times in the list should appear only once.
7. Write a program to read a list of elements. Input an element from the user that has to be inserted in the list. Also input the position at which it is to be inserted. Write a user defined function to insert the element at the desired position in the list.
8. Write a program to read elements of a list.
   a) The program should ask for the position of the element to be deleted from the list. Write a function to delete the element at the desired position in the list.
   b) The program should ask for the value of the element to be deleted from the list. Write a function to delete the element of this value from the list.
9. Read a list of n elements. Pass this list to a function which reverses this list in-place without creating a new list.

Chapter 10

Tuples and Dictionaries

11120CH10

10.1 Introduction to Tuples

A tuple is an ordered sequence of elements of different data types, such as integer, float, string, list or even a tuple. Elements of a tuple are enclosed in parenthesis (round brackets) and are separated by commas. Like list and string, elements of a tuple can be accessed using index values, starting from 0.

Example 10.1

```
#tuple1 is the tuple of integers
>>> tuple1 = (1,2,3,4,5)
>>> tuple1
(1, 2, 3, 4, 5)

#tuple2 is the tuple of mixed data types
>>> tuple2 =('Economics',87,'Accountancy',89.6)
>>> tuple2
('Economics', 87, 'Accountancy', 89.6)

#tuple3 is the tuple with list as an element
>>> tuple3 = (10,20,30,[40,50])
>>> tuple3
(10, 20, 30, [40, 50])

#tuple4 is the tuple with tuple as an element
>>> tuple4 = (1,2,3,4,5,(10,20))
>>> tuple4
(1, 2, 3, 4, 5, (10, 20))
```

If there is only a single element in a tuple then the element should be followed by a comma. If we assign the value without comma it is treated as integer. It should be noted that a sequence without parenthesis is treated as tuple by default.

```
#incorrect way of assigning single element to
#tuple
#tuple5 is assigned a single element
>>> tuple5 = (20)
```

"Computers are to computing as instruments are to music. Software is the score whose interpretations amplifies our reach and lifts our spirits. Leonardo da Vinci called music the shaping of the invisible, and his phrase is even more apt as a description of software."

– A Kay

In this chapter

We generally use list to store elements of the same data types whereas we use tuples to store elements of different data types.

```
>>> tuple5
20
>>>type(tuple5)  #tuple5 is not of type tuple
<class 'int'>    #it is treated as integer

#Correct Way of assigning single element to
#tuple
#tuple5 is assigned a single element
>>> tuple5 = (20,) #element followed by comma
>>> tuple5
(20,)
>>>type(tuple5)    #tuple5 is of type tuple
<class 'tuple'>

#a sequence without parentheses is treated as
#tuple by default
>>> seq = 1,2,3     #comma separated elements
>>> type(seq)       #treated as tuple
<class 'tuple'>
>>> print(seq)      #seq is a tuple
(1, 2, 3)
```

10.1.1 Accessing Elements in a Tuple

Elements of a tuple can be accessed in the same way as a list or string using indexing and slicing.

```
>>> tuple1 = (2,4,6,8,10,12)
#initializes a tuple tuple1
#returns the first element of tuple1
>>> tuple1[0]
2
#returns fourth element of tuple1
>>> tuple1[3]
8
#returns error as index is out of range
>>> tuple1[15]
IndexError: tuple index out of range
#an expression resulting in an integer index
>>> tuple1[1+4]
12
#returns first element from right
>>> tuple1[-1]
12
```

10.1.2 Tuple is Immutable

Tuple is an immutable data type. It means that the elements of a tuple cannot be changed after it has been created. An attempt to do this would lead to an error.

```
>>> tuple1 = (1,2,3,4,5)
```

```
>>> tuple1[4] = 10
TypeError: 'tuple' object does not support
item assignment
```

However an element of a tuple may be of mutable type, e.g., a list.

```
#4th element of the tuple2 is a list
>>> tuple2 = (1,2,3,[8,9])
#modify the list element of the tuple tuple2
>>> tuple2[3][1] = 10
#modification is reflected in tuple2
>>> tuple2
(1, 2, 3, [8, 10])
```

- √ List is mutable but tuple is immutable. So iterating through a tuple is faster as compared to a list.
- √ If we have data that does not change then storing this data in a tuple will make sure that it is not changed accidentally.

10.2 Tuple Operations

10.2.1 Concatenation

Python allows us to join tuples using concatenation operator depicted by symbol +. We can also create a new tuple which contains the result of this concatenation operation.

```
>>> tuple1 = (1,3,5,7,9)
>>> tuple2 = (2,4,6,8,10)
>>> tuple1 + tuple2
#concatenates two tuples
(1, 3, 5, 7, 9, 2, 4, 6, 8, 10)
>>> tuple3 = ('Red','Green','Blue')
>>> tuple4 = ('Cyan', 'Magenta', 'Yellow'
,'Black')
#tuple5 stores elements of tuple3 and tuple4
>>> tuple5 = tuple3 + tuple4
>>> tuple5
('Red','Green','Blue','Cyan','Magenta',
'Yellow','Black')
```

Concatenation operator can also be used for extending an existing tuple. When we extend a tuple using concatenation a new tuple is created.

```
>>> tuple6 = (1,2,3,4,5)

#single element is appended to tuple6
>>> tuple6 = tuple6 + (6,)
>>> tuple6
(1, 2, 3, 4, 5, 6)

#more than one elements are appended
>>> tuple6 = tuple6 + (7,8,9)
>>> tuple6
(1, 2, 3, 4, 5, 6, 7, 8, 9)
```

10.2.2 Repetition

Repetition operation is depicted by the symbol *. It is used to repeat elements of a tuple. We can repeat the tuple elements. The repetition operator requires the first operand to be a tuple and the second operand to be an integer only.

```
>>> tuple1 = ('Hello','World')
>>> tuple1 * 3
('Hello', 'World', 'Hello', 'World', 'Hello',
'World')
#tuple with single element
>>> tuple2 = ("Hello",)
>>> tuple2 * 4
('Hello', 'Hello', 'Hello', 'Hello')
```

10.2.3 Membership

The `in` operator checks if the element is present in the tuple and returns `True`, else it returns `False`.

```
>>> tuple1 = ('Red','Green','Blue')
>>> 'Green' in tuple1
True
```

The `not in` operator returns `True` if the element is not present in the tuple, else it returns `False`.

```
>>> tuple1 = ('Red','Green','Blue')
>>> 'Green' not in tuple1
False
```

10.2.4 Slicing

Like string and list, slicing can be applied to tuples also.

```
#tuple1 is a tuple
>>> tuple1 = (10,20,30,40,50,60,70,80)

#elements from index 2 to index 6
>>> tuple1[2:7]
(30, 40, 50, 60, 70)

#all elements of tuple are printed
>>> tuple1[0:len(tuple1)]
(10, 20, 30, 40, 50, 60, 70, 80)

#slice starts from zero index
>>> tuple1[:5]
(10, 20, 30, 40, 50)

#slice is till end of the tuple
>>> tuple1[2:]
(30, 40, 50, 60, 70, 80)
```

```
#step size 2
>>> tuple1[0:len(tuple1):2]
(10, 30, 50, 70)
#negative indexing
>>> tuple1[-6:-4]
(30, 40)

#tuple is traversed in reverse order
>>> tuple1[::-1]
(80, 70, 60, 50, 40, 30, 20, 10)
```

10.3 TUPLE METHODS AND BUILT-IN FUNCTIONS

Python provides many functions to work on tuples. Table 10.1 list some of the commonly used tuple methods and built-in functions.

Table 10.1 Built-in functions and methods for tuples

Method	Description	Example

len()	Returns the length or the number of elements of the tuple passed as the argument	`>>> tuple1 = (10,20,30,40,50)`
		`>>> len(tuple1)`
		`5`

tuple()	Creates an empty tuple if no argument is passed
	
	Creates a tuple if a sequence is passed as argument
		`>>> tuple1 = tuple()`
		`>>> tuple1`
		`( )`
		`>>> tuple1 = tuple('aeiou')#string`
		`>>> tuple1`
		`('a', 'e', 'i', 'o', 'u')`
		
		`>>> tuple2 = tuple([1,2,3]) #list`
		`>>> tuple2`
		`(1, 2, 3)`
		
		`>>> tuple3 = tuple(range(5))`
		`>>> tuple3`
		`(0, 1, 2, 3, 4)`

count()	Returns the number of times the given element appears in the tuple	`>>> tuple1 = (10,20,30,10,40,10,50)`
		`>>> tuple1.count(10)`
		`3`
		`>>> tuple1.count(90)`
		`0`

index()	Returns the index of the first occurrence of the element in the given tuple	`>>> tuple1 = (10,20,30,40,50)`
		`>>> tuple1.index(30)`
		`2`
		`>>> tuple1.index(90)`
		`ValueError: tuple.index(x): x not in tuple`

sorted()	Takes elements in the tuple and returns a new sorted list. It should be noted that, sorted() does not make any change to the original tuple	>>> tuple1 = ("Rama","Heena","Raj", "Mohsin","Aditya")
		>>> sorted(tuple1)
		['Aditya', 'Heena', 'Mohsin', 'Raj', 'Rama']

min()	Returns minimum or smallest element of the tuple	>>> tuple1 = (19,12,56,18,9,87,34)
		>>> min(tuple1)
		9

max()	Returns maximum or largest element of the tuple	>>> max(tuple1) 87
sum()	Returns sum of the elements of the tuple	>>> sum(tuple1) 235

10.4 TUPLE ASSIGNMENT

Assignment of tuple is a useful feature in Python. It allows a tuple of variables on the left side of the assignment operator to be assigned respective values from a tuple on the right side. The number of variables on the left should be same as the number of elements in the tuple.

Example 10.2

```
#The first element 10 is assigned to num1 and
#the second element 20 is assigned to num2.
>>> (num1,num2) = (10,20)
>>> print(num1)
10
>>> print(num2)
20
>>> record = ( "Pooja",40,"CS")
>>> (name,rollNo,subject) = record
>>> name
'Pooja'
>>> rollNo
40
>>> subject
'CS'
>>> (a,b,c,d) = (5,6,8)
ValueError: not enough values to unpack
(expected 4, got 3)
```

If there is an expression on the right side then first that expression is evaluated and finally the result is assigned to the tuple.

Example 10.3

```
#15 is assigned to num3 and
#25 is assigned to num4
>>> (num3,num4) = (10+5,20+5)
>>> print(num3)
15
>>> print(num4)
25
```

\t is an escape character used for adding horizontal tab space. Another commonly used escape character is \n, used for inserting a new line.

10.5 Nested Tuples

A tuple inside another tuple is called a nested tuple. In the program 10-1, roll number, name and marks (in percentage) of students are saved in a tuple. To store details of many such students we can create a nested tuple.

Program 10-1 This is a program to create a nested tuple to store roll number, name and marks of students

```
#Program 10-1
#To store records of students in tuple and print them
st=((101,"Aman",98),(102,"Geet",95),(103,"Sahil",87),(104,"Pawan",79))
print("S_No"," Roll_No","   Name"," Marks")
for i in range(0,len(st)):
    print((i+1),'\t',st[i][0],'\t',st[i][1],'\t',st[i][2])
```

Output:

```
S_No   Roll_No   Name       Marks
1      101       Aman        98
2      102       Geet        95
3      103       Sahil       87
4      104       Pawan       79
```

10.6 Tuple Handling

Program 10-2 Write a program to swap two numbers without using a temporary variable.

```
#Program 10-2
#Program to swap two numbers
num1 = int(input('Enter the first number: '))
num2 = int(input('Enter the second number: '))
print("\nNumbers before swapping:")
print("First Number:",num1)
print("Second Number:",num2)
(num1,num2) = (num2,num1)
print("\nNumbers after swapping:")
```

```
print("First Number:",num1)
print("Second Number:",num2)
```

Output:

```
Enter the first number: 5
Enter the second number: 10

Numbers before swapping:
First Number: 5
Second Number: 10

Numbers after swapping:
First Number: 10
Second Number: 5
```

Program 10-3 Write a program to compute the area and circumference of a circle using a function.

```
#Program 10-3
#Function to compute area and circumference of the circle.
def circle(r):
    area = 3.14*r*r
    circumference = 2*3.14*r
    #returns a tuple having two elements area and circumference
    return (area,circumference)
#end of function

radius = int(input('Enter radius of circle: '))
area,circumference = circle(radius)
print('Area of circle is:',area)
print('Circumference of circle is:',circumference)
```

Output:

```
Enter radius of circle: 5
Area of circle is: 78.5
Circumference of circle is: 31.400000000000002
```

Program 10-4 Write a program to input *n* numbers from the user. Store these numbers in a tuple. Print the maximum and minimum number from this tuple.

```
#Program 10-4
#Program to input n numbers from the user. Store these numbers
#in a tuple. Print the maximum and minimum number from this tuple.

numbers = tuple()                #create an empty tuple 'numbers'
n = int(input("How many numbers you want to enter?: "))
for i in range(0,n):
    num = int(input())
    #it will assign numbers entered by user to tuple 'numbers'
```

```
    numbers = numbers +(num,)
print('\nThe numbers in the tuple are:')
print(numbers)
print("\nThe maximum number is:")
print(max(numbers))
print("The minimum number is:")
print(min(numbers))
```

Output:

```
How many numbers do you want to enter?: 5
9
8
10
12
15

The numbers in the tuple are:
(9, 8, 10, 12, 15)

The maximum number is:
15
The minimum number is:
8
```

10.7 Introduction to Dictionaries

The data type *dictionary* fall under mapping. It is a mapping between a *set of keys* and a *set of values*. The key-value pair is called an *item*. A key is separated from its value by a colon(:) and consecutive items are separated by commas. Items in dictionaries are unordered, so we may not get back the data in the same order in which we had entered the data initially in the dictionary.

10.7.1 Creating a Dictionary

To create a dictionary, the items entered are separated by commas and enclosed in curly braces. Each item is a key value pair, separated through colon (:). The keys in the dictionary must be unique and should be of any immutable data type, i.e., number, string or tuple. The values can be repeated and can be of any data type.

Example 10.4

```
#dict1 is an empty Dictionary created
#curly braces are used for dictionary
>>> dict1 = {}
>>> dict1
{}
#dict2 is an empty dictionary created using
#built-in function
```

NOTES

```
>>> dict2 = dict()
>>> dict2
{}
#dict3 is the dictionary that maps names
#of the students to respective marks in
#percentage
>>> dict3 = {'Mohan':95,'Ram':89,'Suhel':92,
'Sangeeta':85}
>>> dict3
{'Mohan': 95, 'Ram': 89, 'Suhel': 92,
'Sangeeta': 85}
```

10.7.2 Accessing Items in a Dictionary

We have already seen that the items of a sequence (string, list and tuple) are accessed using a technique called indexing. The items of a dictionary are accessed via the keys rather than via their relative positions or indices. Each key serves as the index and maps to a value.

The following example shows how a dictionary returns the value corresponding to the given key:

```
>>> dict3 = {'Mohan':95,'Ram':89,'Suhel':92,
'Sangeeta':85}
>>> dict3['Ram']
89
>>> dict3['Sangeeta']
85
#the key does not exist
>>> dict3['Shyam']
KeyError: 'Shyam'
```

In the above examples the key `'Ram'` always maps to the value 89 and key `'Sangeeta'` always maps to the value 85. So the order of items does not matter. If the key is not present in the dictionary we get KeyError.

10.8 DICTIONARIES ARE MUTABLE

Dictionaries are mutable which implies that the contents of the dictionary can be changed after it has been created.

10.8.1 Adding a new item

We can add a new item to the dictionary as shown in the following example:

```
>>> dict1 = {'Mohan':95,'Ram':89,'Suhel':92,
'Sangeeta':85}
```

NOTES

```
>>> dict1['Meena'] = 78
>>> dict1
{'Mohan': 95, 'Ram': 89, 'Suhel': 92,
'Sangeeta': 85, 'Meena': 78}
```

10.8.2 Modifying an Existing Item

The existing dictionary can be modified by just overwriting the key-value pair. Example to modify a given item in the dictionary:

```
>>> dict1 = {'Mohan':95,'Ram':89,'Suhel':92,
'Sangeeta':85}
#Marks of Suhel changed to 93.5
>>> dict1['Suhel'] = 93.5
>>> dict1
{'Mohan': 95, 'Ram': 89, 'Suhel': 93.5,
'Sangeeta': 85}
```

10.9 DICTIONARY OPERATIONS

10.9.1 Membership

The membership operator `in` checks if the key is present in the dictionary and returns `True`, else it returns `False`.

```
>>> dict1 = {'Mohan':95,'Ram':89,'Suhel':92,
'Sangeeta':85}
>>> 'Suhel' in dict1
True
```

The `not in` operator returns `True` if the key is not present in the dictionary, else it returns `False`.

```
>>> dict1 = {'Mohan':95,'Ram':89,'Suhel':92,
'Sangeeta':85}
>>> 'Suhel' not in dict1
False
```

10.10 TRAVERSING A DICTIONARY

We can access each item of the dictionary or traverse a dictionary using `for` loop.

```
>>> dict1 ={'Mohan':95,'Ram':89,'Suhel':92,
'Sangeeta':85}
```

Method 1

```
>>> for key in dict1:
        print(key,':',dict1[key])
Mohan: 95
Ram: 89
Suhel: 92
Sangeeta: 85
```

Method 2

```
>>> for key,value in dict1.items():
        print(key,':',value)
Mohan: 95
Ram: 89
Suhel: 92
Sangeeta: 85
```

10.11 DICTIONARY METHODS AND BUILT-IN FUNCTIONS

Python provides many functions to work on dictionaries. Table 10.2 lists some of the commonly used dictionary methods.

Table 10.2 Built-in functions and methods for dictionary

Method	Description	Example

len()	Returns the length or number of key: value pairs of the dictionary passed as the argument	`>>> dict1 = {'Mohan':95,'Ram':89, 'Suhel':92, 'Sangeeta':85}`
		`>>> len(dict1)`
		`4`

dict()	Creates a dictionary from a sequence of key-value pairs	`pair1 = [('Mohan',95),('Ram',89), ('Suhel',92),('Sangeeta',85)]`
		`>>> pair1`
		`[('Mohan', 95), ('Ram', 89), ('Suhel', 92), ('Sangeeta', 85)]`
		`>>> dict1 = dict(pair1)`
		`>>> dict1`
		`{'Mohan': 95, 'Ram': 89, 'Suhel': 92, 'Sangeeta': 85}`

keys()	Returns a list of keys in the dictionary	`>>> dict1 = {'Mohan':95, 'Ram':89, 'Suhel':92, 'Sangeeta':85}`
		`>>> dict1.keys()`
		`dict_keys(['Mohan', 'Ram', 'Suhel', 'Sangeeta'])`

values()	Returns a list of values in the dictionary	`>>> dict1 = {'Mohan':95, 'Ram':89, 'Suhel':92, 'Sangeeta':85}`
		`>>> dict1.values()`
		`dict_values([95, 89, 92, 85])`

items()	Returns a list of tuples(key – value) pair	`>>> dict1 = {'Mohan':95, 'Ram':89, 'Suhel':92, 'Sangeeta':85}`
		`>>> dict1.items()`
		`dict_items([( 'Mohan', 95), ('Ram', 89), ('Suhel', 92), ('Sangeeta', 85)])`

get()	Returns the value corresponding to the key passed as the argument
	
	If the key is not present in the dictionary it will return None
		>>> dict1 = {'Mohan':95, 'Ram':89, 'Suhel':92, 'Sangeeta':85}
		>>> dict1.get('Sangeeta')
		85
		>>> dict1.get('Sohan')
		>>>

update()	appends the key-value pair of the dictionary passed as the argument to the key-value pair of the given dictionary	>>> dict1 = {'Mohan':95, 'Ram':89, 'Suhel':92, 'Sangeeta':85}
		>>> dict2 = {'Sohan':79,'Geeta':89}
		>>> dict1.update(dict2)
		>>> dict1
		{'Mohan': 95, 'Ram': 89, 'Suhel': 92, 'Sangeeta': 85, 'Sohan': 79, 'Geeta': 89}
		>>> dict2
		{'Sohan': 79, 'Geeta': 89}

del()	Deletes the item with the given key
	To delete the dictionary from the memory we write:
	del Dict_name
		>>> dict1 = {'Mohan':95,'Ram':89, 'Suhel':92, 'Sangeeta':85}
		>>> del dict1['Ram']
		>>> dict1
		{'Mohan':95,'Suhel':92, 'Sangeeta': 85}
		
		>>> del dict1 ['Mohan']
		>>> dict1
		{'Suhel': 92, 'Sangeeta': 85}
		>>> del dict1
		>>> dict1
		NameError: name 'dict1' is not defined

clear()	Deletes or clear all the items of the dictionary	>>> dict1 = {'Mohan':95,'Ram':89, 'Suhel':92, 'Sangeeta':85}
		>>> dict1.clear()
		>>> dict1
		{ }

10.12 MANIPULATING DICTIONARIES

In this chapter, we have learnt how to create a dictionary and apply various methods to manipulate it. The following programs show the application of those manipulation methods on dictionaries.

Program 10-5 Create a dictionary 'ODD' of odd numbers between 1 and 10, where the key is the decimal number and the value is the corresponding number in words. Perform the following operations on this dictionary:

(a) Display the keys
(b) Display the values
(c) Display the items
(d) Find the length of the dictionary
(e) Check if 7 is present or not
(f) Check if 2 is present or not
(g) Retrieve the value corresponding to the key 9
(h) Delete the item from the dictionary corresponding to the key 9

```
>>> ODD = {1:'One',3:'Three',5:'Five',7:'Seven',9:'Nine'}
>>> ODD
{1: 'One', 3: 'Three', 5: 'Five', 7: 'Seven', 9: 'Nine'}
```

(a) Display the keys

```
>>> ODD.keys()
dict_keys([1, 3, 5, 7, 9])
```

(b) Display the values

```
>>> ODD.values()
dict_values(['One', 'Three', 'Five', 'Seven', 'Nine'])
```

(c) Display the items

```
>>> ODD.items()
dict_items([(1, 'One'), (3, 'Three'), (5, 'Five'), (7, 'Seven'),
(9, 'Nine')])
```

(d) Find the length of the dictionary

```
>>> len(ODD)
5
```

(e) Check if 7 is present or not

```
>>> 7 in ODD
True
```

(f) Check if 2 is present or not

```
>>> 2 in ODD
False
```

(g) Retrieve the value corresponding to the key 9

```
>>> ODD.get(9)
'Nine'
```

(h) Delete the item from the dictionary corresponding to the key 9

```
>>> del ODD[9]
>>> ODD
{1: 'One', 3: 'Three', 5: 'Five', 7: 'Seven'}
```

Program 10-6 Write a program to enter names of employees and their salaries as input and store them in a dictionary.

```
#Program 10-6
#Program to create a dictionary which stores names of the employee
#and their salary
num = int(input("Enter the number of employees whose data to be
stored: "))
count = 1
employee = dict()                    #create an empty dictionary
while count <= num:
     name = input("Enter the name of the Employee: ")
     salary = int(input("Enter the salary: "))
     employee[name] = salary
     count += 1
print("\n\nEMPLOYEE_NAME\tSALARY")
for k in employee:
     print(k,'\t\t',employee[k])
```

Output:

```
Enter the number of employees to be stored: 5
Enter the name of the Employee: 'Tarun'
Enter the salary: 12000
Enter the name of the Employee: 'Amina'
Enter the salary: 34000
Enter the name of the Employee: 'Joseph'
Enter the salary: 24000
Enter the name of the Employee: 'Rahul'
Enter the salary: 30000
Enter the name of the Employee: 'Zoya'
Enter the salary: 25000
EMPLOYEE_NAME    SALARY
'Tarun'          12000
'Amina'          34000
'Joseph'         24000
'Rahul'          30000
'Zoya'           25000
```

Program 10-7 Write a program to count the number of times a character appears in a given string.

```
#Program 10-7
#Count the number of times a character appears in a given string
```

```
st = input("Enter a string: ")
dic = {}                #creates an empty dictionary
for ch in st:
     if ch in dic:      #if next character is already in the dictionary
          dic[ch] += 1
     else:
          dic[ch] = 1   #if ch appears for the first time

for key in dic:
     print(key,':',dic[key])
```

Output:

```
Enter a string: HelloWorld
H : 1
e : 1
l : 3
o : 2
W : 1
r : 1
d : 1
```

Program 10-8 Write a function to convert a number entered by the user into its corresponding number in words. For example, if the input is 876 then the output should be 'Eight Seven Six'.

```
# Program 10-8
# Write a function to convert number into corresponding number in
# words
def convert(num):
     #numberNames is a dictionary of digits and corresponding number
     #names
     numberNames = {0:'Zero',1:'One',2:'Two',3:'Three',4:'Four',\
                    5:'Five',6:'Six',7:'Seven',8:'Eight',9:'Nine'}

     result = ''
     for ch in num:
          key = int(ch)              #converts character to integer
          value = numberNames[key]
          result = result + ' ' + value
     return result

num = input("Enter any number: ")      #number is stored as string
result = convert(num)
print("The number is:",num)
print("The numberName is:",result)
```

Output:

```
Enter any number: 6512
The number is: 6512
The numberName is: Six Five One Two
```

Summary

- Tuples are immutable sequences, i.e., we cannot change the elements of a tuple once it is created.
- Elements of a tuple are put in round brackets separated by commas.
- If a sequence has comma separated elements without parentheses, it is also treated as a tuple.
- Tuples are ordered sequences as each element has a fixed position.
- Indexing is used to access the elements of the tuple; two way indexing holds in dictionaries as in strings and lists.
- Operator '+' adds one sequence (string, list, tuple) to the end of other.
- Operator '*' repeats a sequence (string, list, tuple) by specified number of times
- Membership operator 'in' tells if an element is present in the sequence or not and 'not in' does the opposite.
- Tuple manipulation functions are: len(), tuple(), count(), index(), sorted(), min(), max(),sum().
- Dictionary is a mapping (non-scalar) data type. It is an unordered collection of key-value pair; key-value pair are put inside curly braces.
- Each key is separated from its value by a colon.
- Keys are unique and act as the index.
- Keys are of immutable type but values can be mutable.

Exercise

1. Consider the following tuples, tuple1 and tuple2:

```
tuple1 = (23,1,45,67,45,9,55,45)
tuple2 = (100,200)
```

Find the output of the following statements:

i. `print(tuple1.index(45))`
ii. `print(tuple1.count(45))`
iii. `print(tuple1 + tuple2)`
iv. `print(len(tuple2))`
v. `print(max(tuple1))`
vi `print(min(tuple1))`

Notes

NOTES

vii. `print(sum(tuple2))`

viii.
```
print(sorted(tuple1))
print(tuple1)
```

2. Consider the following dictionary stateCapital:

```
stateCapital = {"AndhraPradesh":"Hyderabad",
"Bihar":"Patna","Maharashtra":"Mumbai",
"Rajasthan":"Jaipur"}
```

Find the output of the following statements:

i. `print(stateCapital.get("Bihar"))`

ii. `print(stateCapital.keys())`

iii. `print(stateCapital.values())`

iv. `print(stateCapital.items())`

v. `print(len(stateCapital))`

vi `print("Maharashtra" in stateCapital)`

vii. `print(stateCapital.get("Assam"))`

viii.
```
del stateCapital["Rajasthan"]
print(stateCapital)
```

3. "Lists and Tuples are ordered". Explain.
4. With the help of an example show how can you return more than one value from a function.
5. What advantages do tuples have over lists?
6. When to use tuple or dictionary in Python. Give some examples of programming situations mentioning their usefulness.
7. Prove with the help of an example that the variable is rebuilt in case of immutable data types.
8. `TypeError` occurs while statement 2 is running. Give reason. How can it be corrected?

```
>>> tuple1 = (5)        #statement 1
>>> len(tuple1)         #statement 2
```

PROGRAMMING PROBLEMS

1. Write a program to read email IDs of n number of students and store them in a tuple. Create two new tuples, one to store only the usernames from the email IDs and second to store domain names from the email IDs. Print all three tuples at the end of the program. [**Hint:** You may use the function split()]
2. Write a program to input names of n students and store them in a tuple. Also, input a name from the user and find if this student is present in the tuple or not.

We can accomplish these by:

(a) writing a user defined function

(b) using the built-in function

3. Write a Python program to find the highest 2 values in a dictionary.

4. Write a Python program to create a dictionary from a string.

   Note: Track the count of the letters from the string.

   Sample string : 'w3resource'

   Expected output : {'3': 1, 's': 1, 'r': 2, 'u': 1, 'w': 1, 'c': 1, 'e': 2, 'o': 1}

5. Write a program to input your friends' names and their Phone Numbers and store them in the dictionary as the key-value pair. Perform the following operations on the dictionary:

   a) Display the name and phone number of all your friends

   b) Add a new key-value pair in this dictionary and display the modified dictionary

   c) Delete a particular friend from the dictionary

   d) Modify the phone number of an existing friend

   e) Check if a friend is present in the dictionary or not

   f) Display the dictionary in sorted order of names

Case Study-based Question

For the SMIS System given in Chapter 5, let us do the following:

Write a program to take in the roll number, name and percentage of marks for n students of Class X. Write user defined functions to

- accept details of the n students (n is the number of students)
- search details of a particular student on the basis of roll number and display result
- display the result of all the students
- find the topper amongst them
- find the subject toppers amongst them

(**Hint:** use Dictionary, where the key can be roll number and the value is an immutable data type containing name and percentage)

NOTES

Let's peer review the case studies of others based on the parameters given under "DOCUMENTATION TIPS" at the end of Chapter 5 and provide a feedback to them.

CASE STUDY-BASED QUESTIONS

1. A bank is a financial institution which is involved in borrowing and lending of money. With advancement in technology, online banking, also known as internet banking allows customers of a bank to conduct a range of financial transactions through the bank's website anytime, anywhere. As part of initial investigation you are suggested to
   - collect a bank's application form. After careful analysis of the form, identify the information required for opening a savings account. Also enquire about the rate of interest offered for a saving account.
   - The basic two operations performed on an account are Deposit and Withdrawal. Write a menu driven program that accepts either of the two choices of Deposit and Withdrawal, then accepts an amount, performs the transaction and accordingly displays the balance. Remember, every bank has a requirement of minimum balance which needs to be taken care of during withdrawal operations. Enquire about the minimum balance required in your bank.
   - Collect the interest rates for opening a fixed deposit in various slabs in a savings bank account. Remember, rates may be different for senior citizens.

   Finally, write a menu driven program having the following options (use functions and appropriate data types):
   - Open a savings bank account
   - Deposit money
   - Withdraw money
   - Take details, such as amount and period for a Fixed Deposit and display its maturity amount for a particular customer.
2. Participating in a quiz can be fun as it provides a competitive element. Some educational institutes use it as a tool to measure knowledge level, abilities

and/or skills of their pupils either on a general level or in a specific field of study. Identify and analyse popular quiz shows and write a Python program to create a quiz that should also contain the following functionalities besides the one identified by you as a result of your analysis.

- Create an administrative user ID and password to categorically add, modify, delete a question
- Register the student before allowing her or him to play a quiz
- Allow selection of category based on subject area
- Display questions as per the chosen category
- Keep the score as the participant plays
- Display the final score

3. Our heritage monuments are our assets. They are a reflection of our rich and glorious past and an inspiration for our future. UNESCO has identified some of Indian heritage sites as World heritage sites. Collect the following information about these sites:
   - What is the name of the site?
   - Where is it located?
     - District
     - State
   - When was it built?
   - Who built it?
   - Why was it built?
   - Website link (if any).

   Write a Python program to
   - create an administrative user ID and password to add, modify or delete an entered heritage site in the list of sites
   - display the list of world heritage sites in India
   - search and display information of a world heritage site entered by the user
   - display the name(s) of world heritage site(s) on the basis of the state input by the user.
4. Every mode of transport utilises a reservation system to ensure its smooth and efficient functioning. If you analyse you would find many things in common. You are required to identify

any one mode of transportation and prepare a reservation system for it. For example, let us look at the Railway reservation system we talked about earlier. The complex task of designing a good railway reservation system is seen as designing the different components of the system and then making them work with each other efficiently. Possible sub-systems are shown in Figure 1. Each of them may be modelled using functions.

Write a python code to automate the reservation needs of the identified mode of transport.

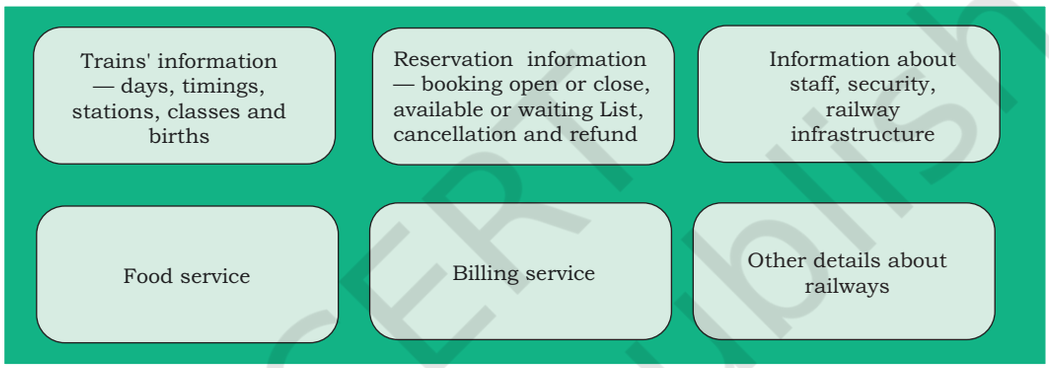


Figure 1: Railway reservation system

Chapter 11

Societal Impact

11120CH11

11.1 Introduction

"I think computer viruses should count as life. I think it says something about human nature that the only form of life we have created so far is purely destructive. We've created life in our own image."

– Stephen Hawking

In recent years, the world around us has seen a lot of changes due to use of 'Digital Technologies'. These changes have made a dramatic impact on our lives, making things more convenient, faster, and easier to handle. In the past, a letter would take days to reach, and every recipient would get his or her own copy and respond separately. Today, one can send and receive emails to more than one person at a time. The instantaneous nature of electronic communications has made us more efficient and productive.

From the banking industry to aviation, industrial production to e-commerce, especially with regard to the delivery of their goods and services, all are now dependent on the use of computers and digital technologies. Applications of digital technologies have redefined and evolved all spheres of human activities. Today more and more people are using digital technologies through smartphones, computers, etc., with the help of high speed Internet.

Why did the digital technologies become so widespread? The introduction of personal computers (PCs) and Internet followed by smartphones has brought these technologies to the common man.

While we reap the benefits of digital technologies, these technologies can also be misused. Let's look at the impact of these technologies on our society and the best practices that can ensure a productive and safe digital environment for us.

In this chapter

11.2 Digital Footprints

Have you ever searched online for any information? Have you ever purchased an online ticket, or responded to your friend's email, or checked the score of a

Think and Reflect

Can your digital footprints be used to judge your attitude and work ethics?

game online? Whenever we surf the Internet using smartphones, tablets, computers, etc., we leave a trail of data reflecting the activities performed by us online, which is our *digital footprint*.

Our digital footprint can be created and used with or without our knowledge. It includes websites we visit, emails we send, and any information we submit online, etc., along with the computer's IP address, location, and other device specific details. Such data could be used for targeted advertisement or could also be misused or exploited. Thus, it is good to be aware of the data trail we might be leaving behind. This awareness should make us cautious about what we write, upload or download or even browse online.

There are two kinds of digital footprints we leave behind. Active digital footprints which includes data that we intentionally submit online. This would include emails we write, or responses or posts we make on different websites or mobile Apps, etc. The digital data trail we leave online unintentionally is called passive digital footprints. This includes the data generated when we visit a website, use a mobile App, browse Internet, etc., as shown in Figure 11.1.

Figure 11.1: Exemplar web applications that result in digital footprints

Everyone who is connected to the Internet may have a digital footprint. With more usage, the trail grows. On examining the browser settings, we can find out how it stores our browsing history, cookies, passwords, auto fills, and many other types of data.

Besides browser, most of our digital footprints are stored in servers where the applications are hosted. We may not have access to remove or erase that data, neither do we have any control on how that data will be used. Therefore, once a data trail is generated, even if we later try to erase data about our online activities, the digital footprints still remain. There is no guarantee that digital footprints will be fully eliminated from the Internet. Therefore, we need to be more cautious while being online! All our online activities leave a data trace on the Internet as well as on the computing device that we use. This can be used to trace the user, his/her location, device and other usage details.

11.3 Digital Society and Netizen

Activity 11.1

As a digital citizen, list various services that you avail online.

As our society is inclined towards using more and more digital technologies, we end up managing most of our tasks digitally. In this era of digital society, our daily activities like communication, social networking, banking, shopping, entertainment, education, transportation, etc., are increasingly being driven by online transactions.

Digital society thus reflects the growing trend of using digital technologies in all spheres of human activities. But while online, all of us need to be aware of how to conduct ourselves, how best to relate with others and what ethics, morals and values to maintain. Anyone who uses digital technology along with Internet is a digital citizen or a netizen. Being a good netizen means practicing safe, ethical and legal use of digital technology. A responsible netizen must abide by net etiquettes, communication etiquettes and social media etiquettes.

11.3.1 Net Etiquettes

We follow certain etiquettes during our social interactions. Similarly, we need to exhibit proper manners and etiquettes while being online as shown in Figure 11.2. One should be ethical, respectful and responsible while surfing the Internet.

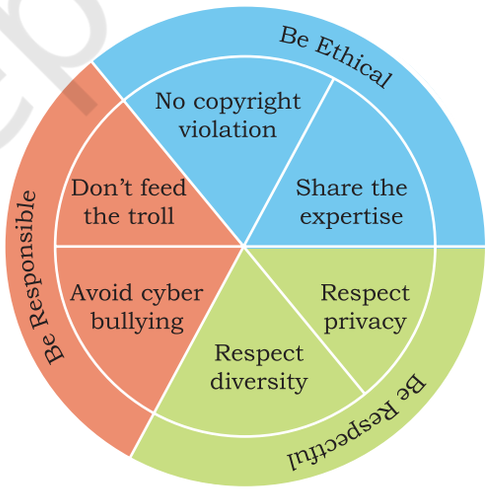


Figure 11.2: Net Etiquettes

(A) Be Ethical

- No copyright violation: we should not use copyrighted materials without the permission of the creator or owner. As an ethical digital citizen, we need to be careful while streaming audio or video or downloading images and files from the Internet. We will learn more about copyright in Section 11.4.
- Share the expertise: it is good to share information and knowledge on Internet so that others can access it. However, prior to sharing information, we need to be sure that we have sufficient knowledge on that topic. The information shared should be true and unambiguous. Also, in order to avoid

redundant information, we should verify that the information is not available already on Internet.

Remember!!

While surfing the Internet, we should be cautious about our personal and confidential data.

√ Think before sharing credentials with others on an online platform.

√ Keep personal information safe and protected through passwords.

(B) Be Respectful

- Respect privacy: as good digital citizens we have the right to privacy and the freedom of personal expression. At the same time, we have to understand that other digital citizens also have the same rights and freedoms. Our personal communication with a digital citizen may include images, documents, files, etc., that are private to both. We should respect this privacy and should not share those images, documents, files, etc., with any other digital citizen without each others' consent.
- Respect diversity: in a group or public forum, we should respect the diversity of the people in terms of knowledge, experience, culture and other aspects.

(C) Be Responsible

- Avoid cyber bullying: any insulting, degrading or intimidating online behaviour like repeated posting of rumours, giving threats online, posting the victim's personal information, sexual harassment or comments aimed to publicly ridicule a victim is termed as cyber bullying. It implies repeatedly targeting someone with intentions to hurt or embarrass. Perhaps new or non-frequent users of the Internet feel that things done online have no effect in the real world. We need to realise that bullying online can have very serious implications on the other person (victim). Also, remember our actions can be traced back using our digital footprints.
- Don't feed the troll: an Internet troll is a person who deliberately sows discord on the Internet by starting quarrels or upsetting people, by posting inflammatory or off topic messages in an online community, just for amusement. Since trolls thrive on attention, the best way to discourage trolls is not to pay any attention to their comments.

Activity 11.2

Find out how to report about an abusive or inappropriate post or about a sender in a social network?

11.3.2 Communication Etiquettes

Digital communication includes email, texting, instant messaging, talking on the cell phone, audio or video

conferencing, posting on forums, social networking sites, etc. All these are great ways to connect with people in order to exchange ideas, share data and knowledge. Good communication over email, chat room and other such forums require a digital citizen to abide by the communication etiquettes as shown in Figure 11.3.

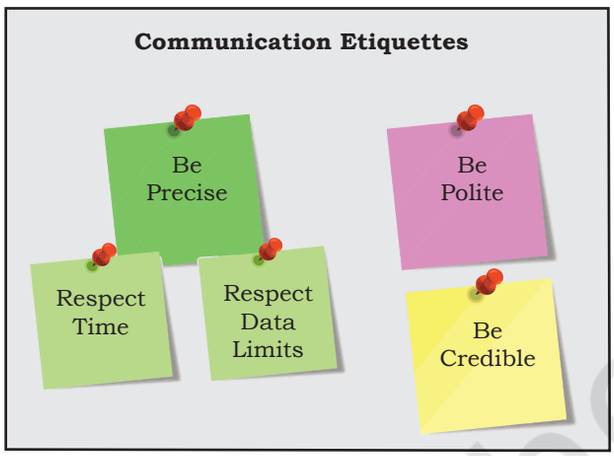


Figure 11.3: Communication etiquettes

(A) Be Precise

- Respect time: we should not waste precious time in responding to unnecessary emails or comments unless they have some relevance for us. Also, we should not always expect an instant response as the recipient may have other priorities.
- Respect data limits: For concerns related to data and bandwidth, very large attachments may be avoided. Rather send compressed files or link of the files through cloud shared storage like Google Drive, Microsoft OneDrive, Yahoo Dropbox, etc.

(B) Be Polite

Whether the communication is synchronous (happening in real time like chat, audio/video calls) or asynchronous (like email, forum post or comments), we should be polite and non-aggressive in our communication. We should avoid being abusive even if we don't agree with others' point of view.

(C) Be Credible

We should be cautious while making a comment, replying or writing an email or forum post as such acts decide our credibility over a period of time. That is how we decide to follow some particular person's forum posts while ignoring posts of other members of the forum. On various discussion forums, we usually try to go through the previous comments of a person and judge their credibility before relying on that person's comments.

Avoid Spam!!

On receiving junk email (called Spam), neither reply nor open any attachment in such email.

No Permanent Deletion!!

We can post or comment anything on Internet, and delete it later.

√ But remember, it cannot be permanently deleted. It is recorded in our Digital Footprint.

√ This is how many culprits who spread hate, bully others or engage in criminal activities are traced and apprehended.

11.3.3 Social Media Etiquettes

In the current digital era, we are familiar with different kinds *social media* and we may have an account on Facebook, Google+, Twitter, Instagram, Pinterest, or the YouTube channel. Social media are websites or

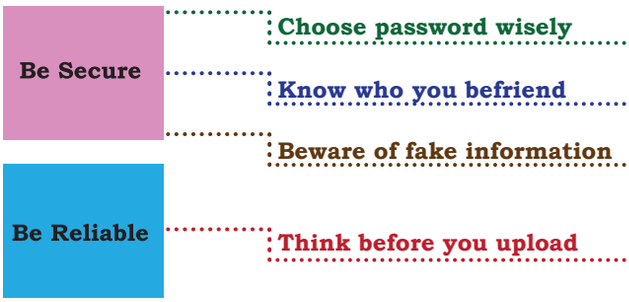


Figure 11.4: Social media etiquettes

applications that enable their users to participate in social networking by creating and sharing content with others in the community. These platforms encourage users to share their thoughts and experiences through posts or pictures. In this way users can interact with other online users of those social media apps or channels. This is why the impact and outreach of social media has grown exponentially. It has begun to shape the outcome of politics, business, culture, education and more. In social media too, there are certain etiquettes we need to follow as shown in Figure 11.4.

Don't Meet Up!!

√ Never arrange to meet an online friend because it may not be safe.

√ No matter how genuine someone is appearing online, they might be pretending and hiding their real identity.

Think and Reflect

Is having the same password for all your accounts on different websites safe?

(A) Be Secure

- Choose password wisely: it is vital for social network users. News of breaching or leakage of user data from social network often attracts headlines. Users should be wary of such possibilities and must know how to safeguard themselves and their accounts. The minimum one can do is to have strong and frequently changed password. Never share personal credentials like username and password with others.
- Know who you befriend: social networks usually encourage connecting with users (making friends), sometime even those whom we don't know or have not met. However, we need to be careful while befriending unknown people as their intentions possibly could be malicious and unsafe.
- Beware of fake information: fake news, messages and posts are common in social networks. As a user, we should be aware of them. With experience, we should be able to figure out whether a news, message or post is genuine or fake. Thus, we should not blindly believe in everything that we come across on such platforms, we should apply our knowledge and experience to validate such news, message or post.

Play Safe!!

Think carefully before sharing personal photos.

(B) Be Reliable

- Think before uploading: we can upload almost anything on social network. However, remember that once uploaded, it is always there in the remote server even if we delete the files. Hence we

need to be cautious while uploading or sending sensitive or confidential files which have a bearing on our privacy.

Activity 11.3

Supose someone's email password is 'tecnnology' which is weak. Can you suggest a stronger password?

11.4 Data Protection

In this digital age, data or information protection is mainly about the privacy of data stored digitally. Elements of data that can cause substantial harm, embarrassment, inconvenience and unfairness to an individual, if breached or compromised, is called sensitive data. Examples of sensitive data include biometric information, health information, financial information, or other personal documents, images or audios or videos. Privacy of sensitive data can be implemented by encryption, authentication, and other secure methods to ensure that such data is accessible only to the authorised user and is for a legitimate purpose.

All over the world, each country has its own data protection policies (laws). These policies are legal documents that provide guidelines to the user on processing, storage and transmission of sensitive information. The motive behind implementation of these policies is to ensure that sensitive information is appropriately protected from modification or disclosure.

Think and Reflect

Why should we always mention the source from which we got an idea or used resources (text, image, audio, video, etc.) to prepare a project or a writeup?

11.4.1 Intellectual Property Right (IPR)

When someone owns a house or a motorcycle, we say that the person owns that property. Similarly, if someone comes out with a new idea, this original idea is that person's intellectual property. Intellectual Property refers to the inventions, literary and artistic expressions, designs and symbols, names and logos. The ownership of such concepts lies with the creator, or the holder of the intellectual property. This enables the creator or copyright owner to earn recognition or financial benefit by using their creation or invention. Intellectual Property is legally protected through copyrights, patents, trademarks,etc.

(A) Copyright

Copyright grants legal rights to creators for their original works like writing, photograph, audio recordings, video, sculptures, architectural works, computer software, and other creative works like literary and artistic work.

Executing IPR: say for a software

- √ Code of the software will be protected by a copyright
- √ Functional expression of the idea will be protected by a patent
- √ The name and logo of the software will come under a registered trademark

Copyrights are automatically granted to creators and authors. Copyright law gives the copyright holder a set of rights that they alone can avail legally. The rights include right to copy (reproduce) a work, right to create derivative works based upon it, right to distribute copies of the work to the public, and right to publicly display or perform the work. It prevents others from copying, using or selling the work. For example, writer Rudyard Kipling holds the copyright to his novel, 'The Jungle Book', which tells the story of Mowgli, the jungle boy. It would be an infringement of the writer's copyright if someone used parts of the novel without permission. To use other's copyrighted material, one needs to obtain a license from them.

(B) Patent

A patent is usually granted for inventions. Unlike copyright, the inventor needs to apply (file) for patenting the invention. When a patent is granted, the owner gets an exclusive right to prevent others from using, selling, or distributing the protected invention. Patent gives full control to the patentee to decide whether or how the invention can be used by others. Thus it encourages inventors to share their scientific or technological findings with others. A patent protects an invention for 20 years, after which it can be freely used. Recognition and/or financial benefit foster the right environment, and provide motivation for more creativity and innovation.

(C) Trademark

Trademark includes any visual symbol, word, name, design, slogan, label, etc., that distinguishes the brand or commercial enterprise, from other brands or commercial enterprises. For example, no company other than Nike can use the Nike brand to sell shoes or clothes. It also prevents others from using a confusingly similar mark, including words or phrases. For example, confusing brands like "Nikke" cannot be used. However, it may be possible to apply for the Nike trademark for unrelated goods like notebooks.

11.4.2 Violation of IPR

Violation of intellectual property right may happen in one of the following ways:

(A) Plagiarism

With the availability of Internet, we can instantly copy or share text, pictures and videos. Presenting someone else's idea or work as one's own idea or work is called plagiarism. If we copy some contents from Internet, but do not mention the source or the original creator, then it is considered as an act of plagiarism. Further, if someone derives an idea or a product from an already existing idea or product, but instead presents it a new idea, then also it is plagiarism. It is a serious ethical offense and sometimes considered as an act of fraud. Even if we take contents that are open for public use, we should cite the author or source to avoid plagiarism.

(B) Copyright Infringement

Copyright infringement is when we use other person's work without obtaining their permission to use or we have not paid for it, if it is being sold. Suppose we download an image from the Internet and use it in our project. But if the owner of the copyright of the image does not permit its free usage, then using such an image even after giving reference of the image in our project is a violation of copyright. Just because it is on the Internet, does not mean that it is free for use. Hence, check the copyright status of writer's work before using it to avoid plagiarism.

(C) Trademark Infringement

Trademark Infringement means unauthorised use of other's trademark on products and services. An owner of a trademark may commence legal proceedings against someone who infringes its registered trademark.

11.4.3 Public Access and Open Source Software

Copyright sometimes put restriction on the usage of the copyrighted works by anyone else. If others are allowed to use and built upon the existing work, it will encourage collaboration and would result in new innovations in the same direction. Licenses provide rules and guidelines for others to use the existing work. When authors share their copyrighted works with others under public license, it allows others to use and even modify the content. Open source licenses help others to contribute to existing work or project without seeking special individual permission to do so.

Activity 11.4

Explore the follouing websites to know about open/public licensing:

(i) creativecommons.org for cc, and

(ii) gnu.org for GNU GPL

Beware!!

- √ Plagiarism means using other's work and not giving adequate citation for use.
- √ Copyright infringement means using another person's work, without permission or without paying for it, if it is being sold.

The GNU General public license (GPL) and the Creative Commons (CC) are two popular categories of public licenses. CC is used for all kind of creative works like websites, music, film, literature, etc. CC enables the free distribution of an otherwise copyrighted work. It is used when an author wants to give people the right to share, use and build upon a work that they have created. GPL is primarily designed for providing public licence to a software. GNU GPL is another free software license, which provides end users the freedom to run, study, share and modify the software, besides getting regular updates.

Users or companies who distribute GPL license works may charge a fee for copies or give them free of charge. This distinguishes the GPL license from freeware software licenses like Skype, Adobe Acrobat reader, etc. that allow copying for personal use but prohibit commercial distribution, or proprietary licenses where copying is prohibited by copyright law.

Many of the proprietary software that we use are sold commercially and their program code (source code) are not shared or distributed. However, there are certain software available freely for anyone and their source code is also open for anyone to access, modify, correct and improve. Free and open source software (FOSS) has a large community of users and developers who are contributing continuously towards adding new features or improving the existing features. For example, Linux kernel-based operating systems like Ubuntu and Fedora come under FOSS. Some of the popular FOSS tools are office packages, like Libre Office, browser like Mozilla Firefox, etc.

Software piracy is the unauthorised use or distribution of software. Those who purchase a license for a copy of the software do not have the rights to make additional copies without the permission of the copyright owner. It amounts to copyright infringement regardless of whether it is done for sale, for free distribution or for copier's own use. One should avoid software piracy. Using a pirated software not only degrades the performance of a computer system, but also affects the software industry which in turn affects the economy of a country.

Remember

- √ CC licenses are a set of copyright licenses that give the recipients, rights to copy, modify and redistribute the creative material, but giving the authors, the liberty to decide the conditions of licensing.
- √ GPL is the most widely used free software license which grants the recipients, rights to copy, modify and redistribute the software and that the same rights are preserved in all derivative works.

11.5 Cyber Crime

Criminal activities or offences carried out in a digital environment can be considered as cyber crime. In such crimes, either the computer itself is the target or the computer is used as a tool to commit a crime. Cyber crimes are carried out against either an individual, or a group, or an organisation or even against a country, with the intent to directly or indirectly cause physical harm, financial loss or mental harassment. A cyber criminal attacks a computer or a network to reach other computers in order to disable or damage data or services. Apart from this, a cyber criminal may spread viruses and other malwares in order to steal private and confidential data for blackmailing and extortion. A computer virus is some lines of malicious code that can copy itself and can have detrimental effect on the computers, by destroying data or corrupting the system. Similarly, malware is a software designed to specifically gain unauthorised access to computer systems. The nature of criminal activities are alarmingly increasing day-by-day, with frequent reports of hacking, ransomware attacks, denial-of-service, phishing, email fraud, banking fraud and identity theft.

Remember!!

Cyber crime is defined as a crime in which computer is the medium of crime (hacking, phishing, spamming), or the computer is used as a tool to commit crimes (extortion, data breaches, theft).

11.5.1 Hacking

Hacking is the act of unauthorised access to a computer, computer network or any digital system. Hackers usually have technical expertise of the hardware and software. They look for bugs to exploit and break into the system.

Hacking, when done with a positive intent, is called ethical hacking. Such ethical hackers are known as white hat hackers. They are specialists in exploring any vulnerability or loophole during testing of the software. Thus, they help in improving the security of a software. An ethical hacker may exploit a website in order to discover its security loopholes or vulnerabilities. He then reports his findings to the website owner. Thus, ethical hacking is actually preparing the owner against any cyber attack.

A non-ethical hacker is the one who tries to gain unauthorised access to computers or networks in order to steal sensitive data with the intent to damage or bring down systems. They are called black hat hackers

Activity 11.5

How can you unsubscribe from a mail group or block an email sender?

Beware !!

Accepting links from untrusted emails can be hazardous, as they may potentially contain a virus or link to malicious website. We should ensure to open any email link or attachment only when it is from a trusted source and doesn't look doubtful.

or crackers. Their primary focus is on security cracking and data stealing. They use their skill for illegal or malicious purposes. Such hackers try to break through system securities for identity theft, monetary gain, to bring a competitor or rival site down, to leak sensitive information, etc.

11.5.2 Phishing and Fraud Emails

Phishing is an unlawful activity where fake websites or emails that look original or authentic are presented to the user to fraudulently collect sensitive and personal details, particularly usernames, passwords, banking and credit card details. The most common phishing method is through email spoofing where a fake or forged email address is used and the user presumes it to be from an authentic source. So you might get an email from an address that looks similar to your bank or educational institution, asking for your information, but if you look carefully you will see their URL address is fake. They will often use logo's of the original, making them difficult to detect from the real! Phishing attempts through phone calls or text messages are also common these days.

(A) Identity Theft

Identity thieves increasingly use personal information stolen from computers or computer networks, to commit fraud by using the data gained unlawfully. A user's identifiable personal data like demographic details, email ID, banking credentials, passport, PAN, Aadhaar number and various such personal data are stolen and misused by the hacker on behalf of the victim. This is one type of phishing attack where the intention is largely for monetary gain. There can be many ways in which the criminal takes advantage of an individual's stolen identity. Given below are a few examples:

- Financial identity theft: when the stolen identity is used for financial gain.
- Criminal identity theft: criminals use a victim's stolen identity to avoid detection of their true identity.
- Medical identity theft: criminals can seek medical drugs or treatment using a stolen identity.

11.5.3 Ransomware

This is another kind of cyber crime where the attacker gains access to the computer and blocks the user from accessing, usually by encrypting the data. The attacker blackmails the victim to pay for getting access to the data, or sometimes threaten to publish personal and sensitive information or photographs unless a ransom is paid.

Ransomware can get downloaded when the users visit any malicious or unsecure websites or download software from doubtful repositories. Some ransomware are sent as email attachments in spam mails. It can also reach our system when we click on a malicious advertisement on the Internet.

Activity 11.6

Explore and find out how to file a complaint with the cyber cell in your area.

11.5.4 Combatting and Preventing Cyber Crime

The challenges of cyber crime can be mitigated with the twin approach of being alert and taking legal help. Following points can be considered as safety measures to reduce the risk of cyber crime:

- Take regular backup of important data
- Use an antivirus software and keep it updated always
- Avoid installing pirated software. Always download software from known and secure (HTTPS) sites
- Always update the system software which include the Internet browser and other application software
- Do not visit or download anything from untrusted websites
- Usually the browser alerts users about doubtful websites whose security certificate could not be verified; avoid visiting such sites
- Use strong password for web login, and change it periodically. Do not use same password for all the websites. Use different combinations of alphanumeric characters including special characters. Ignore common words or names in password
- While using someone else's computer, don't allow browser to save password or auto fill data, and try to browse in your private browser window

- For an unknown site, do not agree to use cookies when asked for, through a Yes/No option.
- Perform online transaction like shopping, ticketing, and other such services only through well-known and secure sites
- Always secure wireless network at home with strong password and regularly change it.

11.6 INDIAN INFORMATION TECHNOLOGY ACT (IT ACT)

Digital signatures are the digital equivalent of a paper certificate. Digital signatures work on a unique digital ID issued by a Certified Authority (CA) to the user. Signing a document digitally means attaching that user's identity which can be used to authenticate.

A licensed CA who has been granted a license to issue it under section 24 of the indian IT-Act 2000, can issue the digital signature.

With the growth of Internet, many cases of cyber crimes, frauds, cyber attacks and cyber bullying are reported. The nature of fraudulent activities and crimes keeps changing. To deal with such menaces, many countries have come up with legal measures for protection of sensitive personal data and to safeguard the rights of Internet users. The Government of India's Information Technology Act, 2000 (also known as IT Act), amended in 2008, provides guidelines to the user on the processing, storage and transmission of sensitive information. In many Indian states, there are cyber cells in police stations where one can report any cyber crime. The act provides legal framework for electronic governance by giving recognition to electronic records and digital signatures. The act outlines cyber crimes and penalties for them.

Cyber Appellate Tribunal has been established to resolve disputes arising from cyber crime, such as tampering with computer source documents, hacking the computer system, using password of another person, publishing sensitive personal data of others without their consent, etc. The act is needed so that people can perform transactions over the Internet through credit cards without fear of misuse. Not only people, the act empowers government departments also to accept filing, creation and storage of official documents in the digital format.

Think and Reflect

Do you follow precautions to stay healthy — physically, mentally as well as emotionally while using digital technologies?

11.7 IMPACT ON HEALTH

As digital technologies have penetrated into different fields, we are spending more time in front of screens, be it mobile, laptop, desktop, television, gaming console,

music or sound device. But interacting in an improper posture can be bad for us — both physically, and mentally. Besides, spending too much time on Internet can be addictive and can have a negative impact on our physical and psychological well being.

However, these health concerns can be addressed to some extent by taking care of the way we position such devices and the way we position our posture. Ergonomics is a branch of science that deals with designing or arranging workplaces including the furniture, equipments and systems so that it becomes safe and comfortable for the user. Ergonomics helps us in reducing the strain on our bodies — including the fatigue and injuries due to prolonged use.

When we continuously look at the screen for watching, typing, chatting or playing games, our eyes are continuously exposed to the glare coming from the screens. Looking at small handheld devices makes it worse. Eye strain is a symptom commonly complained by users of digital devices.

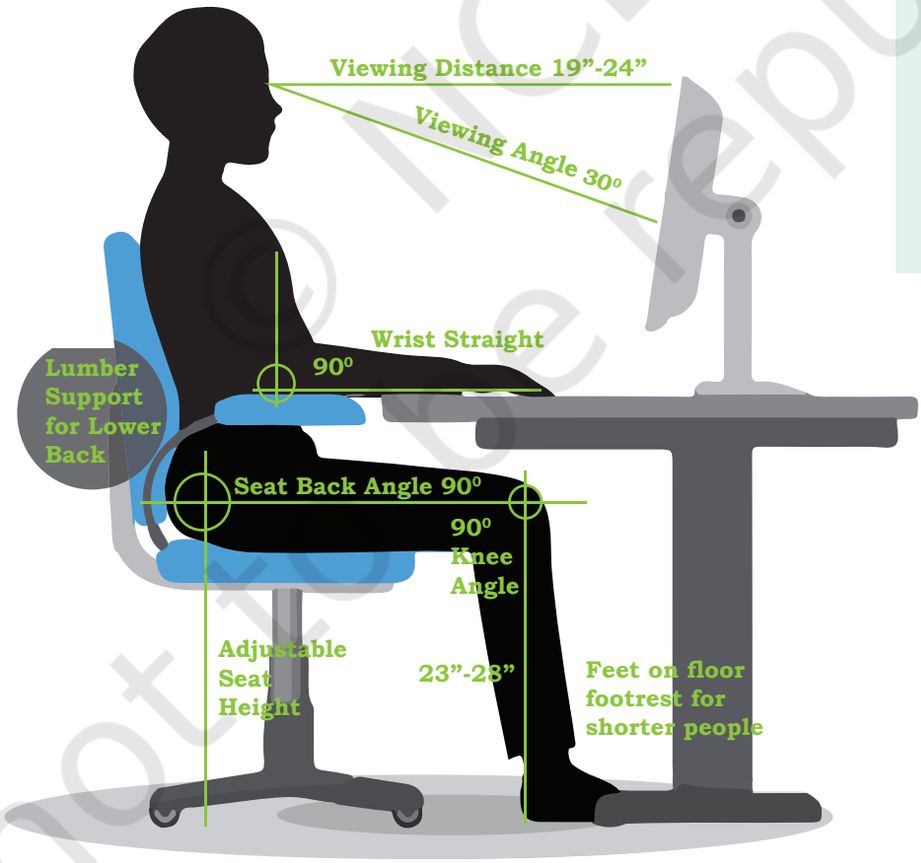


Figure 11.5: Correct posture while sitting in front of a computer

Device Safety: Ensures Good Health of a Computer System

- √ Regularly clean it to keep the dust off. Use a liquid solution specifically formulated for the cleaning of electronic screens.
- √ Wipe monitor's screen often using the regular microfibre soft cloth (the one used for spectacles).
- √ Keep it away from direct heat, sunlight and put it in a room with enough ventilation for air circulation.
- √ Do not eat food or drink over the keyboard. Food crumbs that fall into the gaps between the keys or spilled over liquid can cause issues to the devices.

Maintain a Balance!!

Enjoy the exciting world of digital devices in tandem with other pursuits of thrilling sports and hobbies. Online friends are good, but spending time with friends in real life is very fulfilling. Often the wholesome nature of real interactions cannot be compared to just online social networking.

Ergonomically maintaining the viewing distance and angle, along with the position can be of some help. Figure 11.5 shows the posture to be maintained in order to avoid fatigue caused due to prolonged use of computer system and other digital devices. However, to get rid of dry, watering, or itchy eyes, it is better to periodically focus on distant objects, and take a break for outdoor activities.

Bad posture, backaches, neck and shoulder pains can be prevented by arranging the workspace as recommended by ergonomics. Overuse of keyboards (be it physical keyboard or touchscreen-based virtual keyboard) not aligned ergonomically, can give rise to a painful condition of wrists and fingers, and may require medical help in the long run.

Stress, physical fatigue and obesity are the other related impacts the body may face if one spends too much time using digital devices.

SUMMARY

- Digital footprint is the trail of data we leave behind when we visit any website (or use any online application or portal) to fill-in data or perform any transaction.
- A user of digital technology needs to follow certain etiquettes like net-etiquettes, communication-etiquettes and social media-etiquettes.
- Net-etiquette includes avoiding copyright violations, respecting privacy and diversity of users, and avoiding cyber bullies and cyber trolls, besides sharing of expertise.
- Communication-etiquette requires us to be precise and polite in our conversation so that we remain credible through our remarks and comments.
- While using social media, one needs to take care of security through password, be aware of fake information and be careful while befriending unknowns. Care must be taken while sharing anything on social media as it may create havoc if being mishandled, particularly our personal, sensitive information.
- Intellectual Property Rights (IPR) help in data protection through copyrights, patents and trademarks. There are both ethical and legal

aspects of violating IPR. A good digital citizen should avoid plagiarism, copyright infringement and trademark infringement.

- Certain software are made available for free public access. Free and Open Source Software (FOSS) allow users to not only access but also to modify (or improve) them.
- Cyber crimes include various criminal activities carried out to steal data or to break down important services. These include hacking, spreading viruses or malware, sending phishing or fraudulent emails, ransomware, etc.
- Excessive usage of digital devices has a negative impact on our physical as well as psychological well-being. Ergonomic positioning of devices as well as our posture are important.

Exercise

1. After practicals, Atharv left the computer laboratory but forgot to sign off from his email account. Later, his classmate Revaan started using the same computer. He is now logged in as Atharv. He sends inflammatory email messages to few of his classmates using Atharv's email account. Revaan's activity is an example of which of the following cyber crime? Justify your answer.
   a) Hacking
   b) Identity theft
   c) Cyber bullying
   d) Plagiarism
2. Rishika found a crumpled paper under her desk. She picked it up and opened it. It contained some text which was struck off thrice. But she could still figure out easily that the struck off text was the email ID and password of Garvit, her classmate. What is ethically correct for Rishika to do?
   a) Inform Garvit so that he may change his password.
   b) Give the password of Garvit's email ID to all other classmates.
   c) Use Garvit's password to access his account.
3. Suhana is down with fever. So she decided not to go to school tomorrow. Next day, in the evening she called up her classmate, Shaurya and enquired

NOTES

about the computer class. She also requested him to explain the concept. Shaurya said, "Mam taught us how to use tuples in python". Further, he generously said, "Give me some time, I will email you the material which will help you to understand tuples in python". Shaurya quickly downloaded a 2-minute clip from the Internet explaining the concept of tuples in python. Using video editor, he added the text "Prepared by Shaurya" in the downloaded video clip. Then, he emailed the modified video clip to Suhana. This act of Shaurya is an example of:

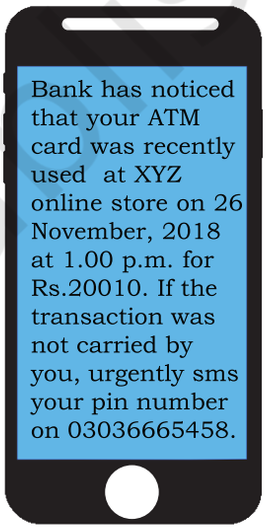
a) Fair use
b) Hacking
c) Copyright infringement
d) Cyber crime

4. After a fight with your friend, you did the following activities. Which of these activities is not an example of cyber bullying?
   a) You sent an email to your friend with a message saying that "I am sorry".
   b) You sent a threatening message to your friend saying "Do not try to call or talk to me".
   c) You created an embarrassing picture of your friend and uploaded on your account on a social networking site.
5. Sourabh has to prepare a project on "Digital India Initiatives". He decides to get information from the Internet. He downloads three web pages (webpage 1, webpage 2, webpage 3) containing information on Digital India Initiatives. Which of the following steps taken by Sourabh is an example of plagiarism or copyright infringement. Give justification in support of your answer.
   a) He read a paragraph on " Digital India Initiatives" from webpage 1 and rephrased it in his own words. He finally pasted the rephrased paragraph in his project.
   b) He downloaded three images of " Digital India Initiatives" from webpage 2. He made a collage for his project using these images.
   c) He downloaded "Digital India Initiative" icon from web page 3 and pasted it on the front page of his project report.

6. Match the following:

Column A	Column B
Plagiarism	Fakers, by offering special rewards or money prize asked for personal information, such as bank account information
Hacking	Copy and paste information from the Internet into your report and then organise it
Credit card fraud	The trail that is created when a person uses the Internet.
Digital Foot Print	Breaking into computers to read private emails and other files

7. You got the below shown SMS from your bank querying a recent transaction. Answer the following:

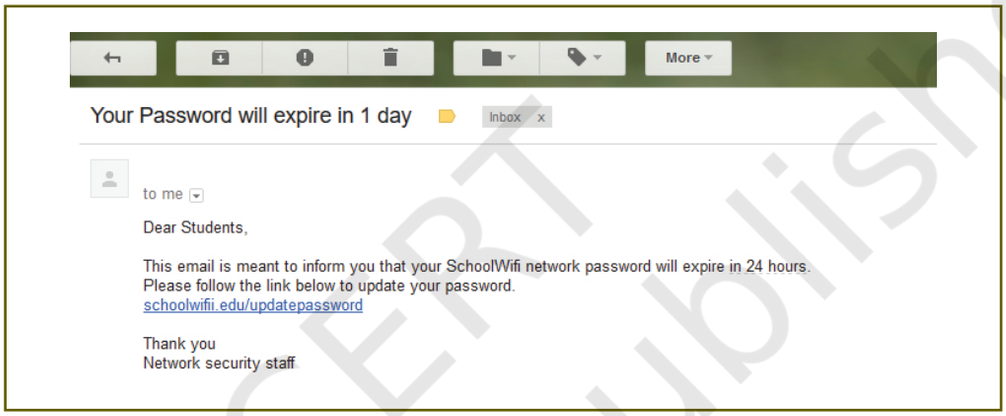


   a) Will you SMS your pin number to the given contact number?
   b) Will you call the bank helpline number to recheck the validity of the SMS received?

8. Preeti celebrated her birthday with her family. She was excited to share the moments with her friend Himanshu. She uploaded selected images of her birthday party on a social networking site so that Himanshu can see them. After few days, Preeti had a fight with Himanshu. Next morning, she deleted her birthday photographs from that social networking site, so that Himanshu cannot access them. Later in the evening, to her surprise, she saw that one of the images which she had already deleted from the social networking site was available with their common friend Gayatri. She hurriedly enquired Gayatri "Where did you get this picture from?". Gayatri replied "Himanshu forwarded this image few minutes back".

   Help Preeti to get answers for the following questions. Give justification for your answers so that Preeti can understand it clearly.
   a) How could Himanshu access an image which I had already deleted?
   b) Can anybody else also access these deleted images?
   c) Had these images not been deleted from my digital footprint?

9. The school offers wireless facility (wifi) to the Computer Science students of Class XI. For communication, the network security staff of the school have a registered URL schoolwifi.edu. On 17 September 2017, the following email was mass distributed to all the Computer Science students of Class XI. The email claimed that the password of the students was about to expire. Instructions were given to go to URL to renew their password within 24 hours.



   a) Do you find any discrepancy in this email?
   b) What will happen if the student will click on the given URL?
   c) Is the email an example of cyber crime? If yes, then specify which type of cyber crime is it. Justify your answer.

10. You are planning to go for a vacation. You surfed the Internet to get answers for the following queries:
   a) Weather conditions
   b) Availability of air tickets and fares
   c) Places to visit
   d) Best hotel deals

   Which of your above mentioned actions might have created a digital footprint?

11. How would you recognise if one of your friends is being cyber bullied?
   a) Cite the online activities which would help you detect that your friend is being cyber bullied?
   b) What provisions are in IT Act 2000, (amended in 2008) to combact such situations.

12. Write the differences between the following-
   a) Copyrights and Patents

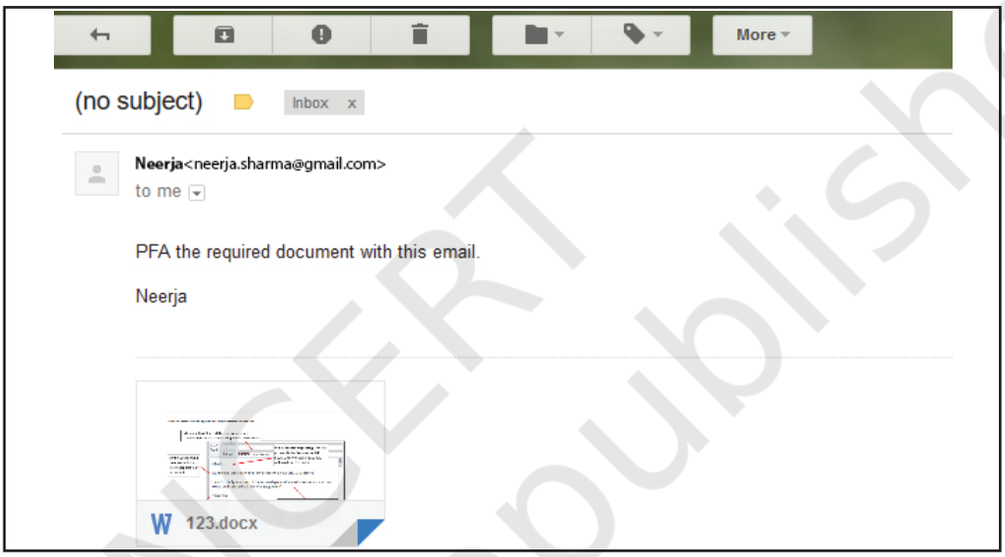
b) Plagiarism and Copyright infringement
c) Non-ethical hacking and Ethical hacking
d) Active and Passive footprints
e) Free software and Free and open source software

13. If you plan to use a short text from an article on the web, what steps must you take in order to credit the sources used?
14. When you search online for pictures, how will you find pictures that are available in the free public domain. How can those pictures be used in your project without copyright violations?
15. Describe why it is important to secure your wireless router at home. Search the Internet to find the rules to create a reasonably secure password. Create an imaginary password for your home router. Will you share your password for home router with following people. Justify your answer.
    a) Parents
    b) Friends
    c) Neighbours
    d) Home Tutors
16. List down the steps you need to take in order to ensure
    a) your computer is in good working condition for a longer time.
    b) smart and safe Internet surfing.
17. What is data privacy? Websites that you visit collect what type of information about you?
18. In the computer science class, Sunil and Jagdish were assigned the following task by their teacher.
    a) Sunil was asked to find information about "India, a Nuclear power". He was asked to use Google Chrome browser and prepare his report using Google Docs.
    b) Jagdish was asked to find information about "Digital India". He was asked to use Mozilla Firefox browser and prepare his report using Libre Office Writer.

    What is the difference between technologies used by Sunil and Jagdish?

19. Cite examples depicting that you were a victim of following cyber crime. Also, cite provisions in IT Act to deal with such a cyber crime.
    a) Identity theft
    b) Credit card account theft

20. Neerja is a student of Class XI. She has opted for Computer Science. Neerja prepared the project assigned to her. She mailed it to her teacher. The snapshot of that email is shown below.



Find out which of the following email etiquettes are missing in it. Justify your answer.
    a) Subject of the mail
    b) Formal greeting
    c) Self-explanatory terms
    d) Identity of the sender
    e) Regards

21. Sumit got good marks in all the subjects. His father gifted him a laptop. He would like to make Sumit aware of health hazards associated with inappropriate and excessive use of laptop. Help his father to list the points which he should discuss with Sumit.

NOTES

NOTES