

# **COMPUTER SCIENCE**

**TEXTBOOK FOR CLASS XI**



11120

विद्या स प्रतमने



**राष्ट्रीय शैक्षिक अनुसंधान और प्रशिक्षण परिषद्**  
**NATIONAL COUNCIL OF EDUCATIONAL RESEARCH AND TRAINING**

**First Edition**

*May 2019 Vaishakha 1941*

**Reprinted**

*June 2021 Jyeshtha 1943*

*November 2021 Agrahayana 1943*

**PD 30T RSP**

© National Council of Educational  
Research and Training, 2019

**₹ 195.00**

*Printed on 80 GSM paper*

Published at the Publication  
Division by the Secretary, National  
Council of Educational Research  
and Training, Sri Aurobindo Marg,  
New Delhi 110 016 and printed  
at Goyal Stationers, B-36/9, G.T.  
Karnal Road Industrial Area,  
Delhi 110 033

**ALL RIGHTS RESERVED**

- ❑ No part of this publication may be reproduced, stored in a retrieval system or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise without the prior permission of the publisher.
- ❑ This book is sold subject to the condition that it shall not, by way of trade, be lent, re-sold, hired out or otherwise disposed of without the publisher's consent, in any form of binding or cover other than that in which it is published.
- ❑ The correct price of this publication is the price printed on this page. Any revised price indicated by a rubber stamp or by a sticker or by any other means is incorrect and should be unacceptable.

**OFFICES OF THE PUBLICATION  
DIVISION, NCERT**

NCERT Campus Sri Aurobindo Marg <b>New Delhi 110 016</b>	Phone : 011-26562708
108, 100 Feet Road Hosdakere Halli Extension Banashankari III Stage <b>Bengaluru 560 085</b>	Phone : 080-26725740
Navjivan Trust Building P.O. Navjivan <b>Ahmedabad 380 014</b>	Phone : 079-27541446
CWC Campus Opp. Dhankal Bus Stop Panighati <b>Kolkata 700 114</b>	Phone : 033-25530454
CWC Complex Maligaon <b>Guwahati 781 021</b>	Phone : 0361-2674869

**Publication Team**

Head, Publication	: <i>Anup Kumar Rajput</i>
Division	
Chief Editor	: <i>Shveta Uppal</i>
Chief Production Officer	: <i>Arun Chitkara</i>
Chief Business Manager	: <i>Vipin Dewan</i>
Editor	: <i>Bijnan Sutar</i>
Production Assistant	: <i>Om Prakash</i>

**Cover and Layout**

*DTP Cell*

# FOREWORD

Computer science as a discipline has evolved over the years and has emerged as a driving force for socio-economic activities. It has made continuous inroads into diverse areas — be it business, commerce, science, technology, sports, health, transportation or education. With the advent of computer and communication technologies, there has been a paradigm shift in teaching learning at the school level. The role and relevance of this discipline is in focus because the expectations from the school pass-outs have grown to be able to meet the challenges of the twenty-first century. Today, we are living in an interconnected world where computer-based applications influence the way we learn, communicate, commute or even socialise!

There is a demand for software engineers in various fields like manufacturing, services, etc. Today, there are a large number of successful startups delivering different services through software applications. All these have resulted in generating interest for this subject among students as well as parents.

Development of logical thinking, reasoning and problem-solving skills are fundamental building blocks for knowledge acquisition at the higher level. Computer plays a key role in problem solving with focus on logical representation or reasoning and analysis.

This book focuses on the fundamental concepts and problem-solving skills while opening a window to the emerging and advanced areas of computer science. The newly developed syllabus has dealt with the dual challenge of reducing curricular load as well as introducing this ever evolving discipline.

As an organisation committed to systemic reforms and continuous improvement in the quality of its products, NCERT welcomes comments and suggestions which will enable us to revise the content of the textbook.

New Delhi  
8 August 2018

HRUSHIKESH SENAPATY  
*Director*  
National Council of Educational  
Research and Training

not to be republished  
© NCERT

# PREFACE

In the present education system of our country, specialised or discipline-based courses are introduced at the higher secondary stage. This stage is crucial as well as challenging because of the transition from general to discipline-based curriculum. The syllabus at this stage needs to have sufficient rigour and depth while remaining mindful of the comprehension level of the learners. Further, the textbook should not be heavily loaded with content.

Computers have permeated in every facet of life. Study of basic concepts of computer science has been desirable in education. There are courses offered in the name of Computer Science, Information and Communication Technology (ICT), Information Technology (IT), etc., by various boards and schools up to secondary stage, as optional. These mainly focus on using computer for word processing, presentation tools and application software.

Computer Science (CS) at the higher secondary stage of school education is also offered as an optional subject. At this stage, students usually opt for CS with an aim of pursuing a career in software development or related areas, after going through professional courses at higher levels. Therefore, at higher secondary stage, the curriculum of CS introduces basics of computing and sufficient conceptual background of Computer Science.

The primary focus is on fostering the development of computational thinking and problem-solving skills. This book has 11 chapters covering the following broader themes:

- Fundamentals: basic understanding of computer system, hardware components and software, data representation, number system, encoding as well as awareness of emerging trends in computer science.
- Problem-solving: problem analysis, algorithm, flowchart, implementation, testing and maintenance.
- Programming: basic constructs of a program using Python programming language — program structure, identifiers, variables, flow of control, advanced data types, functions.
- Societal impact: awareness of digital footprints, data privacy and protection, cyber crime, etiquettes in a digital society and implications on security, privacy, piracy, ethics, values and health concerns.
- Chapters 1, 2, 3, 4 and 11 have two additional components — (i) activities and (ii) think and reflect for self assessment while learning as well as to generate further interest in the learner.

Python programming language is introduced that is easy to learn in interactive and script mode. A number of hands-on examples are given to gradually explain methodology to solve different types of problems across the Chapters 5 to 10. The programming examples as well as the exercises in these chapters are required to be solved in a computer and verify with the given outputs.

Group projects through case studies are proposed to solve complex problems. Peer assessment of these projects will promote peer-learning, team spirit and responsiveness. Some exercises have been made in case-study format to promote problem-finding and problem-solving skills.

Box items (light green background) are pinned inside the chapters either to explain related concepts or to provide additional information related to the topic covered in that section. However, these box items are not to be assessed through examinations.

Unicode encoding scheme for Indic scripts have also been introduced to motivate students to solve problems in public services and the local micro or small businesses in India.

These chapters have been written by involving practicing teachers as well as subject experts. These have been iteratively peer-reviewed.

I would like to place on record appreciation for Professor Om Vikas for leading the review activities of the book as well as for his guidance and motivation to the development team throughout. Several iterations have resulted into this book. Thanks are due to the authors and reviewers for their valuable contribution.

Comments and suggestions are welcome to make this endeavour of par excellence.

New Delhi  
9 August 2018

REJAUL KARIM BARHUIYA  
*Assistant Professor*  
Department of Education  
in Science and Mathematics, NCERT

# TEXTBOOK DEVELOPMENT COMMITTEE

## CHIEF ADVISOR

Om Vikas, *Professor (Retd.)*, Former *Director*, ABV-IIITM, Gwalior, M.P.

## MEMBERS

Anuradha Khattar, *Assistant Professor*, Miranda House, University of Delhi

Ashish Dhalwankar, *PGT(Computer Science)*, Centre Point School, Nagpur, Maharashtra

Chetna Khanna, *Freelance Educationist*, Delhi

Harita Ahuja, *Assistant Professor*, Acharya Narendra Dev College, University of Delhi

Mudasir Wani, *Assistant Professor*, Government College for Women, Nawakadal, Srinagar

Pratiksha Majumdar, *PGT(Computer Science)*, School of Scholars, Nagpur, Maharashtra

Priti Rai Jain, *Assistant Professor*, Miranda House, University of Delhi

Rinku Kumari, *PGT(Computer Science)*, Kendriya Vidyalaya, Sainik Vihar, Delhi

Sajid Yousuf Bhat, *Assistant Professor*, University of Kashmir, J&K

Sarnavi Mahesh, *Research Scholar*, Universita Del Salento, Italy

Sharanjit Kaur, *Associate Professor*, Acharya Narendra Dev College, University of Delhi

Sonali Gogate, *Software Consultant*, Pune, Maharashtra

Tapasi Ray, *Former Global IT Director*, Huntsman Corporation, Singapore

Vandana Tyagi, *PGT(Computer Science)*, Kendriya Vidyalaya, JNU, Delhi

## MEMBER-COORDINATOR

Rejaul Karim Barbhuiya, *Assistant Professor*, DESM, NCERT, Delhi

# **ACKNOWLEDGEMENTS**

The National Council of Educational Research and Training acknowledges the valuable contributions of the individuals and organisations involved in the development of *Computer Science Textbook* for Class XI.

The Council expresses its gratitude to the syllabus development team including MPS Bhatia, *Professor*, Netaji Subhas Institute of Technology, Delhi; T.V. Vijay Kumar, *Professor*, School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi; Zahid Raza, *Associate Professor*, School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi; Vipul Shah, *Principal Scientist*, Tata Consultancy Services, and the CSpa<sup>th</sup>shala team; Aasim Zafar, *Associate Professor*, Department of Computer Science, Aligarh Muslim University, Aligarh; Faisal Anwer, *Assistant Professor*, Department of Computer Science, Aligarh Muslim University, Aligarh; Smruti Ranjan Sarangi, *Associate Professor*, Department of Computer Science and Engineering, Indian Institute of Technology, Delhi; Vikram Goyal, *Associate Professor*, Indraprastha Institute of Information Technology (IIIT), Delhi; Tabrez Nafis, *Assistant Professor*, Jamia Hamdard, New Delhi and Mamur Ali, *Assistant Professor*, Central Institute of Educational Technology, NCERT, New Delhi.

The Council is thankful to the following resource persons for editing, reviewing and refining the manuscript of this book — Mukesh Kumar, DPS RK Puram, Delhi; Gurpreet Kaur, G.D. Goenka Public School, Vasant Kunj, Delhi; Gautam Sarkar, Modern School, Barakhamba Road, Delhi; Aswin K. Dash, Mother's International School, Delhi; Nancy Sehgal, Mata Jai Kaur Public School, Delhi; Ashish Kumar Srivastava, *Assistant Professor*, Department of Education in Science and Mathematics, NCERT, New Delhi; Neelima Gupta, *Professor*, Department of Computer Science, University of Delhi; Anamika Gupta, *Assistant Professor*, Shaheed Sukhdev College of Business Studies, University of Delhi. The Council further acknowledges the contributions of Anuja Krishn, *Freelance Editor*, for language editing.

The Council also gratefully acknowledges the contributions of Meetu Sharma, *Graphic Designer*, Kanika Walecha, *DTP Operator*, and Pooja, *Junior Project Fellow*, in shaping this book. The contributions of the office of the APC, DESM and Publication division, NCERT, New Delhi, in bringing out this book are also duly acknowledged.

The Council also acknowledges the contribution of Shilpa Mohan, *Assistant Editor (Contractual)* Publication Division, NCERT for copy editing this book. The efforts of Sadiq Saeed, *DTP Operator (Contractual)* and Sachin Tanwar, *DTP Operator (Contractual)*, Publication Division, NCERT, are also acknowledged.

# CONTENTS

<b>FOREWORD</b>	<b>iii</b>
<b>PREFACE</b>	<b>v</b>
<b>CHAPTER 1 : COMPUTER SYSTEM</b>	<b>1</b>
1.1 Introduction to Computer System	1
1.2 Evolution of Computer	3
1.3 Computer Memory	5
1.4 Data Transfer between Memory and CPU	7
1.5 Microprocessors	8
1.6 Data and Information	10
1.7 Software	14
1.8 Operating System	20
<b>CHAPTER 2 : ENCODING SCHEMES AND NUMBER SYSTEM</b>	<b>27</b>
2.1 Introduction	27
2.2 Number System	30
2.3 Conversion between Number Systems	34
<b>CHAPTER 3 : EMERGING TRENDS</b>	<b>45</b>
3.1 Introduction	45
3.2 Artificial Intelligence (AI)	45
3.3 Big Data	49
3.4 Internet of Things (IoT)	51
3.5 Cloud Computing	53
3.6 Grid Computing	55
3.7 Blockchains	56
<b>CHAPTER 4 : INTRODUCTION TO PROBLEM SOLVING</b>	<b>61</b>
4.1 Introduction	61
4.2 Steps for Problem Solving	62
4.3 Algorithm	64
4.4 Representation of Algorithms	65
4.5 Flow of Control	70
4.6 Verifying Algorithms	77
4.7 Comparison of Algorithm	79

4.8 Coding	80
4.9 Decomposition	81
<b>CHAPTER 5 : GETTING STARTED WITH PYTHON</b>	<b>87</b>
5.1 Introduction to Python	87
5.2 Python Keywords	90
5.3 Identifiers	91
5.4 Variables	91
5.5 Comments	92
5.6 Everything is an Object	93
5.7 Data Types	94
5.8 Operators	99
5.9 Expressions	104
5.10 Statement	106
5.11 Input and Output	107
5.12 Type Conversion	108
5.13 Debugging	112
<b>CHAPTER 6 : FLOW OF CONTROL</b>	<b>121</b>
6.1 Introduction	121
6.2 Selection	122
6.3 Indentation	126
6.4 Repetition	127
6.5 Break and Continue Statement	132
6.6 Nested Loops	136
<b>CHAPTER 7 : FUNCTIONS</b>	<b>143</b>
7.1 Introduction	143
7.2 Functions	145
7.3 User Defined Functions	146
7.4 Scope of a Variable	158
7.5 Python Standard Library	160
<b>CHAPTER 8 : STRINGS</b>	<b>175</b>
8.1 Introduction	175
8.2 Strings	175
8.3 String Operations	177
8.4 Traversing a String	180
8.5 String Methods and Built-in Functions	180
8.6 Handling Strings	184

<b>CHAPTER 9 : LISTS</b>	<b>189</b>
9.1 Introduction to List	189
9.2 List Operations	190
9.3 Traversing a List	192
9.4 List Methods and Built-in Functions	193
9.5 Nested Lists	195
9.6 Copying Lists	196
9.7 List as Arguments to Function	197
9.8 List Manipulation	199
<b>CHAPTER 10 : TUPLES AND DICTIONARIES</b>	<b>207</b>
10.1 Introduction to Tuples	207
10.2 Tuple Operations	209
10.3 Tuple Methods and Built-in Functions	211
10.4 Tuple Assignment	212
10.5 Nested Tuples	213
10.6 Tuple Handling	213
10.7 Introduction to Dictionaries	215
10.8 Dictionaries are Mutable	216
10.9 Dictionary Operations	217
10.10 Traversing a Dictionary	217
10.11 Dictionary Methods and Built-in functions	218
10.12 Manipulating Dictionaries	219
<b>CHAPTER 11 : SOCIETAL IMPACT</b>	<b>229</b>
11.1 Introduction	229
11.2 Digital Footprints	229
11.3 Digital Society and Netizen	231
11.4 Data Protection	235
11.5 Cyber Crime	239
11.6 Indian Information Technology Act (IT Act)	242
11.7 Impact on Health	242



Empowerment of Girl Child, Responsibility of All

# CHAPTER 1

## COMPUTER SYSTEM



11120CH01

### 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.

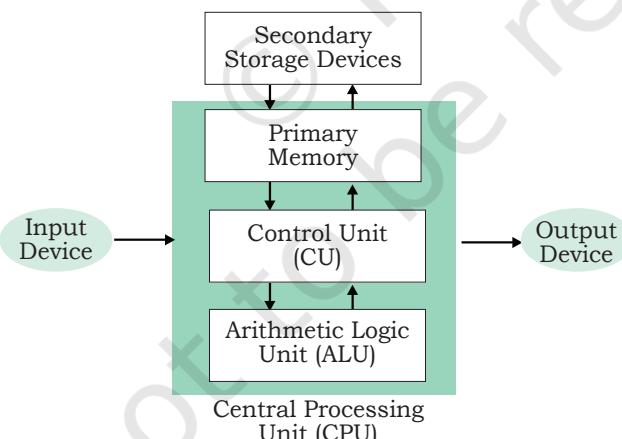


Figure 1.1: Components of a computer system

#### 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.

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

-Alan Turing

#### 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.

### 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



Figure 1.3: 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

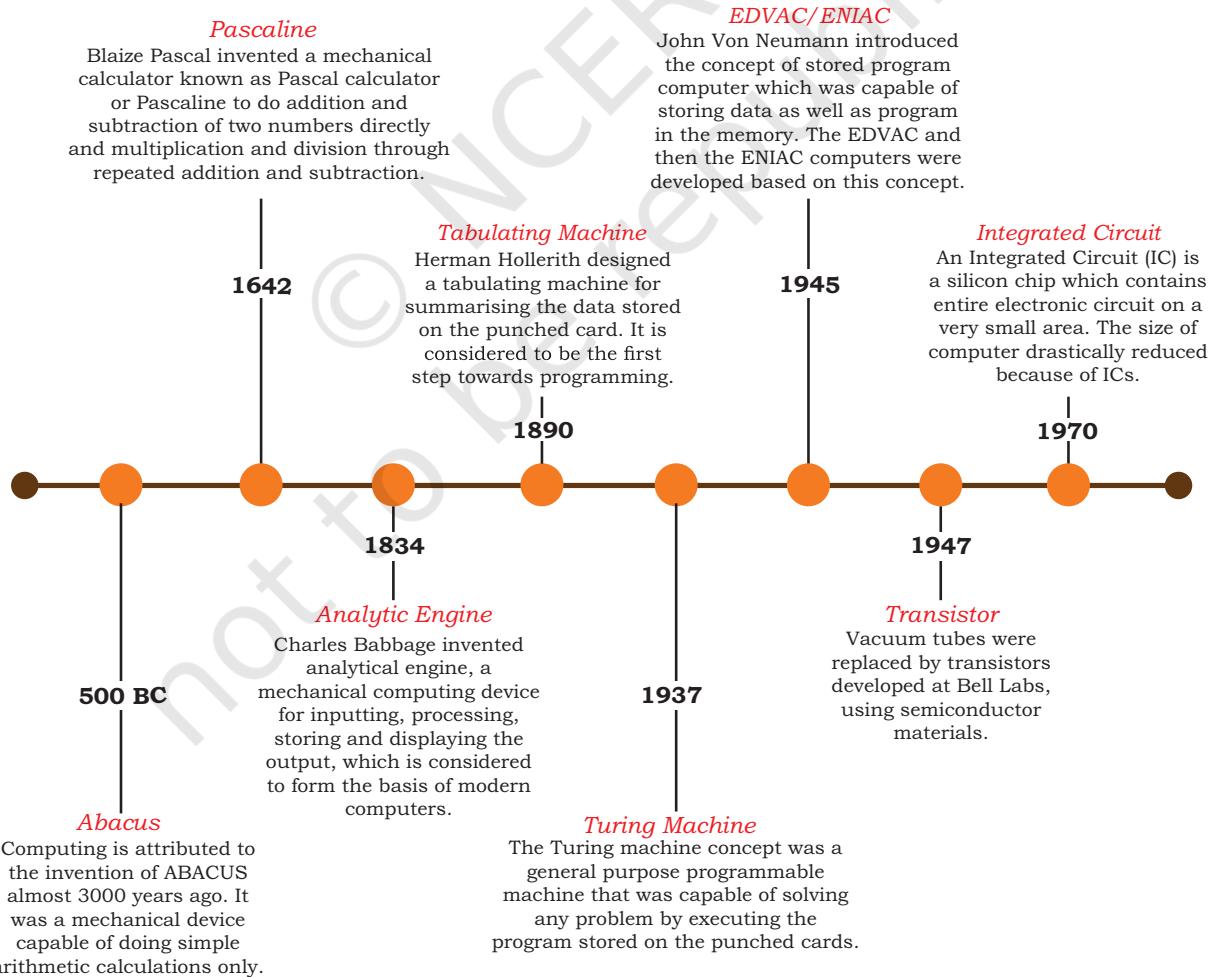


Figure 1.4: Timeline showing key inventions in computing technology

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.

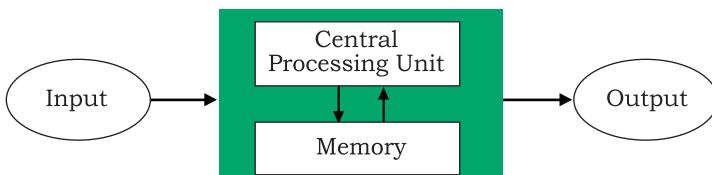


Figure 1.5: Von Neumann architecture for the computer



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.

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 10<sup>6</sup> components) on a single IC called Super Large Scale Integration (SLSI) as shown in Figure 1.6.

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

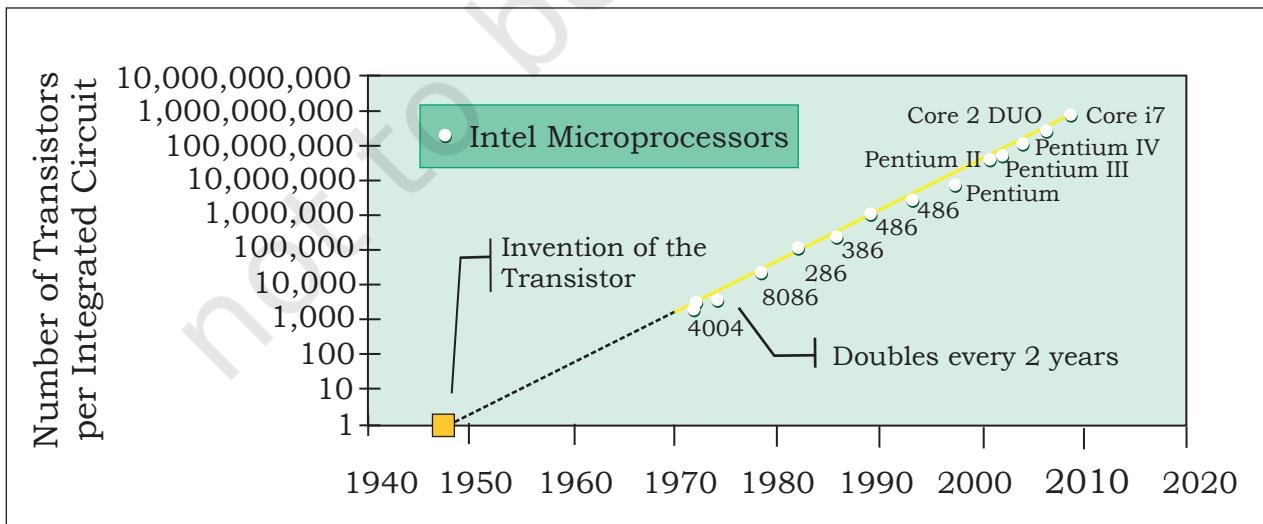


Figure 1.6: Exponential increase in number of transistors used in ICs over time

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**

<b>Unit</b>	<b>Description</b>	<b>Unit</b>	<b>Description</b>
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.

#### (B) Cache Memory

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

#### Think and Reflect

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

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.

## 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 1.7: Storage devices

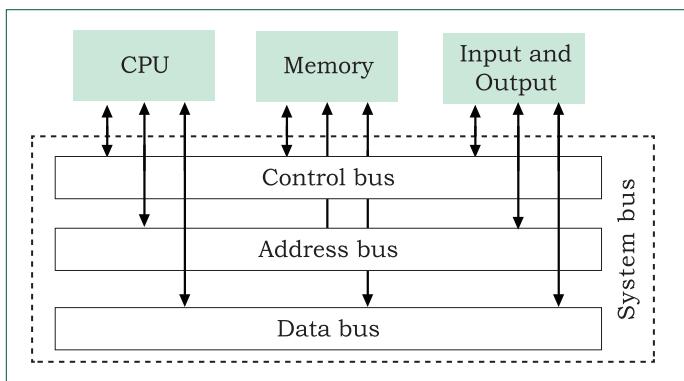


Figure 1.8: Data transfer between components through system bus

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.

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.

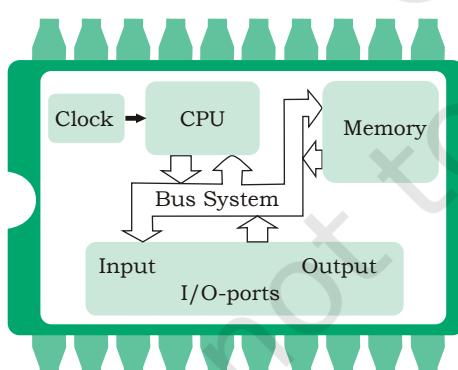


Figure 1.9: Structure of microcontroller

### 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.

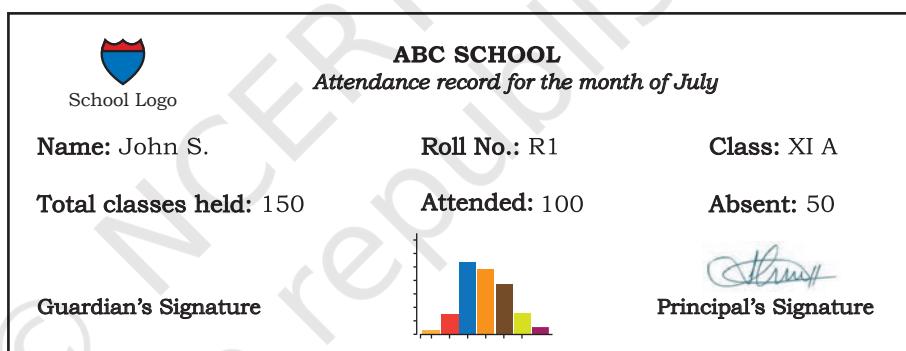


Figure 1.10: Unstructured data: Monthly attendance record

#### (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.

### 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



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.

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.

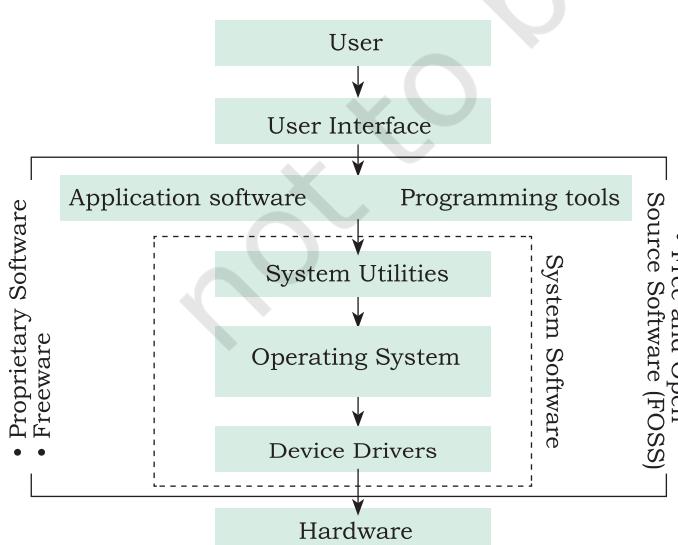


Figure 1.13: Categorisation of software

### **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

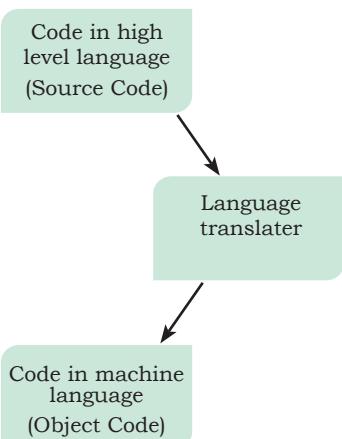


Figure 1.14: Translator to convert source code into object code

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?

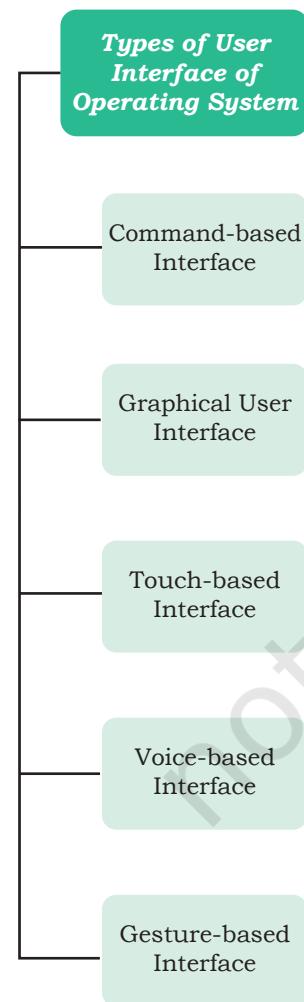


Figure 1.15: Types of user interface of OS

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.

### 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,

**NOTES**

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**

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-



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.

**NOTES**

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.

**NOTES**

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?

not to be republished  
© NCERT

## CHAPTER 2

# ENCODING SCHEMES AND NUMBER SYSTEM



11120CH02

### 2.1 INTRODUCTION

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.

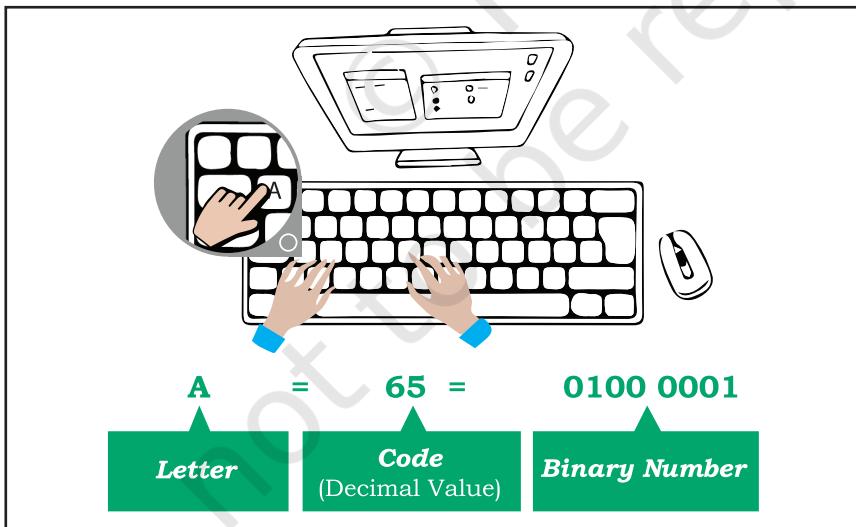


Figure 2.1: Encoding of data entered using keyboard

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

“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

#### 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**

०	१	२	३	४	अ	आ	इ	ঈ	উ	ঊ	ঃ	ঁ	ঁ	ঁ	ঁ
0900	0901	0902	0903	0904	0905	0906	0907	0908	0909	090A	090B	090C	090D	090E	090F
ঁ	ঁ	ঁ	ঁ	ঁ	ক	খ	গ	ঘ	ঙ	চ	ছ	জ	ঝ	ঁ	ট
0910	0911	0912	0913	0914	0915	0916	0917	0918	0919	091A	091B	091C	091D	091E	091F
ঁ	ঁ	ঁ	ঁ	ঁ	থ	দ	ধ	ন	ন	প	ফ	ব	ভ	ম	য
0920	0921	0922	0923	0924	0925	0926	0927	0928	0929	092A	092B	092C	092D	092E	092F
ৰ	ৱ	ল	ঁ	ঁ	ৰ	শ	ঁ	স	হ	ঁ	ঁ	ঁ	ঁ	ৰ	ঁ
0930	0931	0932	0933	0934	0935	0936	0937	0938	0939	093A	093B	093C	093D	093E	093F
ু	ু	৮	৯	৯	ু	৬	৯	৯	৯	৭	৭	৯	৯	৭	৭
0940	0941	0942	0943	0944	0945	0946	0947	0948	0949	094A	094B	094C	094D	094E	094F
ঁ	ঁ	—	—	—	—	—	—	ক	খ	গ	ঝ	ঙ	ঁ	ঁ	ঁ
0950	0951	0952	0953	0954	0955	0956	0957	0958	0959	095A	095B	095C	095D	095E	095F
ঁ	ঁ	ঁ	ঁ	ঁ	।	॥	০	১	২	৩	৪	৫	৬	৭	৮
0960	0961	0962	0963	0964	0965	0966	0967	0968	0969	096A	096B	096C	096D	096E	096F
০	১	২	৩	৪	অ	আ	ই	ঈ	উ	ঊ	ঁ	ঁ	ঁ	ঁ	ঁ
0970	0971	0972	0973	0974	0975	0976	0977	0978	0979	097A	097B	097C	097D	097E	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.

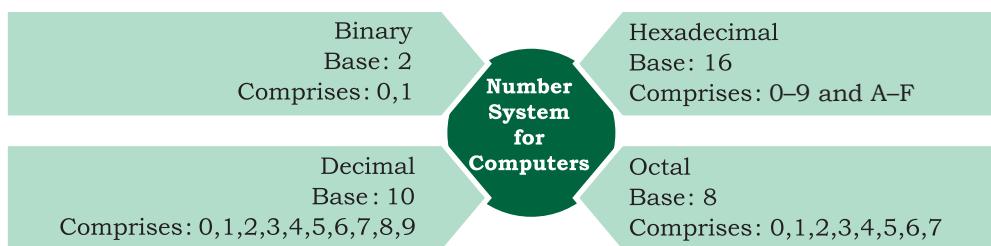


Figure 2.2: Four different number systems

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.

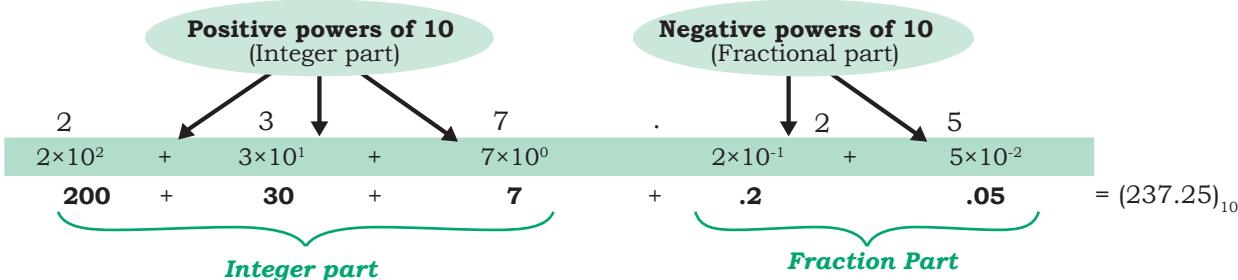


Figure 2.4: Positional value for digits of decimal number represented as power of base 10

### 2.2.2 Binary Number System



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.

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 ( $8=2^3$ ) 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  $(617.24)_8$ .

#### 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 2.5 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

**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)  $(66)_{10}$

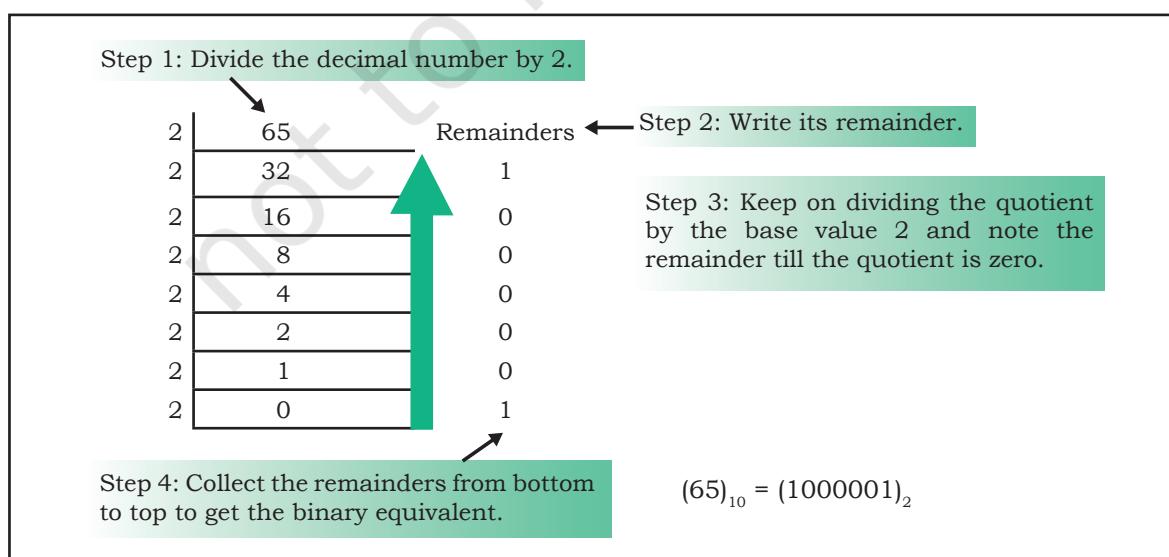


Figure 2.5: Conversion of a decimal number to its equivalent binary number

**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
2	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.

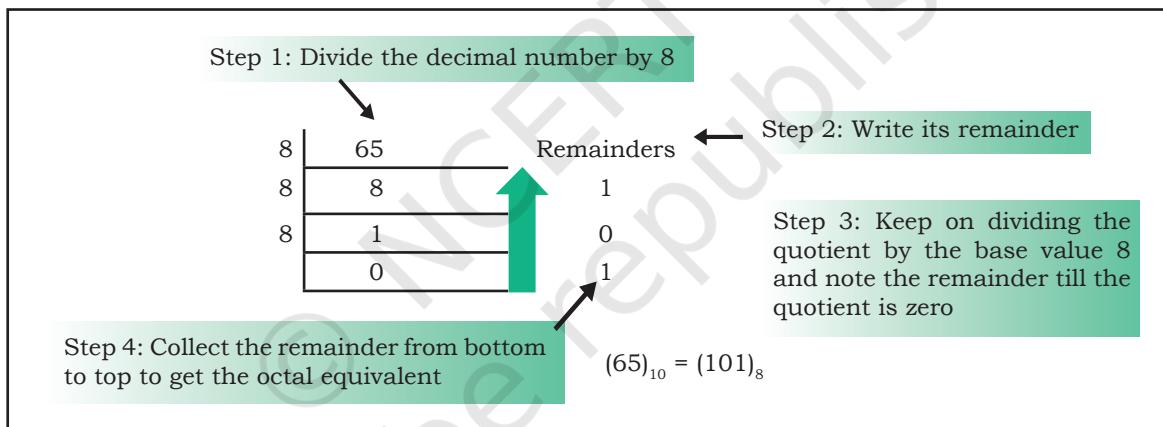


Figure 2.6: Conversion of a decimal number to its equivalent octal number

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
8	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.

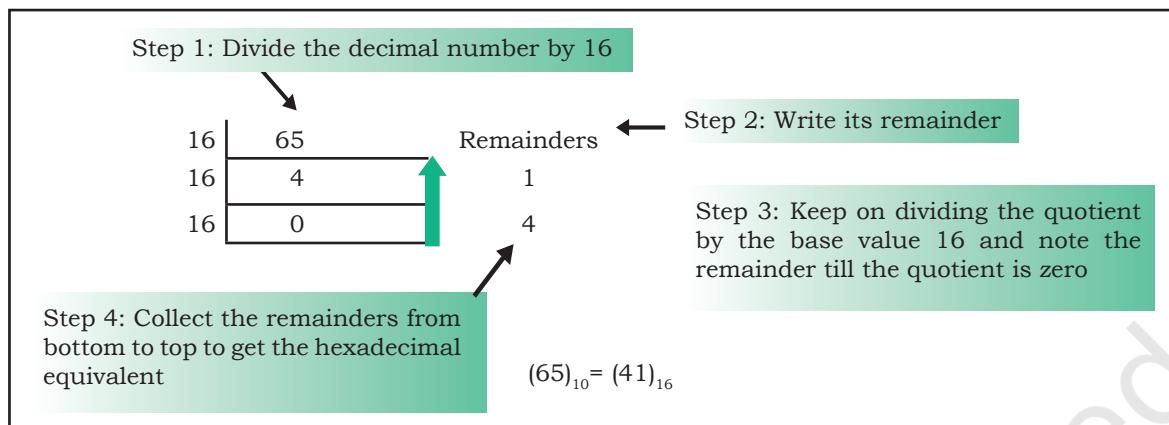
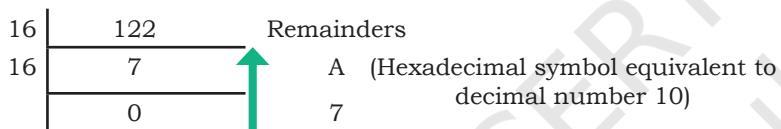


Figure 2.7: Conversion of a decimal number to its equivalent hexadecimal number

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



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

### 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

#### (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

#### Activity 2.4

Convert the following numbers into decimal numbers.

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

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	$1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 8 + 4 + 0 + 1 = (13)_{10}$			

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



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

The base value of octal number system is 8. Convert value 8 in terms of exponent of 2, i.e.,  $8=2^3$ . Hence, three binary digits are sufficient to represent all 8 octal digits.

Simply stated, count all possible combinations of three binary digits, which are  $2 \times 2 \times 2 = 8$ . Therefore, 3 bits are sufficient to represent any octal digit. Hence, 3-bit groups in a binary number are formed to get equivalent octal number.

### (B) Octal Number to 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	$2 \times 8^2 + 5 \times 8^1 + 7 \times 8^0 = 128 + 40 + 7 = (175)_{10}$		

### (C) Hexadecimal Number to Decimal Number

For converting a hexadecimal number into decimal number, use steps given in this section with base value 16 of hexadecimal number system. Use 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	$3 \times 16^2 + 10 \times 16^1 + 5 \times 16^0 = 768 + 160 + 5 = (933)_{10}$		

**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

$$\text{Therefore, } (10101100)_2 = (254)_8$$

### (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

$$\text{Therefore, } (705)_8 = (111000101)_2$$

### (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

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

### 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

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

### 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.

$0.25 \times 2 = 0.50$ $0.50 \times 2 = 1.00$	<span style="font-size: 2em;">Integer part</span> $\downarrow$ <span style="font-size: 2em;">0</span> $\downarrow$ <span style="font-size: 2em;">1</span>
--	---

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.

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

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

	Integer part
$0.675 \times 2 = 1.350$	1
$0.350 \times 2 = 0.700$	0
$0.700 \times 2 = 1.400$	1
$0.400 \times 2 = 0.800$	0
$0.800 \times 2 = 1.600$	1
$0.600 \times 2 = 1.200$	1
$0.200 \times 2 = 0.400$	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
$0.675 \times 8 = 5.400$	5
$0.400 \times 8 = 3.200$	3
$0.200 \times 8 = 1.600$	1
$0.600 \times 8 = 4.800$	4
$0.800 \times 8 = 6.400$	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
$0.675 \times 16 = 10.800$	A (Hexadecimal symbol for 10)
$0.800 \times 16 = 12.800$	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	$1 \times 2^5$	$+0 \times 2^4$	$+0 \times 2^3$	$+1 \times 2^2$	$+0 \times 2^1$	$+1 \times 2^0$	$+$	$1 \times 2^{-1}$	$0 \times 2^{-2}$	$+1 \times 2^{-3}$
=	$\underline{32}$	$+ 0$	$+ 0$	$+ 4$	$+ 0$	$+ 1$	$+$	$\underline{0.5}$	$+ 0$	$+ 0.125$
				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	$6 \times 8^2$	$+ 0 \times 8^1$	$+ 5 \times 8^0$	$+$	$1 \times 8^{-1}$	$+ 2 \times 8^{-2}$
=	$\underline{384}$	$+ 0$	$+ 5$	$+$	$.125$	$+ .03125$
			389			0.15625

Therefore,  $(605.12)_8 = (389.15625)_{10}$

### (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    010 101 100 . 010 110  
 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,  $(10101100.010111)_2 = (AC.5C)_{16}$

## NOTES

### 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



### 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

*“Computer science is no more about computers than astronomy is about telescopes”*

*–Edsger Dijkstra*

#### In this chapter

- » *Introduction*
- » *Artificial Intelligence (AI)*
- » *Big Data*
- » *Internet of Things (IoT)*
- » *Cloud Computing*
- » *Grid Computing*
- » *Blockchains*



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

### Activity 3.1

Find out how NLP is helping differently-abled persons?

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)

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.

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.



Figure 3.1: Use of natural language processing

### 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.

#### **(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.

#### **(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 3.2: Driving simulator



Figure 3.3: Virtual Reality Headset



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.



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.



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

### 3.2.4 Robotics

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 3.6: Sophia is a humanoid



Figure 3.7: Drone

#### 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?

- 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 poaching, besides law-enforcement and border patrolling.

### 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).

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.

#### 3.3.1 Characteristics of Big Data

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

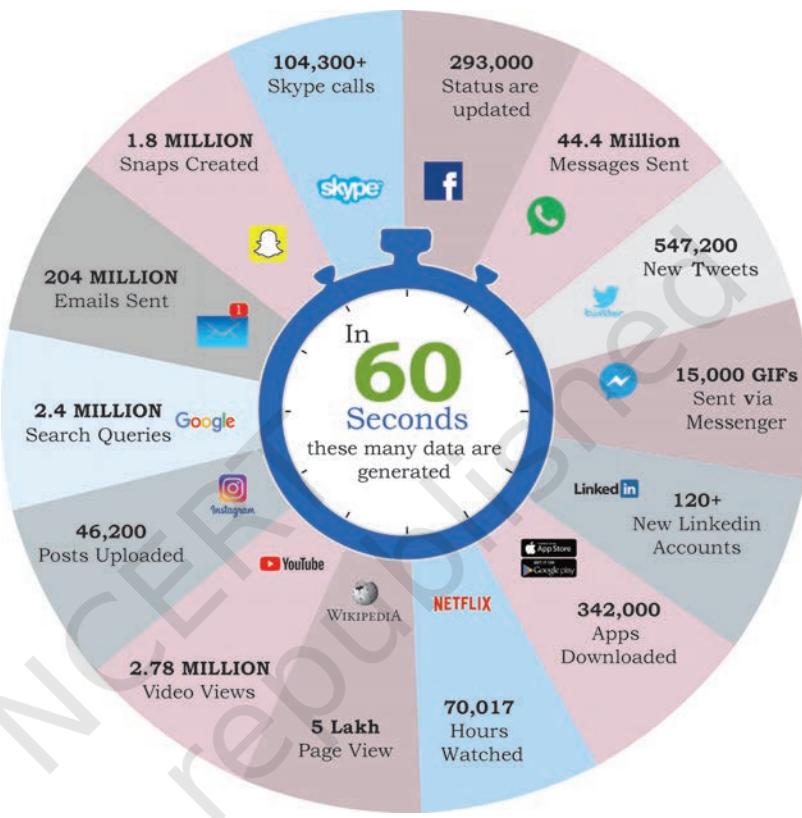


Figure 3.8: Sources of big data (numbers are approximate)

#### Think and Reflect

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

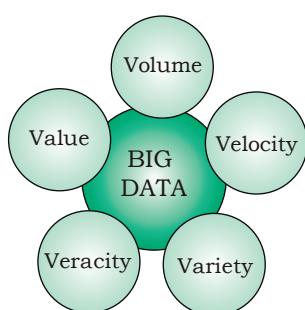


Figure 3.9: Characteristics of big data

#### **(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.

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.

#### 3.4.1 Web of Things (WoT)

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

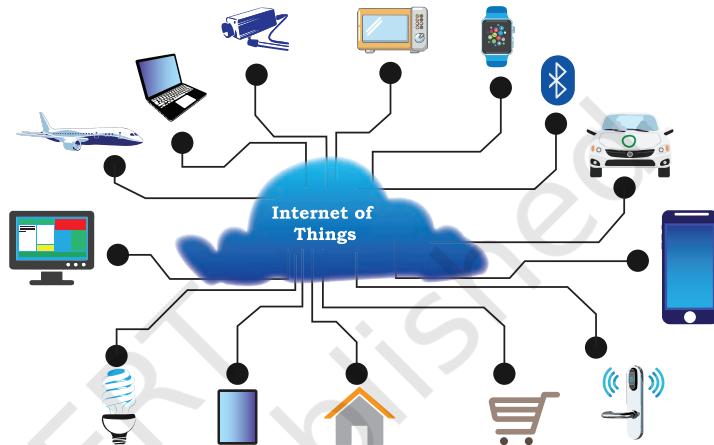


Figure 3.10: Internet of Things (IoT)

#### Activity 3.3

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

**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.

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.

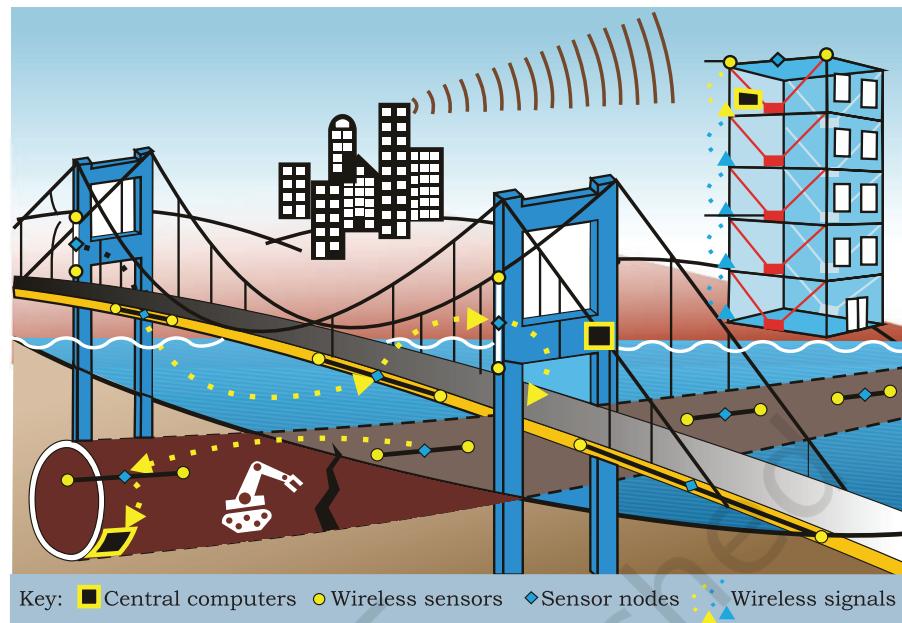


Figure 3.11: Smart city

### 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).

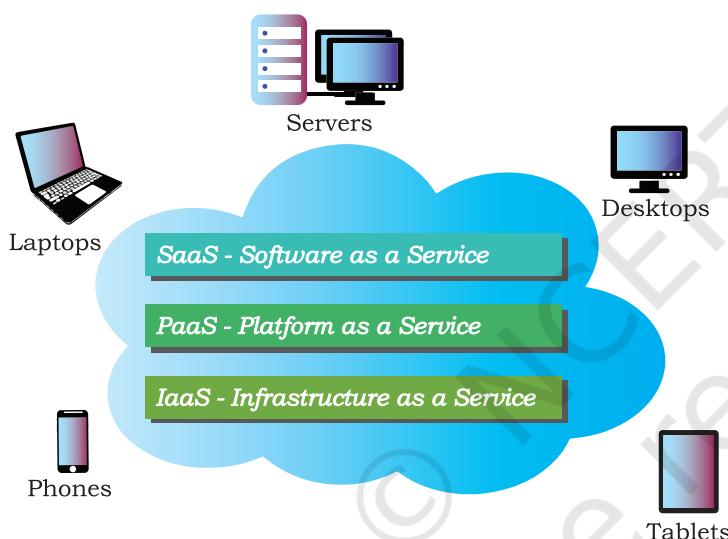


Figure 3.12: Cloud computing

#### **(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.

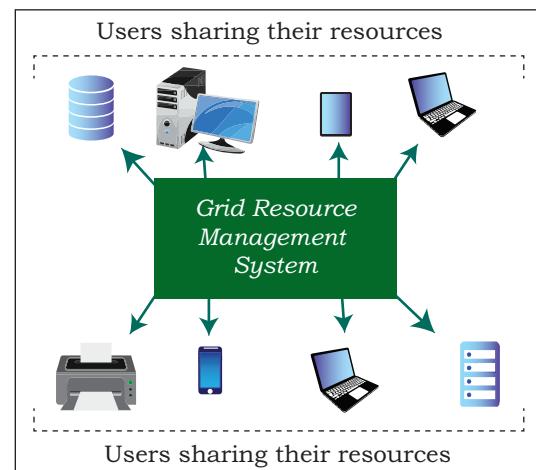


Figure 3.13: Grid computing

### 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

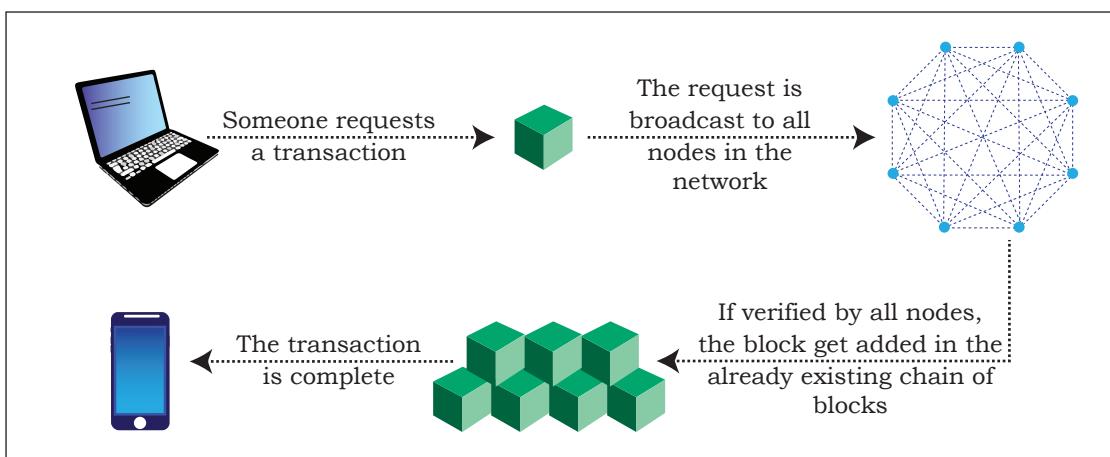


Figure 3.14: Block chain technology

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**

1. You got a reminder to take medication
2. You got an SMS alert that you forgot to lock the door
3. You got an SMS alert that parking space is available near your block
4. You turned off your LED TV from your wrist watch

**Column B**

- A. Smart Parking
- B. Smart Wearable
- C. Home Automation
- D. Smart Health

# CHAPTER 4

## INTRODUCTION TO PROBLEM SOLVING

### 4.1 INTRODUCTION

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.



11120CH04

*“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*

#### In this chapter

- » *Introduction*
- » *Steps for Problem Solving*
- » *Algorithm*
- » *Representation of Algorithms*
- » *Flow of Control*
- » *Verifying Algorithms*
- » *Comparison of Algorithm*
- » *Coding*
- » *Decomposition*



### GIGO (Garbage In Garbage Out)

The correctness of the output that a computer gives depends upon the correctness of input provided.

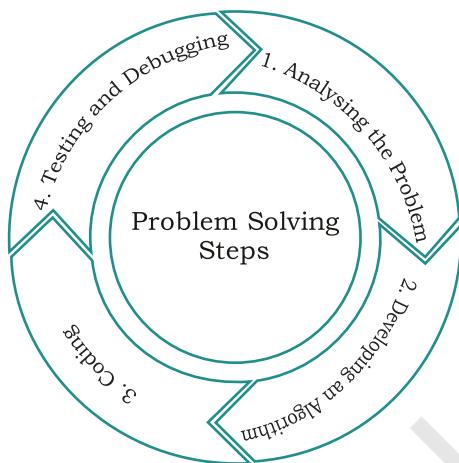


Figure 4.1: Steps for problem solving



### Algorithm

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

## 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.

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.

### 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

**NOTES**

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 54.

**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.

**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



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’.

**NOTES**

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.

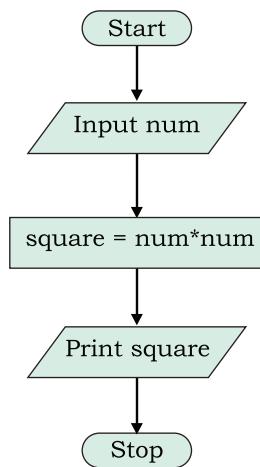


Figure 4.2: Flowchart to calculate square of a number

### Think and Reflect

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

### 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

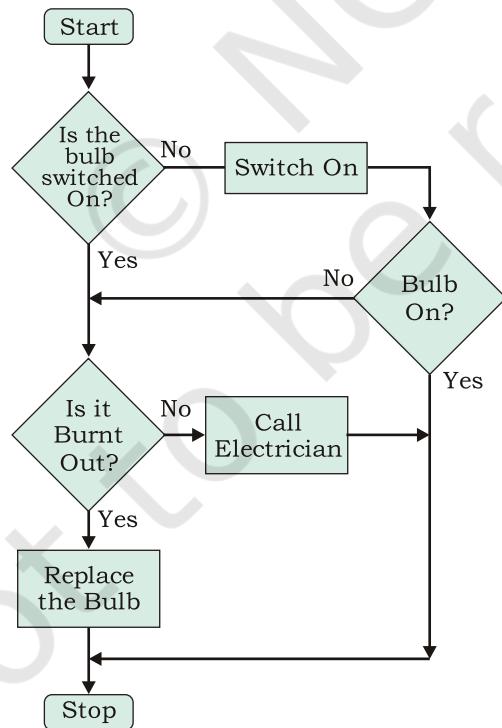


Figure 4.3: Flowchart to solve the problem of a non-functioning light bulb

#### 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

*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.

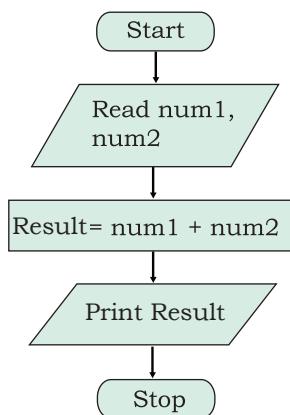


Figure 4.4: Flowchart to display sum of two numbers

**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.

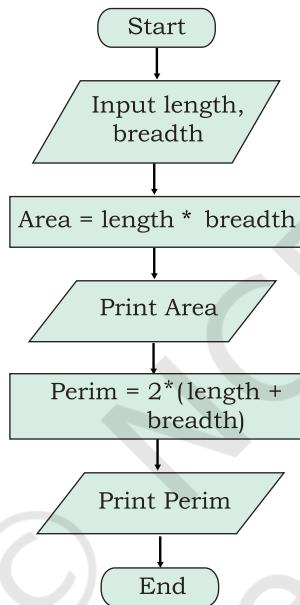


Figure 4.5: Flowchart to calculate area and perimeter of a rectangle

### (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.

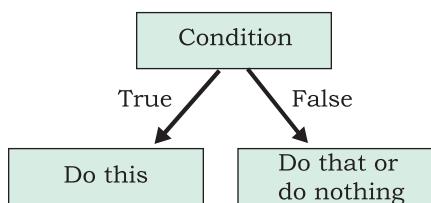


Figure 4.7: Actions depending on true or false of a condition

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 is shown in Figure 4.8.

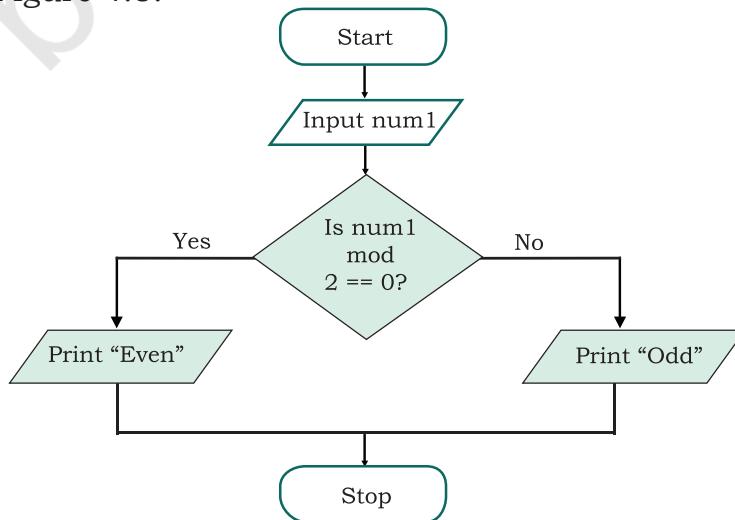


Figure 4.8: Flowchart to check whether a number is even or odd

## 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 ( $\geq 13$  but  $< 20$ ) or adult ( $\geq 20$ ), 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 is shown in Figure 4.9

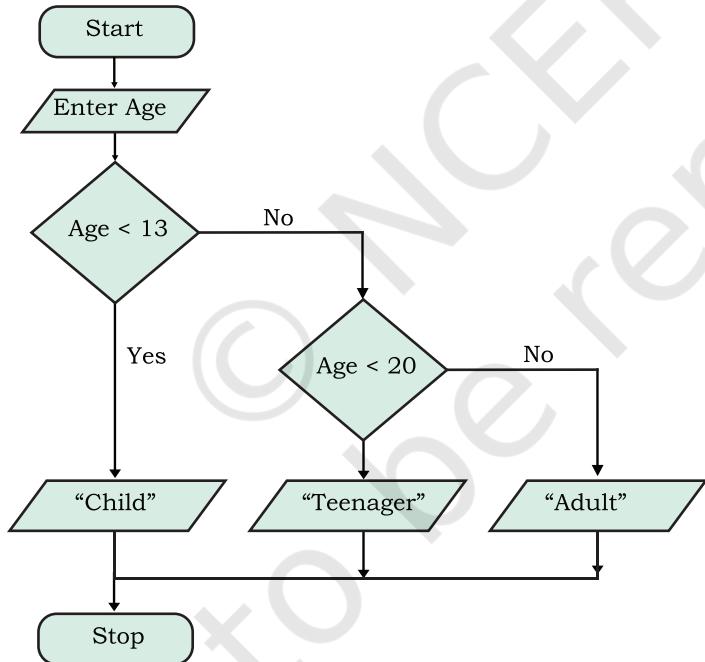


Figure 4.9: Flowchart to check multiple conditions

**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.

**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 iteration of 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

### Think and Reflect

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

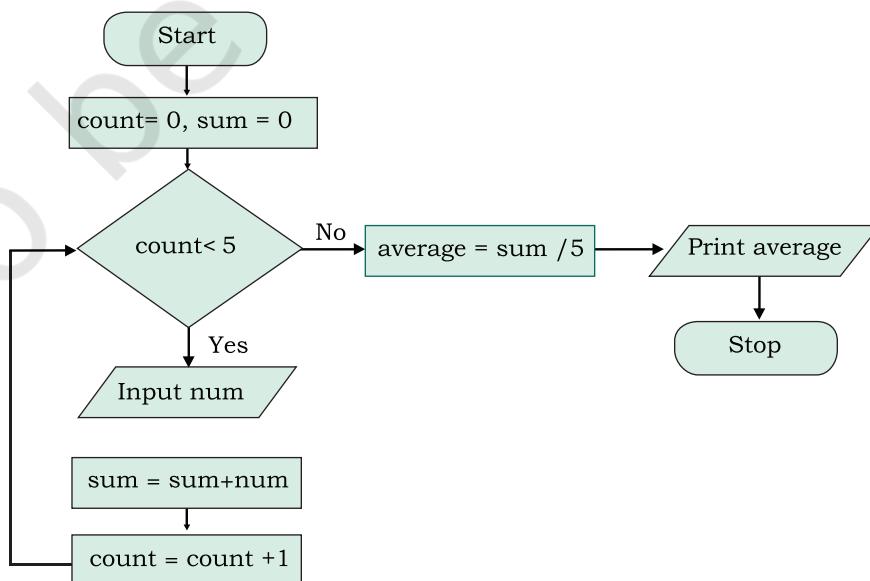


Figure 4.10: Flowchart to Calculate the Average of 5 Numbers

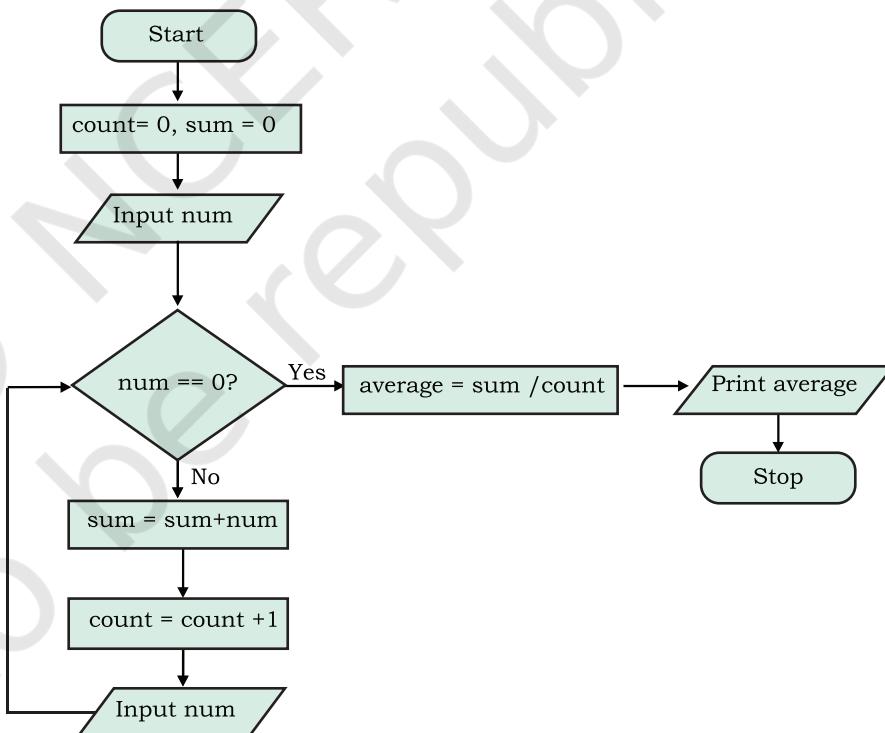
of statements need to be repeated. Such requirements of unknown number of repetitions are handled using WHILE construct.

**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.



#### Activity 4.4

Let us answer the following questions using the pseudocode 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?

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.

## 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.

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 =  $\frac{6(6+1)}{2}$

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

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 ( $T_{\text{total}}$ ) via B where time taken to

### Think and Reflect

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

### 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.

**NOTES**

go from A to B ( $T_1$ ) and B to C ( $T_2$ ) 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  $T_1 = 5$  hrs 20 mins and  $T_2 = 7$  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  $T_1 = 4$  hrs 50 mins and  $T_2 = 2$  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  $mm\_total \geq 60$ , 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  $T_1 = 4$  hrs 50 mins and  $T_2 = 2$  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 processing 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.

**NOTES**

- 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.

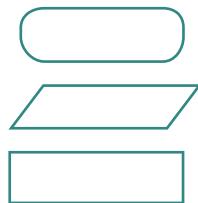
**EXERCISE**

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

**NOTES**

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*



### Downloading Python

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

<https://www.python.org/>

### 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.

### 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

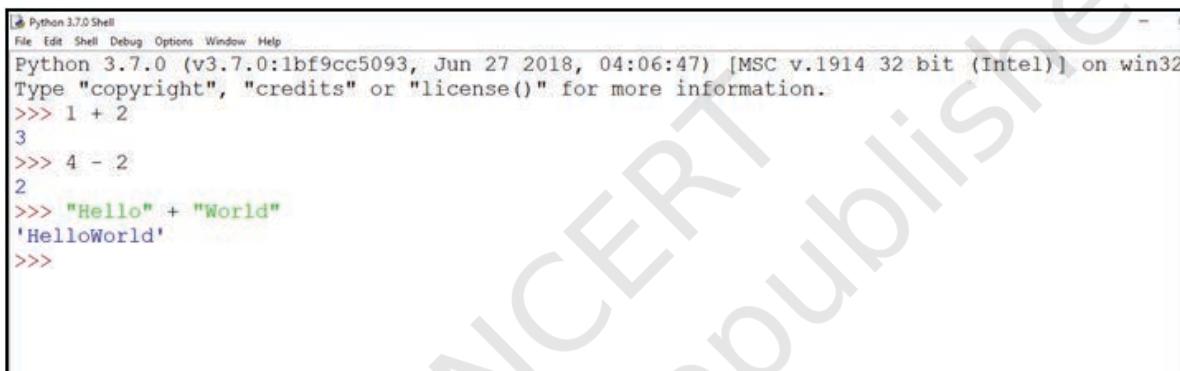
There are two ways to use the Python interpreter:

- 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.



The screenshot shows the Python 3.7.0 Shell window. The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main area displays the following Python session:

```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.

>>> 1 + 2
3
>>> 4 - 2
2
>>> "Hello" + "World"
'HelloWorld'
>>>
```

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.

```
prog5-1.py - C:/NCERT/prog5-1.py (3.7.0)
File Edit Format Run Options Window Help
print("Save Earth")
print("Preserve Future")
```

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



Figure 5.4: Execution of Python in Script mode using IDLE

```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
=====
RESTART: C:/NCERT/prog5-1.py =====
Save Earth
Preserve Future
>>> |
```

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

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

**NOTES**

### 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
                           #object 20
                           refers to
```

## 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.

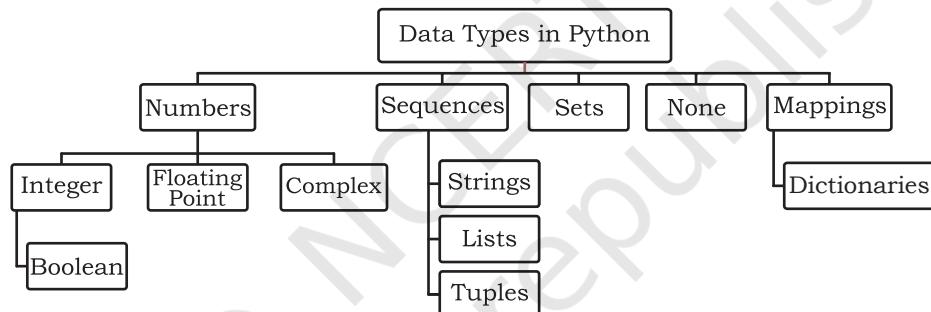


Figure 5.6: Different data types in Python

### 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))
-980.000000000001 <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 same as 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.

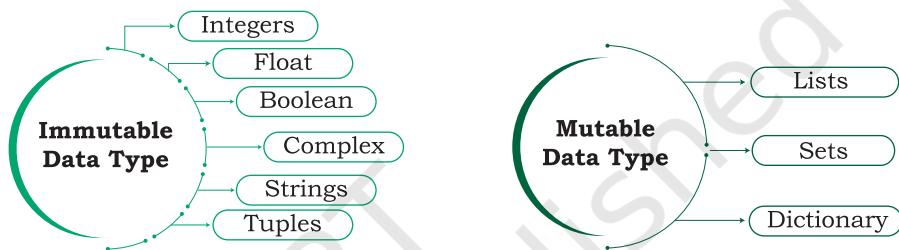


Figure 5.7: Classification of data types

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

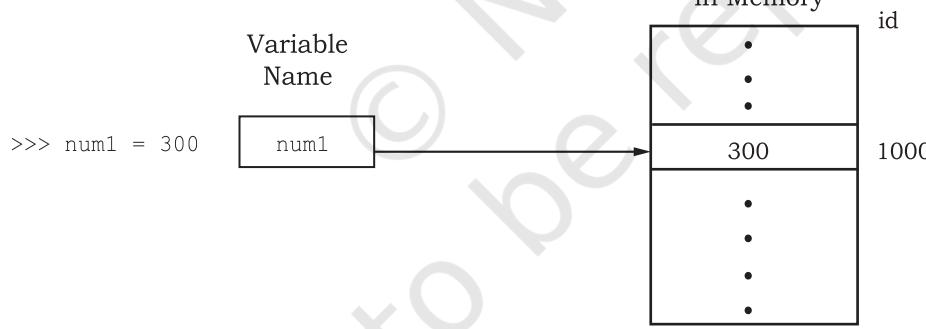


Figure 5.8: Object and its identifier

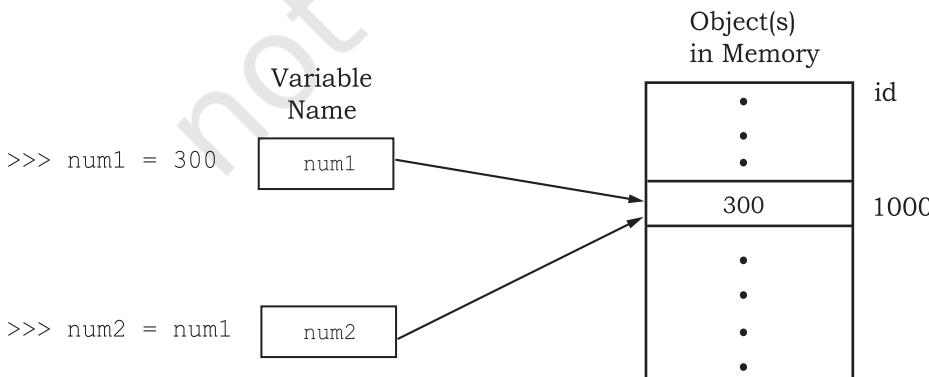


Figure 5.9: Variables with same value have same identifier

`>>> 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.

`>>> 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.

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

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

```
>>> num1 = 300  
>>> num2 = num1
```

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

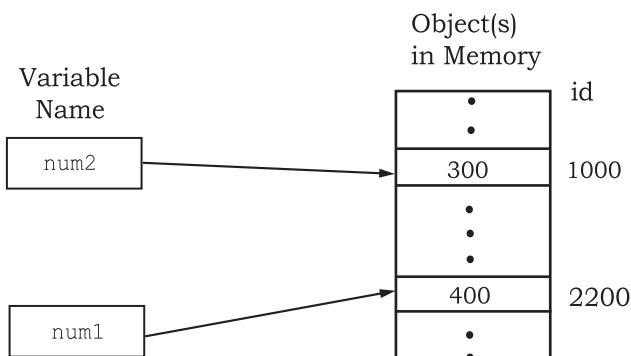


Figure 5.10: Variables with different values have different identifiers

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.

### 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.



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.

## 5.8 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**

<b>Operator</b>	<b>Operation</b>	<b>Description</b>	<b>Example (Try in Lab)</b>
<code>==</code>	Equals to	If the values of two operands are equal, then the condition is True, otherwise it is False	<code>&gt;&gt;&gt; num1 == num2</code> False <code>&gt;&gt; str1 == str2</code> False
<code>!=</code>	Not equal to	If values of two operands are not equal, then condition is True, otherwise it is False	<code>&gt;&gt;&gt; num1 != num2</code> True <code>&gt;&gt;&gt; str1 != str2</code> True <code>&gt;&gt;&gt; num1 != num3</code> False
<code>&gt;</code>	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	<code>&gt;&gt;&gt; num1 &gt; num2</code> True <code>&gt;&gt;&gt; str1 &gt; str2</code> True
<code>&lt;</code>	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	<code>&gt;&gt;&gt; num1 &lt; num3</code> False <code>&gt;&gt;&gt; str2 &lt; str1</code> True
<code>&gt;=</code>	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	<code>&gt;&gt;&gt; num1 &gt;= num2</code> True <code>&gt;&gt;&gt; num2 &gt;= num3</code> False <code>&gt;&gt;&gt; str1 &gt;= str2</code> True
<code>&lt;=</code>	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	<code>&gt;&gt;&gt; num1 &lt;= num2</code> False <code>&gt;&gt;&gt; num2 &lt;= num3</code> True <code>&gt;&gt;&gt; str1 &lt;= str2</code> 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**

<b>Operator</b>	<b>Description</b>	<b>Example (Try in Lab)</b>
<code>=</code>	Assigns value from right-side operand to left-side operand	<code>&gt;&gt;&gt; num1 = 2</code> <code>&gt;&gt;&gt; num2 = num1</code> <code>&gt;&gt;&gt; num2</code> 2 <code>&gt;&gt;&gt; country = 'India'</code> <code>&gt;&gt;&gt; country</code> 'India'

<code>+=</code>	<p>It adds the value of right-side operand to the left-side operand and assigns the result to the left-side operand  <b>Note:</b> <math>x += y</math> is same as <math>x = x + y</math></p>	<pre>&gt;&gt;&gt; num1 = 10 &gt;&gt;&gt; num2 = 2 &gt;&gt;&gt; num1 += num2 &gt;&gt;&gt; num1 12 &gt;&gt;&gt; num2 2 &gt;&gt;&gt; str1 = 'Hello' &gt;&gt;&gt; str2 = 'India' &gt;&gt;&gt; str1 += str2 &gt;&gt;&gt; str1 'HelloIndia'</pre>
<code>-=</code>	<p>It subtracts the value of right-side operand from the left-side operand and assigns the result to left-side operand  <b>Note:</b> <math>x -= y</math> is same as <math>x = x - y</math></p>	<pre>&gt;&gt;&gt; num1 = 10 &gt;&gt;&gt; num2 = 2 &gt;&gt;&gt; num1 -= num2 &gt;&gt;&gt; num1 8</pre>
<code>*=</code>	<p>It multiplies the value of right-side operand with the value of left-side operand and assigns the result to left-side operand  <b>Note:</b> <math>x *= y</math> is same as <math>x = x * y</math></p>	<pre>&gt;&gt;&gt; num1 = 2 &gt;&gt;&gt; num2 = 3 &gt;&gt;&gt; num1 *= 3  &gt;&gt;&gt; num1 6 &gt;&gt;&gt; a = 'India' &gt;&gt;&gt; a *= 3 &gt;&gt;&gt; a 'IndiaIndiaIndia'</pre>
<code>/=</code>	<p>It divides the value of left-side operand by the value of right-side operand and assigns the result to left-side operand  <b>Note:</b> <math>x /= y</math> is same as <math>x = x / y</math></p>	<pre>&gt;&gt;&gt; num1 = 6 &gt;&gt;&gt; num2 = 3 &gt;&gt;&gt; num1 /= num2 &gt;&gt;&gt; num1 2.0</pre>
<code>%=</code>	<p>It performs modulus operation using two operands and assigns the result to left-side operand  <b>Note:</b> <math>x \%= y</math> is same as <math>x = x \% y</math></p>	<pre>&gt;&gt;&gt; num1 = 7 &gt;&gt;&gt; num2 = 3 &gt;&gt;&gt; num1 %= num2 &gt;&gt;&gt; num1 1</pre>
<code>//=</code>	<p>It performs floor division using two operands and assigns the result to left-side operand  <b>Note:</b> <math>x // y</math> is same as <math>x = x // y</math></p>	<pre>&gt;&gt;&gt; num1 = 7 &gt;&gt;&gt; num2 = 3 &gt;&gt;&gt; num1 // y &gt;&gt;&gt; num1 2</pre>
<code>**=</code>	<p>It performs exponential (power) calculation on operators and assigns value to the left-side operand  <b>Note:</b> <math>x **= y</math> is same as <math>x = x ** y</math></p>	<pre>&gt;&gt;&gt; num1 = 2 &gt;&gt;&gt; num2 = 3 &gt;&gt;&gt; num1 **= num2 &gt;&gt;&gt; num1 8</pre>

### 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)
<code>and</code>	Logical AND	If both the operands are True, then condition becomes True	<pre>&gt;&gt;&gt; True and True True &gt;&gt;&gt; num1 = 10 &gt;&gt;&gt; num2 = -20 &gt;&gt;&gt; bool(num1 and num2) True &gt;&gt;&gt; True and False False &gt;&gt;&gt; num3 = 0 &gt;&gt;&gt; bool(num1 and num3) False &gt;&gt;&gt; False and False False</pre>
<code>or</code>	Logical OR	If any of the two operands are True, then condition becomes True	<pre>&gt;&gt;&gt; True or True True &gt;&gt;&gt; True or False True &gt;&gt;&gt; bool(num1 or num3) True &gt;&gt;&gt; False or False False</pre>
<code>not</code>	Logical NOT	Used to reverse the logical state of its operand	<pre>&gt;&gt;&gt; num1 = 10 &gt;&gt;&gt; bool(num1) True &gt;&gt;&gt; not num1 &gt;&gt;&gt; bool(num1) False</pre>

### 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)	<pre>&gt;&gt;&gt; num1 = 5 &gt;&gt;&gt; type(num1) is int True &gt;&gt;&gt; num2 = num1 &gt;&gt;&gt; id(num1) 1433920576 &gt;&gt;&gt; id(num2) 1433920576 &gt;&gt;&gt; num1 is num2 True</pre>
is not	Evaluates to False if the variables on either side of the operator point to the same memory location and True otherwise. var1 is not var2 results to True if id(var1) is not equal to id(var2)	<pre>&gt;&gt;&gt; num1 is not num2 False</pre>

### 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	<pre>&gt;&gt;&gt; a = [1, 2, 3] &gt;&gt;&gt; 2 in a True &gt;&gt;&gt; '1' in a False</pre>
not in	Returns True if the variable/value is not found in the specified sequence and False otherwise	<pre>&gt;&gt;&gt; a = [1, 2, 3] &gt;&gt;&gt; 10 not in a True &gt;&gt;&gt; 1 not in a False</pre>

### 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 as 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	<code>**</code>	Exponentiation (raise to the power)
2	<code>~, +, -</code>	Complement, unary plus and unary minus
3	<code>*, /, %, //</code>	Multiply, divide, modulo and floor division
4	<code>+, -</code>	Addition and subtraction
5	<code>&lt;=, &lt;, &gt;, &gt;=, ==, !=</code>	Relational and Comparison operators
6	<code>=, %=, /=, //=, -=, +=, *=, **=</code>	Assignment operators
7	<code>is, is not</code>	Identity operators
8	<code>in, not in</code>	Membership operators
9	<code>not</code>	Logical operators
10	<code>and</code>	
11	<code>or</code>	

**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:*

$$\begin{aligned}
 &= 20 + (30 * 40) \quad \text{\#Step 1} \\
 \text{\#precedence of * is more than that of +} \\
 &= 20 + 1200 \quad \text{\#Step 2} \\
 &= 1220 \quad \text{\#Step 3}
 \end{aligned}$$

*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).

$$\begin{aligned}
 &= (20 - 30) + 40 \quad \text{\#Step 1} \\
 &= -10 + 40 \quad \text{\#Step 2} \\
 &= 30 \quad \text{\#Step 3}
 \end{aligned}$$

*Example 5.11* How will Python evaluate the following expression?

$$(20 + 30) * 40$$

*Solution:*

$= (20 + 30) * 40 \quad \text{\# Step 1}$   
 #using parenthesis(), we have forced precedence of  $+$  to be more than that of  $*$

$$\begin{aligned}
 &= 50 * 40 \quad \text{\# Step 2} \\
 &= 2000 \quad \text{\# Step 3}
 \end{aligned}$$

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

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

*Solution:*

$$\begin{aligned}
 &= 15.0 / 4 + (8.0 + 3.0) \quad \text{\#Step 1} \\
 &= 15.0 / 4.0 + 11.0 \quad \text{\#Step 2} \\
 &= 3.75 + 11.0 \quad \text{\#Step 3} \\
 &= 14.75 \quad \text{\#Step 4}
 \end{aligned}$$

## 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           |
|-----------------------------------------------------|------------------|
| <code>print("Hello")</code>                         | Hello            |
| <code>print(10*2.5)</code>                          | 25.0             |
| <code>print("I" + "love" + "my" + "country")</code> | Ilovemycountry   |
| <code>print("I'm", 16, "years old")</code>          | 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

**NOTES**

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 |

|                     |                                                        |
|---------------------|--------------------------------------------------------|
| <code>str(x)</code> | Converts x to a string representation                  |
| <code>chr(x)</code> | Converts ASCII value of x to character                 |
| <code>ord(x)</code> | 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 )
```

```

Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
=====
RESTART: C:\NCERT\prog5-7.py =====
Traceback (most recent call last):
  File "C:\NCERT\prog5-7.py", line 7, in <module>
    print("The total is Rs." + totalPrice)
TypeError: can only concatenate str (not "int") to str
>>>

```

*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)
```

```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART: C:\NCERT\prog5-11.py =====
num2 = apple
Traceback (most recent call last):
  File "C:\NCERT\prog5-11.py", line 5, in <module>
    num2 = int(input("num2 = "))
ValueError: invalid literal for int() with base 10: 'apple'
>>>

===== RESTART: C:\NCERT\prog5-11.py =====
num2 = 0
Traceback (most recent call last):
  File "C:\NCERT\prog5-11.py", line 7, in <module>
    print(num1/num2)
ZeroDivisionError: float division by zero
>>> ===== RESTART: C:\NCERT\prog5-11.py =====
num2 = 10
1.0
>>> |
```

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.

**NOTES**

- 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:

- Assign 10 to variable `length` and 20 to variable `breadth`.
  - Assign the average of values of variables `length` and `breadth` to a variable `sum`.
  - Assign a list containing strings ‘Paper’, ‘Gel Pen’, and ‘Eraser’ to a variable `stationery`.
  - Assign the strings ‘Mohandas’, ‘Karamchand’, and ‘Gandhi’ to variables `first`, `middle` and `last`.
  - 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):
- The sum of 20 and -10 is less than 12.
  - `num3` is not more than 24.

**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.
- $0 == 1 == 2$
  - $2 + 3 == 4 + 5 == 7$
  - $1 < -1 == 3 > 4$
5. Write the output of the following:
- ```
num1 = 4
num2 = num1 + 1
num1 = 2
print (num1, num2)
```
  - ```
num1, num2 = 2, 6
num1, num2 = num2, num1 + 2
print (num1, num2)
```
  - ```
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?
- Number of months in a year
  - Resident of Delhi or not
  - Mobile number
  - Pocket money
  - Volume of a sphere
  - Perimeter of a square
  - Name of the student
  - Address of the student
7. Give the output of the following when num1 = 4, num2 = 3, num3 = 2
- ```
num1 += num2 + num3
print (num1)
```
  - ```
num1 = num1 ** (num2 + num3)
print (num1)
```
  - ```
num1 **= num2 + num3
```
  - ```
num1 = '5' + '5'
print (num1)
```

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:  $T(^{\circ}\text{F}) = T(^{\circ}\text{C}) \times 9/5 + 32$ )

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

## NOTES

**NOTES**

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

Amount payable = Principal + SI.

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

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 ( $c = \text{about } 3 \times 10^8 \text{ m/s}$ ) 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

**NOTES**

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.

<b>Name of School</b>	
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

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.

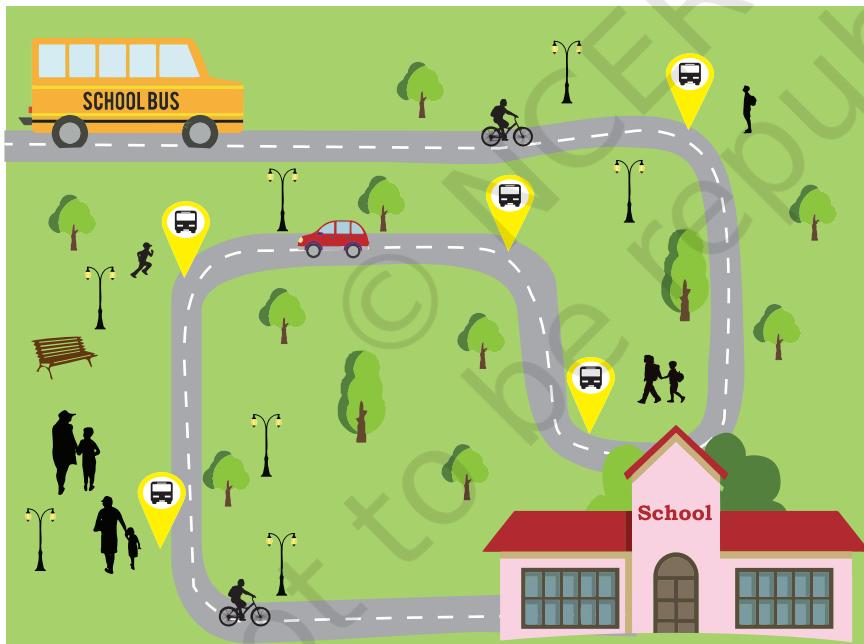


Figure 6.1: Bus carrying students to school

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.

“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 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.

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

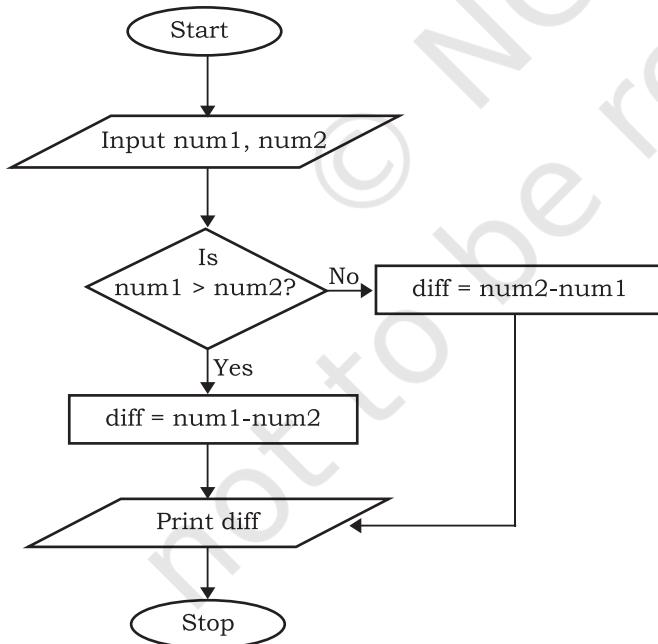


Figure 6.2: Flow chart depicting decision making

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`.

**NOTES**

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.

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).



Figure 6.3: Iterative process occurring in nature

4. Repeat steps 2 and 3 as long as count has a value less than or equal to 100,000 (count  $\leq 100,000$ ).

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  $\leq 100000$ , 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.

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.

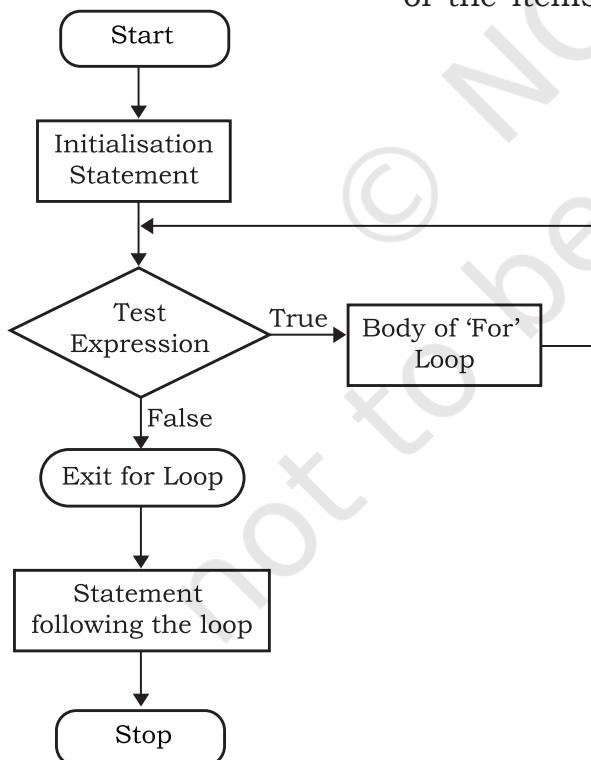


Figure 6.4: Flow chart of for loop

#### (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
>>> list(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.

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
```

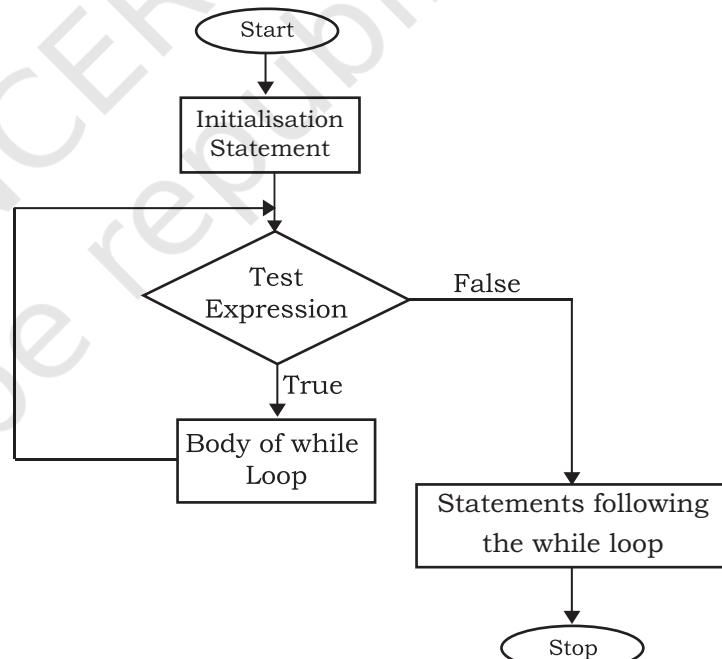


Figure 6.5: Flow chart of while Loop

**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 delimiter
        #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.

### 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.

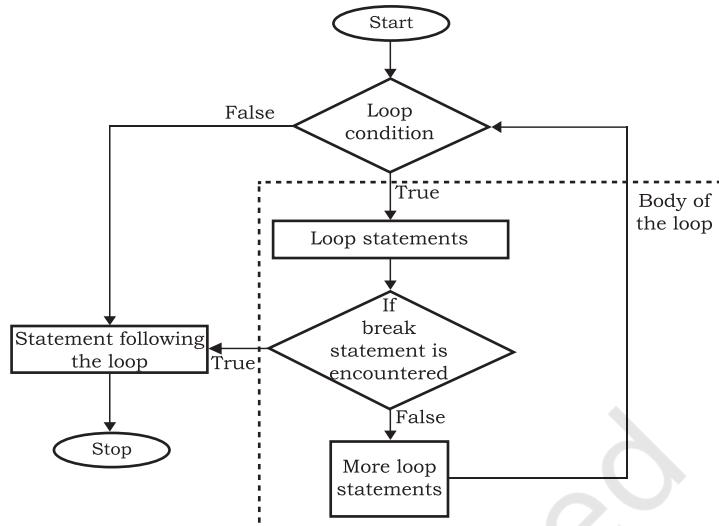


Figure 6.5: Flowchart for using break statement in loop

**Program 6-13** Find the sum of all the positive numbers entered by the user. As soon as the user enters a negative 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.

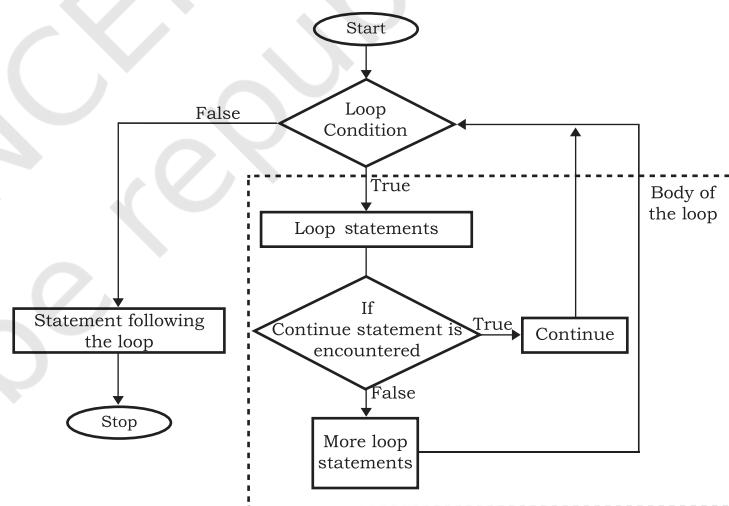


Figure 6.6: Flow chart of continue statement

### 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

```

## 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.

## NOTES

## 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  $1 + \frac{1}{8} + \frac{1}{27} + \dots + \frac{1}{n^3}$ , 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)	1 2 3 4 5 1 2 3 4 1 2 3 1 2 1	iv)	* * * * * * * * * *

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

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.

*“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

#### 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.

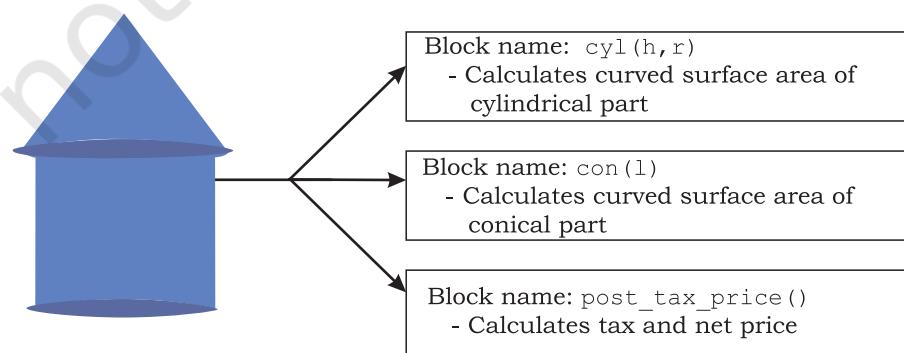


Figure 7.2: Calculation of the cost of the tent

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,....]): Function Header  
    set of instructions to be executed  
    [return <value>] } 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
```

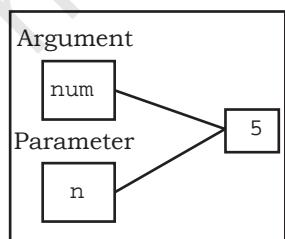


Figure 7.3: Both argument and parameter refers to the same value

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):

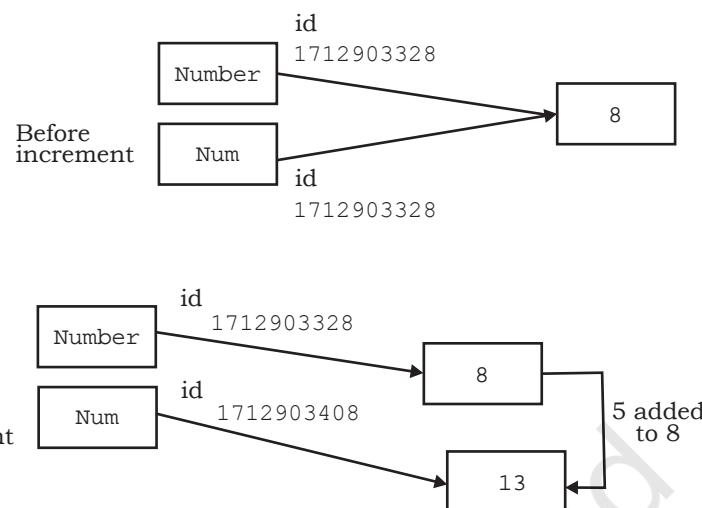


Figure 7.4: ID of argument and parameter before and after increment

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  $\text{Base}^{\text{exponent}}$  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  $\text{Base}^{\text{exponent}}$ 
#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 to the point of function call so that the remaining statements in the program can be executed. Therefore, when we read a program, we should not simply read from top to bottom. Instead, we should follow the flow of control or execution. It is also important to note that a function must be defined before its call within a program.

#### 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 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.

Figure 7.5: Order of execution of statements

**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:

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.

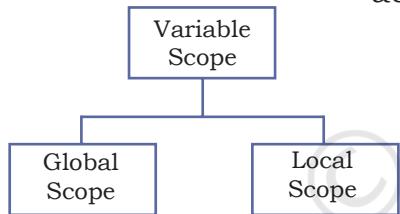


Figure 7.6: Scope of a variable

##### **(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) → y generates error when it is
NameError: name 'y' is not defined
```

→ 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)
```

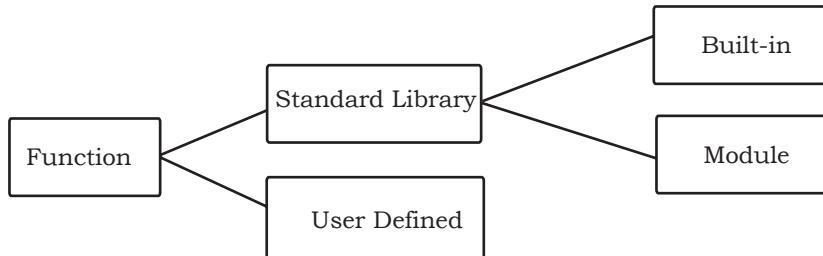
**Output:**

```

Accessing num = 5 ← Global variable num is accessed as the ambiguity is resolved by
num reassigned = 10 prefixing global to it
Accessing num outside myfunc1 10

```

## 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:

Built-in Functions			
Input or Output	Datatype Conversion	Mathematical Functions	Other Functions
<code>input()</code> <code>print()</code>	<code>bool()</code> <code>chr()</code> <code>dict()</code> <code>float()</code> <code>int()</code> <code>list()</code> <code>ord()</code> <code>set()</code> <code>str()</code> <code>tuple()</code>	<code>abs()</code> <code>divmod()</code> <code>max()</code> <code>min()</code> <code>pow()</code> <code>sum()</code>	<code>__import__()</code> <code>len()</code> <code>range()</code> <code>type()</code>

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
<code>abs(x)</code>	x may be an integer or floating point number	Absolute value of x	<code>&gt;&gt;&gt; abs(4)</code> 4 <code>&gt;&gt;&gt; abs(-5.7)</code> 5.7
<code>divmod(x,y)</code>	x and y are integers	A tuple: (quotient, remainder)	<code>&gt;&gt;&gt; divmod(7,2)</code> (3, 1) <code>&gt;&gt;&gt; divmod(7.5,2)</code> (3.0, 1.5) <code>&gt;&gt;&gt; divmod(-7,2)</code> (-4, 1)
<code>max(sequence)</code> or <code>max(x,y,z,...)</code>	x,y,z... may be integer or floating point number	Largest number in the sequence/ largest of two or more arguments	<code>&gt;&gt;&gt; max([1,2,3,4])</code> 4 <code>&gt;&gt;&gt; max("Sincerity")</code> '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

#### 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



Remember, Python is case sensitive. All the module names are in lowercase.

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 <sup>y</sup> (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:

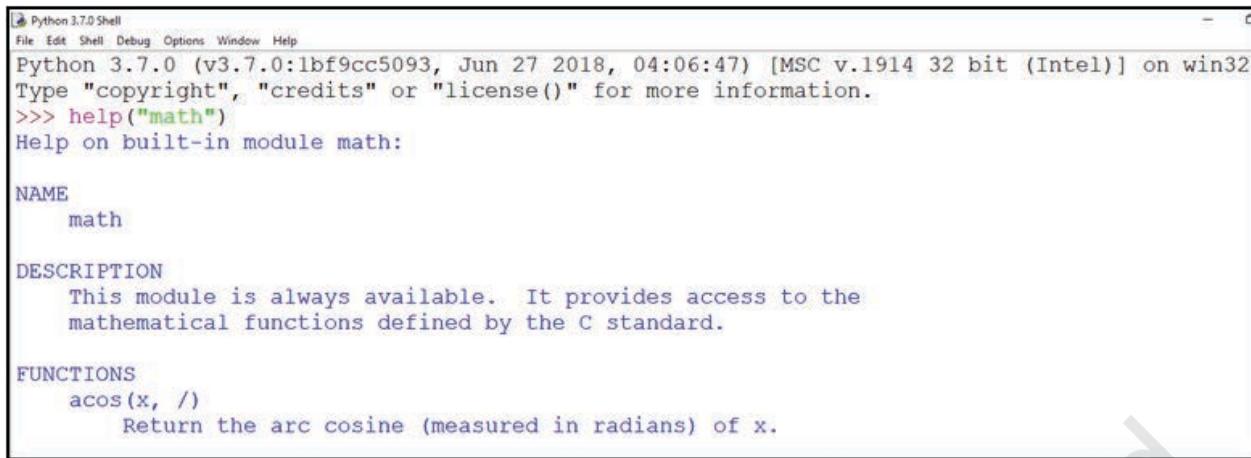
```
import statistics
```

**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")`



```

Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> help("math")
Help on built-in module math:

NAME
    math

DESCRIPTION
    This module is always available. It provides access to the
    mathematical functions defined by the C standard.

FUNCTIONS
    acos(x, /)
        Return the arc cosine (measured in radians) of x.

```

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
                                #basic_math module
7
>>> a = basic_math.subnum(2,5) #Call subnum() function of the
                                #basic_math module
-3
>>> a = basic_math.multnum(2,5) #Call multnum() function of the
                                #basic_math module
10
>>> a = basic_math.divnum(2,5) #Call divnum() function of the
                                #basic_math module
0.4
>>> a = basic_math.divnum(2,0) #Call divnum() function of the
                                #basic_math module
Zero Divide Error
```

`__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
```

## 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

**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( vall = 1.1, val2, val3):
    final = (val2 + val3)/ vall
    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 "StOrE@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
$\geq 500$ and $< 1000$	5%
$\geq 1000$ and $< 2000$	8%
$\geq 2000$	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.

not to be republished  
© NCERT

# CHAPTER 8

## STRINGS



11120CH08

### 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'''
```

*“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

#### 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!'**

<b>Positive Indices</b>	0	1	2	3	4	5	6	7	8	9	10	11
<b>String</b>	H	e	l	l	o		W	o	r	l	d	!
<b>Negative Indices</b>	-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
                                         #after this operation.

>>> str1
'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]
'Hiwl'
```

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	<pre>&gt;&gt;&gt; str1 = 'Hello World!' &gt;&gt;&gt; len(str1) 12</pre>
title()	Returns the string with first letter of every word in the string in uppercase and rest in lowercase	<pre>&gt;&gt;&gt; str1 = 'hello WORLD!' &gt;&gt;&gt; str1.title() 'Hello World!'</pre>

lower()	Returns the string with all uppercase letters converted to lowercase	<pre>&gt;&gt;&gt; str1 = 'hello WORLD!' &gt;&gt;&gt; str1.lower() 'hello world!'</pre>
upper()	Returns the string with all lowercase letters converted to uppercase	<pre>&gt;&gt;&gt; str1 = 'hello WORLD!' &gt;&gt;&gt; str1.upper() 'HELLO WORLD!'</pre>
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	<pre>&gt;&gt;&gt; str1 = 'Hello World! Hello Hello' &gt;&gt;&gt; str1.count('Hello',12,25) 2 &gt;&gt;&gt; str1.count('Hello') 3</pre>
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	<pre>&gt;&gt;&gt; str1 = 'Hello World! Hello Hello' &gt;&gt;&gt; str1.find('Hello',10,20) 13 &gt;&gt;&gt; str1.find('Hello',15,25) 19 &gt;&gt;&gt; str1.find('Hello') 0 &gt;&gt;&gt; str1.find('Hee') -1</pre>
index(str, start, end)	Same as find() but raises an exception if the substring is not present in the given string	<pre>&gt;&gt;&gt; str1 = 'Hello World! Hello Hello' &gt;&gt;&gt; str1.index('Hello') 0 &gt;&gt;&gt; str1.index('Hee') ValueError: substring not found</pre>
endswith()	Returns True if the given string ends with the supplied substring otherwise returns False	<pre>&gt;&gt;&gt; str1 = 'Hello World!' &gt;&gt;&gt; str1.endswith('World!') True &gt;&gt;&gt; str1.endswith('!') True &gt;&gt;&gt; str1.endswith('lde') False</pre>
startswith()	Returns True if the given string starts with the supplied substring otherwise returns False	<pre>&gt;&gt;&gt; str1 = 'Hello World!' &gt;&gt;&gt; str1.startswith('He') True &gt;&gt;&gt; str1.startswith('Hee') False</pre>

isalnum()	<p>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</p>	<pre>&gt;&gt;&gt; str1 = 'HelloWorld' &gt;&gt;&gt; str1.isalnum() True  &gt;&gt;&gt; str1 = 'HelloWorld2' &gt;&gt;&gt; str1.isalnum() True  &gt;&gt;&gt; str1 = 'HelloWorld!!!' &gt;&gt;&gt; str1.isalnum() False</pre>
islower()	<p>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</p>	<pre>&gt;&gt;&gt; str1 = 'hello world!' &gt;&gt;&gt; str1.islower() True  &gt;&gt;&gt; str1 = 'hello 1234' &gt;&gt;&gt; str1.islower() True  &gt;&gt;&gt; str1 = 'hello ??' &gt;&gt;&gt; str1.islower() True  &gt;&gt;&gt; str1 = '1234' &gt;&gt;&gt; str1.islower() False  &gt;&gt;&gt; str1 = 'Hello World!' &gt;&gt;&gt; str1.islower() False</pre>
isupper()	<p>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</p>	<pre>&gt;&gt;&gt; str1 = 'HELLO WORLD!' &gt;&gt;&gt; str1.isupper() True  &gt;&gt;&gt; str1 = 'HELLO 1234' &gt;&gt;&gt; str1.isupper() True  &gt;&gt;&gt; str1 = 'HELLO ??' &gt;&gt;&gt; str1.isupper() True  &gt;&gt;&gt; str1 = '1234' &gt;&gt;&gt; str1.isupper() False  &gt;&gt;&gt; str1 = 'Hello World!' &gt;&gt;&gt; str1.isupper() False</pre>

isspace()	Returns True if the string is non-empty and all characters are white spaces (blank, tab, newline, carriage return)	<pre>&gt;&gt;&gt; str1 = '      \n      \t \r' &gt;&gt;&gt; str1.isspace() True &gt;&gt;&gt; str1 = 'Hello          \n' &gt;&gt;&gt; str1.isspace() False</pre>
istitle()	Returns True if the string is non-empty and title case, i.e., the first letter of every word in the string in uppercase and rest in lowercase	<pre>&gt;&gt;&gt; str1 = 'Hello World!' &gt;&gt;&gt; str1.istitle() True &gt;&gt;&gt; str1 = 'hello World!' &gt;&gt;&gt; str1.istitle() False</pre>
lstrip()	Returns the string after removing the spaces only on the left of the string	<pre>&gt;&gt;&gt; str1 = '           Hello World! ' &gt;&gt;&gt; str1.lstrip() 'Hello World! '</pre>
rstrip()	Returns the string after removing the spaces only on the right of the string	<pre>&gt;&gt;&gt; str1 = '           Hello World!' &gt;&gt;&gt; str1.rstrip() '           Hello World! '</pre>
strip()	Returns the string after removing the spaces both on the left and the right of the string	<pre>&gt;&gt;&gt; str1 = '           Hello World!' &gt;&gt;&gt; str1.strip() 'Hello World! '</pre>
replace(oldstr, newstr)	Replaces all occurrences of old string with the new string	<pre>&gt;&gt;&gt; str1 = 'Hello World!' &gt;&gt;&gt; str1.replace('o','*') 'Hell* W*rld!' &gt;&gt;&gt; str1 = 'Hello World!' &gt;&gt;&gt; str1.replace('World','Country') 'Hello Country!' &gt;&gt;&gt; str1 = 'Hello World! Hello' &gt;&gt;&gt; str1.replace('Hello','Bye') 'Bye World! Bye'</pre>
join()	Returns a string in which the characters in the string have been joined by a separator	<pre>&gt;&gt;&gt; str1 = ('HelloWorld!') &gt;&gt;&gt; str2 = '-'      #separator &gt;&gt;&gt; str2.join(str1) 'H-e-l-l-o-W-o-r-l-d-!'</pre>

<b>partition()</b> Partitions the given string at the first occurrence of the substring (separator) and returns the string partitioned into three parts. 1. Substring before the separator 2. Separator 3. Substring after the separator If the separator is not found in the string, it returns the whole string itself and two empty strings	<pre>&gt;&gt;&gt; str1 = 'India is a Great Country' &gt;&gt;&gt; str1.partition('is') ('India ', 'is', ' a Great Country') &gt;&gt;&gt; str1.partition('are') ('India is a Great Country', ' ', ' ')</pre>
<b>split()</b> Returns a list of words delimited by the specified substring. If no delimiter is given then words are separated by space.	<pre>&gt;&gt;&gt; str1 = 'India is a Great Country' &gt;&gt;&gt; str1.split() ['India', 'is', 'a', 'Great', 'Country'] &gt;&gt;&gt; str1 = 'India is a Great Country' &gt;&gt;&gt; str1.split('a') ['Indi', ' is ', ' Gre', 't Country']</pre>

## 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
```

## 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.

## NOTES

## 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

### 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.

*“Measuring programming progress by lines of code is like measuring aircraft building progress by weight.”*

—Bill Gates

#### 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
<code>len()</code>	Returns the length of the list passed as the argument	<code>&gt;&gt;&gt; list1 = [10, 20, 30, 40, 50]</code> <code>&gt;&gt;&gt; len(list1)</code> <code>5</code>
<code>list()</code>	Creates an empty list if no argument is passed	<code>&gt;&gt;&gt; list1 = list()</code> <code>&gt;&gt;&gt; list1</code>

	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()	Returns minimum or smallest element of the list	>>> list1 = [34,12,63,39,92,44] >>> min(list1) 12
max()	Returns maximum or largest element of the list	>>> max(list1) 92
sum()	Returns sum of the elements of the list	>>> 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

- » Introduction to Tuples
- » Tuple Operations
- » Tuple Methods and Built-in Functions
- » Tuple Assignment
- » Nested Tuples
- » Tuple Handling
- » Introduction to Dictionaries
- » Dictionaries are Mutable
- » Dictionary Operations
- » Traversing a Dictionary
- » Dictionary Methods and Built-in Functions
- » Manipulating Dictionaries



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)<br>>>> len(tuple1)<br>5                                                                                                                                                                                                               |
| tuple() | Creates an empty tuple if no argument is passed<br><br>Creates a tuple if a sequence is passed as argument | >>> tuple1 = tuple()<br>>>> tuple1<br>( )<br>>>> tuple1 = tuple('aeiou')#string<br>>>> tuple1<br>('a', 'e', 'i', 'o', 'u')<br><br>>>> tuple2 = tuple([1,2,3]) #list<br>>>> tuple2<br>(1, 2, 3)<br><br>>>> tuple3 = tuple(range(5))<br>>>> tuple3<br>(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)<br>>>> tuple1.count(10)<br>3<br>>>> tuple1.count(90)<br>0                                                                                                                                                                       |
| index() | Returns the index of the first occurrence of the element in the given tuple                                | >>> tuple1 = (10,20,30,40,50)<br>>>> tuple1.index(30)<br>2<br>>>> tuple1.index(90)<br>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")<br>>>> sorted(tuple1)<br>['Aditya', 'Heena', 'Mohsin', 'Raj', 'Rama'] |
| min()    | Returns minimum or smallest element of the tuple                                                                                            | >>> tuple1 = (19,12,56,18,9,87,34)<br>>>> min(tuple1)<br>9                                                                      |
| max()    | Returns maximum or largest element of the tuple                                                                                             | >>> max(tuple1)<br>87                                                                                                           |
| sum()    | Returns sum of the elements of the tuple                                                                                                    | >>> sum(tuple1)<br>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
```

## 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:")
```



\t is an escape character used for adding horizontal tab space. Another commonly used escape character is \n, used for inserting a new line.

```
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**

| <b>Method</b> | <b>Description</b>                                                                        | <b>Example</b>                                                                                                                                                                                                                                                            |
|---------------|-------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| len()         | Returns the length or number of key: value pairs of the dictionary passed as the argument | <pre>&gt;&gt;&gt; dict1 = {'Mohan':95,'Ram':89, 'Suhel':92, 'Sangeeta':85} &gt;&gt;&gt; len(dict1) 4</pre>                                                                                                                                                                |
| dict()        | Creates a dictionary from a sequence of key-value pairs                                   | <pre>pair1 = [ ('Mohan',95), ('Ram',89), ('Suhel',92), ('Sangeeta',85)] &gt;&gt;&gt; pair1 [ ('Mohan', 95), ('Ram', 89), ('Suhel', 92), ('Sangeeta', 85)] &gt;&gt;&gt; dict1 = dict(pair1) &gt;&gt;&gt; dict1 {'Mohan': 95, 'Ram': 89, 'Suhel': 92, 'Sangeeta': 85}</pre> |
| keys()        | Returns a list of keys in the dictionary                                                  | <pre>&gt;&gt;&gt; dict1 = {'Mohan':95, 'Ram':89, 'Suhel':92, 'Sangeeta':85} &gt;&gt;&gt; dict1.keys() dict_keys(['Mohan', 'Ram', 'Suhel', 'Sangeeta'])</pre>                                                                                                              |
| values()      | Returns a list of values in the dictionary                                                | <pre>&gt;&gt;&gt; dict1 = {'Mohan':95, 'Ram':89, 'Suhel':92, 'Sangeeta':85} &gt;&gt;&gt; dict1.values() dict_values([95, 89, 92, 85])</pre>                                                                                                                               |
| items()       | Returns a list of tuples(key – value) pair                                                | <pre>&gt;&gt;&gt; dict1 = {'Mohan':95, 'Ram':89, 'Suhel':92, 'Sangeeta':85} &gt;&gt;&gt; dict1.items() dict_items([('Mohan', 95), ('Ram', 89), ('Suhel', 92), ('Sangeeta', 85)])</pre>                                                                                    |

|          |                                                                                                                                          |                                                                                                                                                                                                                                                                                                                                                    |
|----------|------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| get()    | Returns the value corresponding to the key passed as the argument<br><br>If the key is not present in the dictionary it will return None | <pre>&gt;&gt;&gt; dict1 = { 'Mohan':95, 'Ram':89, 'Suhel':92, 'Sangeeta':85} &gt;&gt;&gt; dict1.get('Sangeeta') 85 &gt;&gt;&gt; dict1.get('Sohan') &gt;&gt;&gt;</pre>                                                                                                                                                                              |
| update() | appends the key-value pair of the dictionary passed as the argument to the key-value pair of the given dictionary                        | <pre>&gt;&gt;&gt; dict1 = { 'Mohan':95, 'Ram':89, 'Suhel':92, 'Sangeeta':85} &gt;&gt;&gt; dict2 = { 'Sohan':79, 'Geeta':89} &gt;&gt;&gt; dict1.update(dict2) &gt;&gt;&gt; dict1 { 'Mohan': 95, 'Ram': 89, 'Suhel': 92, 'Sangeeta': 85, 'Sohan': 79, 'Geeta': 89} &gt;&gt;&gt; dict2 { 'Sohan': 79, 'Geeta': 89}</pre>                              |
| del()    | Deletes the item with the given key<br><br>To delete the dictionary from the memory we write:<br>del Dict_name                           | <pre>&gt;&gt;&gt; dict1 = { 'Mohan':95, 'Ram':89, 'Suhel':92, 'Sangeeta':85} &gt;&gt;&gt; del dict1['Ram'] &gt;&gt;&gt; dict1 { 'Mohan':95, 'Suhel':92, 'Sangeeta': 85}  &gt;&gt;&gt; del dict1 ['Mohan'] &gt;&gt;&gt; dict1 { 'Suhel': 92, 'Sangeeta': 85} &gt;&gt;&gt; del dict1 &gt;&gt;&gt; dict1 NameError: name 'dict1' is not defined</pre> |
| clear()  | Deletes or clear all the items of the dictionary                                                                                         | <pre>&gt;&gt;&gt; dict1 = { 'Mohan':95, 'Ram':89, 'Suhel':92, 'Sangeeta':85} &gt;&gt;&gt; dict1.clear() &gt;&gt;&gt; dict1 { }</pre>                                                                                                                                                                                                               |

## 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.

## NOTES

## 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**

- 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:
- `print(stateCapital.get("Bihar"))`
  - `print(stateCapital.keys())`
  - `print(stateCapital.values())`
  - `print(stateCapital.items())`
  - `print(len(stateCapital))`
  - `print("Maharashtra" in stateCapital)`
  - `print(stateCapital.get("Assam"))`
  - `del stateCapital["Rajasthan"]`  
`print(stateCapital)`
- “Lists and Tuples are ordered”. Explain.
  - With the help of an example show how can you return more than one value from a function.
  - What advantages do tuples have over lists?
  - When to use tuple or dictionary in Python. Give some examples of programming situations mentioning their usefulness.
  - Prove with the help of an example that the variable is rebuilt in case of immutable data types.
  - TypeError occurs while statement 2 is running. Give reason. How can it be corrected?  
`>>> tuple1 = (5) #statement 1`  
`>>> len(tuple1) #statement 2`

**PROGRAMMING PROBLEMS**

- 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()]
- 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.

**NOTES**

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

**NOTES**

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



### 11.1 INTRODUCTION

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.

### 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

*"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 this chapter

- » Introduction
- » Digital Footprint
- » Digital Society and Netizen
- » Data Protection
- » Cyber Crime
- » Indian IT Act
- » Impact on Health

### 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.

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.



Figure 11.1: Exemplar web applications that result in digital footprints

### 11.3 DIGITAL SOCIETY AND NETIZEN

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.

##### (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

#### Activity 11.1

As a digital citizen, list various services that you avail online.

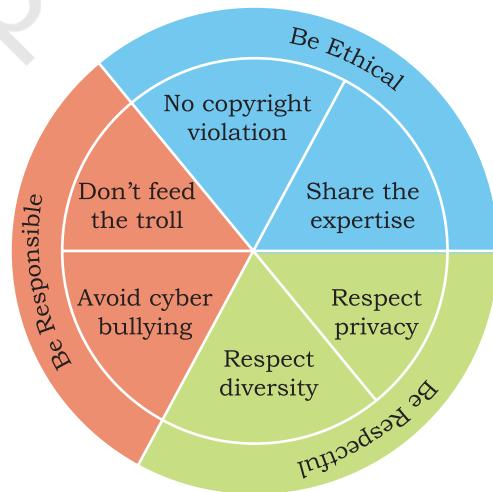


Figure 11.2: Net Etiquettes

**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.

redundant information, we should verify that the information is not available already on Internet.

**(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.

#### **(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.

#### **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

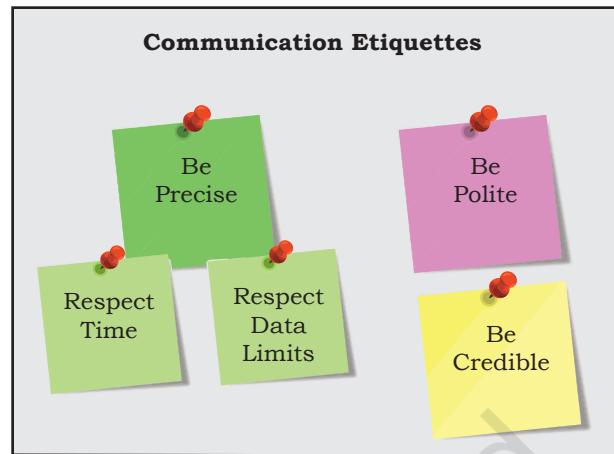


Figure 11.3: Communication etiquettes



#### **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.



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.

#### (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.

#### (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

#### Think and Reflect

Is having the same password for all your accounts on different websites safe?



#### Play Safe!!

Think carefully before sharing personal photos.

need to be cautious while uploading or sending sensitive or confidential files which have a bearing on our privacy.

#### 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.

##### 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.

##### Activity 11.3

Suppose someone's email password is 'tecnology' which is weak. Can you suggest a stronger password?

##### 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?



### **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 as 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 build 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 following 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.

**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.

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.

## 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.

### 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



#### 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).

#### 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.

### 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

#### Activity 11.6

Explore and find out how to file a complaint with the cyber cell in your area.

- 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.



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.

## 11.6 INDIAN INFORMATION TECHNOLOGY ACT (IT ACT)

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.

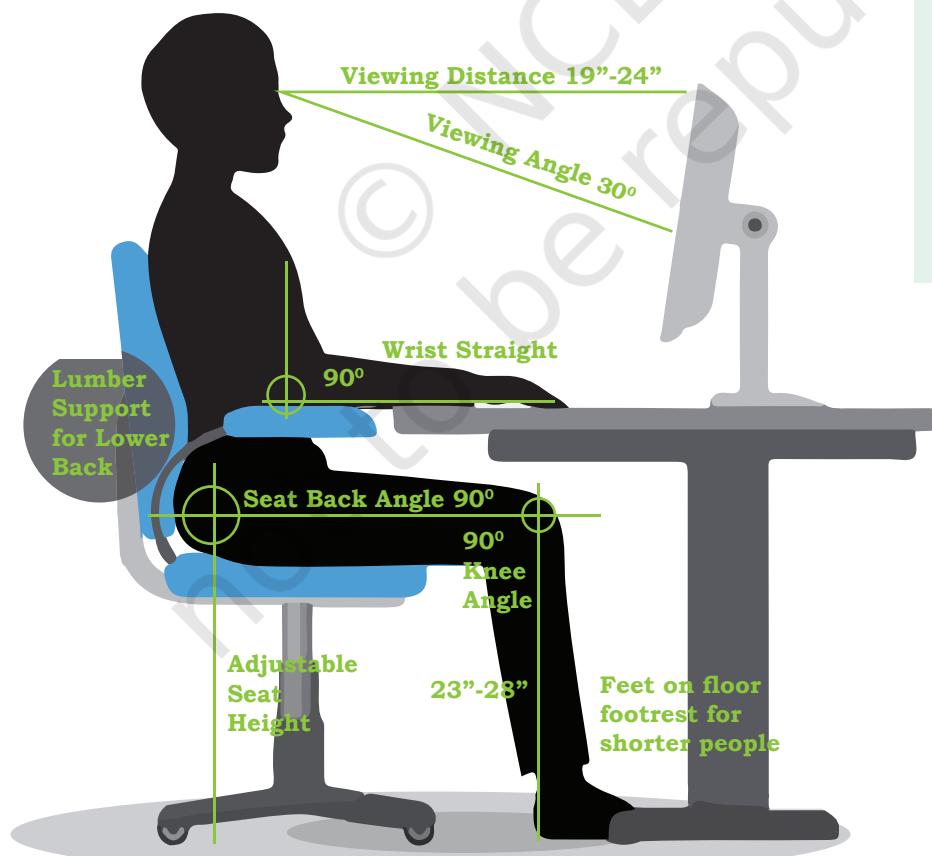


Figure 11.5: Correct posture while sitting in front of a computer



#### 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

**NOTES**

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:

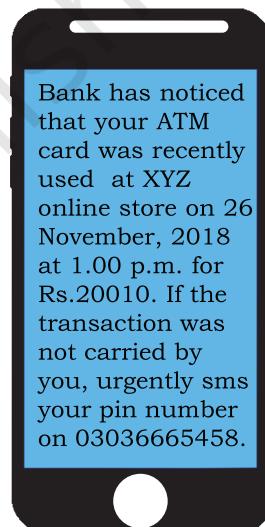
| <b>Column A</b>    | <b>Column B</b>                                                                                                     |
|--------------------|---------------------------------------------------------------------------------------------------------------------|
| Plagiarism         | 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:

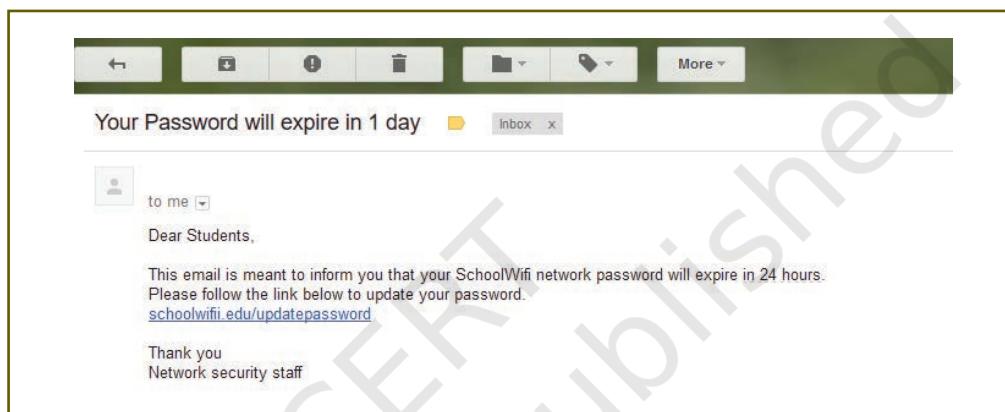
- a) Will you SMS your pin number to the given contact number?
  - b) Will you call the bank helpline number to recheck the validity of the SMS received?
8. Preeti celebrated her birthday with her family. She was excited to share the moments with her friend Himanshu. She uploaded selected images of her 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](http://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.



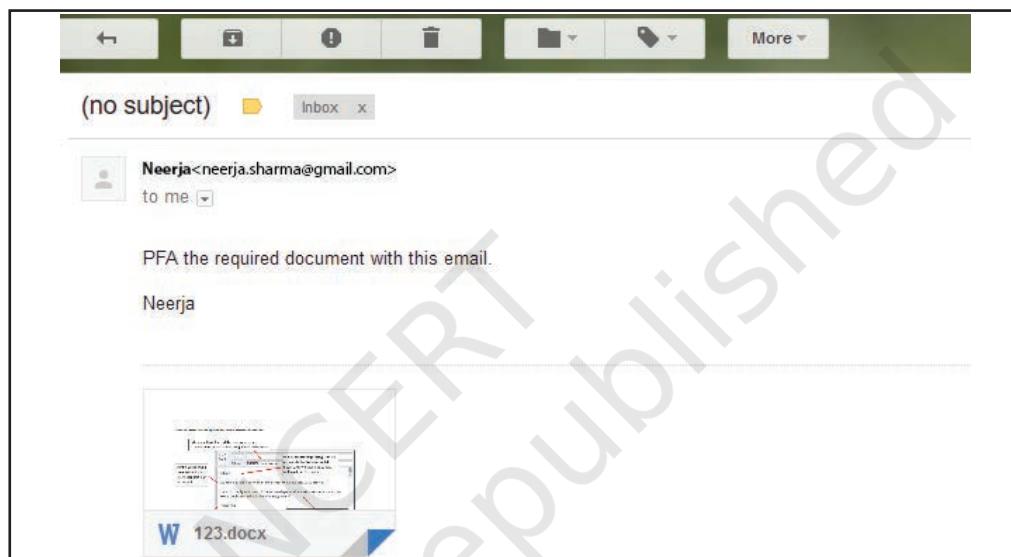
- a) Do you find any discrepancy in this email?
  - b) What will happen if the student will click on the given URL?
  - c) Is the email an example of cyber crime? If yes, then specify which type of cyber crime is it. Justify your answer.
10. You are planning to go for a vacation. You surfed the Internet 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 combat such situations.
12. Write the differences between the following-
- a) Copyrights and Patents

**NOTES**

- 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.
- Identity theft
  - 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.

- Subject of the mail
  - Formal greeting
  - Self-explanatory terms
  - Identity of the sender
  - 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**

---

© NCERT  
not to be republished

## **NOTES**

---

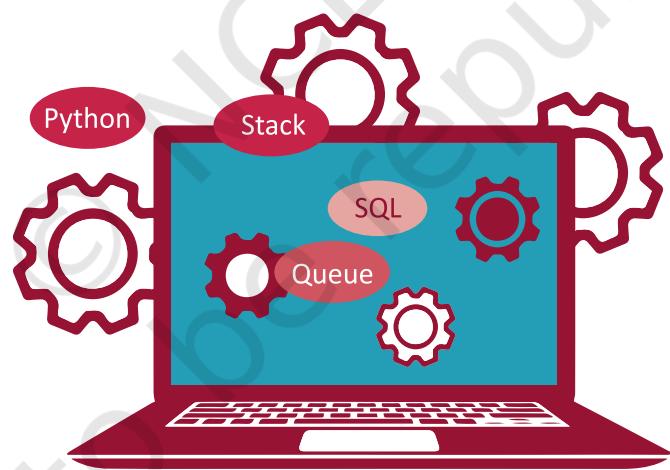
© NCERT  
not to be republished

# **COMPUTER SCIENCE**

**TEXTBOOK FOR CLASS XII**



12130



**राष्ट्रीय शैक्षिक अनुसंधान और प्रशिक्षण परिषद्**  
**NATIONAL COUNCIL OF EDUCATIONAL RESEARCH AND TRAINING**

**12130 – COMPUTER SCIENCE**

Textbook for Class XII

**ISBN 978-93-5292-338-0**

**First Edition**

September 2020 Bhadrapada 1942

**Reprinted**

September 2021 Bhadrapada 1943

December 2021 Agrahayana 1943

October 2022 Kartika 1944

**PD 15T RPS**

**© National Council of Educational  
Research and Training, 2020**

**₹ 240.00**

Printed on 80 GSM paper with NCERT  
watermark

Published at the Publication Division  
by the Secretary, National Council of  
Educational Research and Training,  
Sri Aurobindo Marg, New Delhi 110 016  
and printed at Shiva Printtech Pvt. Ltd.,  
Plot No. 12, Pocket G, Bavana Industrial  
Area, Sector 4, Delhi 110039

**ALL RIGHTS RESERVED**

- No part of this publication may be reproduced, stored in a retrieval system or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise without the prior permission of the publisher.
- This book is sold subject to the condition that it shall not, by way of trade, be lent, re-sold, hired out or otherwise disposed off without the publisher's consent, in any form of binding or cover other than that in which it is published.
- The correct price of this publication is the price printed on this page. Any revised price indicated by a rubber stamp or by a sticker or by any other means is incorrect and should be unacceptable.

**OFFICES OF THE PUBLICATION**

**DIVISION, NCERT**

NCERT Campus  
Sri Aurobindo Marg  
**New Delhi 110 016**

Phone : 011-26562708

108, 100 Feet Road  
Hosdakere Halli Extension  
Banashankari III Stage  
**Bengaluru 560 085**

Phone : 080-26725740

Navjivan Trust Building  
P.O.Navjivan  
**Ahmedabad 380 014**

Phone : 079-27541446

CWC Campus  
Opp. Dhankal Bus Stop  
Panihati  
**Kolkata 700 114**

Phone : 033-25530454

CWC Complex  
Maligaon  
**Guwahati 781 021**

Phone : 0361-2674869

**Publication Team**

Head, Publication Division : *Anup Kumar Rajput*

Chief Production Officer : *Arun Chitkara*

Chief Business Manager : *Vipin Dewan*

Chief Editor (In charge) : *Bijnan Sutar*

Assistant Production Officer : *Mukesh Gaur*

**Cover and Layout**

*Meetu Sharma, DTP Operator, DESM*



## FOREWORD

Computer science as a discipline has evolved over the years and has emerged as a driving force of our socio-economic activities. It has made continuous inroads into diverse areas — be it business, commerce, science, technology, sports, health, transportation or education. With the advent of computer and communication technologies, there has been a paradigm shift in teaching-learning at the school level. The role and relevance of this discipline is in focus because the expectations from the school pass-outs have grown to be able to meet the challenges of the 21st century. Today, we are living in an interconnected world where computer-based applications influence the way we learn, communicate, commute or even socialise!

There is a demand for software engineers in various fields like manufacturing, services, etc. Today, there are a large number of successful startups delivering different services through software applications. All these have resulted in generating interest for this subject among students as well as parents.

Development of logical thinking, reasoning and problem-solving skills are fundamental building blocks for knowledge acquisition at the higher level. Computer plays a key role in problem solving with focus on logical representation or reasoning and analysis.

This textbook focuses on the fundamental concepts and problem-solving skills while opening a window to the emerging and advanced areas of computer science. The newly developed syllabus has dealt with the dual challenge of reducing curricular load as well as introducing this ever evolving discipline. This textbook also provides space to Computational Thinking and Artificial Intelligence, which envisaged in National Education Policy, 2020.

As an organisation committed to systemic reforms and continuous improvement in the quality of its products, NCERT welcomes comments and suggestions which will enable us to revise the content of the textbook.

New Delhi  
August 2020

HRUSHIKESH SENAPATY  
*Director*  
National Council of Educational  
Research and Training

not to be republished  
© NCERT



## PREFACE

In the present education system of our country, specialised or discipline based courses are introduced at the higher secondary stage. This stage is crucial as well as challenging because of the transition from general to discipline-based curriculum. The syllabus at this stage needs to have sufficient rigour and depth while remaining mindful of the comprehension level of the learners. Further, the textbook should not be heavily loaded with content.

Computers have permeated in every facet of life. Study of basic concepts of computer science has been desirable in education. There are courses offered in the name of Computer Science, Information and Communication Technology (ICT), Information Technology (IT), etc., by various boards and schools up to secondary stage, as optional. These mainly focus on using computer for word processing, presentation tools and application software.

Computer Science (CS) at the higher secondary stage of school education is also offered as an optional subject. At this stage, students usually opt for CS with an aim of pursuing a career in software development or related areas, after going through professional courses at higher levels. Therefore, at higher secondary stage, the curriculum of CS introduces basics of computing and sufficient conceptual background of Computer Science.

The primary focus is on fostering the development of computational thinking and problem-solving skills. This book has 13 chapters covering the following broader themes:

- Data Structure: understanding of important data structure Stack, Queue; Searching and Sorting techniques.
- Database: basic understanding of data, database concepts, and relational database management system using MySQL. Structured query language—data definition, data manipulation and data querying.
- Programming: handling errors and exceptions in programs written in Python; handling files and performing file operations in Python.
- Network and Communication: fundamentals of Computers networks, devices, topologies, Internet, Web and IoT, DNS. Basics of Data communication—transmission channel, media; basics of protocols, mobile communication generations.
- Security Aspects: introduction to basic concepts related to network and Internet security, threats and prevention.

Each chapter has two additional components—(i) activities and (ii) think and reflect for self assessment while learning as well as to generate further interest in the learner. A number of hands-on examples are given to gradually explain methodology to solve different types of problems across the Chapters. The programming examples as well as the exercises in the

chapters are required to be solved in a computer and verify with the given outputs.

Box items are pinned inside the chapters either to explain related concepts or to describe additional information related to the topic covered in that section. However, these box-items are not to be assessed through examinations.

Project Based Learning given as the end includes exemplar projects related to real-world problems. Teachers are supposed to assign these or similar projects to be developed in groups. Working in such projects may promote peer-learning, team spirit and responsiveness.

The chapters have been written by involving practicing teachers as well as subject experts. Several iterations have resulted into this book. Thanks are due to the authors and reviewers for their valuable contribution. I would like to place on record appreciation for Professor Om Vikas for leading the review activities of the book as well as for his guidance and motivation to the development team throughout. Comments and suggestions are welcome.

New Delhi  
20 August 2020

Rejaul Karim Barbhuiya  
*Assistant Professor*  
Central Institute of  
Educational Technology



# TEXTBOOK DEVELOPMENT COMMITTEE

## CHIEF ADVISOR

Om Vikas, *Professor (Retd.)*, Former Director, ABV-IIITM, Gwalior, M.P.

## MEMBERS

Anju Gupta, *Freelance Educationist*, Delhi

Anuradha Khattar, *Assistant Professor*, Miranda House, University of Delhi

Chetna Khanna, *Freelance Educationist*, Delhi

Faheem Masoodi, *Assistant Professor*, Department of Computer Science, University of Kashmir

Harita Ahuja, *Assistant Professor*, Acharya Narendra Dev College, University of Delhi

Mohini Arora, *HOD, Computer Science*, Air Force Golden Jubilee Institute, Subroto Park, Delhi

Mudasir Wani, *Assistant Professor*, Govt. College for Women Nawakadal, Sri Nagar, Jammu and Kashmir

Naeem Ahmad, *Assistant Professor*, Madanapalle Institute of Technology and Science, Madanapalle, Andhra Pradesh

Purvi Kumar, *Co-ordinator*, Computer Science Department, Ganga International School, Rohtak Road, Delhi

Priti Rai Jain, *Assistant Professor*, Miranda House, University of Delhi

Sangita Chadha, *HOD, Computer Science*, Ambience Public School, Safdarjung Enclave, Delhi

Sharanjit Kaur, *Associate Professor*, Acharya Narendra Dev College, University of Delhi

## MEMBER-COORDINATOR

Rejaul Karim Barbhuiya, *Assistant Professor*, CIET, NCERT, Delhi



## ACKNOWLEDGEMENTS

The National Council of Educational Research and Training acknowledges the valuable contributions of the individuals and organisations involved in the development of Computer Science textbook for Class XII.

The Council expresses its gratitude to the syllabus development team including MPS Bhatia, *Professor*, Netaji Subhas Institute of Technology, Delhi; T. V. Vijay Kumar, *Professor*, School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi; Zahid Raza, *Associate Professor*, School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi; Vipul Shah, *Principal Scientist*, Tata Consultancy Services, and the CSpahtshala team; Aasim Zafar, *Associate Professor*, Department of Computer Science, Aligarh Muslim University, Aligarh; Faisal Anwer, *Assistant Professor*, Department of Computer Science, Aligarh Muslim University, Aligarh; Smruti Ranjan Sarangi, *Associate Professor*, Department of Computer Science and Engineering, Indian Institute of Technology, Delhi; Vikram Goyal, *Associate Professor*, Indraprastha Institute of Information Technology (IIIT), Delhi; and Mamur Ali, *Assistant Professor*, Department of Teacher Training and Non-formal Education (IASE), Faculty of Education, Jamia Millia Islamia, New Delhi.

The Council is thankful to the following resource persons for providing valuable inputs in developing this book—Veer Sain Dixit, *Assistant Professor*, Atma Ram Sanatan Dharma College, University of Delhi; Mukesh Kumar, DPS RK Puram, Delhi; Aswin K. Dash, Mother's International School, Delhi; Anamika Gupta, *Assistant Professor*, Shaheed Sukhdev College of Business Studies, University of Delhi, Sajid Yousuf Bhat, *Assistant Professor*, University of Kashmir, Jammu and Kashmir.

The council is grateful to Prof. Sunita Farkya, *Head*, Department of Education in Science and Mathematics, NCERT and Prof. Amarendra P. Behera, *Joint Director*, CIET, NCERT for their valuable cooperation and support throughout the development of this book.

The Council also gracefully acknowledges the contributions of Meetu Sharma, *Graphic Designer cum DTP Operator*; Kanika Walecha, *DTP Operator*; Pooja, *Junior Project Fellow*; in shaping this book. The contributions of the office of the APC, DESM and Publication Division, NCERT, New Delhi, in bringing out this book are also duly acknowledged.

The Council also acknowledges the contribution of Ankeeta Bezboruah *Assistant Editor (Contractual)* Publication Division, NCERT for copy editing this book. The efforts of Naresh Kumar, *DTP Operator (Contractual)*, Publication Division, NCERT are also acknowledged.



# CONTENTS

|                                               |            |
|-----------------------------------------------|------------|
| <b>FOREWORD</b>                               | <b>iii</b> |
| <b>PREFACE</b>                                | <b>v</b>   |
| <b>CHAPTER 1 EXCEPTION HANDLING IN PYTHON</b> | <b>1</b>   |
| 1.1 Introduction                              | 1          |
| 1.2 Syntax Errors                             | 1          |
| 1.3 Exceptions                                | 3          |
| 1.4 Built-in Exceptions                       | 3          |
| 1.5 Raising Exceptions                        | 4          |
| 1.6 Handling Exceptions                       | 7          |
| 1.7 Finally Clause                            | 13         |
| <b>CHAPTER 2 FILE HANDLING IN PYTHON</b>      | <b>19</b>  |
| 2.1 Introduction to Files                     | 19         |
| 2.2.Types of Files                            | 20         |
| 2.3 Opening and Closing a Text File           | 21         |
| 2.4 Writing to a Text File                    | 23         |
| 2.5 Reading from a Text File                  | 25         |
| 2.6 Setting Offsets in a File                 | 28         |
| 2.7 Creating and Traversing a Text File       | 29         |
| 2.8 The Pickle Module                         | 32         |
| <b>CHAPTER 3 STACK</b>                        | <b>39</b>  |
| 3.1 Introduction                              | 39         |
| 3.2 Stack                                     | 40         |
| 3.3 Operations on Stack                       | 42         |
| 3.4 Implementation of Stack in Python         | 43         |
| 3.5 Notations for Arithmetic Expressions      | 46         |
| 3.6 Conversion from Infix to Postfix Notation | 47         |
| 3.7 Evaluation of Postfix Expression          | 49         |
| <b>CHAPTER 4 QUEUE</b>                        | <b>53</b>  |
| 4.1 Introduction to Queue                     | 53         |
| 4.2 Operations on Queue                       | 55         |

|                                                  |            |
|--------------------------------------------------|------------|
| 4.3 Implementation of Queue using Python         | 56         |
| 4.4 Introduction to Deque                        | 59         |
| 4.5 Implementation of Deque Using Python         | 61         |
| <b>CHAPTER 5 SORTING</b>                         | <b>67</b>  |
| 5.1 Introduction                                 | 67         |
| 5.2 Bubble Sort                                  | 68         |
| 5.3 Selection Sort                               | 71         |
| 5.4 Insertion Sort                               | 74         |
| 5.5 Time Complexity of Algorithms                | 77         |
| <b>CHAPTER 6 SEARCHING</b>                       | <b>81</b>  |
| 6.1 Introduction                                 | 81         |
| 6.2 Linear Search                                | 82         |
| 6.3 Binary Search                                | 85         |
| 6.4 Search by Hashing                            | 90         |
| <b>CHAPTER 7 UNDERSTANDING DATA</b>              | <b>97</b>  |
| 7.1 Introduction to Data                         | 97         |
| 7.2 Data Collection                              | 101        |
| 7.3 Data Storage                                 | 102        |
| 7.4 Data Processing                              | 102        |
| 7.5 Statistical Techniques for Data Processing   | 103        |
| <b>CHAPTER 8 DATABASE CONCEPTS</b>               | <b>111</b> |
| 8.1 Introduction                                 | 111        |
| 8.2 File System                                  | 112        |
| 8.3 Database Management System                   | 115        |
| 8.4 Relational Data Model                        | 120        |
| 8.5 Keys in a Relational Database                | 123        |
| <b>CHAPTER 9 STRUCTURED QUERY LANGUAGE (SQL)</b> | <b>131</b> |
| 9.1 Introduction                                 | 131        |
| 9.2 Structured Query Language (SQL)              | 131        |
| 9.3 Data Types and Constraints in MySQL          | 133        |
| 9.4 SQL for Data Definition                      | 134        |
| 9.5 SQL for Data Manipulation                    | 141        |
| 9.6 SQL for Data Query                           | 144        |
| 9.7 Data Updation and Deletion                   | 154        |
| 9.8 Functions in SQL                             | 156        |
| 9.9 GROUP BY Clause in SQL                       | 167        |

|                                                     |            |
|-----------------------------------------------------|------------|
| 9.10 Operations on Relations                        | 169        |
| 9.11 Using Two Relations in a Query                 | 172        |
| <b>CHAPTER 10 COMPUTER NETWORKS</b>                 | <b>181</b> |
| 10.1 Introduction to Computer Networks              | 181        |
| 10.2 Evolution of Networking                        | 183        |
| 10.3 Types of Networks                              | 184        |
| 10.4 Network Devices                                | 187        |
| 10.5 Networking Topologies                          | 191        |
| 10.6 Identifying Nodes in a Networked Communication | 194        |
| 10.7 Internet, Web and the Internet of Things       | 195        |
| 10.8 Domain Name System                             | 197        |
| <b>CHAPTER 11 DATA COMMUNICATION</b>                | <b>203</b> |
| 11.1 Concept of Communication                       | 203        |
| 11.2 Components of data Communication               | 204        |
| 11.3 Measuring Capacity of Communication Media      | 205        |
| 11.4 Types of Data Communication                    | 206        |
| 11.5 Switching Techniques                           | 208        |
| 11.6 Transmission Media                             | 209        |
| 11.7 Mobile Telecommunication Technologies          | 215        |
| 11.8 Protocol                                       | 216        |
| <b>CHAPTER 12 SECURITY ASPECTS</b>                  | <b>223</b> |
| 12.1 Threats and Prevention                         | 223        |
| 12.2 Malware                                        | 224        |
| 12.3 Antivirus                                      | 230        |
| 12.4 Spam                                           | 231        |
| 12.5 HTTP vs HTTPS                                  | 231        |
| 12.6 Firewall                                       | 232        |
| 12.7 Cookies                                        | 233        |
| 12.8 Hackers and Crackers                           | 234        |
| 12.9 Network Security Threats                       | 235        |
| <b>CHAPTER 13 PROJECT BASED LEARNING</b>            | <b>241</b> |
| 13.1 Introduction                                   | 241        |
| 13.2 Approaches for Solving Projects                | 242        |
| 13.3 Teamwork                                       | 243        |
| 13.4 Project Descriptions                           | 245        |

# **THE CONSTITUTION OF INDIA**

## **PREAMBLE**

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

**JUSTICE**, social, economic and political;

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

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

**FRATERNITY** assuring the dignity of the individual and the **<sup>2</sup>[unity and integrity of the Nation];**

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

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

# Chapter

1

# Exception Handling in Python



12130CH01

## In this Chapter

- » *Introduction*
- » *Syntax Errors*
- » *Exceptions*
- » *Built-in Exceptions*
- » *Raising Exceptions*
- » *Handling Exceptions*
- » *Finally Clause*

*"I like my code to be elegant and efficient. The logic should be straightforward to make it hard for bugs to hide, the dependencies minimal to ease maintenance, error handling complete according to an articulated strategy, and performance close to optimal so as not to tempt people to make the code messy with unprincipled optimization. Clean code does one thing well."*

— Bjarne Stroustrup

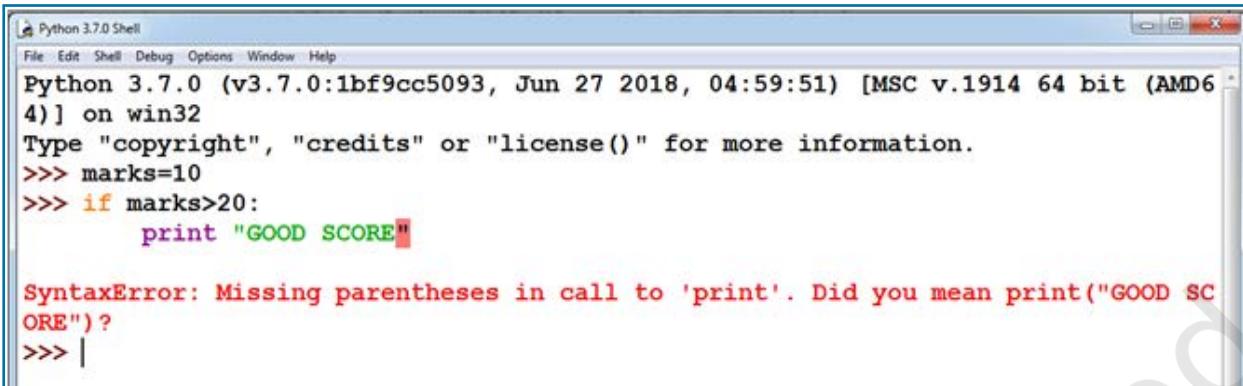
## 1.1 INTRODUCTION

Sometimes while executing a Python program, the program does not execute at all or the program executes but generates unexpected output or behaves abnormally. These occur when there are syntax errors, runtime errors or logical errors in the code. In Python, exceptions are errors that get triggered automatically. However, exceptions can be forcefully triggered and handled through program code. In this chapter, we will learn about exception handling in Python programs.

## 1.2 SYNTAX ERRORS

Syntax errors are detected when we have not followed the rules of the particular programming language while writing a program. These errors are also known as *parsing errors*. On encountering a syntax error, the interpreter does not execute the program unless we rectify the errors, save and

rerun the program. When a syntax error is encountered while working in shell mode, Python displays the name of the error and a small description about the error as shown in Figure 1.1.



The screenshot shows the Python 3.7.0 Shell window. The command line shows:

```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> marks=10
>>> if marks>20:
    print "GOOD SCORE"

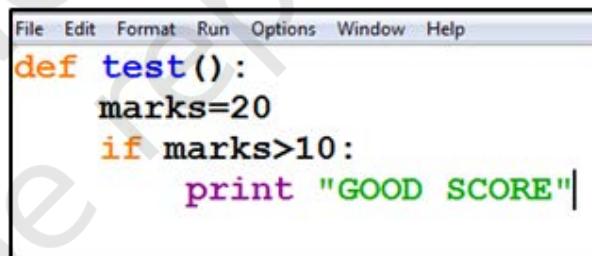
SyntaxError: Missing parentheses in call to 'print'. Did you mean print("GOOD SCORE")?
>>> |
```

A red box highlights the error message: "SyntaxError: Missing parentheses in call to 'print'. Did you mean print("GOOD SCORE")?"

Figure 1.1: A syntax error displayed in Python shell mode

So, a syntax error is reported by the Python interpreter giving a brief explanation about the error and a suggestion to rectify it.

Similarly, when a syntax error is encountered while running a program in script mode as shown in Figure 1.2, a dialog box specifying the name of the error (Figure 1.3) and a small description about the error is displayed.



The screenshot shows a script editor window with the following code:

```
File Edit Format Run Options Window Help
def test():
    marks=20
    if marks>10:
        print "GOOD SCORE"
```

Figure 1.2: An error in the script

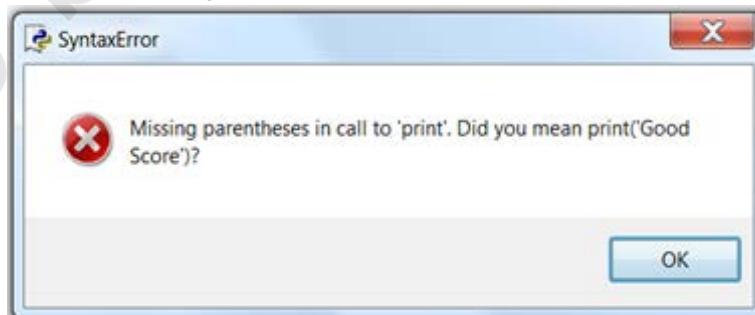


Figure 1.3: Error dialog box

### 1.3 EXCEPTIONS

Even if a statement or expression is syntactically correct, there might arise an error during its execution. For example, trying to open a file that does not exist, division by zero and so on. Such types of errors might disrupt the normal execution of the program and are called exceptions.

An exception is a Python object that represents an error. When an error occurs during the execution of a program, an exception is said to have been raised. Such an exception needs to be handled by the programmer so that the program does not terminate abnormally. Therefore, while designing a program, a programmer may anticipate such erroneous situations that may arise during its execution and can address them by including appropriate code to handle that exception.

It is to be noted that `SyntaxError` shown at Figures 1.1 and 1.3 is also an exception. But, all other exceptions are generated when a program is syntactically correct.

### 1.4 BUILT-IN EXCEPTIONS

Commonly occurring exceptions are usually defined in the compiler/interpreter. These are called built-in exceptions.

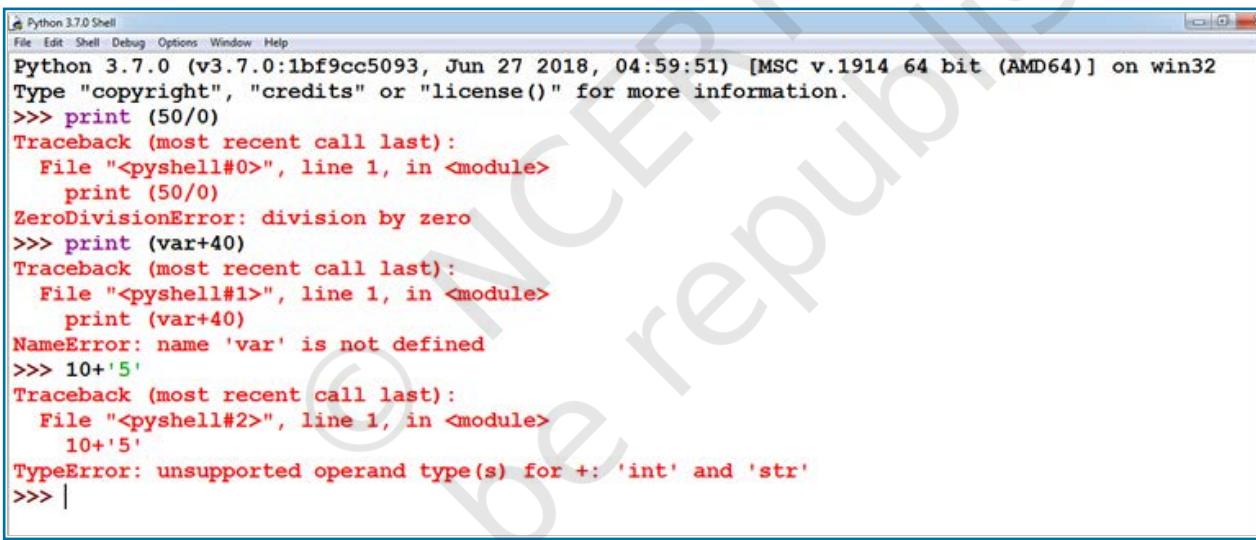
Python's standard library is an extensive collection of built-in exceptions that deals with the commonly occurring errors (exceptions) by providing the standardized solutions for such errors. On the occurrence of any built-in exception, the appropriate exception handler code is executed which displays the reason along with the raised exception name. The programmer then has to take appropriate action to handle it. Some of the commonly occurring built-in exceptions that can be raised in Python are explained in Table 1.1.

**Table 1.1 Built-in exceptions in Python**

| S. No | Name of the Built-in Exception | Explanation                                                                                                                                |
|-------|--------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| 1.    | <code>SyntaxError</code>       | It is raised when there is an error in the syntax of the Python code.                                                                      |
| 2.    | <code>ValueError</code>        | It is raised when a built-in method or operation receives an argument that has the right data type but mismatched or inappropriate values. |
| 3.    | <code>IOError</code>           | It is raised when the file specified in a program statement cannot be opened.                                                              |

|    |                   |                                                                                                                                                          |
|----|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| 4  | KeyboardInterrupt | It is raised when the user accidentally hits the Delete or Esc key while executing a program due to which the normal flow of the program is interrupted. |
| 5  | ImportError       | It is raised when the requested module definition is not found.                                                                                          |
| 6  | EOFError          | It is raised when the end of file condition is reached without reading any data by input().                                                              |
| 7  | ZeroDivisionError | It is raised when the denominator in a division operation is zero.                                                                                       |
| 8  | IndexError        | It is raised when the index or subscript in a sequence is out of range.                                                                                  |
| 9  | NameError         | It is raised when a local or global variable name is not defined.                                                                                        |
| 10 | IndentationError  | It is raised due to incorrect indentation in the program code.                                                                                           |
| 11 | TypeError         | It is raised when an operator is supplied with a value of incorrect data type.                                                                           |
| 12 | OverFlowError     | It is raised when the result of a calculation exceeds the maximum limit for numeric data type.                                                           |

Figure 1.4 shows the built-in exceptions viz, ZeroDivisionError, NameError, and TypeError raised by the Python interpreter in different situations.



```

Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print (50/0)
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    print (50/0)
ZeroDivisionError: division by zero
>>> print (var+40)
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    print (var+40)
NameError: name 'var' is not defined
>>> 10+'5'
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    10+'5'
TypeError: unsupported operand type(s) for +: 'int' and 'str'
>>> |

```

Figure 1.4: Example of built-in exceptions

A programmer can also create custom exceptions to suit one's requirements. These are called user-defined exceptions. We will learn how to handle exceptions in the next section.

## 1.5 RAISING EXCEPTIONS

Each time an error is detected in a program, the Python interpreter raises (throws) an exception. Exception

handlers are designed to execute when a specific exception is raised. Programmers can also forcefully raise exceptions in a program using the raise and assert statements. Once an exception is raised, no further statement in the current block of code is executed. So, raising an exception involves interrupting the normal flow execution of program and jumping to that part of the program (exception handler code) which is written to handle such exceptional situations.

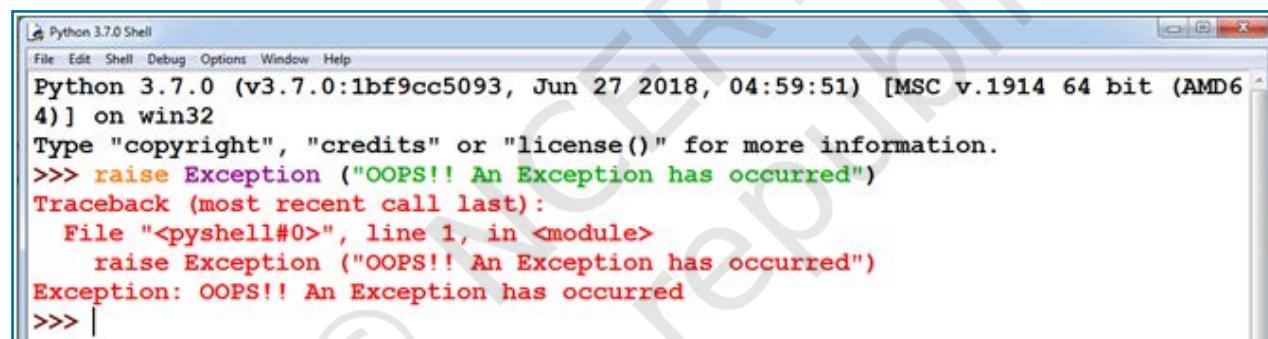
### 1.5.1 The raise Statement

The raise statement can be used to throw an exception.

The syntax of raise statement is:

```
raise exception-name[(optional argument)]
```

The argument is generally a string that is displayed when the exception is raised. For example, when an exception is raised as shown in Figure 1.5, the message “OOPS : An Exception has occurred” is displayed along with a brief description of the error.



A screenshot of the Python 3.7.0 Shell window. The title bar says "Python 3.7.0 Shell". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main window shows the following text:

```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> raise Exception ("OOPS!! An Exception has occurred")
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    raise Exception ("OOPS!! An Exception has occurred")
Exception: OOPS!! An Exception has occurred
>>> |
```

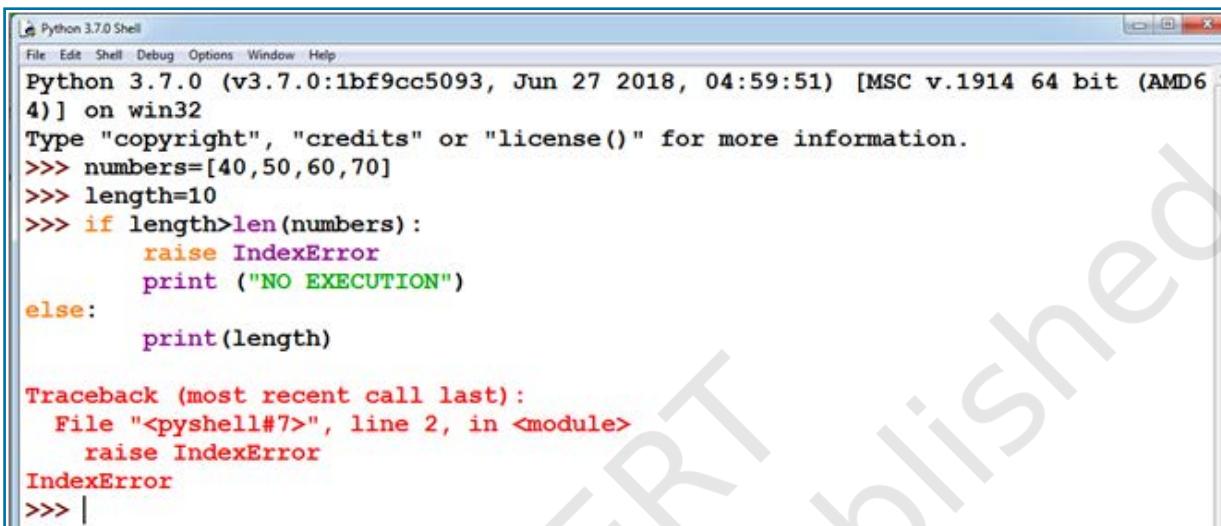
Figure 1.5: Use of the raise statement to throw an exception

The error detected may be a built-in exception or may be a user-defined one. Consider the example given in Figure 1.6 that uses the raise statement to raise a built-in exception called IndexError.

**Note:** In this case, the user has only raised the exception but has not displayed any error message explicitly.

In Figure 1.6, since the value of variable length is greater than the length of the list *numbers*, an IndexError exception will be raised. The statement following the raise statement will not be executed. So the message “NO EXECUTION” will not be displayed in this case.

As we can see in Figure 1.6, in addition to the error message displayed, Python also displays a stack Traceback. This is a structured block of text that contains information about the sequence of function calls that have been made in the branch of execution of code in which the exception was raised. In Figure 1.6, the error has been encountered in the most recently called function that has been executed.



The screenshot shows a Windows desktop with a Python 3.7.0 Shell window open. The window title is "Python 3.7.0 Shell". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main area of the window displays Python code and its execution:

```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.

>>> numbers=[40,50,60,70]
>>> length=10
>>> if length>len(numbers):
    raise IndexError
    print ("NO EXECUTION")
else:
    print(length)

Traceback (most recent call last):
  File "<pyshell#7>", line 2, in <module>
    raise IndexError
IndexError
>>> |
```

Figure 1.6: Use of raise statement with built-in exception

**Note:** We will learn about Stack in Chapter 3.

### 1.5.2 The assert Statement

An assert statement in Python is used to test an expression in the program code. If the result after testing comes false, then the exception is raised. This statement is generally used in the beginning of the function or after a function call to check for valid input. The syntax for assert statement is:

```
assert Expression[, arguments]
```

On encountering an assert statement, Python evaluates the expression given immediately after the assert keyword. If this expression is false, an AssertionError exception is raised which can be handled like any other exception. Consider the code given in Program 1-1.

#### Program 1-1 Use of assert statement

```
print("use of assert statement")
def negativecheck(number):
    assert(number>=0), "OOPS... Negative Number"
```

```
    print(number*number)
print (negativecheck(100))
print (negativecheck(-350))
```

```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:\Users\admin\Desktop\NCERT\python programs for exception handling\assert example.py
use of assert statement
10000
None
Traceback (most recent call last):
  File "C:\Users\admin\Desktop\NCERT\python programs for exception handling\assert example.py", line 6, in <module>
    print (negativecheck(-350))
  File "C:\Users\admin\Desktop\NCERT\python programs for exception handling\assert example.py", line 3, in negativecheck
    assert(number>=0), "OOPS... Negative Number"
AssertionError: OOPS... Negative Number
>>>
```

Figure 1.7: Output of Program 1-1.

In the code, the assert statement checks for the value of the variable number. In case the number gets a negative value, `AssertionError` will be thrown, and subsequent statements will not be executed. Hence, on passing a negative value (-350) as an argument, it results in `AssertionError` and displays the message “OOPS.... Negative Number”. The output of the code is shown in Figure 1.7.

## 1.6 HANDLING EXCEPTIONS

Each and every exception has to be handled by the programmer to avoid the program from crashing abruptly. This is done by writing additional code in a program to give proper messages or instructions to the user on encountering an exception. This process is known as *exception handling*.

### 1.6.1 Need for Exception Handling

Exception handling is being used not only in Python programming but in most programming languages like C++, Java, Ruby, etc. It is a useful technique that helps in capturing runtime errors and handling them so as to avoid the program getting crashed. Following are some

of the important points regarding exceptions and their handling:

- Python categorises exceptions into distinct types so that specific exception handlers (code to handle that particular exception) can be created for each type.
- Exception handlers separate the main logic of the program from the error detection and correction code. The segment of code where there is any possibility of error or exception, is placed inside one block. The code to be executed in case the exception has occurred, is placed inside another block. These statements for detection and reporting the exception do not affect the main logic of the program.
- The compiler or interpreter keeps track of the exact position where the error has occurred.
- Exception handling can be done for both user-defined and built-in exceptions.

### 1.6.2 Process of Handling Exception

When an error occurs, Python interpreter creates an object called the *exception object*. This object contains information about the error like its type, file name and position in the program where the error has occurred. The object is handed over to the runtime system so that it can find an appropriate code to handle this particular exception. This process of creating an exception object and handing it over to the runtime system is called *throwing an exception*. It is important to note that when an exception occurs while executing a particular program statement, the control jumps to an exception handler, abandoning execution of the remaining program statements.

The runtime system searches the entire program for a block of code, called the *exception handler* that can handle the raised exception. It first searches for the method in which the error has occurred and the exception has been raised. If not found, then it searches the method from which this method (in which exception was raised) was called. This hierarchical search in reverse order continues till the exception handler is found. This entire list of methods is known as *call stack*. When a suitable handler is found in the call stack, it is executed by the runtime process. This process of



A runtime system refers to the execution of the statements given in the program. It is a complex mechanism consisting of hardware and software that comes into action as soon as the program, written in any programming language, is put for execution.



executing a suitable handler is known as *catching the exception*. If the runtime system is not able to find an appropriate exception after searching all the methods in the call stack, then the program execution stops.

The flowchart in Figure 1.8 describes the exception handling process.

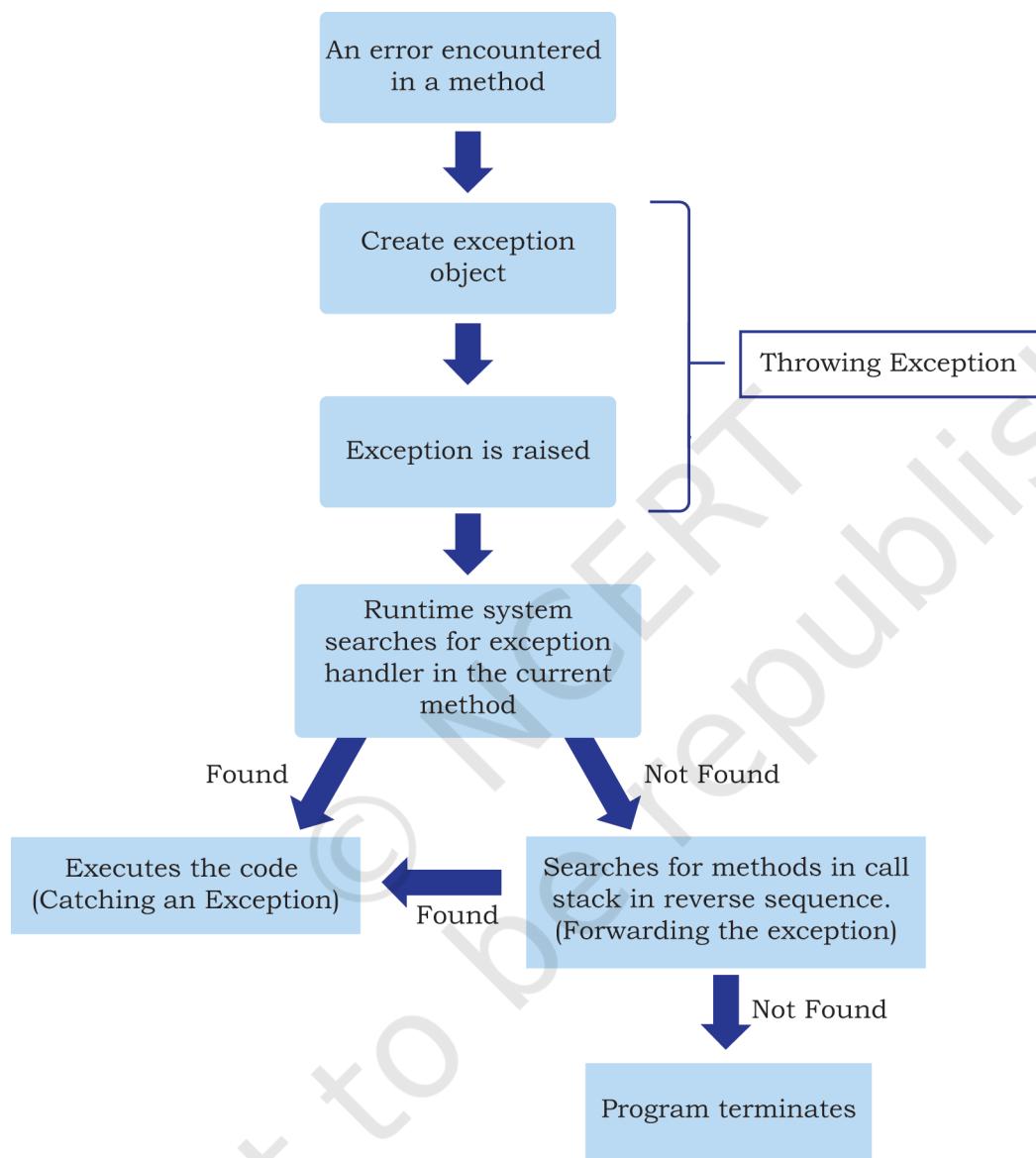


Figure 1.8: Steps of handling exception

### 1.6.3 Catching Exceptions

An exception is said to be caught when a code that is designed to handle a particular exception is executed. Exceptions, if any, are caught in the `try` block and

handled in the except block. While writing or debugging a program, a user might doubt an exception to occur in a particular part of the code. Such suspicious lines of codes are put inside a try block. Every try block is followed by an except block. The appropriate code to handle each of the possible exceptions (in the code inside the try block) are written inside the except clause.

While executing the program, if an exception is encountered, further execution of the code inside the try block is stopped and the control is transferred to the except block. The syntax of try ... except clause is as follows:

```
try:  
    [ program statements where exceptions might occur]  
except [exception-name]:  
    [ code for exception handling if the exception-name error is  
    encountered]
```

Consider the Program 1-2 given below:

#### Program 1-2 Using try..except block

```
print ("Practicing for try block")  
try:  
    numerator=50  
    denom=int(input("Enter the denominator"))  
    quotient=(numerator/denom)  
    print(quotient)  
    print ("Division performed successfully")  
except ZeroDivisionError:  
    print ("Denominator as ZERO.... not allowed")  
print("OUTSIDE try..except block")
```

In Program 1-2, the ZeroDivisionError exception is handled. If the user enters any non-zero value as denominator, the quotient will be displayed along with the message “Division performed successfully”, as shown in Figure 1.10. The except clause will be skipped in this case. So, the next statement after the try..except block is executed and the message “OUTSIDE try..except block” is displayed.

However, if the user enters the value of denom as zero (0), then the execution of the try block will stop. The control will shift to the except block and the message “Denominator as Zero.... not allowed” will be displayed, as shown in Figure 1.11. Thereafter, the

statement following the try..except block is executed and the message “OUTSIDE try..except block” is displayed in this case also.

The screenshot shows the Python 3.7.0 Shell window. The code in the shell is:

```
>>> RESTART: C:/Users/admin/Desktop/NCERT/python programs for exception handling/try_except mar 2020 example.py
Practicing for try block
Enter the denominator12
Division performed successfully
OUTSIDE try..except block
>>> |
```

The output shows the program successfully performing a division and then exiting the try block without an error.

Figure 1.9: Output without an error

The screenshot shows the Python 3.7.0 Shell window. The code in the shell is identical to Figure 1.9:

```
>>> RESTART: C:/Users/admin/Desktop/NCERT/python programs for exception handling/try_except mar 2020 example.py
Practicing for try block
Enter the denominator0
Denominator as ZERO.... not allowed
OUTSIDE try..except block
>>> |
```

The output includes the additional message "Denominator as ZERO.... not allowed", indicating that a ZeroDivisionError was raised and handled by the except clause.

Figure 1.10: Output with exception raised

Sometimes, a single piece of code might be suspected to have more than one type of error. For handling such situations, we can have multiple except blocks for a single try block as shown in the Program 1-3.

### Program 1-3 Use of multiple except clauses

```
print ("Handling multiple exceptions")
try:
    numerator=50
    denom=int(input("Enter the denominator: "))
    print (numerator/denom)
    print ("Division performed successfully")
except ZeroDivisionError:
    print ("Denominator as ZERO is not allowed")
except ValueError:
    print ("Only INTEGERS should be entered")
```

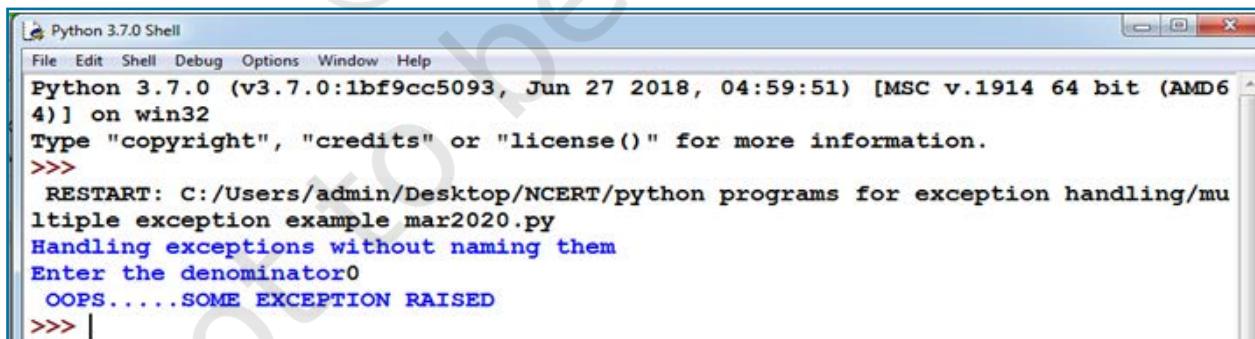
In the code, two types of exceptions (ZeroDivisionError and ValueError) are handled using two except blocks for a single try block. When an exception is raised, a search for the matching except block is made till it is handled. If no match is found, then the program terminates.

However, if an exception is raised for which no handler is created by the programmer, then such an exception can be handled by adding an except clause without specifying any exception. This except clause should be added as the last clause of the try..except block. The Program 1-4 given below along with the output given in Figure 1.11 explains this.

#### Program 1-4 Use of except without specifying an exception

```
print ("Handling exceptions without naming them")
try:
    numerator=50
    denom=int(input("Enter the denominator"))
    quotient=(numerator/denom)
    print ("Division performed successfully")
except ValueError:
    print ("Only INTEGERS should be entered")
except:
    print(" OOPS.....SOME EXCEPTION RAISED")
```

If the above code is executed, and the denominator entered is 0 (zero), the handler for ZeroDivisionError exception will be searched. Since it is not present, the last except clause (without any specified exception) will be executed, so the message “OOPS.....SOME EXCEPTION RAISED” will be displayed.



```
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:/Users/admin/Desktop/NCERT/python programs for exception handling/multiple exception example mar2020.py
Handling exceptions without naming them
Enter the denominator0
OOPS.....SOME EXCEPTION RAISED
>>> |
```

Figure 1.11: Output of Program 1-4

#### 1.6.4 try...except...else clause

We can put an optional else clause along with the try...except clause. An except block will be executed

only if some exception is raised in the try block. But if there is no error then none of the except blocks will be executed. In this case, the statements inside the else clause will be executed. Program 1-5 along with its output explains the use of else block with the try...except block.

#### Program 1-5 Use of else clause

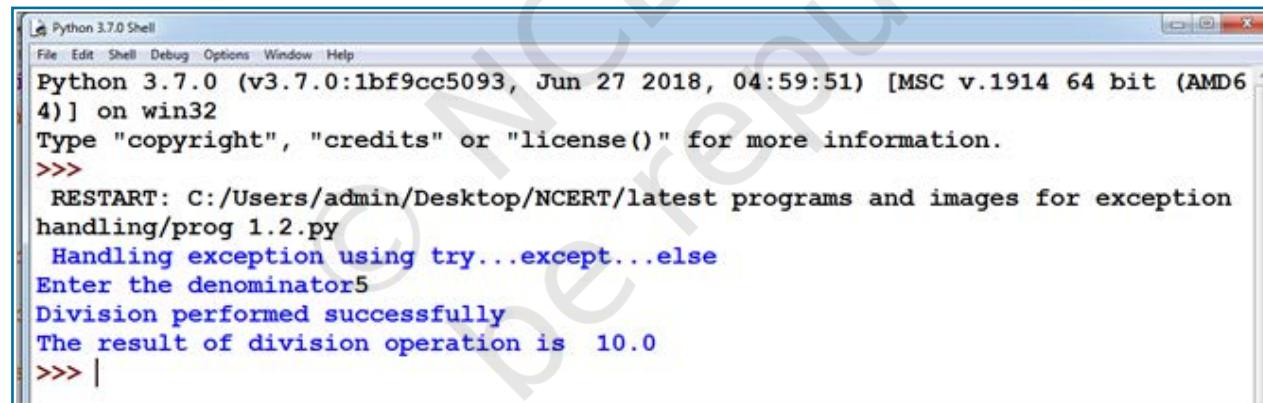
```
print ("Handling exception using try...except...else")
try:
    numerator=50
    denom=int(input("Enter the denominator: "))
    quotient=(numerator/denom)
    print ("Division performed successfully")

except ZeroDivisionError:
    print ("Denominator as ZERO is not allowed")

except ValueError:
    print ("Only INTEGERS should be entered")

else:
    print ("The result of division operation is ", quotient)
```

#### Output:

A screenshot of the Python 3.7.0 Shell window. The title bar says "Python 3.7.0 Shell". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main window shows the following text:

```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:/Users/admin/Desktop/NCERT/latest programs and images for exception handling/prog 1.2.py
Handling exception using try...except...else
Enter the denominator5
Division performed successfully
The result of division operation is  10.0
>>> |
```

The text is color-coded: blue for keywords like "RESTART", "copyright", etc., and red for the output of the print statement.

Figure 1.12: Output of Program 1-5.

### 1.7 FINALLY CLAUSE

The try statement in Python can also have an optional finally clause. The statements inside the finally block are always executed regardless of whether an exception has occurred in the try block or not. It is a common practice to use finally clause while working with files to ensure that the file object is closed. If used, finally should always be placed at the end of try clause, after all except blocks and the else block.

### Program 1-6 Use of finally clause

```
print ("Handling exception using try...except...else...finally")
try:
    numerator=50
    denom=int(input("Enter the denominator: "))
    quotient=(numerator/denom)
    print ("Division performed successfully")
except ZeroDivisionError:
    print ("Denominator as ZERO is not allowed")
except ValueError:
    print ("Only INTEGERS should be entered")
else:
    print ("The result of division operation is ", quotient)
finally:
    print ("OVER AND OUT")
```

In the above program, the message “OVER AND OUT” will be displayed irrespective of whether an exception is raised or not.

#### 1.6.1 Recovering and continuing with finally clause

If an error has been detected in the try block and the exception has been thrown, the appropriate except block will be executed to handle the error. But if the exception is not handled by any of the except clauses, then it is re-raised after the execution of the finally block. For example, Program 1.4 contains only the except block for ZeroDivisionError. If any other type of error occurs for which there is no handler code (except clause) defined, then also the finally clause will be executed first. Consider the code given in Program 1-7 to understand these concepts.

### Program 1-7 Recovering through finally clause

```
print (" Practicing for try block")
try:
    numerator=50
    denom=int(input("Enter the denominator"))
    quotient=(numerator/denom)
    print ("Division performed successfully")
except ZeroDivisionError:
    print ("Denominator as ZERO is not allowed")
else:
    print ("The result of division operation is ", quotient)
finally:
    print ("OVER AND OUT")
```

While executing the above code, if we enter a non-numeric data as input, the finally block will be executed. So, the message “OVER AND OUT” will be displayed. Thereafter the exception for which handler is not present will be re-raised. The output of Program 1-7 is shown in Figure 1.13.

```
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:/Users/admin/Desktop/NCERT/latest programs and images for exception handling/try finally example fig 1.12.py
Practicing for try block
Enter the denominatorvar
OVER AND OUT
Traceback (most recent call last):
  File "C:/Users/admin/Desktop/NCERT/latest programs and images for exception handling/try finally example fig 1.12.py", line 4, in <module>
    denom=int(input("Enter the denominator"))
ValueError: invalid literal for int() with base 10: 'var'
>>> |
```

Figure 1.13: Output of Program 1-7

After execution of finally block, Python transfers the control to a previously entered try or to the next higher level default exception handler. In such a case, the statements following the finally block is executed. That is, unlike except, execution of the finally clause does not terminate the exception. Rather, the exception continues to be raised after execution of finally.

To summarise, we put a piece of code where there are possibilities of errors or exceptions to occur inside a try block. Inside each except clause we define handler codes to handle the matching exception raised in the try block. The optional else clause contains codes to be executed if no exception occurs. The optional finally block contains codes to be executed irrespective of whether an exception occurs or not.

### SUMMARY

- Syntax errors or parsing errors are detected when we have not followed the rules of the particular programming language while writing a program.

## NOTES

- When syntax error is encountered, Python displays the name of the error and a small description about the error.
- The execution of the program will start only after the syntax error is rectified.
- An exception is a Python object that represents an error.
- Syntax errors are also handled as exceptions.
- The exception needs to be handled by the programmer so that the program does not terminate abruptly.
- When an exception occurs during execution of a program and there is a built-in exception defined for that, the error message written in that exception is displayed. The programmer then has to take appropriate action and handle it.
- Some of the commonly occurring built-in exceptions are `SyntaxError`, `ValueError`, `IOError`, `KeyboardInterrupt`, `ImportError`, `EOFError`, `ZeroDivisionError`, `IndexError`, `NameError`, `IndentationError`, `TypeError`, and `OverflowError`.
- When an error is encountered in a program, Python interpreter raises or throws an exception.
- Exception Handlers are the codes that are designed to execute when a specific exception is raised.
- Raising an exception involves interrupting the normal flow of the program execution and jumping to the exception handler.
- Raise and assert statements are used to raise exceptions.
- The process of exception handling involves writing additional code to give proper messages or instructions to the user. This prevents the program from crashing abruptly. The additional code is known as an exception handler.
- An exception is said to be caught when a code that is designed to handle a particular exception is executed.
- An exception is caught in the try block and handles in except block.

- The statements inside the finally block are always executed regardless of whether an exception occurred in the try block or not.



## EXERCISE

- "Every syntax error is an exception but every exception cannot be a syntax error." Justify the statement.
- When are the following built-in exceptions raised? Give examples to support your answers.
  - ImportError
  - IOError
  - NameError
  - ZeroDivisionError
- What is the use of a raise statement? Write a code to accept two numbers and display the quotient. Appropriate exception should be raised if the user enters the second number (denominator) as zero (0).
- Use assert statement in Question No. 3 to test the division expression in the program.
- Define the following:
  - Exception Handling
  - Throwing an exception
  - Catching an exception
- Explain catching exceptions using try and except block.
- Consider the code given below and fill in the blanks.

```
print (" Learning Exceptions...")  
try:  
    num1= int(input ("Enter the first number"))  
    num2=int(input("Enter the second number"))  
    quotient=(num1/num2)  
    print ("Both the numbers entered were correct")  
except _____: # to enter only integers  
    print (" Please enter only numbers")  
except _____: # Denominator should not be zero  
    print(" Number 2 should not be zero")  
else:  
    print(" Great .. you are a good programmer")  
_____ : # to be executed at the end  
    print(" JOB OVER... GO GET SOME REST")
```

## NOTES

8. You have learnt how to use math module in Class XI. Write a code where you use the wrong number of arguments for a method (say sqrt() or pow()). Use the exception handling process to catch the ValueError exception.
9. What is the use of finally clause? Use finally clause in the problem given in Question No. 7.

not to be republished © NCERT

# Chapter

# 2

# File Handling in Python



12130CH02

## In this Chapter

- » *Introduction to Files*
- » *Types of Files*
- » *Opening and Closing a Text File*
- » *Writing to a Text File*
- » *Reading from a Text File*
- » *Setting Offsets in a File*
- » *Creating and Traversing a Text File*
- » *The Pickle Module*

*There are many ways of trying to understand programs. People often rely too much on one way, which is called "debugging" and consists of running a partly-understood program to see if it does what you expected. Another way, which ML advocates, is to install some means of understanding in the very programs themselves.*

— Robin Milner

## 2.1 INTRODUCTION TO FILES

We have so far created programs in Python that accept the input, manipulate it and display the output. But that output is available only during execution of the program and input is to be entered through the keyboard. This is because the variables used in a program have a lifetime that lasts till the time the program is under execution. What if we want to store the data that were input as well as the generated output permanently so that we can reuse it later? Usually, organisations would want to permanently store information about employees, inventory, sales, etc. to avoid repetitive tasks of entering the same data. Hence, data are stored permanently on secondary storage devices for reusability. We store Python programs written in script mode with a .py extension. Each program is stored on the secondary device as a file. Likewise, the data entered, and the output can be stored permanently into a file.

  
Text files contain only the ASCII equivalent of the contents of the file whereas a .docx file contains many additional information like the author's name, page settings, font type and size, date of creation and modification, etc.

### Activity 2.1

Create a text file using notepad and write your name and save it. Now, create a .docx file using Microsoft Word and write your name and save it as well. Check and compare the file size of both the files. You will find that the size of .txt file is in bytes whereas that of .docx is in KBs.



So, what is a file? A file is a named location on a secondary storage media where data are permanently stored for later access.

## 2.2. TYPES OF FILES

Computers store every file as a collection of 0s and 1s i.e., in binary form. Therefore, every file is basically just a series of bytes stored one after the other. There are mainly two types of data files — text file and binary file. A text file consists of human readable characters, which can be opened by any text editor. On the other hand, binary files are made up of non-human readable characters and symbols, which require specific programs to access its contents.

### 2.2.1 Text file

A text file can be understood as a sequence of characters consisting of alphabets, numbers and other special symbols. Files with extensions like .txt, .py, .csv, etc. are some examples of text files. When we open a text file using a text editor (e.g., Notepad), we see several lines of text. However, the file contents are not stored in such a way internally. Rather, they are stored in sequence of bytes consisting of 0s and 1s. In ASCII, UNICODE or any other encoding scheme, the value of each character of the text file is stored as bytes. So, while opening a text file, the text editor translates each ASCII value and shows us the equivalent character that is readable by the human being. For example, the ASCII value 65 (binary equivalent 1000001) will be displayed by a text editor as the letter 'A' since the number 65 in ASCII character set represents 'A'.

Each line of a text file is terminated by a special character, called the End of Line (EOL). For example, the default EOL character in Python is the newline (\n). However, other characters can be used to indicate EOL. When a text editor or a program interpreter encounters the ASCII equivalent of the EOL character, it displays the remaining file contents starting from a new line. Contents in a text file are usually separated by whitespace, but comma (,) and tab (\t) are also commonly used to separate values in a text file.

## 2.2.2 Binary Files

Binary files are also stored in terms of bytes (0s and 1s), but unlike text files, these bytes do not represent the ASCII values of characters. Rather, they represent the actual content such as image, audio, video, compressed versions of other files, executable files, etc. These files are not human readable. Thus, trying to open a binary file using a text editor will show some garbage values. We need specific software to read or write the contents of a binary file.

Binary files are stored in a computer in a sequence of bytes. Even a single bit change can corrupt the file and make it unreadable to the supporting application. Also, it is difficult to remove any error which may occur in the binary file as the stored contents are not human readable. We can read and write both text and binary files through Python programs.

## 2.3 OPENING AND CLOSING A TEXT FILE

In real world applications, computer programs deal with data coming from different sources like databases, CSV files, HTML, XML, JSON, etc. We broadly access files either to write or read data from it. But operations on files include creating and opening a file, writing data in a file, traversing a file, reading data from a file and so on. Python has the `io` module that contains different functions for handling files.

### 2.3.1 Opening a file

To open a file in Python, we use the `open()` function. The syntax of `open()` is as follows:

```
file_object= open(file_name, access_mode)
```

This function returns a file object called file handle which is stored in the variable `file_object`. We can use this variable to transfer data to and from the file (read and write) by calling the functions defined in the Python's `io` module. If the file does not exist, the above statement creates a new empty file and assigns it the name we specify in the statement.

The `file_object` has certain attributes that tells us basic information about the file, such as:

- `<file.closed>` returns true if the file is closed and false otherwise.



The `file_object` establishes a link between the program and the data file stored in the permanent storage.



- `<file.mode>` returns the access mode in which the file was opened.
- `<file.name>` returns the name of the file.

### Activity 2.2

Some of the other file access modes are `<rb+>`, `<wb>`, `<w+>`, `<ab>`, `<ab+>`. Find out for what purpose each of these are used. Also, find the file offset positions in each case.



The `file_name` should be the name of the file that has to be opened. If the file is not in the current working directory, then we need to specify the complete path of the file along with its name.

The `access_mode` is an optional argument that represents the mode in which the file has to be accessed by the program. It is also referred to as processing mode. Here mode means the operation for which the file has to be opened like `<r>` for reading, `<w>` for writing, `<+>` for both reading and writing, `<a>` for appending at the end of an existing file. The default is the read mode. In addition, we can specify whether the file will be handled as binary (`<b>`) or text mode. By default, files are opened in text mode that means strings can be read or written. Files containing non-textual data are opened in binary mode that means read/write are performed in terms of bytes. Table 2.1 lists various file access modes that can be used with the `open()` method. The file offset position in the table refers to the position of the file object when the file is opened in a particular mode.

**Table 2.1 File Open Modes**

| File Mode                                            | Description                                                                                                                                                             | File Offset position  |
|------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|
| <code>&lt;r&gt;</code>                               | Opens the file in read-only mode.                                                                                                                                       | Beginning of the file |
| <code>&lt;rb&gt;</code>                              | Opens the file in binary and read-only mode.                                                                                                                            | Beginning of the file |
| <code>&lt;r+&gt;</code> or <code>&lt;+r&gt;</code>   | Opens the file in both read and write mode.                                                                                                                             | Beginning of the file |
| <code>&lt;w&gt;</code>                               | Opens the file in write mode. If the file already exists, all the contents will be overwritten. If the file doesn't exist, then a new file will be created.             | Beginning of the file |
| <code>&lt;wb+&gt;</code> or <code>&lt;+wb&gt;</code> | Opens the file in read,write and binary mode. If the file already exists, the contents will be overwritten. If the file doesn't exist, then a new file will be created. | Beginning of the file |
| <code>&lt;a&gt;</code>                               | Opens the file in append mode. If the file doesn't exist, then a new file will be created.                                                                              | End of the file       |
| <code>&lt;a+&gt;</code> or <code>&lt;+a&gt;</code>   | Opens the file in append and read mode. If the file doesn't exist, then it will create a new file.                                                                      | End of the file       |

Consider the following example.

```
myObject=open("myfile.txt", "a+")
```

In the above statement, the file *myfile.txt* is opened in append and read modes. The file object will be at the end of the file. That means we can write data at the end of the file and at the same time we can also read data from the file using the file object named *myObject*.

### 2.3.2 Closing a file

Once we are done with the read/write operations on a file, it is a good practice to close the file. Python provides a `close()` method to do so. While closing a file, the system frees the memory allocated to it. The syntax of `close()` is:

```
file_object.close()
```

Here, `file_object` is the object that was returned while opening the file.

Python makes sure that any unwritten or unsaved data is flushed off (written) to the file before it is closed. Hence, it is always advised to close the file once our work is done. Also, if the file object is re-assigned to some other file, the previous file is automatically closed.

### 2.3.3 Opening a file using with clause

In Python, we can also open a file using `with` clause. The syntax of `with` clause is:

```
with open (file_name, access_mode) as file_
object:
```

The advantage of using `with` clause is that any file that is opened using this clause is closed automatically, once the control comes outside the `with` clause. In case the user forgets to close the file explicitly or if an exception occurs, the file is closed automatically. Also, it provides a simpler syntax.

```
with open("myfile.txt","r+") as myObject:
    content = myObject.read()
```

Here, we don't have to close the file explicitly using `close()` statement. Python will automatically close the file.

## 2.4 WRITING TO A TEXT FILE

For writing to a file, we first need to open it in write or append mode. If we open an existing file in write mode, the previous data will be erased, and the file object will be positioned at the beginning of the file. On the other

## Think and Reflect

For a newly created file, is there any difference between write() and append() methods?



hand, in append mode, new data will be added at the end of the previous data as the file object is at the end of the file. After opening the file, we can use the following methods to write data in the file.

- write() - for writing a single string
- writelines() - for writing a sequence of strings

### 2.4.1 The write() method

write() method takes a string as an argument and writes it to the text file. It returns the number of characters being written on single execution of the write() method. Also, we need to add a newline character (\n) at the end of every sentence to mark the end of line.

Consider the following piece of code:

```
>>> myobject=open("myfile.txt",'w')
>>> myobject.write("Hey I have started
#using files in Python\n")
41
>>> myobject.close()
```

On execution, write() returns the number of characters written on to the file. Hence, 41, which is the length of the string passed as an argument, is displayed.

**Note:** '\n' is treated as a single character

If numeric data are to be written to a text file, the data need to be converted into string before writing to the file. For example:

```
>>>myobject=open("myfile.txt",'w')
>>> marks=58
#number 58 is converted to a string using
#str()
>>> myobject.write(str(marks))
2
>>>myobject.close()
```

The write() actually writes data onto a buffer. When the close() method is executed, the contents from this buffer are moved to the file located on the permanent storage.

### 2.4.2 The writelines() method

This method is used to write multiple strings to a file. We need to pass an iterable object like lists, tuple, etc. containing strings to the writelines() method. Unlike



We can also use the flush() method to clear the buffer and write contents in buffer to the file. This is how programmers can forcefully write to the file as and when required.



`write()`, the `writelines()` method does not return the number of characters written in the file. The following code explains the use of `writelines()`.

```
>>> myobject=open("myfile.txt", 'w')
>>> lines = ["Hello everyone\n", "Writing
#multiline strings\n", "This is the
#third line"]
>>> myobject.writelines(lines)
>>>myobject.close()
```

On opening `myfile.txt`, using notepad, its content will appear as shown in Figure 2.1.

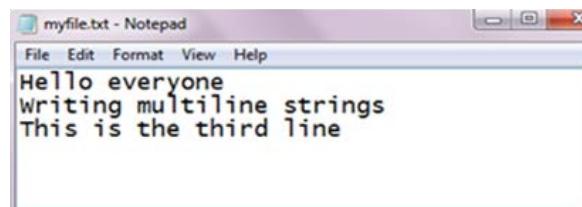


Figure 2.1: Contents of `myfile.txt`

### Activity 2.3

Run the above code by replacing `writelines()` with `write()` and see what happens.



### Think and Reflect

Can we pass a tuple of numbers as an argument to `writelines()`? Will it be written to the file or an error will be generated?



## 2.5 READING FROM A TEXT FILE

We can write a program to read the contents of a file. Before reading a file, we must make sure that the file is opened in “r”, “r+”, “w+” or “a+” mode. There are three ways to read the contents of a file:

### 2.5.1 The `read()` method

This method is used to read a specified number of bytes of data from a data file. The syntax of `read()` method is:

```
file_object.read(n)
```

Consider the following set of statements to understand the usage of `read()` method:

```
>>>myobject=open("myfile.txt", 'r')
>>> myobject.read(10)
'Hello ever'
>>> myobject.close()
```

If no argument or a negative number is specified in `read()`, the entire file content is read. For example,

```
>>> myobject=open("myfile.txt", 'r')
>>> print(myobject.read())
Hello everyone
Writing multiline strings
This is the third line
>>> myobject.close()
```

### 2.5.2 The readline([n]) method

This method reads one complete line from a file where each line terminates with a newline (\n) character. It can also be used to read a specified number (n) of bytes of data from a file but maximum up to the newline character (\n). In the following example, the second statement reads the first ten characters of the first line of the text file and displays them on the screen.

```
>>> myobject=open("myfile.txt",'r')
>>> myobject.readline(10)
'Hello ever'
>>> myobject.close()
```

If no argument or a negative number is specified, it reads a complete line and returns string.

```
>>>myobject=open("myfile.txt",'r')
>>> print (myobject.readline())
'Hello everyone\n'
```

To read the entire file line by line using the readline(), we can use a loop. This process is known as looping/ iterating over a file object. It returns an empty string when EOF is reached.

#### Activity 2.4

Create a file having multiline data and use readline() with an iterator to read the contents of the file line by line



### 2.5.3 The readlines() method

The method reads all the lines and returns the lines along with newline as a list of strings. The following example uses readlines() to read data from the text file *myfile.txt*.

```
>>> myobject=open("myfile.txt",'r')
>>> print(myobject.readlines())
['Hello everyone\n', 'Writing multiline
strings\n', 'This is the third line']
>>> myobject.close()
```

As shown in the above output, when we read a file using readlines() function, lines in the file become members of a list, where each list element ends with a newline character ('\n').

In case we want to display each word of a line separately as an element of a list, then we can use split() function. The following code demonstrates the use of split() function.

```
>>> myobject=open("myfile.txt",'r')
>>> d=myobject.readlines()
```

```

>>> for line in d:
    words=line.split()
    print(words)

['Hello', 'everyone']
['Writing', 'multiline', 'strings']
['This', 'is', 'the', 'third', 'line']

```

In the output, each string is returned as elements of a list. However, if *splitlines()* is used instead of *split()*, then each line is returned as element of a list, as shown in the output below:

```

>>> for line in d:
    words=line.splitlines()
    print(words)

['Hello everyone']
['Writing multiline strings']
['This is the third line']

```

Let us now write a program that accepts a string from the user and writes it to a text file. Thereafter, the same program reads the text file and displays it on the screen.

#### Program 2-1 Writing and reading to a text file

```

fobject=open("testfile.txt","w")      # creating a data file
sentence=input("Enter the contents to be written in the file: ")
fobject.write(sentence)              # Writing data to the file
fobject.close()                     # Closing a file

print("Now reading the contents of the file: ")
fobject=open("testfile.txt","r")
#looping over the file object to read the file
for str in fobject:
    print(str)
fobject.close()

```

In Program 2.1, the file named *testfile.txt* is opened in write mode and the file handle named *fobject* is returned. The string is accepted from the user and written in the file using *write()*. Then the file is closed and again opened in read mode. Data is read from the file and displayed till the end of file is reached.

## Output of Program 2-1:

```
>>>
      RESTART: Path_to_file\Program2-1.py
Enter the contents to be written in the file:
roll_numbers = [1, 2, 3, 4, 5, 6]
Now reading the contents of the file:
roll_numbers = [1, 2, 3, 4, 5, 6]
>>>
```

## 2.6 SETTING OFFSETS IN A FILE

The functions that we have learnt till now are used to access the data sequentially from a file. But if we want to access data in a random fashion, then Python gives us seek() and tell() functions to do so.

### 2.6.1 The tell() method

This function returns an integer that specifies the current position of the file object in the file. The position so specified is the byte position from the beginning of the file till the current position of the file object. The syntax of using tell() is:

```
file_object.tell()
```

### 2.6.2 The seek() method

This method is used to position the file object at a particular position in a file. The syntax of seek() is:

```
file_object.seek(offset [, reference_point])
```

In the above syntax, offset is the number of bytes by which the file object is to be moved. reference\_point indicates the starting position of the file object. That is, with reference to which position, the offset has to be counted. It can have any of the following values:

- 0 - beginning of the file
- 1 - current position of the file
- 2 - end of file

By default, the value of reference\_point is 0, i.e. the offset is counted from the beginning of the file. For example, the statement `fileObject.seek(5,0)` will position the file object at 5<sup>th</sup> byte position from the beginning of the file. The code in Program 2-2 below demonstrates the usage of seek() and tell().

### Think and Reflect

Does the seek() function work in the same manner for text and binary files?



## Program 2-2 Application of seek() and tell()

```
print("Learning to move the file object")
fileobject=open("testfile.txt", "r+")
str=fileobject.read()
print(str)
print("Initially, the position of the file object is: ",fileobject.tell())
fileobject.seek(0)
print("Now the file object is at the beginning of the file:
",fileobject.tell())
fileobject.seek(10)
print("We are moving to 10th byte position from the beginning of
file")
print("The position of the file object is at", fileobject.tell())
str=fileobject.read()
print(str)
```

### Output of Program 2-2:

```
>>>
RESTART: Path_to_file\Program2-2.py
Learning to move the file object
roll_numbers = [1, 2, 3, 4, 5, 6]
Initially, the position of the file object is: 33
Now the file object is at the beginning of the file: 0
We are moving to 10th byte position from the beginning of file

The position of the file object is at 10
numbers = [1, 2, 3, 4, 5, 6]
>>>
```

## 2.7 CREATING AND TRAVERSING A TEXT FILE

Having learnt various methods that help us to open and close a file, read and write data in a text file, find the position of the file object and move the file object at a desired location, let us now perform some basic operations on a text file. To perform these operations, let us assume that we will be working with practice.txt.

### 2.7.1 Creating a file and writing data

To create a text file, we use the `open()` method and provide the filename and the mode. If the file already exists with the same name, the `open()` function will behave differently depending on the mode (write or append) used. If it is in write mode (`w`), then all the existing contents of file will be lost, and an empty file will be created with the same name. But, if the file is

created in append mode (a), then the new data will be written after the existing data. In both cases, if the file does not exist, then a new empty file will be created. In Program 2-3, a file, *practice.txt* is opened in write (w) mode and three sentences are stored in it as shown in the output screen that follows it

#### Program 2-3 To create a text file and write data in it

```
# program to create a text file and add data
fileobject=open("practice.txt","w+")
while True:
    data= input("Enter data to save in the text file: ")
    fileobject.write(data)
    ans=input("Do you wish to enter more data?(y/n): ")
    if ans=='n': break
fileobject.close()
```

#### Output of Program 2-3:

```
>>>
RESTART: Path_to_file\Program2-3.py
Enter data to save in the text file: I am interested to learn about
Computer Science
Do you wish to enter more data?(y/n): y
Enter data to save in the text file: Python is easy to learn
Do you wish to enter more data?(y/n): n
>>>
```

### 2.7.2 Traversing a file and displaying data

To read and display data that is stored in a text file, we will refer to the previous example where we have created the file *practice.txt*. The file will be opened in read mode and reading will begin from the beginning of the file.

#### Program 2-4 To display data from a text file

```
fileobject=open("practice.txt","r")
str = fileobject.readline()
while str:
    print(str)
    str=fileobject.readline()
fileobject.close()
```

In Program 2-4, the `readline()` is used in the while loop to read the data line by line from the text file. The lines are displayed using the `print()`. As the end of file is reached, the `readline()` will return an empty string. Finally, the file is closed using the `close()`.

### Output of Program 2-4:

```
>>>
I am interested to learn about Computer SciencePython is easy to learn
```

Till now, we have been creating separate programs for writing data to a file and for reading the file. Now let us create one single program to read and write data using a single file object. Since both the operations have to be performed using a single file object, the file will be opened in w+ mode.

### Program 2-5 To perform reading and writing operation in a text file

```
fileobject=open("report.txt", "w+")
print ("WRITING DATA IN THE FILE")
print() # to display a blank line
while True:
    line= input("Enter a sentence ")
    fileobject.write(line)
    fileobject.write('\n')
    choice=input("Do you wish to enter more data? (y/n): ")
    if choice in ('n', 'N'): break
print("The byte position of file object is ",fileobject.tell())
fileobject.seek(0) #places file object at beginning of file
print()
print("READING DATA FROM THE FILE")
str=fileobject.read()
print(str)
fileobject.close()
```

In Program 2-5, the file will be read till the time end of file is not reached and the output as shown in below is displayed.

### Output of Program 2-5:

```
>>>
RESTART: Path_to_file\Program2-5.py
WRITING DATA IN THE FILE

Enter a sentence I am a student of class XII
Do you wish to enter more data? (y/n): y
Enter a sentence my school contact number is 4390xxxx8
Do you wish to enter more data? (y/n): n
The byte position of file object is 67

READING DATA FROM THE FILE
I am a student of class XII
my school contact number is 4390xxxx8
>>>
```

## 2.8 THE PICKLE MODULE

We know that Python considers everything as an object. So, all data types including list, tuple, dictionary, etc. are also considered as objects. During execution of a program, we may require to store current state of variables so that we can retrieve them later to its present state. Suppose you are playing a video game, and after some time, you want to close it. So, the program should be able to store the current state of the game, including current level/stage, your score, etc. as a Python object. Likewise, you may like to store a Python dictionary as an object, to be able to retrieve later. To save any object structure along with data, Python provides a module called Pickle. The module Pickle is used for serializing and de-serializing any Python object structure. Pickling is a method of preserving food items by placing them in some solution, which increases the shelf life. In other words, it is a method to store food items for later consumption.

Serialization is the process of transforming data or an object in memory (RAM) to a stream of bytes called byte streams. These byte streams in a binary file can then be stored in a disk or in a database or sent through a network. Serialization process is also called pickling.

De-serialization or unpickling is the inverse of pickling process where a byte stream is converted back to Python object.

The pickle module deals with binary files. Here, data are not written but dumped and similarly, data are not read but loaded. The Pickle Module must be imported to load and dump data. The pickle module provides two methods - `dump()` and `load()` to work with binary files for pickling and unpickling, respectively.

### 2.8.1 The `dump()` method

This method is used to convert (pickling) Python objects for writing data in a binary file. The file in which data are to be dumped, needs to be opened in binary write mode (`wb`).

Syntax of `dump()` is as follows:

```
dump(data_object, file_object)
```

where `data_object` is the object that has to be dumped to the file with the file handle named `file_`

object. For example, Program 2-6 writes the record of a student (roll\_no, name, gender and marks) in the binary file named *mybinary.dat* using the *dump()*. We need to close the file after pickling.

#### Program 2-6 Pickling data in Python

```
import pickle
listvalues=[1,"Geetika",'F', 26]
fileobject=open("mybinary.dat", "wb")
pickle.dump(listvalues,fileobject)
fileobject.close()
```

#### 2.8.2 The *load()* method

This method is used to load (unpickling) data from a binary file. The file to be loaded is opened in binary read (rb) mode. Syntax of *load()* is as follows:

```
Store_object = load(file_object)
```

Here, the pickled Python object is loaded from the file having a file handle named *file\_object* and is stored in a new file handle called *store\_object*. The program 2-7 demonstrates how to read data from the file *mybinary.dat* using the *load()*.

#### Program 2-7 Unpickling data in Python

```
import pickle
print("The data that were stored in file are: ")
fileobject=open("mybinary.dat","rb")
objectvar=pickle.load(fileobject)
fileobject.close()
print(objectvar)
```

Output of Program 2-7:

```
>>>
RESTART: Path_to_file\Program2-7.py
The data that were stored in file are:
[1, 'Geetika', 'F', 26]
>>>
```

#### 2.8.3 File handling using *pickle* module

As we read and write data in a text file, similarly we will be adding and displaying data for a binary file. Program 2-8 accepts a record of an employee from the user and appends it in the binary file *tv*. Thereafter, the records are read from the binary file and displayed on the screen using the same object. The user may enter

as many records as they wish to. The program also displays the size of binary files before starting with the reading process.

#### Program 2-8 To perform basic operations on a binary file using pickle module

```
# Program to write and read employee records in a binary file
import pickle
print("WORKING WITH BINARY FILES")
bfile=open("empfile.dat", "ab")
recno=1
print ("Enter Records of Employees")
print()
#taking data from user and dumping in the file as list object
while True:
    print("RECORD No.", recno)
    eno=int(input("\tEmployee number : "))
    ename=input("\tEmployee Name : ")
    ebasic=int(input("\tBasic Salary : "))
    allow=int(input("\tAllowances : "))
    totsal=ebasic+allow
    print("\tTOTAL SALARY : ", totsal)
    edata=[eno,ename,ebasic,allow,totsal]
    pickle.dump(edata,bfile)
    ans=input("Do you wish to enter more records (y/n) ? ")
    recno=recno+1
    if ans.lower()=='n':
        print("Record entry OVER ")
        print()
        break
# retrieving the size of file
print("Size of binary file (in bytes):",bfile.tell())
bfile.close()
# Reading the employee records from the file using load() module
print("Now reading the employee records from the file")
print()
readrec=1
try:
    with open("empfile.dat", "rb") as bfile:
        while True:
            edata=pickle.load(bfile)
            print("Record Number : ",readrec)
            print(edata)
            readrec=readrec+1
except EOFError:
    pass
bfile.close()
```

### Output of Program 2-8:

```
>>>
      RESTART: Path_to_file\Program2-8.py
      WORKING WITH BINARY FILES
      Enter Records of Employees

      RECORD No. 1
          Employee number : 11
          Employee Name : D N Ravi
          Basic Salary : 32600
          Allowances : 4400
          TOTAL SALARY : 37000
      Do you wish to enter more records (y/n) ? y
      RECORD No. 2
          Employee number : 12
          Employee Name : Farida Ahmed
          Basic Salary : 38250
          Allowances : 5300
          TOTAL SALARY : 43550
      Do you wish to enter more records (y/n) ? n
      Record entry OVER

      Size of binary file (in bytes): 216
      Now reading the employee records from the file

      Record Number : 1
      [11, 'D N Ravi', 32600, 4400, 37000]
      Record Number : 2
      [12, 'Farida Ahmed', 38250, 5300, 43550]
>>>
```

As each employee record is stored as a list in the file empfile.dat, hence while reading the file, a list is displayed showing record of each employee. Notice that in Program 2-8, we have also used try.. except block to handle the end-of-file exception.

### SUMMARY

- A file is a named location on a secondary storage media where data are permanently stored for later access.
- A text file contains only textual information consisting of alphabets, numbers and other

## NOTES

special symbols. Such files are stored with extensions like `.txt`, `.py`, `.c`, `.csv`, `.html`, etc. Each byte of a text file represents a character.

- Each line of a text file is stored as a sequence of ASCII equivalent of the characters and is terminated by a special character, called the End of Line (EOL).
- Binary file consists of data stored as a stream of bytes.
- `open()` method is used to open a file in Python and it returns a file object called file handle. The file handle is used to transfer data to and from the file by calling the functions defined in the Python's `io` module.
- `close()` method is used to close the file. While closing a file, the system frees up all the resources like processor and memory allocated to it.
- `write()` method takes a string as an argument and writes it to the text file.
- `writelines()` method is used to write multiple strings to a file. We need to pass an iterable object like lists, tuple etc. containing strings to `writelines()` method.
- `read([n])` method is used to read a specified number of bytes (`n`) of data from a data file.
- `readline([n])` method reads one complete line from a file where lines are ending with a newline (`\n`). It can also be used to read a specified number (`n`) of bytes of data from a file but maximum up to the newline character (`\n`).
- `readlines()` method reads all the lines and returns the lines along with newline character, as a list of strings.
- `tell()` method returns an integer that specifies the current position of the file object. The position so specified is the byte position from the beginning of the file till the current position of the file object.
- `seek()` method is used to position the file object at a particular position in a file.

- Pickling is the process by which a Python object is converted to a byte stream.
- `dump()` method is used to write the objects in a binary file.
- `load()` method is used to read data from a binary file.



## EXERCISE

1. Differentiate between:
  - a) text file and binary file
  - b) readline() and readlines()
  - c) write() and writelines()
2. Write the use and syntax for the following methods:
  - a) `open()`
  - b) `read()`
  - c) `seek()`
  - d) `dump()`
3. Write the file mode that will be used for opening the following files. Also, write the Python statements to open the following files:
  - a) a text file “example.txt” in both read and write mode
  - b) a binary file “bfile.dat” in write mode
  - c) a text file “try.txt” in append and read mode
  - d) a binary file “btry.dat” in read only mode.
4. Why is it advised to close a file after we are done with the read and write operations? What will happen if we do not close it? Will some error message be flashed?
5. What is the difference between the following set of statements (a) and (b):
  - a) `P = open("practice.txt", "r")`  
`P.read(10)`
  - b) with `open("practice.txt", "r") as P:`  
`x = P.read()`
6. Write a command(s) to write the following lines to the text file named `hello.txt`. Assume that the file is opened in append mode.
 

“Welcome my class”  
 “It is a fun place”  
 “You will learn and play”

## NOTES

7. Write a Python program to open the file *hello.txt* used in question no 6 in read mode to display its contents. What will be the difference if the file was opened in write mode instead of append mode?
8. Write a program to accept string/sentences from the user till the user enters “END” to. Save the data in a text file and then display only those sentences which begin with an uppercase alphabet.
9. Define pickling in Python. Explain serialization and deserialization of Python object.
10. Write a program to enter the following records in a binary file:

|           |         |
|-----------|---------|
| Item No   | integer |
| Item_Name | string  |
| Qty       | integer |
| Price     | float   |

Number of records to be entered should be accepted from the user. Read the file to display the records in the following format:

Item No:  
Item Name :  
Quantity:  
Price per item:  
Amount: ( to be calculated as Price \* Qty)

Chapter

3

# Stack



12130CH03

## In this Chapter

- » *Introduction*
- » *Stack*
- » *Operations on Stack*
- » *Implementation of Stack in Python*
- » *Notations for Arithmetic Expressions*
- » *Conversion From Infix To Postfix Notation*
- » *Evaluation of Postfix Expression*

*“We’re going to be able to ask our computers to monitor things for us, and when certain conditions happen, are triggered, the computers will take certain actions and inform us after the fact.”*

— Steve Jobs

### 3.1 INTRODUCTION

We have learnt about different data types in Python for handling values in Class XI. Recall that String, List, Set, Tuple, etc. are the sequence data types that can be used to represent collection of elements either of the same type or different types. Multiple data elements are grouped in a particular way for faster accessibility and efficient storage of data. That is why we have used different data types in python for storing data values. Such grouping is referred as a data structure.

A data structure defines a mechanism to store, organise and access data along with operations (processing) that can be efficiently performed on the data. For example, string is a data structure containing a sequence of elements where each element is a character. On the other hand, list is a sequence data structure in which each element may be of different types. We can apply different operations like reversal, slicing, counting of



Other important data structures in Computer Science include Array, Linked List, Binary Trees, Heaps, Graph, Sparse Matrix, etc.



A data structure in which elements are organised in a sequence is called linear data structure.

elements, etc. on list and string. Hence, a data structure organises multiple elements in a way so that certain operations on each element as well as the collective data unit could be performed easily.

*Stack* and *Queue* are two other popular data structures used in programming. Although not directly available in Python, it is important to learn these concepts as they are extensively used in a number of programming languages. In this chapter, we will study about stack, its implementation using Python as well as its applications.

### 3.2 STACK

We have seen piles of books in the library or stack of plates at home (Figure 3.1). To put another book or another plate in such a pile, we always place (add to the pile) the object at the top only. Likewise, to remove a book or a plate from such a pile, we always remove (delete from the pile) the object from the top only. This is because in a large pile, it is inconvenient to add or remove an object from in between or bottom. Such an arrangement of elements in a linear order is called a stack. We add new elements or remove existing elements from the same end, commonly referred to as the *top* of the stack. It thus follows the Last-In-First-out (LIFO) principle. That is, the element which was inserted last (the most recent element) will be the first one to be taken out from the stack.

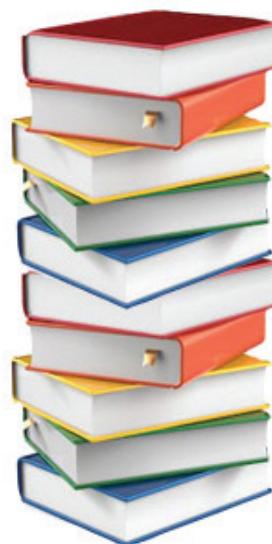


Figure 3.1: Stack of plates and books

### 3.2.1 APPLICATIONS OF STACK

Some of the applications of stack in real-life are:

- Pile of clothes in an almirah
- Multiple chairs in a vertical pile
- Bangles worn on wrist
- Pile of boxes of eatables in pantry or on a kitchen shelf

Some examples of application of stack in programming are as follows:

- When we need to reverse a string, the string is traversed from the last character till the first character. i.e. characters are traversed in the reverse order of their appearance in the string. This is very easily done by putting the characters of a string in a stack.
- We use text/image editor for editing the text/image where we have options to redo/undo the editing done. When we click on the redo /undo icon, the most recent editing is redone/undone. In this scenario, the system uses a stack to keep track of changes made.
- While browsing the web, we move from one web page to another by accessing links between them. In order to go back to the last visited web page, we may use the back button on the browser. Let us say we accessed a web page P1 from where we moved to web page P2 followed by browsing of web page P3. Currently, we are on web page P3 and want to revisit web page P1. We may go to a previously visited web page by using the BACK button of the browser. On clicking the BACK button once, we are taken from web page P3 to web page P2, another click on BACK shows web page P1. In this case, the history of browsed pages is maintained as stack.
- While writing any arithmetic expression in a program, we may use parentheses to order the evaluation of operators. While executing the program, the compiler checks for matched parentheses i.e. each opening parenthesis should have a corresponding closing parenthesis and the pairs of parentheses are properly nested. In case of parentheses are

### Think and Reflect

How does a compiler or an interpreter handle function calls in a program?



### Think and Reflect

The operating system in computer or mobile allocates memory to different applications for their execution. How does an operating system keep track of the free memory that can be allocated among programs/ applications to be executed?



mismatched, the compiler needs to throw an error. To handle matching of parentheses, stack is used.

### 3.3 OPERATIONS ON STACK

As explained in the previous section, a stack is a mechanism that implements LIFO arrangement hence elements are added and deleted from the stack at one end only. The end from which elements are added or deleted is called TOP of the stack. Two fundamental operations performed on the stack are PUSH and POP. In this section, we will learn about them and implement them using Python.

#### 3.3.1 PUSH and POP Operations

- PUSH adds a new element at the TOP of the stack. It is an insertion operation. We can add elements to a stack until it is full. A stack is full when no more elements can be added to it. Trying to add an element to a full stack results in an exception called ‘overflow’.
- POP operation is used to remove the top most element of the stack, that is, the element at the TOP of the stack. It is a delete operation. We can delete elements from a stack until it is empty i.e. there is no element in it. Trying to delete an element from an empty stack results in an exception called ‘underflow’.

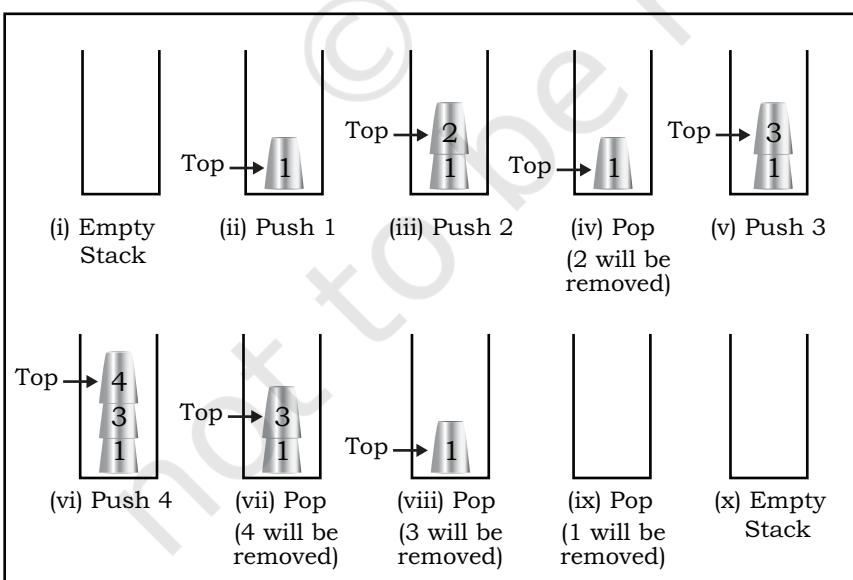


Figure 3.2: PUSH and POP operations on the stack of glasses

A stack is used to insert and delete elements in LIFO order. Same principle is followed in adding and removing glasses from a pile of glasses. Let us create a stack of glasses assuming that each glass is numbered. Visual representations of PUSH and POP operations on a stack of glasses are shown in Figure 3.2.

### 3.4 IMPLEMENTATION OF STACK IN PYTHON

### NOTES

We have learnt so far that a stack is a linear and ordered collection of elements. The simple way to implement a stack in Python is using the data type *list*. We can fix either of the sides of the list as TOP to insert/remove elements. It is to be noted that we are using built-in methods *append()* and *pop()* of the list for implementation of the stack. As these built-in methods insert/delete elements at the rightmost end of the list, hence explicit declaration of TOP is not needed.

Let us write a program to create a STACK (stack of glasses as given in Figure 3.2) in which we will:

- insert/delete elements (glasses)
- check if the STACK is empty (no glasses in the stack)
- find the number of elements (glasses) in the STACK
- read the value of the topmost element (number on the topmost glass) in the STACK

The program shall define the following functions to perform these operations:

- Let us create an empty stack named *glassStack*. We will do so by assigning an empty list to the identifier named *glassStack*:

```
glassStack = list()
```

- A function named *isEmpty* to check whether the stack *glassStack* is empty or not. Remember trying to remove an element from an empty stack would result in ‘underflow’. This function returns True if the stack is empty, else returns False.

```
def isEmpty(glassStack):  
    if len(glassStack)==0:  
        return True  
    else:  
        return False
```

- A function named *opPush* to insert (PUSH) a new element in stack. This function has two parameters - the name of the stack in which the element is to be inserted (*glassStack*) and the element that needs to be inserted. We know that insertion of an element is always done at the TOP of the stack. Hence, we

## NOTES

shall use the built-in method `append()` of list to add an element to the stack that always adds at the end of the list. As there is no limit on the size of list in Python, the implemented stack will never be full unless there is no more space available in memory. Hence, we will never face ‘overflow’ (no space for new element) condition for stack.

```
def opPush(glassStack,element):  
    glassStack.append(element)
```

- A function named `size` to read the number of elements in the `glassStack`. We will use the `len()` function of list in Python to find the number of elements in the `glassStack`.

```
def size(glassStack):  
    return len(glassStack)
```

- A function named `top` to read the most recent element (`TOP`) in the `glassStack`.

```
def top(glassStack):  
    if isEmpty(glassStack):  
        print('Stack is empty')  
        return None  
    else:  
        x = len(glassStack)  
        element=glassStack[x-1]  
        return element
```

- A function named `opPop` to delete the topmost element from the stack. It takes one parameter - the name of the stack (`glassStack`) from which element is to be deleted and returns the value of the deleted element. The function first checks whether the stack is empty or not. If it is not empty, it removes the topmost element from it. We shall use the built-in method `pop()` of Python list that removes the element from the end of the list.

```
def opPop(glassStack):  
    if isEmpty(glassStack):  
        print('underflow')  
        return None  
    else:  
        return(glassStack.pop())
```

- A function named `display` to show the contents of the stack.

## NOTES

```
def display(glassStack):
    x=len(glassStack)
    print("Current elements in the stack
are: ")
    for i in range(x-1,-1,-1):
        print(glassStack[i])
```

Once we define the above functions we can use the following Python code to implement a stack of glasses.

```
glassStack = list() # create empty stack

#add elements to stack
element='glass1'
print("Pushing element ",element)
opPush(glassStack,element)
element='glass2'
print("Pushing element ",element)
opPush(glassStack,element)

#display number of elements in stack
print("Current number of elements in stack
is",size(glassStack))

#delete an element from the stack
element=opPop(glassStack)
print("Popped element is",element)

#add new element to stack
element='glass3'
print("Pushing element ",element)
opPush(glassStack,element)

#display the last element added to the
#stack
print("top element is",top(glassStack))

#display all elements in the stack
display(glassStack)
```

```
#delete all elements from stack
while True:
    item=opPop(glassStack)
    if item == None:
        print("Stack is empty now")
        break
    else:
        print("Popped element is",item)
```

The output of the above program will be as follows:

```
Pushing element glass1
Pushing element glass2
Current number of elements in stack is 2
Popped element is glass2
Pushing element glass3
top element is glass3
Current elements in the stack are:
glass3
glass1
Popped element is glass3
Popped element is glass1
Underflow
Stack is empty now
```

### 3.5 NOTATIONS FOR ARITHMETIC EXPRESSIONS

We write arithmetic expressions using operators in between operands, like  $x + y$ ,  $2 - 3 * y$ , etc. and use parentheses () to order the evaluation of operators in complex expressions. These expressions follow infix representation and are evaluated using BODMAS rule.

Polish mathematician Jan Lukasiewicz in the 1920's introduced a different way of representing arithmetic expression, called polish notation. In such notation, operators are written before their operands. So the order of operations and operands determines the result, making parentheses unnecessary. For example, we can write  $x+y$  in polish notation as  $+xy$ . This is also called prefix notation as we prefix the operator before operands.

By reversing this logic, we can write an expression by putting operators after their operands. For example,  $x+y$  can be written as  $xy+$ . This is called reverse polish

notation or postfix notation. To summarise, any arithmetic expression can be represented in any of the three notations viz. Infix, Prefix and Postfix and are listed in Table 3.1 with examples.

**Table 3.1 Infix, Prefix and Postfix Notations**

| Type of Expression       | Description                                            | Example                                             |
|--------------------------|--------------------------------------------------------|-----------------------------------------------------|
| Infix                    | Operators are placed in between the operands           | $x * y + z$<br>$3 * (4 + 5)$<br>$(x + y) / (z * 5)$ |
| Prefix (Polish)          | Operators are placed before the corresponding operands | $+z*xy$<br>$*3+45$<br>$/+xy*z5$                     |
| Postfix (Reverse Polish) | Operators are placed after the corresponding operands  | $xy*z+$<br>$345+*$<br>$xy+z5*/$                     |

### 3.6 CONVERSION FROM INFIX TO POSTFIX NOTATION

It is easy for humans to evaluate an *infix* expression. Consider an *infix* expression  $x + y / z$ . While going from left to right we first encounter + operator, but we do not add  $x + y$  and rather evaluate  $y/z$ , followed by addition operation. This is because we know the order of precedence of operators that follows BODMAS rule. But, how do we pass this precedence knowledge to the computer through an expression?

In contrast, *prefix/postfix* expressions do not have to deal with such precedence because the operators are already positioned according to their order of evaluation. Hence a single traversal from left to right is sufficient to evaluate the expression. In this section, we will learn about the conversion of an arithmetic expression written in *infix* notation to its equivalent expression in *postfix* notation using a stack.

During such conversion, a stack is used to keep track of the operators encountered in the *infix* expression. A variable of string type is used to store the equivalent *postfix* expression. Algorithm 3.1 converts an expression in *infix* notation to *postfix* notation:

#### Algorithm 3.1: Conversion of expression from infix to postfix notation

Step 1: Create an empty string named postExp to store the converted postfix expression.

Step 2: INPUT infix expression in a variable, say inExp

Step 3: For each character in inExp, REPEAT Step 4

STACK

#### Think and Reflect

Write an algorithm to convert an infix expression into equivalent prefix expression using stack.



**Step 4:** IF character is a left parenthesis THEN PUSH on the Stack  
ELSE IF character is a right parenthesis  
THEN POP the elements from the Stack and append to string  
postExp until the corresponding left parenthesis is popped  
while discarding both left and right parentheses

ELSE IF character is an operator  
THEN IF its precedence is lower than that of operator at the top of Stack  
THEN POP elements from the Stack till an  
operator with precedence less than the current  
operator is encountered and append to string  
postExp before pushing this operator on the  
postStack

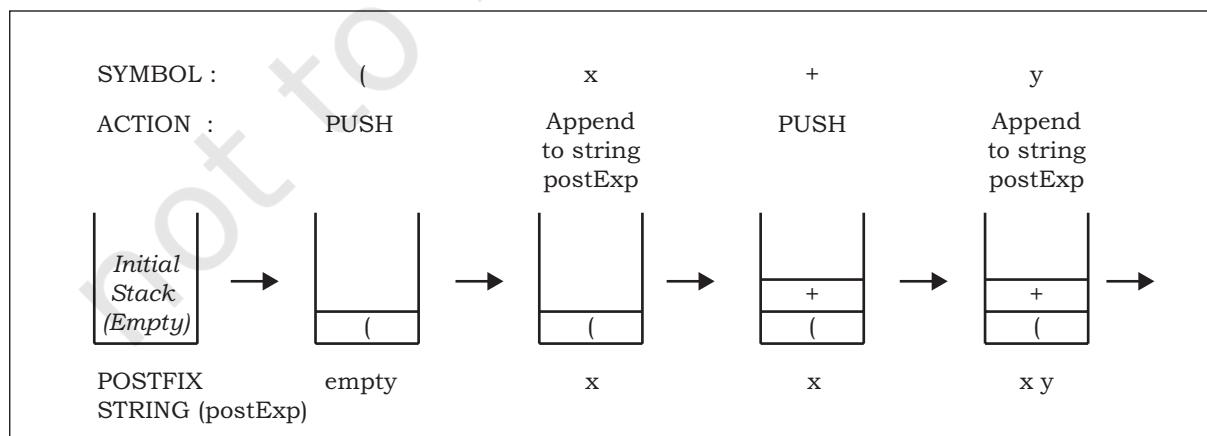
ELSE PUSH operator on the Stack  
ELSE Append the character to postExp

**Step 5:** Pop elements from the Stack and append to postExp until Stack is empty

**Step 6:** OUTPUT postExp

### Example 3.1

Let us now use this algorithm to convert a given infix expression  $(x + y) / (z^8)$  into equivalent postfix expression using a stack. Figure 3.3 shows the steps to be followed on encountering an operator or an operand in the given infix expression. Note here that stack is used to track the operators and parentheses, and a string variable contains the equivalent postfix expression. Initially both are empty. Each character in the given infix expression is processed from left to right and the appropriate action is taken as detailed in the algorithm. When each character in the given infix expression has been processed, the string will contain the equivalent postfix expression.



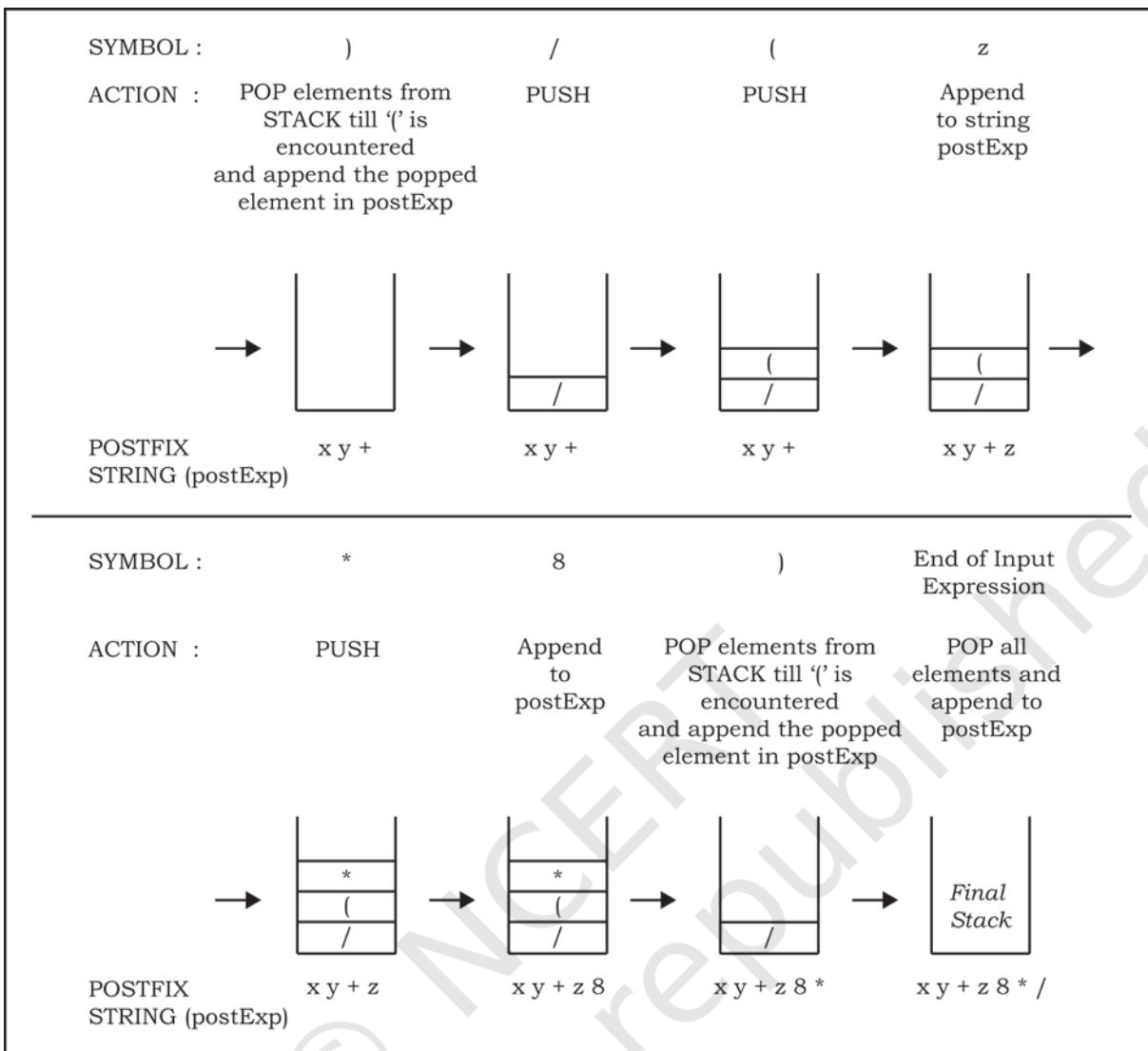


Figure 3.3: Conversion of infix expression  $(x + y)/(z^*8)$  to postfix notation

### 3.7 EVALUATION OF POSTFIX EXPRESSION

Stacks can be used to evaluate an expression in postfix notation. For simplification, we are assuming that operators used in expressions are binary operators. The detailed step by step procedure is given in Algorithm 3.2.

#### Algorithm 3.2: Evaluation of postfix expression

Step 1: INPUT postfix expression in a variable, say postExp

Step 2: For each character in postExp, REPEAT Step 3

Step 3: IF character is an operand

THEN PUSH character on the Stack

ELSE POP two elements from the Stack, apply the operator on

the popped elements and PUSH the computed value onto the Stack

*Step 4:* IF Stack has a single element

THEN POP the element and OUTPUT as the net result

ELSE OUTPUT “Invalid Postfix expression”

### Example 3.2

Figure 3.4 shows the step-by-step process of evaluation of the postfix expression  $7\ 8\ 2\ *\ 4\ / +$  using Algorithm 3.2 .

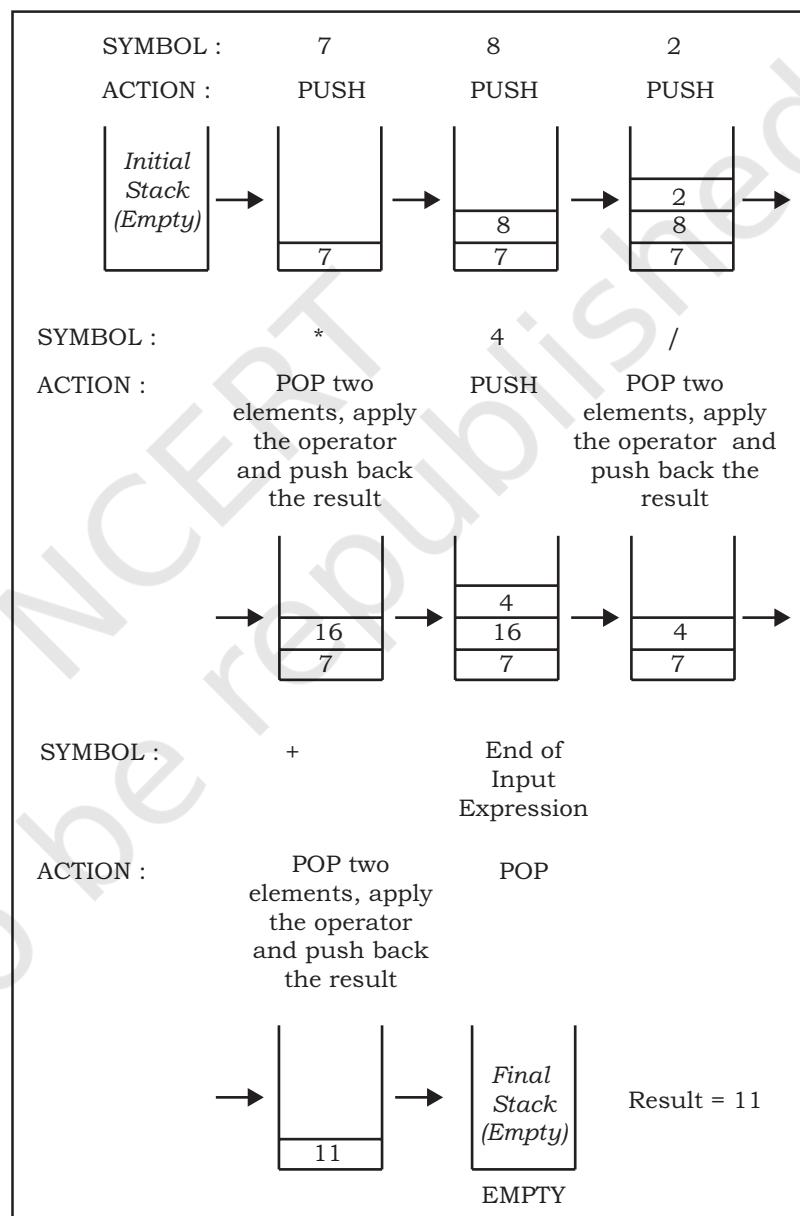


Figure 3.4: Evaluation of postfix expression  $7\ 8\ 2\ *\ 4\ / +$

## SUMMARY

- Stack is a data structure in which insertion and deletion is done from one end only, usually referred to as TOP.
- Stack follows LIFO principle using which an element inserted in the last will be the first one to be out.
- PUSH and POP are two basic operations performed on a stack for insertion and deletion of elements, respectively.
- Trying to pop an element from an empty stack results into a special condition *underflow*.
- In Python, list is used for implementing a stack and its built-in-functions *append* and *pop* are used for insertion and deletion, respectively. Hence, no explicit declaration of TOP is needed.
- Any arithmetic expression can be represented in any of the three notations viz. Infix, Prefix and Postfix.
- While programming, Infix notation is used for writing an expression in which binary operators are written in between the operands.
- A single traversal from left to right of Prefix/ Postfix expression is sufficient to evaluate the expression as operators are correctly placed as per their order of precedence.
- Stack is commonly used data structure to convert an Infix expression into equivalent Prefix/Postfix notation.
- While conversion of an Infix notation to its equivalent Prefix/Postfix notation, only operators are PUSHed onto the Stack.
- When evaluating any Postfix expression using Stack, only operands are PUSHed onto it.

## NOTES

## EXERCISE

1. State TRUE or FALSE for the following cases:
  - a) Stack is a linear data structure
  - b) Stack does not follow LIFO rule
  - c) PUSH operation may result into underflow condition
  - d) In POSTFIX notation for expression, operators are placed after operands
2. Find the output of the following code:
  - a) result=0  

```
numberList=[10,20,30]
numberList.append(40)
result=result+numberList.pop()
result=result+numberList.pop()
print("Result=",result)
```
  - b) answer=[]; output=""
 

```
answer.append('T')
answer.append('A')
answer.append('M')
ch=answer.pop()
output=output+ch
ch=answer.pop()
output=output+ch
ch=answer.pop()
output=output+ch
print("Result=",output)
```
3. Write a program to reverse a string using stack.
4. For the following arithmetic expression:  
 $((2+3)*(4/2))+2$   
 Show step-by-step process for matching parentheses using stack data structure.
5. Evaluate following postfix expressions while showing status of stack after each operation given A=3, B=5, C=1, D=4
  - a) A B + C \*
  - b) A B \* C / D \*
6. Convert the following infix notations to postfix notations, showing stack and string contents at each step.
  - a) A + B - C \* D
  - b) A \* (( C + D)/E)
7. Write a program to create a Stack for storing only odd numbers out of all the numbers entered by the user. Display the content of the Stack along with the largest odd number in the Stack. (Hint. Keep popping out the elements from stack and maintain the largest element retrieved so far in a variable. Repeat till Stack is empty)

# Chapter

# 4

# Queue



12130CH04

## In this Chapter

- » Introduction to Queue
- » Operations on Queue
- » Implementation of Queue using Python
- » Introduction to Deque
- » Implementation of Deque using Python

*"We could say we want the Web to reflect a vision of the world where everything is done democratically. To do that, we get computers to talk with each other in such a way as to promote that ideal."*

— Tim Berners-Lee

## 4.1 INTRODUCTION TO QUEUE

In the previous chapter we learned about a data structure called Stack, which works on Last-In-First-Out (LIFO) principle. In this chapter, we will learn about another data structure called Queue which works on First-In-First-Out (FIFO) principle. Queue is an ordered linear list of elements, having different ends for adding and removing elements in it.

Examples of queue in our everyday life include students standing in a queue for morning assembly, customers forming a queue at the cash counter in a bank (Figure 4.1), vehicles queued at fuel pumps (Figure 4.2), etc.

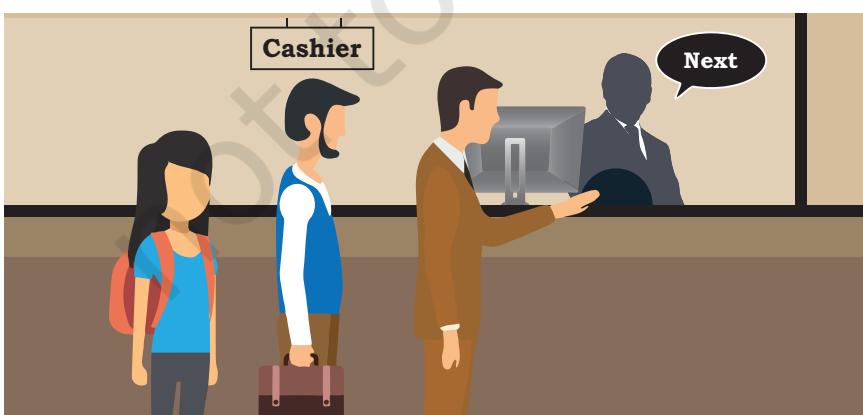


Figure 4.1: Queue of people at a bank

## Petrol Pump

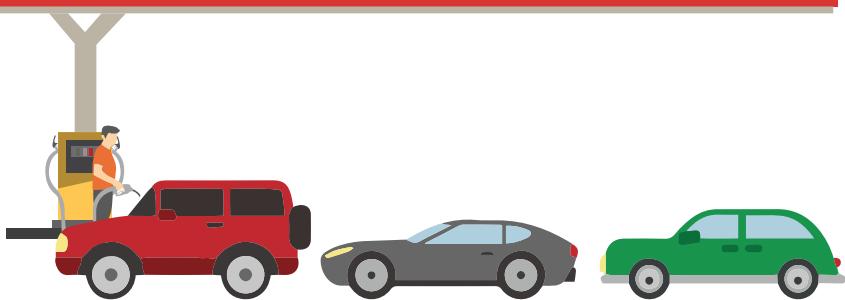


Figure 4.2: Queue of cars in a petrol pump

### 4.1.1 First In First Out (FIFO)

Queue follows the principle of First In First Out (FIFO), since the element entering first in the queue will be the first one to come out of it. Thus, the element that has been longest in the queue will be removed first. It is also known as a First Come First Served (FCFS) approach. Queue is an arrangement in which new objects/items always get added at one end, usually called the REAR, and objects/items always get removed from the other end, usually called the FRONT of the queue. REAR is also known as TAIL and FRONT as HEAD of a queue.

### 4.1.2 Applications of Queue

#### (A) The concept of queue has many applications in real-life:

- If a train ticket is in the waiting list (such as W/L1), it means the ticket is in a queue of tickets waiting to get confirmed, as per the increasing order of waiting numbers. If a confirmed ticket is cancelled, the W/L1 numbered ticket is removed from the FRONT of the waiting queue and confirmed.
- Sometimes on calling a customer service centre, the Interactive Voice Response System (IVRS) tells us to wait till a support person is available. Here the call is put into a queue of customers waiting to be serviced.
- Imagine there is a single-lane one-way road, then the vehicle that entered first will exit first, following the concept of queue. Likewise, vehicles in a highway toll tax booth are served following the principle of FIFO.

**(B) Following are some examples of application of queue in computer science:**

- Suppose there is a web-server hosting a web-site to declare result(s). This server can handle a maximum of 50 concurrent requests to view result(s). So, to serve thousands of user requests, a Queue would be the most appropriate data structure to use.
- Some Operating Systems (OS) are required to handle multiple tasks called - jobs, seeking to use the processor. But we know that a processor can handle only one task at a time. Therefore, in a multitasking operating system, jobs are lined up (queued) and then given access to the processor according to some order. The simplest way is to give access to the processor on a FIFO basis, that is according to the order in which the jobs arrive with a request for the processor.
- When we send print commands from multiple files from the same computer or from different computers using a shared printer. The OS puts these print requests in a queue and sends them to the printer one by one on a FIFO basis.

## 4.2 OPERATIONS ON QUEUE

Following the FIFO approach, data structure queue supports the following operations:

- ENQUEUE: is used to insert a new element to the queue at the rear end. We can insert elements in the queue till there is space in the queue for adding more elements. Inserting elements beyond capacity of the queue will result in an exception - known as Overflow.
- DEQUEUE: is used to remove one element at a time from the front of the queue. We can delete elements from a queue until it is empty, trying to delete an element from an empty queue will result in exception - known as Underflow.

To perform enqueue and dequeue efficiently on a queue, following operations are also required:

- IS EMPTY : used to check whether the queue has any element or not, so as to avoid Underflow exception while performing dequeue operation.

### Think and Reflect

In the web-server example (for result declaration), suppose the server receives a request from an Administrator to access the result of a school on an urgent basis, along with other requests from students to check individual results. Can you suggest some strategy to ensure service to all as per their urgency?



While using a list to implement queue, we can designate either end of the list as Front or Rear of the queue. But we have to fix either of the ends index[0] or index[n-1] as Front and fix the opposite end as Rear.

- **PEEK** : used to view elements at the front of the queue, without removing it from the queue.
- **IS FULL** : used to check whether any more elements can be added to the queue or not, to avoid Overflow exceptions while performing enqueue operation.

Figure 4.3 shows the various stages of a simple queue containing alphabets. In the figure, Front of the queue is on the left and Rear on the right.

| Operation performed | Status of queue after operation |
|---------------------|---------------------------------|
| enqueue(z)          | F → [Z] ← R                     |
| enqueue(x)          | F → [Z] [X] ← R                 |
| enqueue(c)          | F → [Z] [X] [C] ← R             |
| dequeue()           | F → [X] [C] ← R                 |
| enqueue(v)          | F → [X] [C] [V] ← R             |
| dequeue()           | F → [C] [V] ← R                 |
| dequeue()           | F → [V] ← R                     |

Figure 4.3: Various Stages of Stack Operations

### 4.3 IMPLEMENTATION OF QUEUE USING PYTHON

There are many ways in which queues can be implemented in a computer program, one way is using the list data type of Python. For creating a queue structure in the program, following functions need to be defined:

- Let's create a queue named myQueue. We can create it by assigning an empty list.

```
myQueue = list()
```

- A function (enqueue) to insert a new element at the end of queue. The function has two parameters - name of the queue and element which is to be inserted in the queue.

```
def enqueue(myQueue, element):
    myQueue.append(element)
```

**Note:** append() function always adds an element at the end of the list, hence Rear of queue.

- We don't need to implement Is Full, as Python being a dynamic language, does not ask for the

creation of list having fixed size. Hence, we will never encounter a situation when the queue is full.

- A function (`isEmpty`) to check, if the queue has an element or not? This can be done by checking the length of the queue. The function has a parameter -- name of the queue and returns True if the queue is empty False otherwise.

```
def isEmpty(myQueue):  
    if len(myQueue)==0:  
        return True  
    else:  
        return False
```

- A function (`dequeue`) to delete an element from the front of the queue. It has one parameter - name of the queue and returns the deleted element. The function first checks if the queue is empty or not, for successful deletion.

```
def dequeue(myQueue):  
    if not (isEmpty(myQueue)):  
        return myQueue.pop(0)  
    else :  
        print("Queue is empty")
```

**Note:** The `pop()` function with index[0] will delete the element from the beginning of the list, hence Front of queue.

- A function (`size`) to get the number of elements in the queue. We can use the `len()` function of Python's list to find the number of elements in the queue. The function has one parameter - name of the queue and returns the number of elements in the queue.

```
def size(myQueue):  
    return len(myQueue)
```

- A function (`peek`) to simply read, but not to delete, the element at the front end of the queue. For this, we can read the element at index[0] of the queue. The function has one parameter - name of the queue and returns the value of element at Front if queue is not empty, None otherwise.

```
def peek(myQueue):  
    if isEmpty(myQueue):  
        print('Queue is empty')  
        return None  
    else:  
        return myQueue[0]
```

### Think and Reflect

Can you implement a queue data structure using tuple or dictionary?



While choosing the name of above functions general naming convention w.r.t. the queue is followed. As these are user defined functions any other name can also be used.



#### Activity 4.1

How can you avoid printing of None, when trying to print an empty queue?



#### Activity 4.2

What if the content of the complete queue is to be listed?  
Write a function for it.



Let us consider the example of a queue that people form while waiting at a bank cash counter. Usually, following are the events that occur in queue:

- Two friends come together and go to the cash counter, i.e. they form a queue - enqueue operation is performed two times.
- As soon as the person at the front is serviced, he will be removed from the queue - thus dequeue operation is performed. Cashier calls Next to serve the next person who is now at the front of the queue.
- Cashier wants to know the length of the queue - size of the queue is checked.
- Meanwhile, a few more people walk in the bank, and three of them join the queue at the cash counter, i.e. enqueue happens 3 times.
- Another person gets served and leaves the counter, i.e. dequeue is performed. Cashier calls Next to serve another person.
- The Next three people get served one after another, i.e. dequeue is performed thrice.
- Cashier calls Next and realises that there are no more people to be served - underflow situation happens

Now, let us write the code for the above scenario of the bank.

#### Program 4-1

```
myQueue = list()
# each person to be assigned a code as P1, P2, P3, ...
element = input("enter person's code to enter in queue :")
enqueue(myQueue,element)
element = input("enter person's code for insertion in queue :")
enqueue(myQueue,element)
print("person removed from queue is:", dequeue(myQueue))
print("Number of people in the queue is :",size(myQueue))
element = input("enter person's code to enter in queue :")
enqueue(myQueue,element)
element = input("enter person's code to enter in queue :")
enqueue(myQueue,element)
element = input("enter person's code to enter in queue :")
enqueue(myQueue,element)
```

```

print("Now we are going to remove remaining people from the
queue")
while not isEmpty(myQueue):
    print("person removed from queue is ",
dequeue(myQueue))

```

### Output

```

enter person's code to enter in queue :P1
enter person's code to enter in queue :P2
person removed from the queue is :p1
number of people in the queue is :1
enter person's code to enter in queue :P3
enter person's code to enter in queue :P4
enter person's code to enter in queue :P5
Now we are going to remove remaining people from the queue
person removed from the queue is :p2
person removed from the queue is :p3
person removed from the queue is :p4
person removed from the queue is :p5
Queue is empty

```

## 4.4 INTRODUCTION TO DEQUE

Deque (pronounced as “deck”) is an arrangement in which addition and removal of element(s) can happen from any end, i.e. head/front or tail/rear. This data structure does not apply any restriction on the side from which addition/removal of elements should happen, so it can be used to implement stack or queue in the program. It is also known as Double ended queue, because it permits insertion, deletion operations from any end.

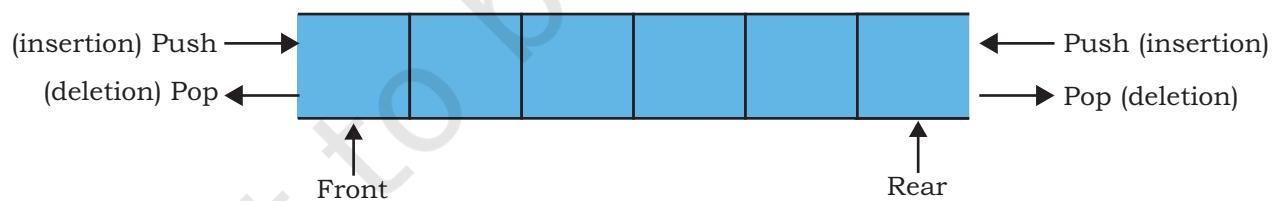


Figure 4.4: Basic deque structure displaying head and tail to implement stack or queue.

### 4.4.1 Applications of Deque

- At a train ticket purchasing counter, a normal queue of people is formed for purchasing a ticket. A person at the front purchased the ticket and left the counter. After a while they return back to the counter to ask

something. As they have already purchased a ticket, they may have the privilege to join the queue from the front.

- Vehicles in a highway toll tax booth are served following the principle of queue. There are multiple queues if there are parallel booths at the toll gate. In case all vehicles of a booth are served then vehicles from the other booth(s) are asked to form a queue in front of the vacant booth. So, vehicles at the end of those queues will leave (removed from the end from where queue was joined) current booth and join queue at the vacant booth.

#### Activity 4.3

In a deque, if insertion and deletion of elements is done from the same end, it will behave as

- 1) Queue
- 2) Stack
- 3) List
- 4) None of the above



#### Activity 4.4

In a deque, if insertion and deletion of elements is done from the opposite end, it will behave as

- 1) Queue
- 2) Stack
- 3) List
- 4) None of the above



Following are some examples where data structure deque maybe applied in computer science:

- To maintain browser history (URL), usually a stack is used, because once a tab is closed and if you press **ctrl+shift+T**, the most recently closed URL is opened first. As the number of URLs which can be stored in history is fixed, so when this list of URLs becomes large, URLs from the end of the list (i.e. which were least visited) gets deleted.
- Same happens for providing the Do and Undo option in any text editor.
- To check whether a given string is palindrome or not? Process string left to right (character wise) and insert it in deque from tail/rear like a normal queue. Once the entire string is processed (i.e. inserted in deque) we will take out (delete) a character from both the ends and match them till there is no character left or only one character left in deque. In either case, string is palindrome.

#### 4.4.2 Operations on Deque

- **INSERTFRONT:** This operation is used to insert a new element at the front of the deque.
- **INSERTREAR:** This operation is the same as a normal queue, i.e. insert a new element at the rear of the deque.
- **DELETIONFRONT:** This operation is the same as normal queue, i.e. to remove an element from the front of the deque.
- **DELETIONREAR:** This operation is used to remove one element at a time from the rear of the deque.

To perform above operations efficiently on a deque, we will need all supporting operations used in normal queue viz Is Empty, Peek, Size.

Let's understand how these operations work for checking whether a string is palindrome or not, using a deque through the following algorithm.

#### Algorithm 4.1

*Step 1:* Start traversing string (madam) from left side, a character at a time.

*Step 2:* Insert the character in deque as normal queue using INSERTREAR.

*Step 3:* Repeat Step 1 and Step 2 for all characters of string (madam)

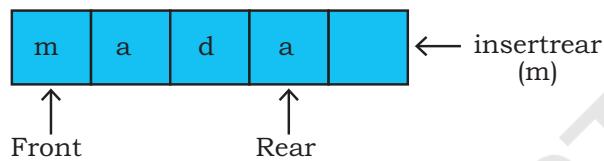


Figure 4.5: Status of Deque after 4th iteration

*Step 4:* Remove one character from the front and one character from the rear end of deque using DELETIONFRONT and DELETIONREAR we can do it.

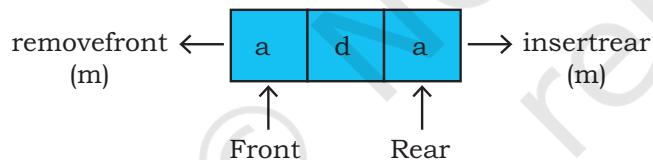


Figure 4.6: Status of Deque after removing one character from both the ends.

*Step 5:* Match these two removed characters.

*Step 6:* If they are same then

repeat Step 4 and 5 till deque is empty or left with only one character,

eventually string is Palindrome

else stop as string is not palindrome

## 4.5 IMPLEMENTATION OF DEQUE USING PYTHON

Like queue, deque is also an ordered linear list, hence we use list data type to create deque in our program. The program should have the following functions/statement(s) defined in it:

## NOTES

- A statement to create deque, with name myDeque.  
myDeque = list()
- A function insertFront(), to insert an element at the front of deque having two parameters - name of deque and element to be inserted. As the element is to be inserted in the beginning, we will use insert() with index 0 for it.

```
def insertFront(myDeque, element):  
    myDeque.insert(0,element)
```

- A function insertRear(), to insert an element at the rear of deque. It's implementation will be the same as enqueue() of normal queue requiring two parameters same as insertFront().
- A function isEmpty(), to check the presence of element(s) in deque will be the same as the function, with the same name, defined for normal queue.
- A function deletionRear(), to delete an element from the rear of the deque. It only requires the name of deque and returns the deleted element. We will use pop() without parameter(s) to delete the last element of the deque.

```
def deletionRear(myDeque):  
    if not (isEmpty()):  
        return myDeque.pop()  
        # removing data from end of list  
    else :  
        print("Deque empty")
```

- A function deletionFront(), to delete an element from the front of deque. It's implementation will be the same as dequeue() of normal queue.
- A function getFront(), to read value from the front of deque, without removing it from the queue when the queue is not empty. It accepts the name of deque as parameter and returns a copy of value.

```
def getFront(mydeque):  
    if not (isEmpty()):  
        return mydeque[0]  
    else :  
        print(" Queue empty")
```

- A function getRear(), to read value from the rear of the deque, without removing it from the deque. The

function accepts deque as argument and returns a copy of value, when the queue is not empty.

```
def getRear(mydeque):
    if not (isempty()):
        return mydeque[len(mydeque)-1]
    else :
        print(" Deque empty")
```

Let us write a main(), function to invoke various Deque functions :

#### Program 4-2 Implementation of Deque in Python

```
def insertFront(myDeque,element):
    myDeque.insert(0,element)

def getFront(myDeque):
    if not (isEmpty(myDeque)):
        return myDeque[0]
    else:
        print("Queue underflow")

def getRear(myDeque):
    if not (isEmpty(myDeque)):
        return myDeque[len(myDeque)-1]
    else:
        print ("Queue underflow")

def insertRear(myDeque,element):
    myDeque.append(element)

def isEmpty(myDeque):
    if len(myDeque) == 0:
        return True
    else:
        return False

def deletionRear(myDeque):
    if not isEmpty(myDeque):
        return myDeque.pop()
    else:
        print("Queue underflow")

def deletionFront(myDeque):
    if isEmpty(myDeque):
```

```

        print("Queue underflow")
    else:
        return myDeque.pop(0)

def main():
    dQu = list()
    choice = int(input('enter 1 to use as normal queue 2 otherwise
 : '))
    if choice == 1:
        element = input("data for insertion at rear ")
        insertRear(dQu,element)
        element = getFront(dQu)
        print("data at the beginning of queue is ", element)
        element = input("data for insertion at front ")
        insertRear(dQu,element)
        print('data removed from front of queue is ', deletionFront(dQu))
        print('data removed from front of queue is ', deletionFront(dQu))

```

### Output

```

enter 1 to use as normal queue 2 otherwise : 1
data for insertion at rear      23
data at the beginning of queue is 23
data for insertion at rear      45
data removed from front of queue is 23
data removed from front of queue is 45
Queue underflow
data removed from front of queue is None

enter 1 to use as normal queue 2 otherwise : 2
data for insertion at front     34
data at the end of queue is     34
data for insertion at front     56
data removed from rear of queue is 34
data removed from rear of queue is 56
Queue underflow
data removed from rear of queue is None

```

### SUMMARY

- Queue is an ordered linear data structure, following FIFO strategy.
- Front and Rear are used to indicate beginning and end of queue.
- In Python, the use of predefined methods takes care of Front and Rear.

- Insertion in a queue happens at the rear end. Deletion happens at the front.
- Insertion operation is known as enqueue and deletion operation is known as dequeue.
- To support enqueue and dequeue operations, isEmpty, isfull and peek operations are used
- Deque is a version of queue, which allows insertion and deletion at both ends.
- A deque can support both stack and queue operations.
- Other operations supported by deque are insertfront, insertrear, deletefront, deleterear, getfront, getrear, isempty and isfull.



## EXERCISE

1. Fill in the blank
  - a) \_\_\_\_\_ is a linear list of elements in which insertion and deletion takes place from different ends.
  - b) Operations on a queue are performed in \_\_\_\_\_ order.
  - c) Insertion operation in a queue is called \_\_\_\_\_ and deletion operation in a queue is called \_\_\_\_\_.
  - d) Deletion of elements is performed from \_\_\_\_\_ end of the queue.
  - e) Elements 'A', 'S', 'D' and 'F' are present in the queue, and they are deleted one at a time, \_\_\_\_\_ is the sequence of element received.
  - f) \_\_\_\_\_ is a data structure where elements can be added or removed at either end, but not in the middle.
  - g) A deque contains 'z', 'x', 'c', 'v' and 'b'. Elements received after deletion are 'z', 'b', 'v', 'x' and 'c'. \_\_\_\_\_ is the sequence of deletion operation performed on deque.
2. Compare and contrast queue with stack.
3. How does FIFO describe queue?

## NOTES

4. Write a menu driven python program using queue, to implement movement of shuttlecock in it's box.
5. How is queue data type different from deque data type?
6. Show the status of queue after each operation  
`enqueue(34)`  
`enqueue(54)`  
`dequeue()`  
`enqueue(12)`  
`dequeue()`  
`enqueue(61)`  
`peek()`  
`dequeue()`  
`dequeue()`  
`dequeue()`  
`dequeue()`  
`enqueue(1)`
7. Show the status of deque after each operation  
`peek()`  
`insertFront(12)`  
`insertRear(67)`  
`deletionFront()`  
`insertRear(43)`  
`deletionRear()`  
`deletionFront()`  
`deletionRear()`
8. Write a python program to check whether the given string is palindrome or not, using deque. (Hint : refer to algorithm 4.1)

Chapter

5

# Sorting



12130CH05

## In this Chapter

- » *Introduction*
- » *Bubble Sort*
- » *Selection Sort*
- » *Insertion Sort*
- » *Time Complexity of Algorithms*

*“Every one of today's smartphones has thousands of times more processing power than the computers that guided astronauts to the moon.”*

— Peter Thiel

## 5.1 INTRODUCTION

Sorting is the process of ordering or arranging a given collection of elements in some particular order. We can sort a collection of numbers in ascending (increasing) or descending (decreasing) order. If the collection is of strings, we can sort it in an alphabetical order (a-z or z-a) or according to the length of the string. For example, words in a dictionary are sorted in alphabetical order; seats in an examination hall are ordered according to candidates' roll number. We can also sort a list of students based on their height or weight.

Imagine finding the meaning of a word from a dictionary that is not ordered. We will have to search for the word on each page till we find the word, which will be very tedious. That is why dictionaries have the words in alphabetical order and it ease the process of searching.

### **Think and Reflect**

Can you identify other examples where sorting plays an important role in computers?



Sorting a large number of items can take a substantial amount of time. However, this extra time (called overhead) is worth when compared to the amount of time needed to find an element from an unsorted list. Sorting is an important area of study in computer science, and many sorting algorithms have been developed and analysed from their performance point of view. In this chapter, we will learn about three sorting methods and implement them using Python. Bubble sort is discussed in section 5.2, followed by discussion on selection sort and insertion sort in section 5.3 and 5.4, respectively.

## **5.2 BUBBLE SORT**

The first sorting technique we are going to understand is Bubble sort. It sorts a given list of elements by repeatedly comparing the adjacent elements and swapping them if they are unordered. Swapping two elements means changing their positions with each other. In algorithm, every iteration through each element of a list is called a pass. For a list with  $n$  elements, the bubble sort makes a total of  $n - 1$  passes to sort the list. In each pass, the required pairs of adjacent elements of the list will be compared. In order to arrange elements in ascending order, the largest element is identified after each pass and placed at the correct position in the list. This can be considered as the largest element being ‘bubbled up’. Hence the name Bubble sort. This sorted element is not considered in the remaining passes and thus the list of elements gets reduced in successive passes.

### **Think and Reflect**

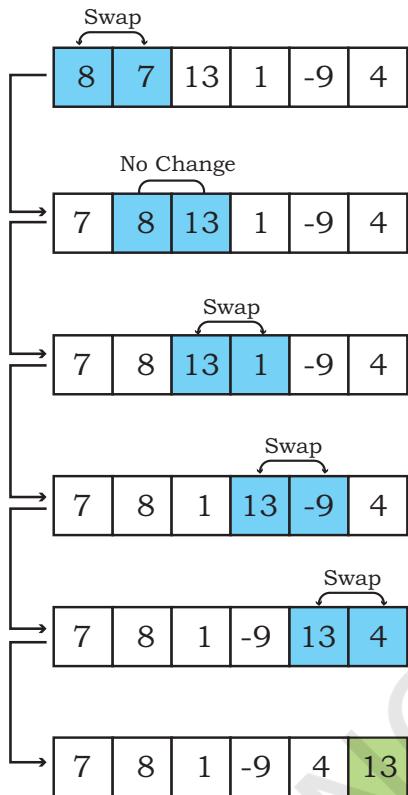
In Figure 5.1, we can see that the list got sorted in the 4th pass itself. Still the bubble sort technique made a redundant 5th pass which did not result in any swap. If there is no swapping in any pass, it means the list is already sorted, hence the sorting operation needs to be stopped. Can you think of making any improvement in the Algorithm 5.1 so that it stops when the list becomes sorted?



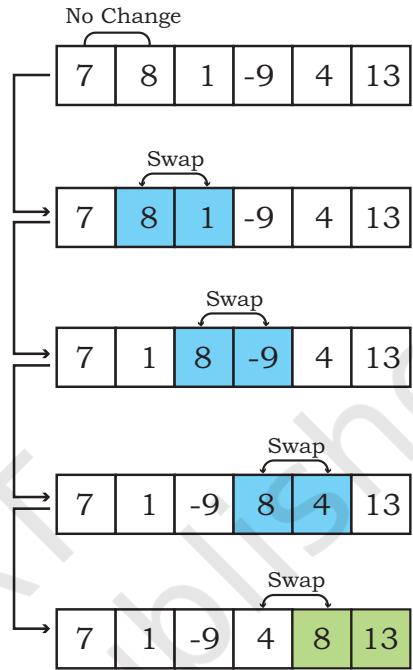
Figure 5.1 demonstrates the working of the bubble sort method to arrange a list in ascending order. Let us consider a list having 6 elements as `numList = [8, 7, 13, 1, -9, 4]`. In the figure, elements being compared are highlighted with blue colour and sorted elements are highlighted with green colour. To begin sorting, the element at index 0 is compared with the element at index 1. If the first element is bigger, it is swapped with the second. Else, no change is done. Next, the element at index 1 is compared with the element at index 2. This continues till the end of the list is reached. After the first pass, the largest element will reach the end of the list as shown in Figure 5.1 with green colour.

|         |   |   |    |   |    |   |
|---------|---|---|----|---|----|---|
| numList | 8 | 7 | 13 | 1 | -9 | 4 |
| Index   | 0 | 1 | 2  | 3 | 4  | 5 |

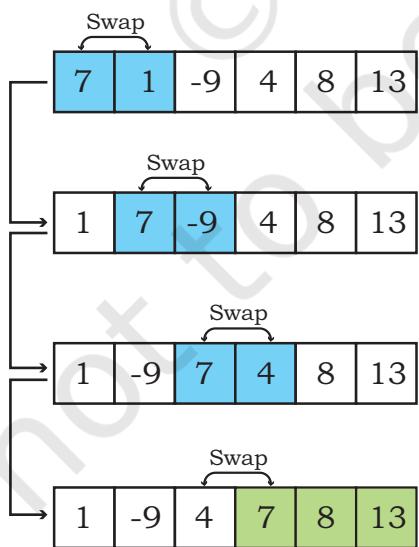
Comparison in Pass 1



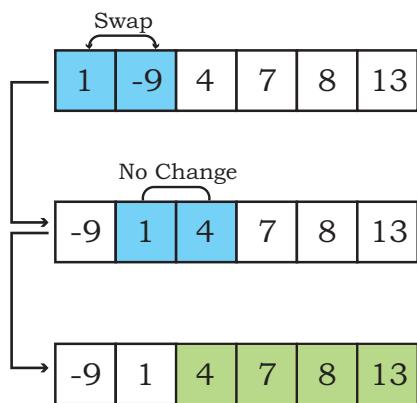
Comparison in Pass 2



Comparison in Pass 3



Comparison in Pass 4



## Comparison in Pass 5

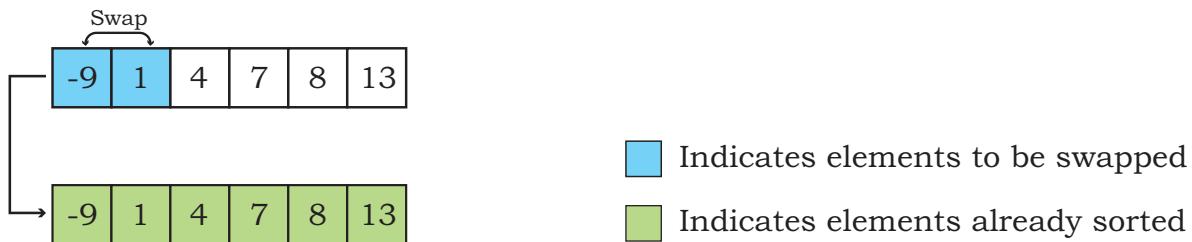


Figure 5.1: Comparisons done in different passes of Bubble sort

Algorithm 5.1 shows the steps followed for the bubble sort that takes numList as a list of n elements, and sorts the list in ascending order:

### Activity 5.1

Algorithm 5.1 sorts a list in ascending order. Write a bubble sort algorithm to sort a list in descending order?



### Algorithm 5.1: Bubble Sort

BUBBLESORT( numList, n)

Step 1: SET i = 0

Step 2: WHILE i < n REPEAT STEPS 3 to 8

Step 3: SET j = 0

Step 4: WHILE j < n-i-1, REPEAT STEPS 5 to 7

Step 5: IF numList[j] > numList[j+1] THEN

Step 6: swap(numList[j],numList[j+1])

Step 7: SET j=j+1

Step 8: SET i=i+1

### Program 5-1 Implementation of bubble sort using Python.

```
def bubble_Sort(list1):
    n = len(list1)
    for i in range(n):      # Number of passes
        for j in range(0, n-i-1):
            # size -i-1 because last i elements are already sorted
            # in previous passes
            if list1[j] > list1[j+1] :
                # Swap element at jth position with (j+1)th position
                list1[j], list1[j+1] = list1[j+1], list1[j]
numList = [8, 7, 13, 1, -9, 4]
bubble_Sort(numList)
```

```
print ("The sorted list is :")
for i in range(len(numList)):
    print (numList[i], end=" ")
```

Output:

```
The sorted list is :
-9 1 4 7 8 13
```

### 5.3 SELECTION SORT

Selection sort is another sorting technique. To sort a list having  $n$  elements, the selection sort makes  $(n-1)$  number of passes through the list. The list is considered to be divided into two lists -- the left list containing the sorted elements, and the right list containing the unsorted elements. Initially, the left list is empty, and the right list contains all the elements.

For arranging elements in ascending order, in the first pass, all the elements in the unsorted list are traversed to find the smallest element. The smallest element is then swapped with the leftmost element of the unsorted list. This element occupies the first position in the sorted list, and it is not considered in further passes. In the second pass, the next smallest element is selected from the remaining elements in the unsorted list and swapped with the leftmost element of the unsorted list. This element occupies the second position in the sorted list, and the unsorted list reduces by one element for the third pass.

This process continues until  $n-1$  smallest elements are found and moved to their respective places. The  $n$ th element is the last, and it is already in place. Figure 5.2 demonstrates the working of selection sort method to arrange a list in ascending order. In this Figure, elements being compared are shown using arrows and the smaller element in a comparison is highlighted with blue colour. The sorted elements are highlighted—

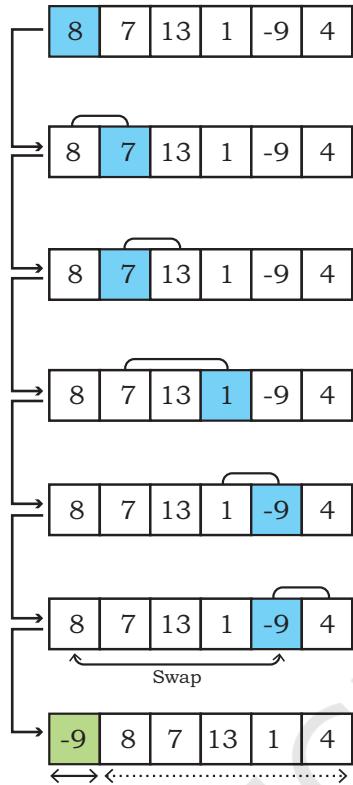
#### Activity 5.2

Apply bubble sort technique to sort a list of elements numList2 = [8, 7, 6, 5, 4]. Show the positions of elements in the list after each pass. In which pass the last swap happens?

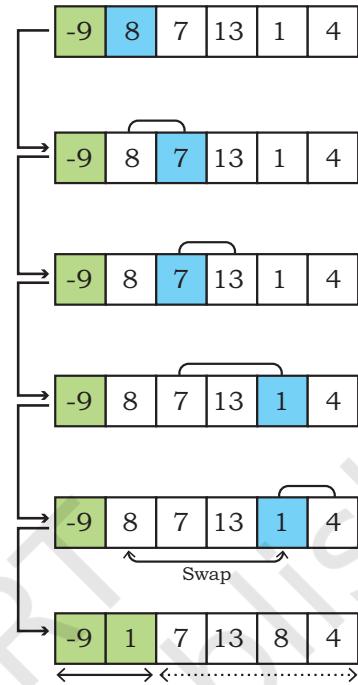


|         |   |   |    |   |    |   |
|---------|---|---|----|---|----|---|
| numList | 8 | 7 | 13 | 1 | -9 | 4 |
| Index   | 0 | 1 | 2  | 3 | 4  | 5 |
|         |   |   |    |   |    |   |

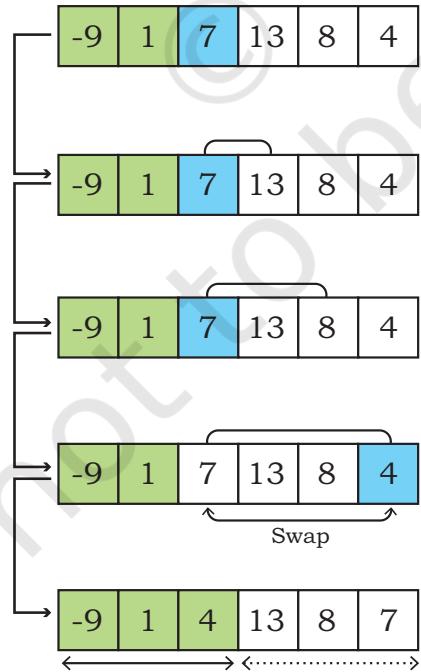
Comparison in Pass 1



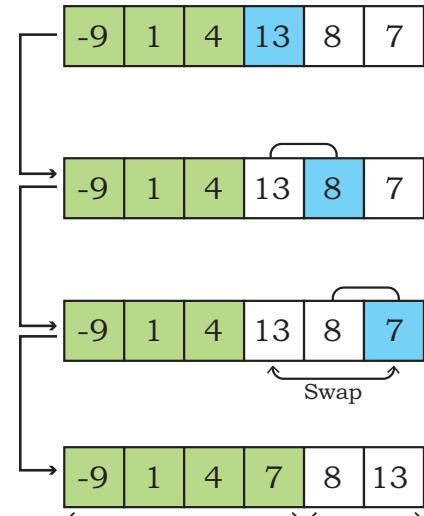
Comparison in Pass 2



Comparison in Pass 3



Comparison in Pass 4



## Comparison in Pass 5

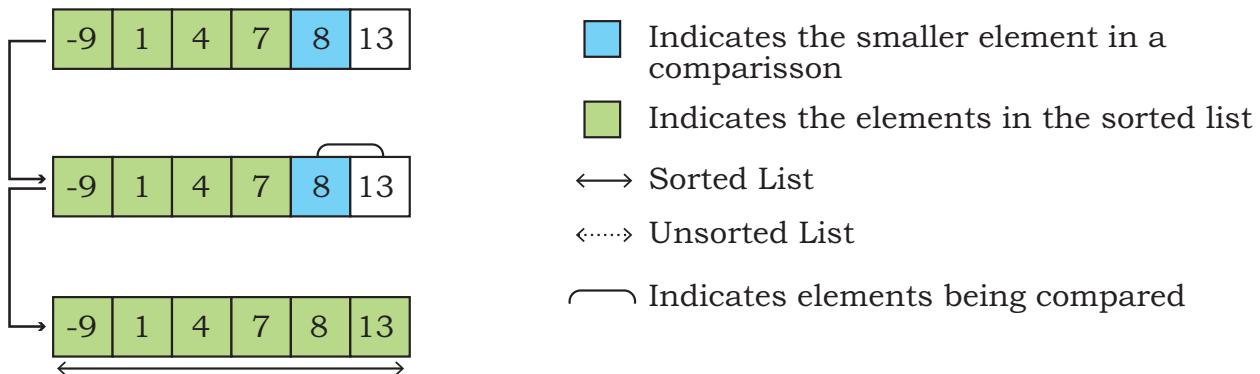


Figure 5.2: Comparisons done in different passes of Selection sort

The following is an algorithm for the selection sort that takes numList as a list consisting of n elements, and sorts the list in ascending order:

Algorithm 5.2 shows the steps followed for the selection sort that takes numList as a list of n elements, and sorts the list in ascending order:

### Algorithm 5.2: Selection Sort

**SELECTIONSORT( numList, n)**

*Step 1:* SET i=0

*Step 2:* WHILE i < n REPEAT STEPS 3 to 11

*Step 3:*      SET min = i, flag = 0

*Step 4:*      SET j= i+1

*Step 5:*      WHILE j < n, REPEAT STEPS 6 to 10

*Step 6:*      IF numList[j] < numList[min] THEN

*Step 7:*              min = j

*Step 8:*              flag = 1

*Step 9:*      IF flag = 1 THEN

*Step 10:*     swap(numList[i],numList[min])

*Step 11:* SET i=i+1

### Activity 5.3

Consider a list of 10 elements:  
randList = [7,11,3,10,17,23,1,4,21,5]. Determine the partially sorted list after four complete passes of selection sort.



### Program 5-2 Implementation of selection sort using Python.

```
def selection_Sort(list2):
    flag = 0          #to decide when to swap
    n=len(list2)
    for i in range(n): # Traverse through all list elements
        min = i
        for j in range(i + 1, len(list2)): #the left elements
            #are already sorted in previous passes
            if list2[j] < list2[min]: # element at j is smaller
                #than the current min element
                min = j
                flag = 1
        if flag == 1 : # next smallest element is found
            list2[min], list2[i] = list2[i], list2[min]

numList = [8, 7, 13, 1, -9, 4]
selection_Sort(numList)
print ("The sorted list is :")
for i in range(len(numList)):
    print (numList[i], end=" ")
```

#### Output:

```
The sorted list is :
-9 1 4 7 8 13
```

## 5.4 INSERTION SORT

Insertion sort is another sorting algorithm that can arrange elements of a given list in ascending or descending order. Like Selection sort, in Insertion sort also, the list is divided into two parts - one of sorted elements and another of unsorted elements. Each element in the unsorted list is considered one by one and is inserted into the sorted list at its appropriate position. In each pass, the sorted list is traversed from the backward direction to find the position where the unsorted element could be inserted. Hence the sorting method is called insertion sort.

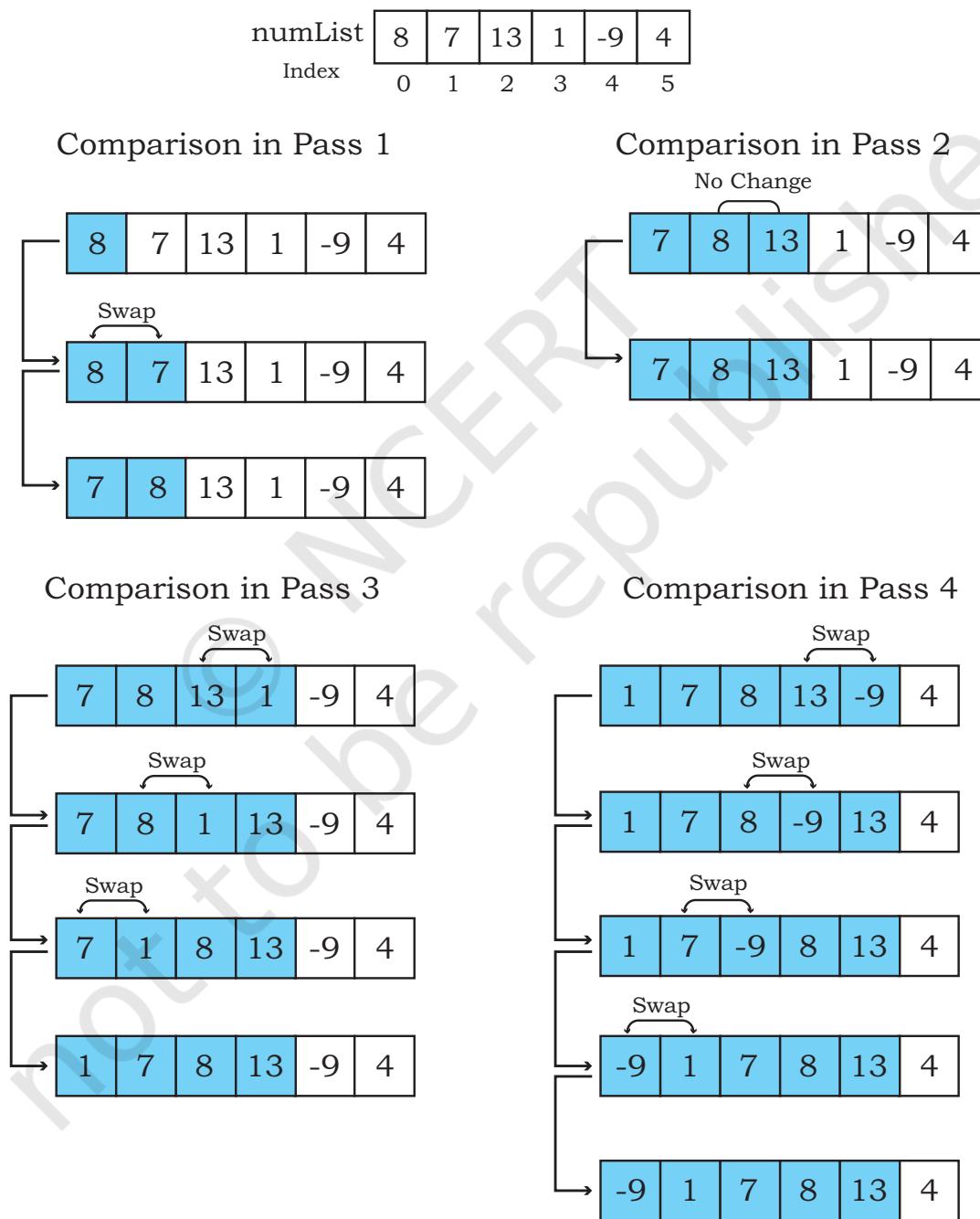
In pass 1, the unsorted list has  $n-1$  elements and the sorted list has a single element (say element s). The first element of the unsorted list (say element e) is compared with the element s of sorted list. If element e is smaller than element s, then element s is shifted to the right making space for inserting element e. This shifting will now make sorted list of size 2 and unsorted list of size  $n-2$ .

In pass 2, the first element (say element e) of unsorted list will be compared with each element of sorted list

starting from the backward direction till the appropriate position for insertion is found. The elements of sorted list will be shifted towards right making space for the element e where it could be inserted.

This continues till all the elements in unsorted lists are inserted at appropriate positions in the sorted list. This results into a sorted list in which elements are arranged in ascending order.

Figure 5.3 demonstrates the working of the insertion sort to arrange a list in ascending order.



## Comparison in Pass 5

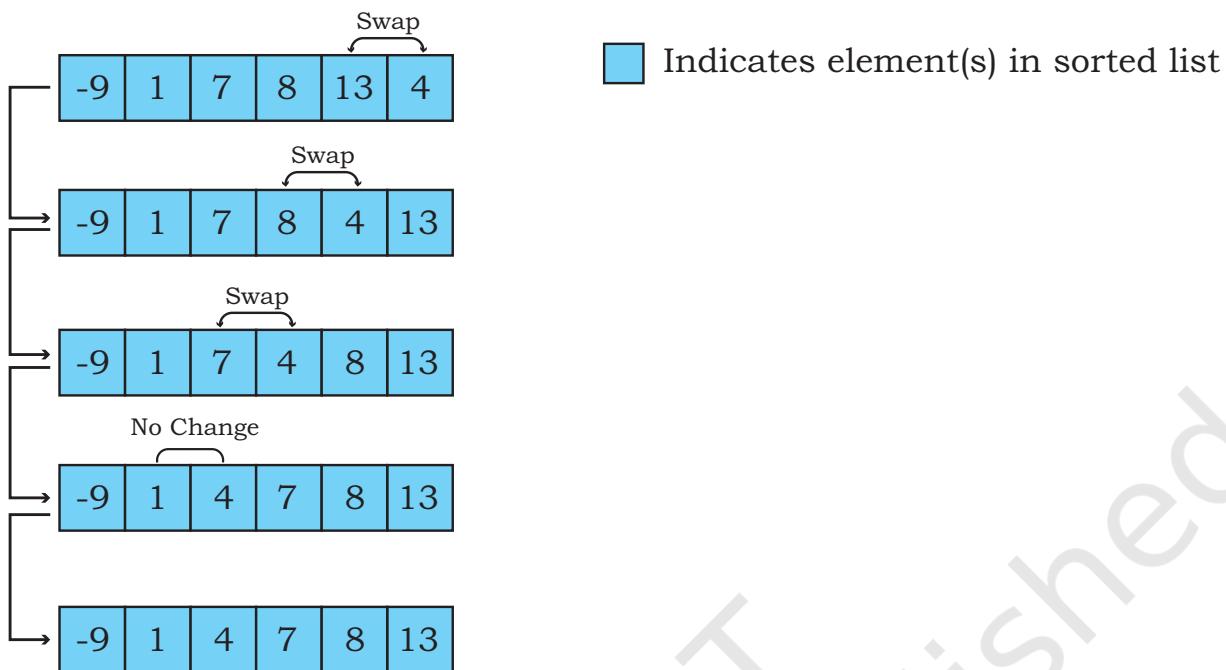


Figure 5.3: Comparisons done in different passes of Insertion sort

Let us consider that numList is a list consisting of n elements. Algorithm 5.3 sorts the list numList in ascending order using insertion sort technique.

### Algorithm 5.3: Insertion Sort

INSERTIONSORT( numList, n)

*Step 1:* SET i=1

*Step 2:* WHILE i< n REPEAT STEPS 3 to 9

*Step 3:* temp = numList[i]

*Step 4:* SET j = i-1

*Step 5:* WHILE j> = 0 and numList[j]>temp,REPEAT STEPS 6 to 7

*Step 6:* numList[j+1] = numList[j]

*Step 7:* SET j=j-1

*Step 8:* numList[j+1] = temp #insert temp at position j

*Step 9:* set i=i+1

### Activity 5.4

Consider a list of 10 elements:

Array =

[7,11,3,10,17,23,1,4,21,5]

Determine the partially sorted list after three complete passes of insertion sort.



Program 5-3 Implementation of insertion sort using Python.

```
def insertion_Sort(list3):
    n= len(list3)
    for i in range(n): # Traverse through all elements
        temp = list3[i]
        j = i-1
        while j >=0 and temp< list3[j] :
            list3[j+1] = list3[j]
            j = j-1
        list3[j+1] = temp

numList = [8, 7, 13, 1, -9, 4]
insertion_Sort(numList)
print ("The sorted list is :")
for i in range(len(numList)):
    print (numList[i], end=" ")
```

Output:

```
The sorted list is :
-9 1 4 7 8 13
```

## 5.5 TIME COMPLEXITY OF ALGORITHMS

We have studied that there can be more than one approach to solve a problem using a computer. In Class XI, we compared four different algorithms to check whether a given number is prime or not. For the same problem, one algorithm may require more processing time than the other. The amount of time an algorithm takes to process a given data can be called its time complexity.

For a small set of data elements shown in examples of this chapter so far, the time and memory required by different algorithms do not differ significantly. However, in the real world, sorting algorithms are required to work on huge amounts of data. In such cases, the total time utilisation becomes significant, and therefore it is important to consider the time complexity of an algorithm before being used in a real world data set.

Computer scientists proposing different techniques for sorting are always interested to find out their time complexity. The aim is to find out how a sorting algorithm behaves if the order of input elements changes or if the number of elements in the list increases or decreases. Such comparisons help to decide which algorithm is more suitable for which kind of data and application.

## NOTES

Calculating the complexity of different algorithms involves mathematical calculations and detailed analysis, and it is beyond the scope of this textbook to discuss them in detail. However, we will discuss some basics of complexity to get some ideas. The following tips will guide us in estimating the time complexity of an algorithm.

- Any algorithm that does not have any loop will have time complexity as 1 since the number of instructions to be executed will be constant, irrespective of the data size. Such algorithms are known as Constant time algorithms.
- Any algorithm that has a loop (usually 1 to n) will have the time complexity as  $n$  because the loop will execute the statement inside its body  $n$  number of times. Such algorithms are known as Linear time algorithms.
- A loop within a loop (nested loop) will have the time complexity as  $n^2$ . Such algorithms are known as Quadratic time algorithms.
- If there is a nested loop and also a single loop, the time complexity will be estimated on the basis of the nested loop only.

Now, look at the Python programs of the three sorting techniques discussed in this chapter, you will notice that in each of the three programs, there is a nested loop, i.e., one inside another. So according to the above rules, all the sorting algorithms namely, bubble sort, selection sort and insertion sort have a time complexity of  $n^2$ .

### SUMMARY

- The process of placing or rearranging a collection of elements into a particular order is known as sorting.
- Bubble sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements in case they are unordered in  $n-1$  passes.
- In Selection Sort, the smallest element is selected from the unsorted array and swapped with the

leftmost element, and that element becomes a part of the sorted array. The process continues for the next element in the unsorted array till the list is sorted.

- Insertion Sort places the element of a list at its suitable place in each pass. It is similar to the placing of cards at its right position while playing cards.
- Complexity analysis is performed to explain how an algorithm will perform when the input grows larger.



## EXERCISE

1. Consider a list of 10 elements:

`numList =[7,11,3,10,17,23,1,4,21,5].`

Display the partially sorted list after three complete passes of Bubble sort.

2. Identify the number of swaps required for sorting the following list using selection sort and bubble sort and identify which is the better sorting technique with respect to the number of comparisons.

List 1 : 

|    |    |    |   |
|----|----|----|---|
| 63 | 42 | 21 | 9 |
|----|----|----|---|

3. Consider the following lists:

List 1: 

|   |   |   |   |    |
|---|---|---|---|----|
| 2 | 3 | 5 | 7 | 11 |
|---|---|---|---|----|

List 2: 

|    |   |   |   |   |
|----|---|---|---|---|
| 11 | 7 | 5 | 3 | 2 |
|----|---|---|---|---|

If the lists are sorted using Insertion sort then which of the lists List1 or List 2 will make the minimum number of comparisons? Justify using diagrammatic representation.

4. Write a program using user defined functions that accepts a List of numbers as an argument and finds its median. (Hint : Use bubble sort to sort the accepted list. If there are odd number of terms, the median is the center term. If there are even number of terms, add the two middle terms and divide by 2 get median)

## NOTES

5. All the branches of XYZ school conducted an aptitude test for all the students in the age group 14 - 16. There were a total of n students. The marks of n students are stored in a list. Write a program using a user defined function that accepts a list of marks as an argument and calculates the ' $x^{\text{th}}$ ' percentile (where x is any number between 0 and 100). You are required to perform the following steps to be able to calculate the ' $x^{\text{th}}$ ' percentile.

**Note:** Percentile is a measure of relative performance i.e. It is calculated based on a candidate's performance with respect to others. For example : If a candidate's score is in the 90<sup>th</sup> percentile, that means she/he scored better than 90% of people who took the test.

Steps to calculate the  $x^{\text{th}}$  percentile:

- I. Order all the values in the data set from smallest to largest using Selection Sort. In general any of the sorting methods can be used.

- II. Calculate index by multiplying  $x$  percent by the total number of values, n.

*For example:* to find 90<sup>th</sup> percentile for 120 students:

$$0.90 \times 120 = 108$$

- III. Ensure that the index is a whole number by using `math.round()`

- VI. Display the value at the index obtained in Step 3.

The corresponding value in the list is the  $x^{\text{th}}$  percentile.

6. During admission in a course, the names of the students are inserted in ascending order. Thus, performing the sorting operation at the time of inserting elements in a list. Identify the type of sorting technique being used and write a program using a user defined function that is invoked every time a name is input and stores the name in ascending order of names in the list.

Chapter

6

# Searching



## In this Chapter

- » *Introduction*
- » *Linear Search*
- » *Binary Search*
- » *Search by Hashing*

*“Even though most people won't be directly involved with programming, everyone is affected by computers, so an educated person should have a good understanding of how computer hardware, software, and networks operate.”*

— Brian Kernighan

### 6.1 INTRODUCTION

We store many things in our home and find them out later as and when required. Sometimes we remember the exact location of a required item. But, sometimes we do not remember the exact location and in that case we need to search for the required item. A computer also stores lots of data to be retrieved later as and when demanded by a user or a program.

Searching means locating a particular element in a collection of elements. Search result determines whether that particular element is present in the collection or not. If it is present, we can also find out the position of that element in the given collection. Searching is an important technique in computer science. In order to design algorithms, programmers need to understand the different ways in which a collection of data can be searched for retrieval.

## 6.2 LINEAR SEARCH

Linear search is the most fundamental and the simplest search method. It is an exhaustive searching technique where every element of a given list is compared with the item to be searched (usually referred to as ‘key’). So, each element in the list is compared one by one with the key. This process continues until an element matching the key is found and we declare that the search is successful. If no element matches the key and we have traversed the entire list, we declare the search is unsuccessful i.e., the key is not present in the list. This item by item comparison is done in the order, in which the elements are present in the list, beginning at the first element of the list and moving towards the last. Thus, it is also called sequential search or serial search. This technique is useful for collection of items that are small in size and are unordered.

Given a list numList of n elements and key value K, Algorithm 6.1 uses a linear search algorithm to find the position of the key K in numList.

### Algorithm 6.1 : Linear Search

LinearSearch(numList, key, n)

Step 1: SET index = 0

Step 2: WHILE index < n, REPEAT Step 3

Step 3: IF numlist[index]= key THEN

        PRINT “Element found at position”, index+1

        STOP

        ELSE

            index = index+1

Step 4: PRINT “Search unsuccessful”

**Example 6.1** Assume that the numList has seven elements [8, -4, 7, 17, 0, 2, 19] so, n = 7. We need to search for the key, say 17 in numList. Table 6.1 shows the elements in the given list along with their index values.

**Table 6.1 Elements in numList alongwith their index value**

| Index in numList | 0 | 1  | 2 | 3  | 4 | 5 | 6  |
|------------------|---|----|---|----|---|---|----|
| Value            | 8 | -4 | 7 | 17 | 0 | 2 | 19 |

The step-by-step process of linear search using Algorithm 6.1. is given in Table 6.2.

**Table 6.2 Linear search for key 17 in numList of Table 6.1**

| index | index < n   | numList[index]= key | index=index+1 |
|-------|-------------|---------------------|---------------|
| 0     | 0 < 7 ? Yes | 8 = 17? No          | 1             |
| 1     | 1 < 7 ? Yes | -4 = 17? No         | 2             |
| 2     | 2 < 7 ? Yes | 7 = 17? No          | 3             |
| 3     | 3 < 7 ? Yes | 17 = 17? Yes        |               |

Observe that after four comparisons, the algorithm found the key 17 and will display ‘Element found at position 4’.

Let us now assume another arrangement of the elements in numList as [17, 8, -4, 7, 0, 2, 19] and search for the key K=17 in numList.

**Table 6.3 Elements in numList alongwith their index value**

| Index in numList | 0  | 1 | 2  | 3 | 4 | 5 | 6  |
|------------------|----|---|----|---|---|---|----|
| Value            | 17 | 8 | -4 | 7 | 0 | 2 | 19 |

**Table 6.4 Linear search for key 17 in numList given in Table 6.3**

| index | index < n   | numList[index]= key | index=index+1 |
|-------|-------------|---------------------|---------------|
| 0     | 0 < 7 ? Yes | 17 = 17? Yes        | 1             |

From Table 6.4, it is clear that the algorithm had to make only 1 comparison to display ‘Element found at position 1’. Thus, if the key to be searched is the first element in the list, the linear search algorithm will always have to make only 1 comparison. This is the minimum amount of work that the linear search algorithm would have to do.

Let us now assume another arrangement of the elements in numList as [8, -4, 7, 0, 2, 19, 17] and search for the key K =17 in numList.

On a dry run, we can find out that the linear search algorithm has to compare each element in the list till the end to display ‘Element found at position 7’. Thus, if the key to be searched is the last element in the list, the linear search algorithm will have to make n comparisons, where n is the number of elements in the list. This is in fact the maximum amount of work the linear search algorithm would have to do.

### Activity 6.2

In the list : L = [7,-1, 11,32,17,19,23,29,31, 37,43]

Determine the number of comparisons linear search takes to search for key = 43.



Let us now assume another case, where the key being searched is not present in the list. For example, we are searching for key = 10 in the numList.

In this case also, the algorithm has to compare each element in the list till the end to display 'Element is not found in the list'. Thus, if the key is not present in the list, the linear search algorithm will have to make n comparisons. This again is a case where maximum work is done. Let us now understand the program of a Linear Search. It takes a list of elements and the key to be searched as input and returns either the position of the element in the list or display that the key is not present in the list.

### Program 6-1 Linear Search

```
def linearSearch(list, key):      #function to perform the search
    for index in range(0,len(list)):
        if list[index] == key:      #key is present
            return index+1         #position of key in list
    return None #key is not in list
#end of function

list1 = [] #Create an empty list
maximum = int(input("How many elements in your list? "))
print("Enter each element and press enter: ")
for i in range(0,maximum):
    n = int(input())
    list1.append(n) #append elements to the list
print("The List contents are:", list1)

key = int(input("Enter the number to be searched: "))
position = linearSearch(list1, key)
if position is None:
    print("Number",key,"is not present in the list")
else:
    print("Number",key,"is present at position",position)
```

### Output

```
How many elements in your list? 4
Enter each element and press enter:
12
23
3
-45
The List contents are: [12, 23, 3, -45]
Enter the number to be searched:23
Number 23 is present at position 2
```

### 6.3 BINARY SEARCH

Consider a scenario where we have to find the meaning of the word Zoology in an English dictionary. Where do we search it in the dictionary?

1. in the first half?
2. around the middle?
3. in the second half?

It is certainly more prudent to look for the word in the second half of the dictionary as the word starts with the alphabet 'Z'. On the other hand, if we were to find the meaning of the word Biology, we would have searched in the first half of the dictionary.

We were able to decide where to search in the dictionary because we are aware of the fact that all words in an English dictionary are placed in alphabetical order. Taking advantage of this, we could avoid unnecessary comparison through each word beginning from the first word of the dictionary and moving towards the end till we found the desired word. However, if the words in the dictionary were not alphabetically arranged, we would have to do linear search to find the meaning of a word.

The binary search is a search technique that makes use of the ordering of elements in the list to quickly search a key. For numeric values, the elements in the list may be arranged either in ascending or descending order of their key values. For textual data, it may be arranged alphabetically starting from a to z or from z to a.

In binary search, the key to be searched is compared with the element in the middle of a sorted list. This could result in either of the three possibilities:

- i) the element at the middle position itself matches the key or
- ii) the element at the middle position is greater than the key or
- iii) the element at the middle position is smaller than the key

If the element at the middle position matches the key, we declare the search successful and the searching process ends.

We are using the term iteration and not comparison in binary search,

because after every unsuccessful comparison, we change the search area redefining the first, mid and last position before making subsequent comparisons.

If the middle element is greater than the key it means that if the key is present in the list, it must surely be in the first half. We can thus straight away ignore the second half of the list and repeat the searching process only in the first half.

If the middle element is less than the key, it means if the key is present in the list, it must be in the second half. We can thus straight away ignore the first half of the list and repeat the searching process only in the second half. This splitting and reduction in list size continued till the key is either found or the remaining list consists of only one item. If that item is not the key, then the search is unsuccessful as the key is not in the list.

Thus, it is evident that unlike linear search of elements one-by-one, we can search more efficiently using binary search provided the list from which we want to search is arranged in some order. That is, the list needs to be sorted.

If the list to be searched contains an even number of elements, the mid value is calculated using the floor division (`//`) operator. If there are 10 elements in the list, then the middle position (`mid`) =  $10//2 = 5$ . Therefore, the sixth element in the list is considered the middle element as we know that the first element in list has index value 0. If required, the list is further divided into two parts where the first half contains 5 elements and the second half contains 4 elements.

It is interesting to note that the intermediate comparisons which do not find the key still give us information about the part of the list where the key may be found! They reveal whether the key is before or after the current middle position in the list, and we use this information to narrow down or reduce our search area. Thus, each unsuccessful comparison reduces the number of elements remaining to be searched by half, hence the name binary search. Let us now discuss the algorithm of binary search.

Given a list `numList` of  $n$  elements and key value  $K$ , Algorithm 6.2 shows steps for finding position of the key  $K$  in the `numList` using binary search algorithm.

#### Activity 6.3

Consider the numList [17, 8, -4, 7, 0, 2, 19]. Sort it using the `sort()` function of Python's Lists. Now apply binary search to search for the key 7. Determine the number of iterations required.



#### Activity 6.4

Consider a list [-4, 0, 2, 7, 8, 17, 19]. Apply binary search to find element -4. Determine the number of key comparisons required.



### Algorithm 6.2: Binary Search

```

BinarySearch(numList, key)
Step 1: SET first = 0, last = n-1
Step 2: Calculate mid = (first+last) // 2
Step 3: WHILE first <= last REPEAT Step 4
Step 4:   IF numList[mid] = key
           PRINT "Element found at position",
           " mid+1
           STOP
       ELSE
           IF numList[mid] > key, THEN last
           = mid-1
           ELSE first = mid + 1
Step 5: PRINT "Search unsuccessful"

```



The binary search algorithm does not change the list. Rather, after every pass of the algorithm, the search area gets reduced by half. That is, only the index of the element to be compared with the key changes in each iteration.



**Example 6.2** Consider a sorted list comprising of 15 elements:

numList = [2, 3, 5, 7, 10, 11, 12, 17, 19, 23, 29, 31, 37, 41, 43]

We need to search for the key, say 17 in numList. The first, middle and last element identified in numList alongwith their index values are shown in Table 6.5.

**Table 6.5 Elements in sorted numList alongwith their index value**

|                  | first |   |   |   |    |    |    | mid |    |    |    |    |    |    | last |
|------------------|-------|---|---|---|----|----|----|-----|----|----|----|----|----|----|------|
| Index in numList | 0     | 1 | 2 | 3 | 4  | 5  | 6  | 7   | 8  | 9  | 10 | 11 | 12 | 13 | 14   |
| Value            | 2     | 3 | 5 | 7 | 10 | 11 | 12 | 17  | 19 | 23 | 29 | 31 | 37 | 41 | 43   |

**Table 6.6 Working of binary search using steps given in Algorithm 6.2.**

|             | first | last | mid           | numList <sub>[mid]</sub> == K | key < L <sub>mid</sub> ?            | first <= last    |
|-------------|-------|------|---------------|-------------------------------|-------------------------------------|------------------|
| At Start    | 0     | 14   | (0+14) // 2=7 | Not known                     | Not known                           | 0 <= 14?<br>True |
| Iteration 1 | 0     | 14   | 7             | 17 = 17?<br>Yes               | Key is found. The search terminates |                  |

Note that the algorithm had to make only 1 iteration to display 'Element found at position 8'. This is because the key being searched is the middle element in the list. Thus, binary search requires only 1 iteration when the key to be searched is the middle value in the list. This

is the minimum amount of work binary search would have to do to confirm that a key is present in the list.

Now, let us search for key 2 in the list.

```
numList = [2, 3, 5, 7, 10, 11, 12, 17, 19, 23, 29, 31, 37, 41, 43]
```

### Activity 6.5

For  $L = [2, 3, 5, 7, 10, 11, 12, 17, 19, 23, 29, 31, 37, 41, 43]$ . Fill up the Table 6.8 for the given key values 2, 43, 17 and 9. What do you infer from Table 6.8 regarding performance of both the algorithms in different cases?



In the first iteration, we have the mid value as 17. As 2 is smaller than the mid value (17), we have to search for the first half of the list in the second iteration. We now consider only 7 elements. As 2 is smaller than the new mid value (7), we have to search for the first half of the remaining list in the third iteration. We now consider only 3 elements. Observe that the number of elements in the numList is halved each time. It reduces from 15 elements in iteration 1 to 7 elements in iteration 2, and to 3 elements in iteration 3. In the 3rd iteration, the algorithm finds that key 2 is smaller than the new mid value (3), we have to search in the first half of the remaining list. The list now has only 1 element in the fourth iteration and on comparison, it is found that the element is the same as key. Hence, the search terminates successfully and returns the position of key. Steps followed for binary search are given in Table 6.7.

**Table 6.7 Searching key = 2 in the numList using binary search**

|                    | first | last | mid          | $\text{numList}_{\text{mid}} == K$ | $K < \text{numList}_{\text{mid}}$                          | first $\leq$ last    |
|--------------------|-------|------|--------------|------------------------------------|------------------------------------------------------------|----------------------|
| <b>At Start</b>    | 0     | 14   | $(0+14)/2=7$ | Not known                          | Not known                                                  | True                 |
| <b>Iteration 1</b> | 0     | 14   | $(0+14)/2=7$ | $17 = 2?$<br>No                    | $2 < 17?$<br>True                                          | $0 \leq 14?$<br>True |
| <b>Iteration 2</b> | 0     | 6    | $(0+6)/2=3$  | $7 = 2?$<br>No                     | $2 < 7?$<br>True                                           | $0 \leq 6?$<br>True  |
| <b>Iteration 3</b> | 0     | 2    | $(0+2)/2=1$  | $3 = 2?$<br>No                     | $2 < 3?$<br>True                                           | $0 \leq 2?$<br>True  |
| <b>Iteration 4</b> | 0     | 0    | $(0+0)/2=0$  | $2 = 2?$<br>Yes                    | Key found, search Terminates, return position as (mid+1)=1 |                      |

As we can see, the binary search algorithm had to make 4 iterations to narrow down the list to a single element and decide that the search key is the first element of list. This is clearly the maximum work required to find a key in the given list.

## Program 6-2 Binary Search

```
def binarySearch(list, key):
    first = 0
    last = len(list) - 1
    while(first <= last):
        mid = (first + last)//2
        if list[mid] == key:
            return mid
        elif key > list[mid]:
            first = mid + 1
        elif key < list[mid]:
            last = mid - 1
    return -1

list1 = [] #Create an empty list
print ("Create a list by entering elements in ascending order")
print ("press enter after each element, press -999 to stop")
num = int(input())
while num!==-999:
    list1.append(num)
    num = int(input())
n = int(input("Enter the key to be searched: "))
pos = binarySearch(list1,n)
if(pos != -1):
    print( n,"is found at position", pos+1)
else:
    print (n,"is not found in the list ")
```

### Output

```
Create a list by entering elements in ascending order
press enter after each element, press -999 to stop
1
3
4
5
-999
Enter the number to be searched: 4
4 is found at position 3

Second run of the program with different data:
Create a list by entering elements in ascending order
press enter after each element, press -999 to stop
12
8
3
-999
Enter the number to be searched: 4
4 is not found in the list
```

### 6.3.1 Applications of Binary Search

- Binary search has numerous applications including
  - searching a dictionary or a telephone directory, finding the element with minimum value or maximum value in a sorted list, etc.
- Modified binary search techniques have far reaching applications such as indexing in databases, implementing routing tables in routers, data compression code, etc.

## 6.4 SEARCH BY HASHING

Hashing is a technique which can be used to know the presence of a key in a list in just one step. The idea is if we already know the value at every index position in a list, it would require only a single comparison to check the presence or absence of a key in that list. Hashing makes searching operations very efficient. A formula called hash function is used to calculate the value at an index in the list.

Thus, a *hash function* takes elements of a list one by one and generates an index value for every element. This will generate a new list called the hash table. Each index of the hash table can hold only one item and the positions are indexed by integer values starting from 0. Note that the size of the hash table can be larger than the size of the list.

A simple hash function that works with numeric values is known as the *remainder method*. It takes an element from a list and divides it by the size of the hash table. The remainder so generated is called the hash value.

$$h(\text{element}) = \text{element \% size(hash table)}$$

We can easily implement a hash table using a Python's List. Let us consider an empty hash table having 10 positions as shown in Table 6.8:

**Table 6.8 An Empty hash table with 10 positions**

| Index/<br>position | 0    | 1    | 2    | 3    | 4    | 5    | 6    | 7    | 8    | 9    |
|--------------------|------|------|------|------|------|------|------|------|------|------|
| value              | None |

Let us consider a list of numbers (34, 16, 2, 93, 80, 77, 51). We can use the hash function remainder

method explained earlier to create a hash table as shown in Table 6.9.

**Table 6.9 hash function element % 10 applied on the elements of list**

|            |                |                |               |                |                |                |                |
|------------|----------------|----------------|---------------|----------------|----------------|----------------|----------------|
| Element    | 34             | 16             | 2             | 93             | 80             | 77             | 51             |
| Hash Value | $34 \% 10 = 4$ | $16 \% 10 = 6$ | $2 \% 10 = 2$ | $93 \% 10 = 3$ | $80 \% 10 = 0$ | $77 \% 10 = 7$ | $51 \% 10 = 1$ |

After computing the hash values, each element is inserted at its designated position in the hash table shown in Table 6.10

**Table 6.10 hash table generated for elements given in Table 6.10**

|       |    |    |   |    |    |      |    |    |      |      |
|-------|----|----|---|----|----|------|----|----|------|------|
| index | 0  | 1  | 2 | 3  | 4  | 5    | 6  | 7  | 8    | 9    |
| value | 80 | 51 | 2 | 93 | 34 | None | 16 | 77 | None | None |

Now, to search for a key, we can calculate its index using the hashing function and compare the element at that index with the key to declare whether the element is present in the list or not. This search operation involves just one comparison and hence the same amount of time is always required to search a key irrespective of the size of the list.

**Program 6-3 Use of hashing to find a key in the given list L**

```
#Function to check if a key is present or not
def hashFind(key,hashTable):
    if (hashTable[key % 10] == key): #key is present
        return ((key % 10)+1) #return the position
    else:
        return None #key is not present
#end of function

#create hashTable with 10 empty positions
hashTable=[None, None, None, None, None, None, None, None, None, None]
print("We have created a hashTable of 10 positions:")
print(hashTable)

L = [34, 16, 2, 93, 80, 77, 51]
print("The given list is", L[:])

# Apply hash function
for i in range(0,len(L)):
    hashTable[L[i]%10] = L[i]
```

```

print("The hash table contents are: ")
for i in range(0,len(hashTable)):
    print("hashindex= ", i, " , value =", hashTable[i])

key = int(input("Enter the number to be searched:"))

position = hashFind(key,hashTable)
if position is None:
    print("Number",key,"is not present in the hash table")
else:
    print("Number ",key," present at ",position, " position")

```

#### Output:

We have created a hashTable of 10 positions:  
[None, None, None, None, None, None, None, None, None, None]  
The given list is [34, 16, 2, 93, 80, 77, 51]  
The hash table contents are:  
hashindex= 0 , value = 80  
hashindex= 1 , value = 51  
hashindex= 2 , value = 2  
hashindex= 3 , value = 93  
hashindex= 4 , value = 34  
hashindex= 5 , value = None  
hashindex= 6 , value = 16  
hashindex= 7 , value = 77  
hashindex= 8 , value = None  
hashindex= 9 , value = None  
Enter the number to be searched:16  
Number 16 present at 7 position

#### 6.4.1 COLLISION

The hashing technique works fine if each element of the list maps to a unique location in the hash table. Consider a list [34, 16, 2, 26, 80]. While applying the hash function say, list [i] % 10, two elements (16 and 26) would have a hash value 6. This is a problematic situation, because according to our definition, two or more elements cannot be in the same position in the list. This situation is called *collision* in hashing.

We must have a mechanism for placing the other items with the same hash value in the hash table. This process is called *collision resolution*. Collision can be resolved in many ways, but it is beyond the scope of this book to discuss collision resolution methods.

If every item of the list maps to a unique index in the hash table, the hash function is called a *perfect hash function*. If a hash function is perfect, collision will never occur.

## NOTES

Apart from modulo division method, hash functions may be based on several other techniques like integer division, shift folding, boundary folding, mid-square function, extraction, radix transformation, etc. Again, it is beyond the scope of this book to discuss these methods.

The time taken by different hash functions may be different, but it remains constant for a particular hash function. The advantage of hashing is that the time required to compute the index value is independent of the number of items in the search list. It is to remember that the cost of computing a hash function must be small enough to make a hashing-based searching more efficient than other search methods.

### SUMMARY

- Searching means trying to locate a particular element called key in a collection of elements. Search specifies whether that key is present in the collection or not. Also, if the key is present, it tells the position of that key in the given collection.
- Linear search checks the elements of a list, one at a time, without skipping any element. It is useful when we need to search for an item in a small unsorted list, but it is slow and time-consuming when the list contains a large number of items. The time taken to search the list increases as the size of the list increases.
- Binary search takes a sorted/ordered list and divides it in the middle. It then compares the middle element with the key to be searched. If the middle element matches the key, the search is declared successful and the program ends. If the middle element is greater than the key, the search repeats only in the first half of the list. If the middle element is lesser than the key, the search repeats only in the second half of the list.

## NOTES

This splitting and reduction in list size continue till the key is found or the remaining list consists of only one item.

- In binary search, comparisons that do not find the key still give us idea about the location where the key may probably be found! They reveal whether the key is before or after the current middle position in the list, and we can use this information to narrow down or reduce our searching efforts.
- Hash based searching requires only one key comparison to discover the presence or absence of a key, provided every element is present at its designated position decided by a hash function. It calculates the position of the key in the list using a formula called the hash function and the key itself.
- When two elements map to the same slot in the hash table, it is called collision.
- The process of identifying a slot for the second and further items in the hash table in the event of collision, is called collision resolution.
- A perfect hash function maps every input key to a unique index in the hash table. If the hash function is perfect, collisions will never occur.



## EXERCISE

1. Using linear search determine the position of 8, 1, 99 and 44 in the list:

[1, -2, 32, 8, 17, 19, 42, 13, 0, 44]

Draw a detailed table showing the values of the variables and the decisions taken in each pass of linear search.

2. Use the linear search program to search the key with value 8 in the list having duplicate values such as [42, -2, 32, 8, 17, 19, 42, 13, 8, 44]. What is the position returned? What does this mean?
3. Write a program that takes as input a list having a mix of 10 negative and positive numbers and a key value.

Apply linear search to find whether the key is present in the list or not. If the key is present it should display the position of the key in the list otherwise it should print an appropriate message. Run the program for at least 3 different keys and note the result.

4. Write a program that takes as input a list of 10 integers and a key value and applies binary search to find whether the key is present in the list or not. If the key is present it should display the position of the key in the list otherwise it should print an appropriate message. Run the program for at least 3 different key values and note the results.
5. Following is a list of unsorted/unordered numbers:  
 [50, 31, 21, 28, 72, 41, 73, 93, 68, 43, 45, 78, 5, 17, 97, 71, 69, 61, 88, 75, 99, 44, 55, 9]
  - Use linear search to determine the position of 1, 5, 55 and 99 in the list. Also note the number of key comparisons required to find each of these numbers in the list.
  - Use a Python function to sort/arrange the list in ascending order.
  - Again, use linear search to determine the position of 1, 5, 55 and 99 in the list and note the number of key comparisons required to find these numbers in the list.
  - Use binary search to determine the position of 1, 5, 55 and 99 in the sorted list. Record the number of iterations required in each case.
6. Write a program that takes as input the following unsorted list of English words:  
 [Perfect, Stupendous, Wondrous, Gorgeous, Awesome, Mirthful, Fabulous, Splendid, Incredible, Outstanding, Propitious, Remarkable, Stellar, Unbelievable, Super, Amazing].
  - Use linear search to find the position of Amazing, Perfect, Great and Wondrous in the list. Also note the number of key comparisons required to find these words in the list.
  - Use a Python function to sort the list.
  - Again, use linear search to determine the position of Amazing, Perfect, Great and Wondrous in the list and note the number of key comparisons required to find these words in the list.
  - Use binary search to determine the position of Amazing, Perfect, Great and Wondrous in the sorted list. Record the number of iterations required in each case.

## NOTES

7. Estimate the number of key comparisons required in binary search and linear search if we need to find the details of a person in a sorted database having 230 (1,073,741,824) records when details of the person being searched lies at the middle position in the database. What do you interpret from your findings?
8. Use the hash function:  $h(\text{element}) = \text{element} \% 11$  to store the collection of numbers: [44, 121, 55, 33, 110, 77, 22, 66] in a hash table. Display the hash table created. Search if the values 11, 44, 88 and 121 are present in the hash table, and display the search results.
9. Write a Python program by considering a mapping of list of countries and their capital cities such as:

```
CountryCapital= {'India':'New Delhi','UK':  
                 'London','France':'Paris',  
                 'Switzerland': 'Berne',  
                 'Australia': 'Canberra'}
```

Let us presume that our hash function is the length of the Country Name. Take two lists of appropriate size: one for keys (Country) and one for values (Capital). To put an element in the hash table, compute its hash code by counting the number of characters in Country, then put the key and value in both the lists at the corresponding indices. For example, India has a hash code of 5. So, we store India at the 5th position (index 4) in the keys list, and New Delhi at the 5th position (index 4) in the values list and so on. So that we end up with:

| hash index = length of key - 1 | List of Keys | List of Values |
|--------------------------------|--------------|----------------|
| 0                              | None         | None           |
| 1                              | UK           | London         |
| 2                              | None         | None           |
| 3                              | Cuba         | Havana         |
| 4                              | India        | New Delhi      |
| 5                              | France       | Paris          |
| 6                              | None         | None           |
| 7                              | None         | None           |
| 8                              | Australia    | Canberra       |
| 9                              | None         | None           |
| 10                             | Switzerland  | Berne          |

Now search the capital of India, France and the USA in the hash table and display your result.

Chapter

7

# Understanding Data



## In this Chapter

- » Introduction to Data
- » Data Collection
- » Data Storage
- » Data Processing
- » Statistical Techniques for Data Processing

*“Data is not information, Information is not knowledge, Knowledge is not understanding, Understanding is not wisdom.”*

— Gary Schubert

## 7.1 INTRODUCTION TO DATA

Many a time, people take decisions based on certain data or information. For example, while choosing a college for getting admission, one looks at placement data of previous years of that college, educational qualification and experience of the faculty members, laboratory and hostel facilities, fees, etc. So we can say that identification of a college is based on various data and their analysis. Governments systematically collect and record data about the population through a process called census. Census data contains valuable information which are helpful in planning and formulating policies. Likewise, the coaching staff of a sports team analyses previous performances of opponent teams for making strategies. Banks maintain data about the customers, their account details and transactions. All these examples highlight the need of data in various fields. Data are indeed crucial for decision making.

In the previous examples, one cannot make decisions by looking at the data itself. In our example of choosing a college, suppose the placement cell of the college has maintained data of about 2000 students placed with different companies at different salary packages in the last 3 years. Looking at such data, one cannot make any remark about the placement of students of that college. The college processes and analyses this data and the results are given in the placement brochure of the college through summarisation as well as visuals for easy understanding. Hence, data need to be gathered, processed and analysed for making decisions.

In general, data is a collection of characters, numbers, and other symbols that represents values of some situations or variables. Data is plural and singular of the word data is “datum”. Using computers, data are stored in electronic forms because data processing becomes faster and easier as compared to manual data processing done by people. The Information and Communication Technology (ICT) revolution led by computer, mobile and Internet has resulted in generation of large volume of data and at a very fast pace. The following list contains some examples of data that we often come across.

- Name, age, gender, contact details, etc., of a person
- Transactions data generated through banking, ticketing, shopping, etc. whether online or offline
- Images, graphics, animations, audio, video
- Documents and web pages
- Online posts, comments and messages
- Signals generated by sensors
- Satellite data including meteorological data, communication data, earth observation data, etc.

### 7.1.1 Importance of Data

Human beings rely on data for making decisions. Besides, large amount of data when processed with the help of a computer, show us the possibilities or hidden traits which are otherwise not visible to humans. When one withdraws money from ATM, the bank needs to debit the withdrawn amount from the linked account. So the bank needs to maintain data and update it as and when required. The meteorological offices continuously keep on monitoring satellite data for any upcoming cyclone or heavy rain.



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



In a competitive business environment, it is important for business organisations to continuously monitor and analyse market behaviour with respect to their products and take actions accordingly. Besides, companies identify customer demands as well as feedbacks, and make changes in their products or services accordingly.

The dynamic pricing concept used by airlines and railway is another example where they decide the price based on relationships between demand and supply. The cab booking Apps increase or decrease the price based on demand for cabs at a particular time. Certain restaurants offer discounted price (called happy hours), they decide when and how much discount to offer by analysing sales data at different time periods.

Besides business, following are some other scenarios where data are also stored and analysed for making decisions:

- The electronic voting machines are used for recording the votes cast. Subsequently, the voting data from all the machines are accumulated to declare election results in a short time as compared to manual counting of ballot papers.
- Scientists record data while doing experiments to calculate and compare results.
- Pharmaceutical companies record data while trying out a new medicine to see its effectiveness.
- Libraries maintain data about books in the library and the membership of the library.
- The search engines give us results after analysing large volume of data available on the websites across World Wide Web (www).
- Weather alerts are generated by analysing data received from various satellites.

### **7.1.2 Types of Data**

As data come from different sources, they can be in different formats. For example, an image is a collection of pixels; a video is made up of frames; a fee slip is made up of few numeric and non-numeric entries; and messages/chats are made up of texts, icons (emoticons) and images/videos. Two broad categories in which data can be classified on the basis of their format are:

#### **(A) Structured Data**

Data which is organised and can be recorded in a well defined format is called structured data. Structured

### Activity 7.1

Observe Voter Identity cards of your family members and identify the data fields under which data are organised. Are they same for all?



Data is usually stored in computer in a tabular (in rows and columns) format where each column represents different data for a particular parameter called attribute/characteristic/variable and each row represents data of an observation for different attributes. Table 7.1 shows structured data related to an inventory of kitchen items maintained by a shop.

**Table 7.1 Structured data about kitchen items in a shop**

| ModelNo | ProductName     | Unit Price | Discount(%) | Items_in_Inventory |
|---------|-----------------|------------|-------------|--------------------|
| ABC1    | Water bottle    | 126        | 8           | 13                 |
| ABC2    | Melamine Plates | 320        | 5           | 45                 |
| ABC3    | Dinner Set      | 4200       | 10          | 8                  |
| GH67    | Jug             | 80         | 0           | 10                 |
| GH78    | Table Spoon     | 120        | 5           | 14                 |
| GH81    | Bucket          | 190        | 12          | 6                  |
| NK2     | Kitchen Towel   | 25         | 0           | 32                 |

Given this data, using a spreadsheet or other such software, the shop owner can find out how many total items are there by summing the column Items\_in\_Inventory of Table 7.1. The owner of the shop can also calculate the total value of all items in the inventory by multiplying each entry of column 3 (Unit Price) with corresponding entry of column 5 (Items\_in\_Inventory) and finding their sum.

Table 7.2 shows more examples of structured data recorded for different attributes.

**Table 7.2 Attributes maintained for different activities**

| Entity/Activities           | Data Fields/Parameters/Attributes                                                               |
|-----------------------------|-------------------------------------------------------------------------------------------------|
| Books at a shop             | BookTitle, Author, Price, YearofPublication                                                     |
| Depositing fees in a school | StudentName, Class, RollNo, FeesAmount, DepositDate                                             |
| Amount withdrawal from ATM  | AccHolderName, AccountNo, TypeofAcc, DateofWithdrawal, AmountWithdrawn, ATMId, TimeOfWithdrawal |

### (B) Unstructured Data

A newspaper contains various types of news items which are also called data. But there is no fixed pattern that a newspaper follows in placing news articles. One day there might be three images of different sizes on a page along with five news items and one or more advertisements. While on another day there, might be one big image with three textual news items. So there is

no particular format nor any fixed structure for printing news. Another example is the content of an email. There is no fixed structure about how many lines or paragraphs one has to write in an email or how many files are to be attached with an email. In summary, data which are not in the traditional row and column structure is called unstructured data.

Examples of unstructured data include web pages consisting of text as well as multimedia contents (image, graphics, audio/video). Other examples include text documents, business reports, books, audio/video files, social media messages. Although there are ways to process unstructured data, we are going to focus on handling structured data only in this book.

Unstructured data are sometimes described with the help of some other data called metadata. Metadata is basically data about data. For example, we describe different parts of an email as subject, recipient, main body, attachment, etc. These are the metadata for the email data. Likewise, we can have some metadata for an image file as image size (in KB or MB), image type (for example, JPEG, PNG), image resolution, etc.

### Think and Reflect

When we click a photograph using our digital or mobile camera, does it have some metadata associated with it?



## 7.2 DATA COLLECTION

For processing data, we need to collect or gather data first. We can then store the data in a file or database for later use. Data collection here means identifying already available data or collecting from the appropriate sources. Suppose there are three different scenarios where sales data in a grocery store are available:

- Sales data are available with the shopkeeper in a diary or register. In this case we should enter the data in a digital format for example, in a spreadsheet.
- Data are already available in a digital format, say in a CSV (comma separated values) file.
- The shopkeeper has so far not recorded any data in either form but wants to get a software developed for maintaining sales data and accounts. The software may be developed using a programming language such as Python which can be used to store and retrieve data from a CSV file or a database management system like MySQL, which will be discussed further.

### **Think and Reflect**

Identify attributes needed for creating an Aadhaar Card.



Data are continuously being generated at different sources. Our interactions with digital medium are continuously generating huge volumes of data. Hospitals are collecting data about patients for improving their services. Shopping malls are collecting data about the items being purchased by people. On analysing such data, suppose it appears that bedsheets and groceries are frequently bought together. Hence, the shop owner may decide to display bedsheets near the grocery section in the mall to increase the sales. Likewise, a political analyst may look at the data contained in the posts and messages at a social media platform and analyse to see public opinion before an election. Organisations like World Bank and International Monetary Fund (IMF) are collecting data related to various economic parameters from different countries for making economic forecasts.

### **7.3 DATA STORAGE**

Once we gather data and process them to get results, we may not then simply discard the data. Rather, we would like to store them for future use as well. Data storage is the process of storing data on storage devices so that data can be retrieved later. Now a days large volume of data are being generated at a very high rate. As a result, 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, Hard Disk Drive (HDD), Solid State Drive (SSD), CD/DVD, Tape Drive, Pen Drive, Memory Card, etc.

We store data like images, documents, audios/videos, etc. as files in our computers. Likewise, school/hospital data are stored in data files. We use computers to add, modify or delete data in these files or process these data files to get results. However, file processing has certain limitations, which can be overcome through Database Management System (DBMS).

### **Think and Reflect**

Is it necessary to store data in files before processing?



### **7.4 DATA PROCESSING**

We are interested in understanding data as they hold valuable facts and information that can be useful in our decision making process. However, by looking at the vast or large amount of data, one cannot arrive at a conclusion. Rather, data need to be processed to

get results and after analysing those results, we make conclusions or decisions.

We find automated data processing in situations like online bill payment, registration of complaints, booking tickets, etc. Figure 7.1 illustrates basic steps used to process the data to get the output.

Figure 7.2 shows some tasks along with data, processing and generated output/information.

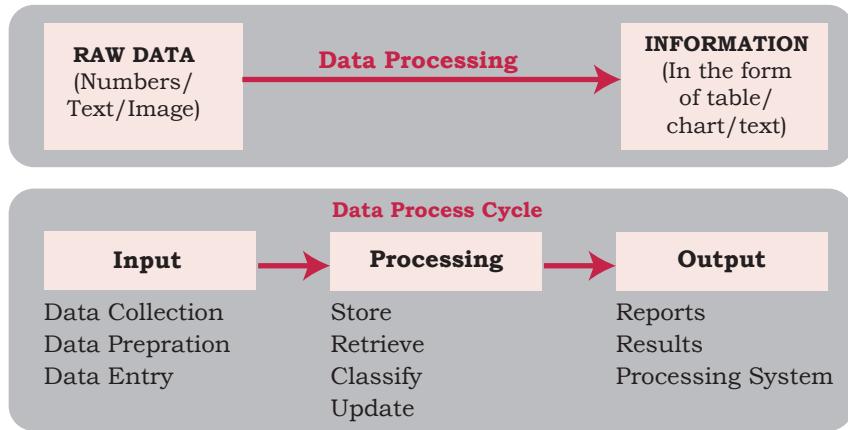


Figure 7.1: Steps in data processing

## 7.5 STATISTICAL TECHNIQUES FOR DATA PROCESSING

Given a set of data values, we need to process them to get information. There are various techniques which help us to have preliminary understanding about the data.

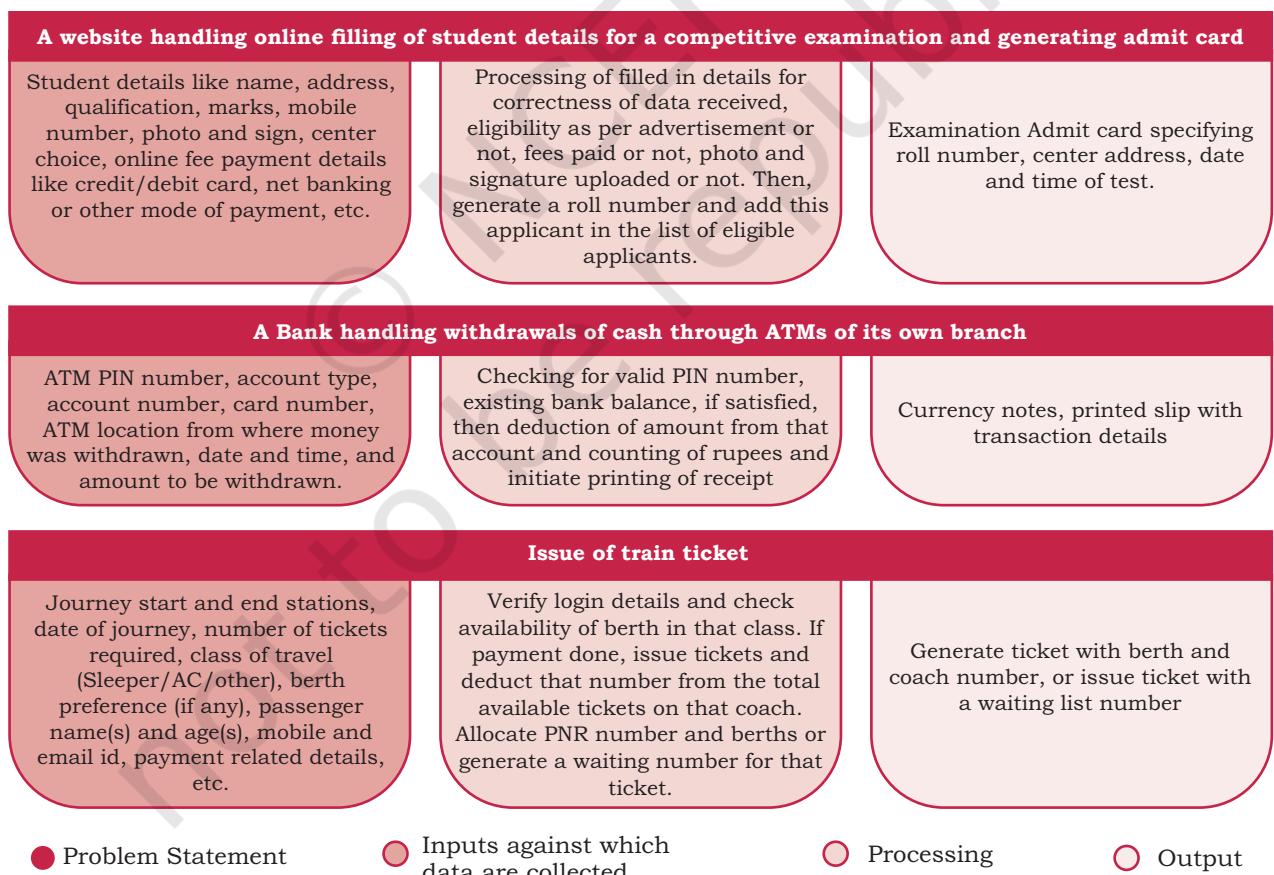


Figure 7.2: Data based problem statements

Summarisation methods are applied on tabular data for its easy comprehension. Commonly used statistical techniques for data summarisation are given below:

### 7.5.1 Measures of Central Tendency

A measure of central tendency is a single value that gives us some idea about the data. Three most common measures of central tendency are the *mean*, *median*, and *mode*. Instead of looking at each individual data values, we can calculate the mean, median and mode of the data to get an idea about average, middle value and frequency of occurrence of a particular value, respectively. Selection of a measure of central tendency depends on certain characteristics of data.

#### (A) Mean

*Mean* is simply the average of numeric values of an attribute. Mean is also called *average*. Suppose there are data on weight of 40 students in a class. Instead of looking at each of the data values, we can calculate the average to get an idea about the average weight of students in that class.

**Definition:** Given  $n$  values  $x_1, x_2, x_3, \dots, x_n$ , mean is computed as 
$$\frac{\sum_{i=1}^n x_i}{n}$$
.

#### Example 7.1

Assume that height (in cm) of students in a class are as follows [90, 102, 110, 115, 85, 90, 100, 110, 110]. Mean or average height of the class is

$$\frac{90 + 102 + 110 + 115 + 85 + 90 + 100 + 110 + 110}{9} = \frac{912}{9} = 101.33 \text{ cm}$$

Mean is not a suitable choice if there are outliers in the data. To calculate mean, the outliers or extreme values should be removed from the given data and then calculate mean of the remaining data.

**Note:** An outlier is an exceptionally large or small value, in comparison to other values of the data. Usually, outliers are considered as error since they can influence/affect the average or other statistical calculation based on the data.

#### (B) Median

Median is also computed for a single attribute/variable at a time. When all the values are sorted in ascending or descending order, the middle value is called the *Median*. When there are odd number of values, then median is

the value at the middle position. If the list has even number of values, then median is the average of the two middle values. Median represents the central value at which the given data is equally divided into two parts.

#### *Example 7.2*

Consider the previous data of height of students used in calculation of mean value. In order to compute the median, the first step is to sort data in ascending or descending order. We have sorted the height data in ascending order as [85,90,90,100,102,110,110,110,115]. As there are total 9 values (odd number), the median is the value at position 5, that is 102 cm, whether counted from left to right or from right to left. Median represents the actual central value at which the given data is equally divided into two parts.

#### **(C) Mode**

Value that appears most number of times in the given data of an attribute/variable is called *Mode*. It is computed on the basis of frequency of occurrence of distinct values in the given data. A data set has no mode if each value occurs only once. There may be multiple modes in the data if more than one values have same highest frequency. Mode can be found for numeric as well as non-numeric data.

#### *Example 7.3*

In the list of height of students, mode is 110 as its frequency of occurrence in the list is 3, which is larger than the frequency of rest of the values.

### **7.5.2 Measures of Variability**

The measures of variability refer to the spread or variation of the values around the mean. They are also called measures of dispersion that indicate the degree of diversity in a data set. They also indicate difference within the group. Two different data sets can have the same mean, median or mode but completely different levels of dispersion, or vice versa. Common measures of dispersion or variability are Range and Standard Deviation.

#### **(A) Range**

It is the difference between maximum and minimum values of the data (the largest value minus the smallest value). *Range* can be calculated only for numerical data. It is a measure of dispersion and tells about coverage/spread of data values. For

#### **Think and Reflect**

Out of Mean and Median, which one is more sensitive to outliers in data?



example difference in salaries of employees, marks of a student, price of toys, etc. As range is calculated based on the two extreme values, any outlier in the data badly influences the result.

Let  $M$  be the largest or maximum value and  $S$  is the smallest or minimum value in the data, then Range is the difference between two extreme values i.e.  $M - S$  or  $\text{Maximum} - \text{Minimum}$ .

#### *Example 7.4*

In the above example, minimum height value is 85 cm and maximum height value is 115 cm. Hence, range is  $115 - 85 = 30$  cm.

#### **(B) Standard deviation**

Standard deviation refers to differences within the group or set of data of a variable. Like Range, it also measures the spread of data. However, unlike Range which only uses two extreme values in the data, calculation of standard deviation considers all the given data. It is calculated as the positive square root of the average of squared difference of each value from the mean value of data. Smaller value of standard deviation means data are less spread while a larger value of standard deviation means data are more spread.

Given  $n$  values  $x_1, x_2, x_3, \dots, x_n$ , and their mean  $\bar{x}$ , the standard deviation, represented as  $\sigma$  (greek letter sigma) is computed as

$$\sigma = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n}}$$

#### *Example 7.5*

Let us compute the standard deviation of the height of nine students that we used while calculating Mean. The Mean ( $\bar{x}$ ) was calculated to be 101.33 cm. Subtract each value from the mean and take square of that value. Dividing the sum of square values by total number of values and taking its square root gives the standard deviation in data. See Table 7.3 for details.

**Table 7.3 Standard deviation of attendance of 9 students**

| Height ( $x$ ) in cm | $x - \bar{x}$ | $(x - \bar{x})^2$ | $\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n}$ |
|----------------------|---------------|-------------------|--------------------------------------------|
| 90                   | -11.33        | 128.37            |                                            |
| 102                  | 0.67          | 0.36              |                                            |
| 110                  | 8.67          | 75.17             |                                            |
| 115                  | 13.67         | 186.87            |                                            |

|                    |                            |                                |                                            |
|--------------------|----------------------------|--------------------------------|--------------------------------------------|
| 85                 | -16.33                     | 266.67                         | $= \frac{938}{9} = 104.22$                 |
| 90                 | -11.33                     | 128.37                         |                                            |
| 100                | -1.33                      | 1.77                           |                                            |
| 110                | 8.67                       | 75.17                          |                                            |
| 110                | 8.67                       | 75.17                          |                                            |
| $n = 9$            | $\sum(x - \bar{x}) = 0.03$ | $\sum(x - \bar{x})^2 = 938.00$ | $\sigma = \sqrt{104.22} = 10.2 \text{ cm}$ |
| $\bar{x} = 101.33$ |                            |                                |                                            |

Let us look at the following problems and select the suitable statistical technique to be applied (Mean/Median/Mode/Range/Standard Deviation):

| Problem Statement                                                                       | Choose suitable statistical method |
|-----------------------------------------------------------------------------------------|------------------------------------|
| The management of a company wants to know about disparity in salaries of all employees. |                                    |
| Teacher wants to know about the average performance of the whole class in a test.       |                                    |
| Compare height of residents of two cities                                               |                                    |
| Find the dominant value from a set of values                                            |                                    |
| Compare income of residents of two cities                                               |                                    |
| Find the popular color for car after surveying the car owners of a small city.          |                                    |

It is important to understand statistical techniques so that one can decide which statistical technique to use to arrive at a decision. Different programming tools are available for efficient analysis of large volumes of data. These tools make use of statistical techniques for data analysis. One such programming tool is Python and it has libraries specially built for data processing and analysis. We will be covering some of them in the following chapters.

## SUMMARY

- Data refer to unorganised facts that can be processed to generate meaningful result or information.
- Data can be structured or unstructured.
- Hard Disk, SSD, CD/DVD, Pen Drive, Memory Card, etc. are some of the commonly used storage devices.

- Data Processing cycle involves input and storage of data, its processing and generating output.
- Summarising data using statistical techniques aids in revealing data characteristics.
- Mean, Median, Mode, Range, and Standard Deviation are some of the statistical techniques used for data summarisation.
- Mean is the average of given values.
- Median is the mid value when data are sorted in ascending/descending order.
- Mode is the data value that appears most number of times.
- Range is the difference between the maximum and minimum values.
- Standard deviation is the positive square root of the average of squared difference of each value from the mean.

## EXERCISE

1. Identify data required to be maintained to perform the following services:
  - a) Declare exam results and print e-certificates
  - b) Register participants in an exhibition and issue biometric ID cards
  - c) To search for an image by a search engine
  - d) To book an OPD appointment with a hospital in a specific department
2. A school having 500 students wants to identify beneficiaries of the merit-cum means scholarship, achieving more than 75% for two consecutive years and having family income less than 5 lakh per annum. Briefly describe data processing steps to be taken by the to beneficial prepare the list of school.
3. A bank 'xyz' wants to know about its popularity among the residents of a city 'ABC' on the basis of number of bank accounts each family has and the average monthly account balance of each person. Briefly describe the steps to be taken for collecting data and what results can be checked through processing of the collected data.

## NOTES

4. Identify type of data being collected/generated in the following scenarios:
  - a) Recording a video
  - b) Marking attendance by teacher
  - c) Writing tweets
  - d) Filling an application form online
5. Consider the temperature (in Celsius) of 7 days of a week as 34, 34, 27, 28, 27, 34, 34. Identify the appropriate statistical technique to be used to calculate the following:
  - a) Find the average temperature.
  - b) Find the temperature Range of that week.
  - c) Find the standard deviation temperature.
6. A school teacher wants to analyse results. Identify the appropriate statistical technique to be used along with its justification for the following cases:
  - a) Teacher wants to compare performance in terms of division secured by students in Class XII A and Class XII B where each class strength is same.
  - b) Teacher has conducted five unit tests for that class in months July to November and wants to compare the class performance in these five months.
7. Suppose annual day of your school is to be celebrated. The school has decided to felicitate those parents of the students studying in classes XI and XII, who are the alumni of the same school. In this context, answer the following questions:
  - a) Which statistical technique should be used to find out the number of students whose both parents are alumni of this school?
  - b) How varied are the age of parents of the students of that school?
8. For the annual day celebrations, the teacher is looking for an anchor in a class of 42 students. The teacher would make selection of an anchor on the basis of singing skill, writing skill, as well as monitoring skill.
  - a) Which mode of data collection should be used?
  - b) How would you represent the skill of students as data?
9. Differentiate between structured and unstructured data giving one example.

The principal of a school wants to do following analysis on the basis of food items procured and sold in the canteen:

  - a) Compare the purchase and sale price of fruit juice and biscuits.

## NOTES

- b) Compare sales of fruit juice, biscuits and samosa.
- c) Variation in sale price of fruit juices of different companies for same quantity (in ml).

Create an appropriate dataset for these items (fruit juice, biscuits, samosa) by listing their purchase price and sale price. Apply basic statistical techniques to make the comparisons.

Chapter

8

# Database Concepts



12130CH08

## In this Chapter

- » *Introduction*
- » *File System*
- » *Database Management System*
- » *Rational Data Model*
- » *Keys in a Relational Database*

*"Inconsistency of your mind... Can damage your memory... Remove the inconsistent data... And keep the original one !!!"*

— Nisarga Jain

## 8.1 INTRODUCTION

After learning about importance of data in the previous chapter, we need to explore the methods to store and manage data electronically. Let us take an example of a school that maintains data about its students, along with their attendance record and guardian details.

The class teacher marks daily attendance of the students in the attendance register. The teacher records 'P' for present or 'A' for absent against each student's roll number on each working day. If class strength is 50 and total working days in a month are 26, the teacher needs to record  $50 \times 26$  records manually in the register every month. As the volume of data increases, manual data entry becomes tedious. Following are some of the limitations of manual record keeping in this example:

### Activity 8.1

Visit a few shops where records are maintained manually and identify a few limitations of manual record keeping faced by them.



- 1) Entry of student details (Roll number and name) in the new attendance register when the student is promoted to the next class.
- 2) Writing student details on each month's attendance page where inconsistency may happen due to incorrectly written names, skipped student records, etc.
- 3) Loss of data in case attendance register is lost or damaged.
- 4) Erroneous calculation while consolidating attendance record manually.

The office staff also manually maintain student details viz. Roll Number, Name and Date of Birth with respective guardian details viz. Guardian name, Contact Number and Address. This is required for correspondence with guardian regarding student attendance and result.

Finding information from a huge volume of papers or deleting/modifying an entry is a difficult task in pen and paper based approach. To overcome the hassles faced in manual record keeping, it is desirable to store attendance record and student details on separate data files on a computerised system, so that office staff and teachers can:

- 1) Simply copy the student details to the new attendance file from the old attendance file when students are promoted to next class.
- 2) Find any data about student or guardian.
- 3) Add more details to existing data whenever a new student joins the school.
- 4) Modify stored data like details of student or guardian whenever required.
- 5) Remove/delete data whenever a student leaves the school.

## 8.2 FILE SYSTEM

A file can be understood as a container to store data in a computer. Files can be stored on the storage device of a computer system. Contents of a file can be texts, computer program code, comma separated values (CSV), etc. Likewise, pictures, audios/videos, web pages are also files.

Files stored on a computer can be accessed directly and searched for desired data. But to access data of a

file through software, for example, to display monthly attendance report on school website, one has to write computer programs to access data from files.

Continuing the example of attendance at school, we need to store data about students and attendance in two separate files. Table 8.1 shows the contents of STUDENT file which has six columns, as detailed below:

- RollNumber – Roll number of the student
- SName – Name of the student
- SDateofBirth – Date of birth of the student
- GName – Name of the guardian
- GPhone – Phone number of the student guardian
- GAddress – Address of the guardian of the student

**Table 8.1 STUDENT file maintained by office staff**

| Roll Number | SName        | SDateof Birth | GName           | GPhone     | GAddress                               |
|-------------|--------------|---------------|-----------------|------------|----------------------------------------|
| 1           | Atharv Ahuja | 2003-05-15    | Amit Ahuja      | 5711492685 | G-35, Ashok Vihar, Delhi               |
| 2           | Daizy Bhutia | 2002-02-28    | Baichung Bhutia | 7110047139 | Flat no. 5, Darjeeling Appt., Shimla   |
| 3           | Taleem Shah  | 2002-02-28    | Himanshu Shah   | 9818184855 | 26/77, West Patel Nagar, Ahmedabad     |
| 4           | John Dsouza  | 2003-08-18    | Danny Dsouza    |            | S -13, Ashok Village, Daman            |
| 5           | Ali Shah     | 2003-07-05    | Himanshu Shah   | 9818184855 | 26/77, West Patel Nagar, Ahmedabad     |
| 6           | Manika P.    | 2002-03-10    | Sujata P.       | 7802983674 | HNO-13, B- block, Preet Vihar, Madurai |

Table 8.2 shows another file called ATTENDANCE which has four columns, as detailed below:

- AttendanceDate – Date for which attendance was marked
- RollNumber – Roll number of the student
- SName – Name of the student
- AttendanceStatus – Marked as P (present) or A (absent)

**Table 8.2 ATTENDANCE file maintained by class teacher**

| AttendanceDate | RollNumber | SName        | AttendanceStatus |
|----------------|------------|--------------|------------------|
| 2018-09-01     | 1          | Atharv Ahuja | P                |
| 2018-09-01     | 2          | Daizy Bhutia | P                |
| 2018-09-01     | 3          | Taleem Shah  | A                |
| 2018-09-01     | 4          | John Dsouza  | P                |
| 2018-09-01     | 5          | Ali Shah     | A                |
| 2018-09-01     | 6          | Manika P.    | P                |

|            |   |              |   |
|------------|---|--------------|---|
| 2018-09-02 | 1 | Atharv Ahuja | P |
| 2018-09-02 | 2 | Daizy Bhutia | P |
| 2018-09-02 | 3 | Taleem Shah  | A |
| 2018-09-02 | 4 | John Dsouza  | A |
| 2018-09-02 | 5 | Ali Shah     | P |
| 2018-09-02 | 6 | Manika P.    | P |

### 8.2.1 Limitations of a File System

File system becomes difficult to handle when number of files increases and volume of data also grows. Following are some of the limitations of file system:

#### (A) Difficulty in Access

Files themselves do not provide any mechanism to retrieve data. Data maintained in a file system are accessed through application programs. While writing such programs, the developer may not anticipate all the possible ways in which data may be accessed. So, sometimes it is difficult to access data in the required format and one has to write application program to access data.

#### (B) Data Redundancy

Redundancy means same data are duplicated in different places (files). In our example, student names are maintained in both the files. Besides, in Table 8.1, students with roll numbers 3 and 5 have same guardian name and therefore same guardian name is maintained twice. Both these are examples of redundancy which is difficult to avoid in a file system. Redundancy leads to excess storage use and may cause data inconsistency also.

#### (C) Data Inconsistency

Data inconsistency occurs when same data maintained in different places do not match. If a student wants to get changed the spelling of her name, it needs to be changed in SName column in both the files. Likewise, if a student leaves school, the details need to be deleted from both the files. As the files are being maintained by different people, the changes may not happen in one of the files. In that case, the student name will be different (inconsistent) in both the files.

#### (D) Data Isolation

Both the files presented at Table 8.1 (STUDENT) and at Table 8.2 (ATTENDANCE) are related to students. But

there is no link or mapping between them. The school will have to write separate programs to access these two files. This is because data mapping is not supported in file system. In a more complex system where data files are generated by different person at different times, files being created in isolation may be of different formats. In such case, it is difficult to write new application programs to retrieve data from different files maintained at multiple places, as one has to understand the underlying structure of each file as well.

#### **(E) Data Dependence**

Data are stored in a specific format or structure in a file. If the structure or format itself is changed, all the existing application programs accessing that file also need to be changed. Otherwise, the programs may not work correctly. This is data dependency. Hence, updating the structure of a data file requires modification in all the application programs accessing that file.

#### **(F) Controlled Data Sharing**

There can be different category of users like teacher, office staff and parents. Ideally, not every user should be able to access all the data. As an example, guardians and office staff can only see the student attendance data but should not be able to modify/delete it. It means these users should be given limited access (read only) to the ATTENDANCE file. Only the teacher should be able to update the attendance data. It is very difficult to enforce this kind of access control in a file system while accessing files through application programs.

### **8.3 DATABASE MANAGEMENT SYSTEM**

Limitations faced in file system can be overcome by storing the data in a database where data are logically related. We can organise related data in a database so that it can be managed in an efficient and easy way.

A database management system (DBMS) or database system in short, is a software that can be used to create and manage databases. DBMS lets users to create a database, store, manage, update/modify and retrieve data from that database by users or application programs. Some examples of open source and commercial DBMS include MySQL, Oracle, PostgreSQL, SQL Server, Microsoft Access, MongoDB.



Some database management systems include a graphical user interface for users to create and manage databases. Other database systems use a command line interface that requires users to use programming commands to create and manage databases.



A database system hides certain details about how data are actually stored and maintained. Thus, it provides users with an abstract view of the data. A database system has a set of programs through which users or other programs can access, modify and retrieve the stored data.

The DBMS serves as an interface between the database and end users or application programs. Retrieving data from a database through special type of commands is called querying the database. In addition, users can modify the structure of the database itself through a DBMS.

Databases are widely used in various fields. Some applications are given in Table 8.3.

**Table 8.3 Use of Database in Real-life Applications**

| Application                      | Database to maintain data about                                                                                    |
|----------------------------------|--------------------------------------------------------------------------------------------------------------------|
| Banking                          | customer information, account details, loan details, transaction details, etc.                                     |
| Crop Loan                        | kisan credit card data, farmer's personal data, land area and cultivation data, loan history, repayment data, etc. |
| Inventory Management             | product details, customer information, order details, delivery data, etc.                                          |
| Organisation Resource Management | employee records, salary details, department information, branch locations, etc.                                   |
| Online Shopping                  | items description, user login details, users preferences details, etc.                                             |

### 8.3.1 File System to DBMS

Let us revisit our school example where two data files were maintained (Table 8.1 by office and Table 8.2 by teacher). Let us now design a database to store data of those two files. We know that tables in a database are linked or related through one or more common columns or fields. In our example, the STUDENT (Table 8.1) file and ATTENDANCE (Table 8.2) file have RollNumber and SName as common field names. In order to convert these two files into a database, we need to incorporate the following changes:

- a) SName need not be maintained in ATTENDANCE file as it is already there in STUDENT. Details for a student can be retrieved through the common field RollNumber in both the files.

- b) If two siblings are in the same class, then same guardian details (GName, GPhone and GAddress) are maintained for both the siblings. We know this is a redundancy and by using a database we can avoid this. So let us split the STUDENT file into two file (STUDENT file and GUARDIAN) file so that each guardian data are maintained only once.
- c) One and more guardians can have the same name. So it will not be possible to identify which guardian is related to which student. In such case, we need to create an additional column, say GUID (Guardian ID) that will take unique value for each record in the GUARDIAN file. The column GUID will also be kept with STUDENT file for relating these two files.

**Note:** We could distinguish guardians by their phone numbers also. But, phone number can change, and therefore may not truly distinguish guardian.

Figure 8.1 shows the related data files for the STUDENT, GUARDIAN and ATTENDANCE details. Note that this is not the complete database schema since it does not show any relationship among tables.

| STUDENT                                     | GUARDIAN                            | ATTENDANCE                                       |
|---------------------------------------------|-------------------------------------|--------------------------------------------------|
| RollNumber<br>SName<br>SDateofBirth<br>GUID | GUID<br>GName<br>GPhone<br>GAddress | AttendanceDate<br>RollNumber<br>AttendanceStatus |

Figure 8.1: Record structure of three files in STUDENTATTENDANCE database

The tables shown at Figure 8.1 are empty, which are to be populated with actual data as shown in Table 8.4, 8.5 and 8.6.

**Table 8.4 Snapshot of STUDENT table**

| RollNumber | SName        | SDateofBirth | GUID         |
|------------|--------------|--------------|--------------|
| 1          | Atharv Ahuja | 2003-05-15   | 444444444444 |
| 2          | Daizy Bhutia | 2002-02-28   | 111111111111 |
| 3          | Taleem Shah  | 2002-02-28   |              |
| 4          | John Dsouza  | 2003-08-18   | 333333333333 |
| 5          | Ali Shah     | 2003-07-05   | 101010101010 |
| 6          | Manika P.    | 2002-03-10   | 466444444666 |



High Cost is incurred while shifting from file system to DBMS:

- Purchasing sophisticated hardware and software.
- Training users for querying.
- Recurrent cost to take regular backup and perform recovery operations.



**Table 8.5 Snapshot of GUARDIAN table**

| <b>GUID</b>  | <b>GName</b>    | <b>GPhone</b> | <b>GAddress</b>                        |
|--------------|-----------------|---------------|----------------------------------------|
| 444444444444 | Amit Ahuja      | 5711492685    | G-35, Ashok Vihar, Delhi               |
| 111111111111 | Baichung Bhutia | 3612967082    | Flat no. 5, Darjeeling Appt., Shimla   |
| 101010101010 | Himanshu Shah   | 4726309212    | 26/77, West Patel Nagar, Ahmedabad     |
| 333333333333 | Danny Dsouza    |               | S -13, Ashok Village, Daman            |
| 466444444666 | Sujata P.       | 3801923168    | HNO-13, B- block, Preet Vihar, Madurai |

**Table 8.6 Snapshot of ATTENDANCE table**

| <b>Date</b> | <b>RollNumber</b> | <b>Status</b> |
|-------------|-------------------|---------------|
| 2018-09-01  | 1                 | P             |
| 2018-09-01  | 2                 | P             |
| 2018-09-01  | 3                 | A             |
| 2018-09-01  | 4                 | P             |
| 2018-09-01  | 5                 | A             |
| 2018-09-01  | 6                 | P             |
| 2018-09-02  | 1                 | P             |
| 2018-09-02  | 2                 | P             |
| 2018-09-02  | 3                 | A             |
| 2018-09-02  | 4                 | A             |
| 2018-09-02  | 5                 | P             |
| 2018-09-02  | 6                 | P             |

Figure 8.2 shows a simplified database called STUDENTATTENDANCE, which is used to maintain data about the student, guardian and attendance. As shown here, the DBMS maintains a single repository of data at a centralised location and can be used by multiple users (office staff, teacher) at the same time.

### 8.3.2 Key Concepts in DBMS

In order to efficiently manage data using a DBMS, let us understand certain key terms:

#### (A) Database Schema

Database Schema is the design of a database. It is the skeleton of the database that represents the structure (table names and their fields/columns), the type of data each column can hold, constraints on the data to be stored (if any), and the relationships among the tables. Database schema is also called the visual or logical architecture as it tells us how the data are organised in a database.

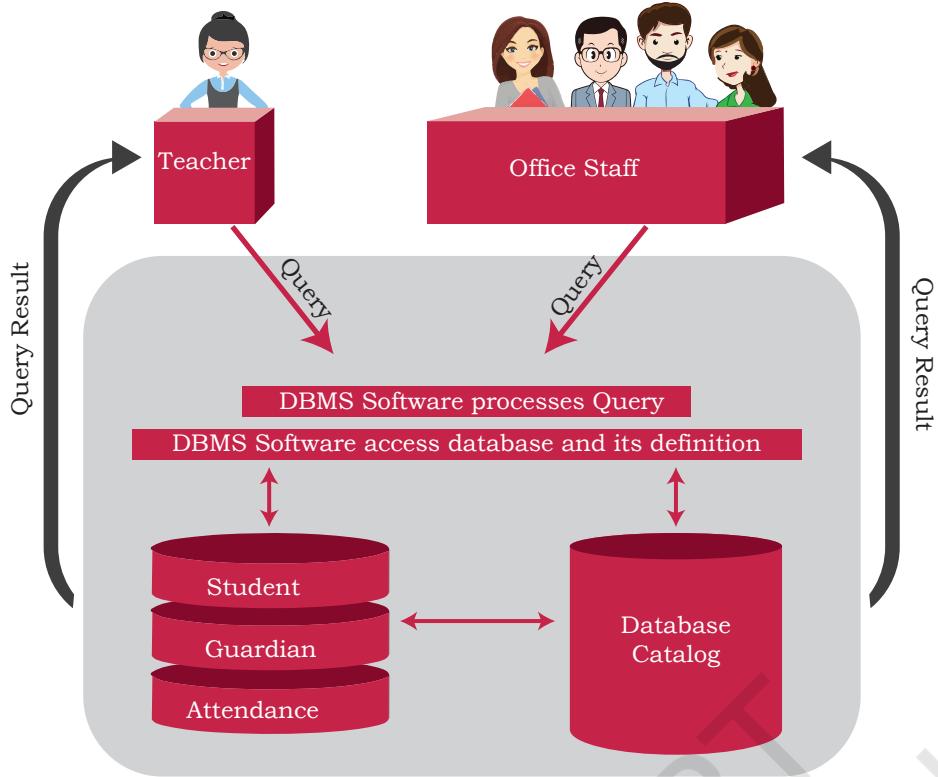


Figure 8.2: STUDENTATTENDANCE database environment

#### **(B) Data Constraint**

Sometimes we put certain restrictions or limitations on the type of data that can be inserted in one or more columns of a table. This is done by specifying one or more constraints on that column(s) while creating the tables. For example, one can define the constraint that the column mobile number can only have non-negative integer values of exactly 10 digits. Since each student shall have one unique roll number, we can put the NOT NULL and UNIQUE constraints on the RollNumber column. Constraints are used to ensure accuracy and reliability of data in the database.

#### **(C) Meta-data or Data Dictionary**

The database schema along with various constraints on the data is stored by DBMS in a database catalog or dictionary, called meta-data. A meta-data is data about the data.

#### **(D) Database Instance**

When we define database structure or schema, state of database is empty i.e. no data entry is there. After loading data, the state or snapshot of the database at any given time is the database instance. We may then retrieve data through queries or manipulate data

through updation, modification or deletion. Thus, the state of database can change, and thus a database schema can have many instances at different times.

#### **(E) Query**

A query is a request to a database for obtaining information in a desired way. Query can be made to get data from one table or from a combination of tables. For example, “find names of all those students present on Attendance Date 2000-01-02” is a query to the database. To retrieve or manipulate data, the user needs to write query using a query language called, which is discussed in chapter 8.

#### **(F) Data Manipulation**

Modification of database consists of three operations viz. Insertion, Deletion or Update. Suppose, Rivaan joins as a new student in the class then the student details need to be added in STUDENT as well as in GUARDIAN files of the Student Attendance database. This is called Insertion operation on the database. In case a student leaves the school, then his/her data as well as her guardian details need to be removed from STUDENT, GUARDIAN and ATTENDANCE files, respectively. This is called Deletion operation on the database. Suppose Atharv’s Guardian has changed his mobile number, his GPhone should be updated in GUARDIAN file. This is called Update operation on the database.

#### **(G) Database Engine**

Database engine is the underlying component or set of programs used by a DBMS to create database and handle various queries for data retrieval and manipulation.

### **8.4 RELATIONAL DATA MODEL**

Different types of DBMS are available and their classification is done based on the underlying data model. A data model describes the structure of the database, including how data are defined and represented, relationships among data, and the constraints. The most commonly used data model is Relational Data Model. Other types of data models include object-oriented data model, entity-relationship data model, document model and hierarchical data model. This book discusses the DBMS based on relational data model.

In relational model, tables are called relations that store data for different columns. Each table can have

multiple columns where each column name should be unique. For example, each row in the table represents a related set of values. Each row of Table 8.5 represents a particular guardian and has related values viz. guardian's ID with guardian name, address and phone number. Thus, a table consists of a collection of relationships.

It is important to note here that relations in a database are not independent tables, but are associated with each other. For example, relation ATTENDANCE has attribute RollNumber which links it with corresponding student record in relation STUDENT. Similarly, attribute GUID is placed with STUDENT table for extracting guardian details of a particular student. If linking attributes are not there in appropriate relations, it will not be possible to keep the database in correct state and retrieve valid information from the database.

Figure 8.3 shows the relational database Student Attendance along with the three relations (tables) STUDENT, ATTENDANCE and GUARDIAN.

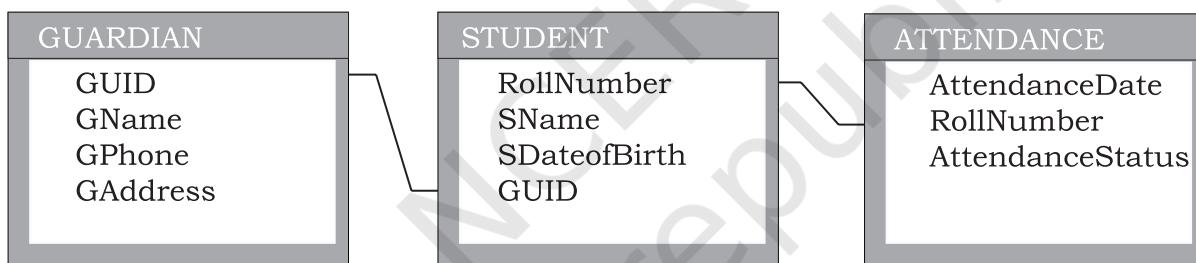


Figure 8.3: Representing STUDENTATTENDANCE database using Relational Data Model

**Table 8.7 Relation schemas along with its description of Student Attendance database**

| Relation Scheme                                          | Description of attributes                                                                                                                                                                                                                              |
|----------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| STUDENT(RollNumber, SName, SDateofBirth, GUID)           | RollNumber: unique id of the student<br>SName: name of the student<br>SDateofBirth: date of birth of the student<br>GUID: unique id of the guardian of the student                                                                                     |
| ATTENDANCE(AttendanceDate, RollNumber, AttendanceStatus) | AttendanceDate: date on which attendance is taken<br>RollNumber: roll number of the student<br>AttendanceStatus: whether present (P) or absent(A)<br>Note that combination of AttendanceDate and RollNumber will be unique in each record of the table |
| GUARDIAN(GUID, GName, GPhone, GAddress)                  | GUID: unique id of the guardian<br>GName: name of the guardian<br>GPhone: contact number of the guardian<br>GAddress: contact address of the guardian                                                                                                  |

Each tuple (row) in a relation (table) corresponds to data of a real world entity (for example, Student, Guardian, and Attendance). In the GUARDIAN relation (Table 8.5), each row represents the facts about the guardian and each column name in the GUARDIAN table is used to interpret the meaning of data stored under that column. A database that is modeled on relational data model concept is called Relational Database. Figure 8.4 shows relation GUARDIAN with some populated data.

Let us now understand the commonly used terminologies in relational data model using Figure 8.4.

| GUID         | GName           | GPhone     | GAddress                               |
|--------------|-----------------|------------|----------------------------------------|
| 444444444444 | Amit Ahuja      | 5711492685 | G-35, Ashok Vihar, Delhi               |
| 111111111111 | Baichung Bhutia | 3612967082 | Flat no. 5, Darjeeling Appt., Shimla   |
| 101010101010 | Himanshu Shah   | 4726309212 | 26/77, West Patel Nagar, Ahmedabad     |
| 333333333333 | Danny Dsouza    |            | S -13, Ashok Village, Daman            |
| 466444444666 | Sujata P.       | 3801923168 | HNO-13, B- block, Preet Vihar, Madurai |

Facts about RELATION GUARDIAN:

1. Degree (Number of attributes) = 4
2. Cardinality (Number of rows/tuples/records) = 5
3. Relation is a flat file i.e, each column has a single value and each record has same number of columns

Relation GUARDIAN  
with 4 attribute/  
columns

Relation  
State

Record/tuple/row

Figure 8.4: Relation GUARDIAN with its attributes and tuples

- i) **ATTRIBUTE:** Characteristic or parameters for which data are to be stored in a relation. Simply stated, the columns of a relation are the attributes which are also referred as fields. For example, GUID, GName, GPhone and GAddress are attributes of relation GUARDIAN.
- ii) **TUPLE:** Each row of data in a relation (table) is called a tuple. In a table with n columns, a tuple is a relationship between the n related values.
- iii) **DOMAIN:** It is a set of values from which an attribute can take a value in each row. Usually, a data type is used to specify domain for an attribute. For example, in STUDENT relation, the attribute RollNumber takes integer values and hence its domain is a set of integer values. Similarly, the set of character strings constitutes the domain of the attribute SName.
- iv) **DEGREE:** The number of attributes in a relation is called the Degree of the relation. For example, relation GUARDIAN with four attributes is a relation of degree 4.

- v) **CARDINALITY:** The number of tuples in a relation is called the Cardinality of the relation. For example, the cardinality of relation GUARDIAN is 5 as there are 5 tuples in the table.

#### **8.4.1 Three Important Properties of a Relation**

In relational data model, following three properties are observed with respect to a relation which makes a relation different from a data file or a simple table.

**Property 1:** imposes following rules on an attribute of the relation.

- Each attribute in a relation has a unique name.
- Sequence of attributes in a relation is immaterial.

**Property 2:** governs following rules on a tuple of a relation.

- Each tuple in a relation is distinct. For example, data values in no two tuples of relation ATTENDANCE can be identical for all the attributes. Thus, each tuple of a relation must be uniquely identified by its contents.
- Sequence of tuples in a relation is immaterial. The tuples are not considered to be ordered, even though they appear to be in tabular form.

**Property 3:** imposes following rules on the state of a relation.

- All data values in an attribute must be from the same domain (same data type).
- Each data value associated with an attribute must be atomic (cannot be further divisible into meaningful subparts). For example, GPhone of relation GUARDIAN has ten digit numbers which is indivisible.
- No attribute can have many data values in one tuple. For example, Guardian cannot specify multiple contact numbers under GPhone attribute.
- A special value “NULL” is used to represent values that are unknown or non-applicable to certain attributes. For example, if a guardian does not share his or her contact number with the school authorities, then GPhone is set to NULL (data unknown).

#### **8.5 KEYS IN A RELATIONAL DATABASE**

The tuples within a relation must be distinct. It means no two tuples in a table should have same value for all attributes. That is, there should be at least one attribute

in which data are distinct (unique) and not NULL. That way, we can uniquely distinguish each tuple of a relation. So, relational data model imposes some restrictions or constraints on the values of the attributes and how the contents of one relation be referred through another relation. These restrictions are specified at the time of defining the database through different types of keys as given below:

### **8.5.1 Candidate Key**

A relation can have one or more attributes that takes distinct values. Any of these attributes can be used to uniquely identify the tuples in the relation. Such attributes are called candidate keys as each of them are candidates for the primary key.

As shown in Figure 8.4, the relation GUARDIAN has four attributes out of which GUID and GPhone always take unique values. No two guardians will have same phone number or same GGUID. Hence, these two attributes are the candidate keys as they both are candidates for primary key.

### **8.5.2 Primary Key**

Out of one or more candidate keys, the attribute chosen by the database designer to uniquely identify the tuples in a relation is called the primary key of that relation. The remaining attributes in the list of candidate keys are called the alternate keys.

In the relation GUARDIAN, suppose GGUID is chosen as primary key, then GPhone will be called the alternate key.

### **8.5.3 Composite Primary Key**

If no single attribute in a relation is able to uniquely distinguish the tuples, then more than one attribute are taken together as primary key. Such primary key consisting of more than one attribute is called Composite Primary key.

In relation ATTENDANCE, Roll Number cannot be used as primary key as roll number of same student will appear in another row for a different date. Similarly, in relation Attendance, AttendanceDate cannot be used as primary key because same date is repeated for each roll number. However combination of these two attributes RollNumber and AttendanceDate together would always have unique value in ATTENDANCE table as on any working day, of a student would be

marked attendance only once. Hence {RollNumber, AttendanceDate} will make the of ATTENDANCE relation composite primary key.

#### 8.5.4 Foreign Key

A foreign key is used to represent the relationship between two relations. A foreign key is an attribute whose value is derived from the primary key of another relation. This means that any attribute of a relation (referencing), which is used to refer contents from another (referenced) relation, becomes foreign key if it refers to the primary key of referenced relation. The referencing relation is called Foreign Relation. In some cases, foreign key can take NULL value if it is not the part of primary key of the foreign table. The relation in which the referenced primary key is defined is called primary relation or master relation.

In Figure 8.5, two foreign keys in Student Attendance database are shown using schema diagram where the foreign key is displayed as a directed arc (arrow) originating from it and ending at the corresponding attribute of the primary key of the referenced table. The underlined attributes make the primary key of that table.

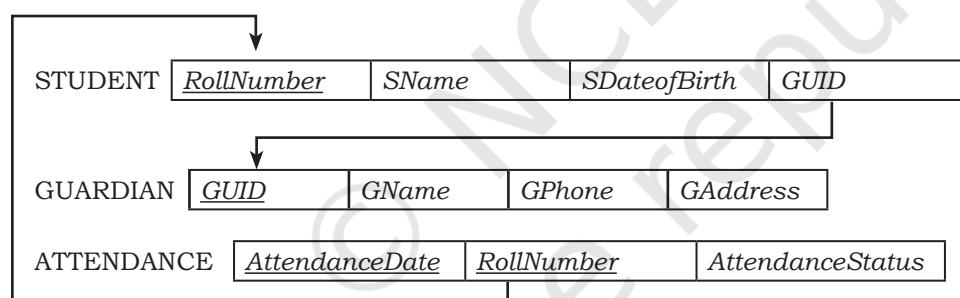


Figure 8.5: STUDENTATTENDANCE database with the primary and foreign keys

#### SUMMARY

- A file in a file system is a container to store data in a computer.
- File system suffers from Data Redundancy, Data Inconsistency, Data Isolation, Data Dependence and Controlled Data sharing.
- Database Management System (DBMS) is a software to create and manage databases. A database is a collection of tables.
- Database schema is the design of a database.
- A database constraint is a restriction on the type of data that can be inserted into the table.

- Database schema and database constraints are stored in database catalog.
- The snapshot of the database at any given time is the database instance.
- A query is a request to a database for information retrieval and data manipulation (insertion, deletion or update). It is written in Structured Query Language (SQL).
- Relational DBMS (RDBMS) is used to store data in related tables. Rows and columns of a table are called tuples and attributes respectively. A table is referred to as a relation.
- Destructions on data stored in a RDBMS is applied by use of keys such as Candidate Key, Primary Key, Composite Primary Key, Foreign Key.
- Primary key in a relation is used for unique identification of tuples.
- Foreign key is used to relate two tables or relations.
- Each column in a table represents a feature (attribute) of a record. Table stores the information for an entity whereas a row represents a record.
- Each row in a table represents a record. A tuple is a collection of attribute values that makes a record unique.
- A tuple is a unique entity whereas attribute values can be duplicate in the table.
- SQL is the standard language for RDBMS systems like MySQL.



## EXERCISE

1. Give the terms for each of the following:
  - a) Collection of logically related records.
  - b) DBMS creates a file that contains description about the data stored in the database.
  - c) Attribute that can uniquely identify the tuples in a relation.
  - d) Special value that is stored when actual data value is unknown for an attribute.
  - e) An attribute which can uniquely identify tuples of the table but is not defined as primary key of the table.
  - f) Software that is used to create, manipulate and maintain a relational database.
2. Why foreign keys are allowed to have NULL values? Explain with an example.
3. Differentiate between:
  - a) Database state and database schema
  - b) Primary key and foreign key
  - c) Degree and cardinality of a relation

4. Compared to a file system, how does a database management system avoid redundancy in data through a database?
5. What are the limitations of file system that can be overcome by a relational DBMS?
6. A school has a rule that each student must participate in a sports activity. So each one should give only one preference for sports activity. Suppose there are five students in a class, each having a unique roll number. The class representative has prepared a list of sports preferences as shown below. Answer the following:

**Table: Sports Preferences**

| <b>Roll_no</b> | <b>Preference</b> |
|----------------|-------------------|
| 9              | Cricket           |
| 13             | Football          |
| 17             | Badminton         |
| 17             | Football          |
| 21             | Hockey            |
| 24             | NULL              |
| NULL           | Kabaddi           |

- a) Roll no 24 may not be interested in sports. Can a NULL value be assigned to that student's preference field?
- b) Roll no 17 has given two preferences in sports. Which property of relational DBMC is violated here? Can we use any constraint or key in the relational DBMS to check against such violation, if any?
- c) Kabaddi was not chosen by any student. Is it possible to have this tuple in the Sports Preferences relation?
7. In another class having 2 sections, the two respective class representatives have prepared 2 separate Sports Preferences tables, as shown below:

Sports preference of section 1 (arranged on roll number column)

**Table: Sports Preferences**

| <b>Roll_no</b> | <b>Sports</b> |
|----------------|---------------|
| 9              | Cricket       |
| 13             | Football      |
| 17             | Badminton     |
| 21             | Hockey        |
| 24             | Cricket       |

Sports preference of section 2 (arranged on Sports name column, and column order is also different)

**Table: Sports Preferences**

| Sports    | Roll_no |
|-----------|---------|
| Badminton | 17      |
| Cricket   | 9       |
| Cricket   | 24      |
| Football  | 13      |
| Hockey    | 21      |

Are the states of both the relations equivalent? Justify.

8. The school canteen wants to maintain records of items available in the school canteen and generate bills when students purchase any item from the canteen. The school wants to create a canteen database to keep track of items in the canteen and the items purchased by students. Design a database by answering the following questions:
  - a) To store each item name along with its price, what relation should be used? Decide appropriate attribute names along with their data type. Each item and its price should be stored only once. What restriction should be used while defining the relation?
  - b) In order to generate bill, we should know the quantity of an item purchased. Should this information be in a new relation or a part of the previous relation? If a new relation is required, decide appropriate name and data type for attributes. Also, identify appropriate primary key and foreign key so that the following two restrictions are satisfied:
    - i) The same bill cannot be generated for different orders.
    - ii) Bill can be generated only for available items in the canteen.
  - c) The school wants to find out how many calories students intake when they order an item. In which relation should the attribute ‘calories’ be stored?
9. An organisation wants to create a database EMP-DEPENDENT to maintain following details about its employees and their dependent.

**EMPLOYEE(AadharNumber, Name, Address,  
Department, EmployeeID)**

**DEPENDENT(EmployeeID, DependentName,  
Relationship)**

- a) Name the attributes of EMPLOYEE, which can be used as candidate keys.
- b) The company wants to retrieve details of dependent of a particular employee. Name the tables and the key which are required to retrieve this detail.

- c) What is the degree of EMPLOYEE and DEPENDENT relation?
10. School uniform is available at M/s Sheetal Private Limited. They have maintained SCHOOL\_UNIFORM Database with two relations viz. UNIFORM and COST. The following figure shows database schema and its state.

| School Uniform Database    |                       |          |        |
|----------------------------|-----------------------|----------|--------|
| Attributes and Constraints |                       |          |        |
| Table: UNIFORM             |                       |          |        |
| Attribute                  | UCode                 | UName    | UColor |
| Constraints                | Primary Key           | Not Null | -      |
| Table: COST                |                       |          |        |
| Attribute                  | UCode                 | Size     | Price  |
| Constraints                | Composite Primary Key | >0       |        |
| Table: UNIFORM             |                       |          |        |
| UCode                      | UName                 | UColor   |        |
| 1                          | Shirt                 | White    |        |
| 2                          | Pant                  | Grey     |        |
| 3                          | Skirt                 | Grey     |        |
| 4                          | Tie                   | Blue     |        |
| 5                          | Socks                 | Blue     |        |
| 6                          | Belt                  | Blue     |        |

| Table: COST |      |            |
|-------------|------|------------|
| UCode       | Size | COST Price |
| 1           | M    | 500        |
| 1           | L    | 580        |
| 1           | XL   | 620        |
| 2           | M    | 810        |
| 2           | L    | 890        |
| 2           | XL   | 940        |
| 3           | M    | 770        |
| 3           | L    | 830        |
| 3           | XL   | 910        |
| 4           | S    | 150        |
| 4           | L    | 170        |
| 5           | S    | 180        |
| 5           | L    | 210        |
| 6           | M    | 110        |
| 6           | L    | 140        |
| 6           | XL   | 160        |

- a) Can they insert the following tuples to the UNIFORM Relation? Give reasons in support of your answer.
- i) 7, Handkerchief, NULL
  - ii) 4, Ribbon, Red
  - iii) 8, NULL, White
- b) Can they insert the following tuples to the COST Relation? Give reasons in support of your answer.
- i) 7, S, 0
  - ii) 9, XL, 100
11. In a multiplex, movies are screened in different auditoriums. One movie can be shown in more than one auditorium. In order to maintain the record of movies, the multiplex maintains a relational database consisting of two relations viz. MOVIE and AUDI respectively as shown below:

**Movie (Movie\_ID, MovieName, ReleaseDate)**

**Audi (Audi\_No, Movie\_ID, Seats, ScreenType, TicketPrice)**

- a) Is it correct to assign Movie\_ID as the primary key in the MOVIE relation? If no, then suggest an appropriate primary key.
- b) Is it correct to assign AudiNo as the primary key in the AUDI relation? If no, then suggest appropriate primary key.
- c) Is there any foreign key in any of these relations?

| Student Project Database |       |       |         |                 |
|--------------------------|-------|-------|---------|-----------------|
| Table: STUDENT           |       |       |         |                 |
| Roll No                  | Name  | Class | Section | Registration_ID |
| 11                       | Mohan | XI    | 1       | IP-101-15       |
| 12                       | Sohan | XI    | 2       | IP-104-15       |
| 21                       | John  | XII   | 1       | CS-103-14       |
| 22                       | Meena | XII   | 2       | CS-101-14       |
| 23                       | Juhi  | XII   | 2       | CS-101-10       |

| Table: PROJECT |                    |                |
|----------------|--------------------|----------------|
| ProjectNo      | PName              | SubmissionDate |
| 101            | Airline Database   | 12/01/2018     |
| 102            | Library Database   | 12/01/2018     |
| 103            | Employee Database  | 15/01/2018     |
| 104            | Student Database   | 12/01/2018     |
| 105            | Inventory Database | 15/01/2018     |
| 106            | Railway Database   | 15/01/2018     |

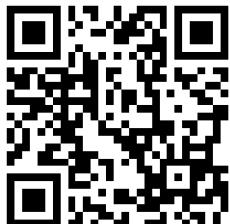
| Table: PROJECT ASSIGNED |           |
|-------------------------|-----------|
| Registration_ID         | ProjectNo |
| IP-101-15               | 101       |
| IP-104-15               | 103       |
| CS-103-14               | 102       |
| CS-101-14               | 105       |
| CS-101-10               | 104       |

12. For the above given database STUDENT-PROJECT, answer the following:
- a) Name primary key of each table.
  - b) Find foreign key(s) in table PROJECT-ASSIGNED.
  - c) Is there any alternate key in table STUDENT? Give justification for your answer.
  - d) Can a user assign duplicate value to the field RollNo of STUDENT table? Justify.
13. For the above given database STUDENT-PROJECT, can we perform the following operations?
- a) Insert a student record with missing roll number value.
  - b) Insert a student record with missing registration number value.
  - c) Insert a project detail without submission-date.
  - d) Insert a record with registration ID IP-101-19 and ProjectNo 206 in table PROJECT-ASSIGNED.

# Chapter

9

# Structured Query Language (SQL)



12130CH09

## In this Chapter

- » *Introduction*
- » *Structured Query Language (SQL)*
- » *Data Types and Constraints in MySQL*
- » *SQL for Data Definition*
- » *SQL for Data Manipulation*
- » *SQL for Data Query*
- » *Data Updation and Deletion*
- » *Functions in SQL*
- » *GROUP BY Clause in SQL*
- » *Operations on Relations*
- » *Using Two Relations in a Query*

“Any unique image that you desire probably already exists on the internet or in some database... The problem today is no longer how to create the right image, but how to find an already existing one.”

— Lev Manovich

## 9.1 INTRODUCTION

We have learnt about Relational Database Management Systems (RDBMS) and its purpose in the previous chapter. There are many RDBMS such as MySQL, Microsoft SQL Server, PostgreSQL, Oracle, etc. that allow us to create a database consisting of relations. These RDBMS also allow us to store, retrieve and manipulate data on that database through queries. In this chapter, we will learn how to create, populate and query databases using MySQL.

## 9.2 STRUCTURED QUERY LANGUAGE (SQL)

One has to write application programs to access data in case of a file system. However, for database management systems there are special kinds of languages called query language that can be used to access and manipulate data from the database. The Structured Query Language (SQL) is the most popular query language used by major relational

database management systems such as MySQL, ORACLE, SQL Server, etc.

### Activity 9.1

Find and list other types of databases other than RDBMS.

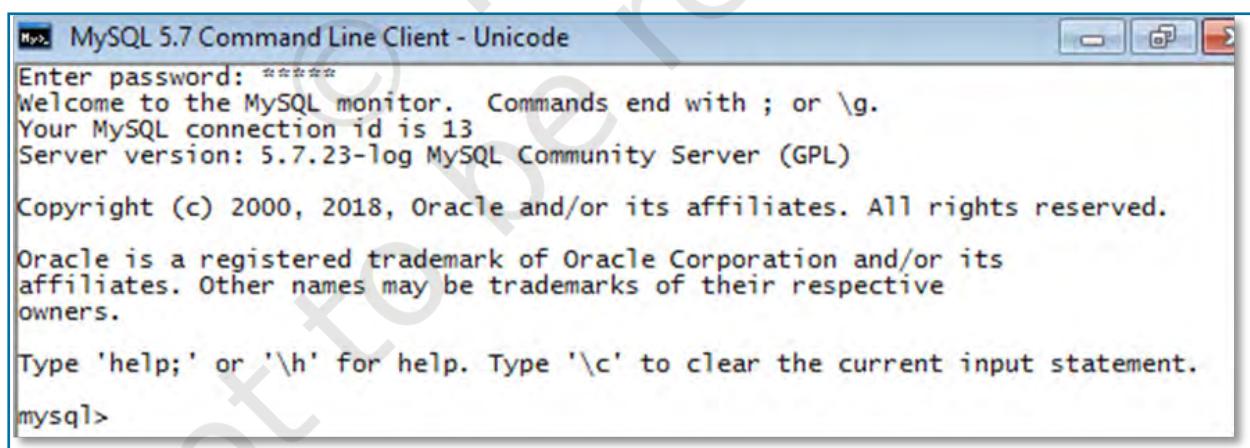


SQL is easy to learn as the statements comprise of descriptive English words and are not case sensitive. We can create and interact with a database using SQL easily. Benefit of using SQL is that we do not have to specify how to get the data from the database. Rather, we simply specify what is to be retrieved, and SQL does the rest. Although called a query language, SQL can do much more, besides querying. SQL provides statements for defining the structure of the data, manipulating data in the database, declaring constraints and retrieving data from the database in various ways, depending on our requirements.

In this chapter, we will use the StudentAttendance discussed in chapter 8 and create a database. We will also learn how to populate databases with data, manipulate data and retrieve data from a database through SQL queries.

#### 9.2.1 Installing MySQL

MySQL is an open source RDBMS software which can be easily downloaded from the official website <https://dev.mysql.com/downloads>. After installing MySQL, start MySQL service. The appearance of mysql> prompt (Figure 9.1) means that MySQL is ready to accept SQL statements.



```
MySQL 5.7 Command Line Client - Unicode
Enter password: *****
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 13
Server version: 5.7.23-log MySQL Community Server (GPL)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Figure 9.1: MySQL Shell

Following are some important points to be kept in mind while using SQL:

- SQL is case insensitive. For example, the column names ‘salary’ and ‘SALARY’ are the same for SQL.
- Always end SQL statements with a semicolon (;).
- To enter multiline SQL statements, we don’t write “;” after the first line. We press the Enter key to continue on the next line. The prompt mysql> then changes to “->”, indicating that statement is continued to the next line. After the last line, put “;” and press enter.

## 9.3 DATA TYPES AND CONSTRAINTS IN MySQL

We know that a database consists of one or more relations and each relation (table) is made up of attributes (column). Each attribute has a data type. We can also specify constraints for each attribute of a relation.

### 9.3.1 Data type of Attribute

Data type of an attribute indicates the type of data value that an attribute can have. It also decides the operations that can be performed on the data of that attribute. For example, arithmetic operations can be performed on numeric data but not on character data. Commonly used data types in MySQL are numeric types, date and time types, and string types as shown in Table 9.1.

**Table 9.1 Commonly used data types in MySQL**

| Data type   | Description                                                                                                                                                                                                                                                                                                                                                                               |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CHAR (n)    | Specifies character type data of length n where n could be any value from 0 to 255. CHAR is of fixed length, means, declaring CHAR (10) implies to reserve spaces for 10 characters. If data does not have 10 characters (e.g., ‘city’ has four characters), MySQL fills the remaining 6 characters with spaces padded on the right.                                                      |
| VARCHAR (n) | Specifies character type data of length where n could be any value from 0 to 65535. But unlike CHAR, VARCHAR(n) is a variable-length data type. That is, declaring VARCHAR (30) means a maximum of 30 characters can be stored but the actual allocated bytes will depend on the length of entered string. So ‘city’ in VARCHAR (30) will occupy space needed to store 4 characters only. |
| INT         | INT specifies an integer value. Each INT value occupies 4 bytes of storage. The range of unsigned values allowed in a 4 byte integer type are 0 to 4,294,967,295. For values larger than that, we have to use BIGINT, which occupies 8 bytes.                                                                                                                                             |
| FLOAT       | Holds numbers with decimal points. Each FLOAT value occupies 4 bytes.                                                                                                                                                                                                                                                                                                                     |
| DATE        | The DATE type is used for dates in 'YYYY-MM-DD' format. YYYY is the 4 digit year, MM is the 2 digit month and DD is the 2 digit date. The supported range is '1000-01-01' to '9999-12-31'.                                                                                                                                                                                                |

### Activity 9.2

What are the other data types supported in MySQL? Are there other variants of integer and float data type?



### **Think and Reflect**

Which two constraints when applied together will produce a Primary Key constraint?



### **9.3.2 Constraints**

Constraints are the certain types of restrictions on the data values that an attribute can have. Table 9.2 lists some of the commonly used constraints in SQL. They are used to ensure correctness of data. However, it is not mandatory to define constraints for each attribute of a table.

**Table 9.2 Commonly used SQL Constraints**

| Constraint   | Description                                                                                          |
|--------------|------------------------------------------------------------------------------------------------------|
| NOT NULL     | Ensures that a column cannot have NULL values where NULL means missing/unknown/not applicable value. |
| UNIQUE       | Ensures that all the values in a column are distinct/unique                                          |
| DEFAULT      | A default value specified for the column if no value is provided                                     |
| PRI MARY KEY | The column which can uniquely identify each row/record in a table.                                   |
| FOREIGN KEY  | The column which refers to value of an attribute defined as primary key in another table             |

## **9.4 SQL FOR DATA DEFINITION**

In order to be able to store data we need to first define the relation schema. Defining a schema includes creating a relation and giving name to a relation, identifying the attributes in a relation, deciding upon the datatype for each attribute and also specify the constraints as per the requirements. Sometimes, we may require to make changes to the relation schema also. SQL allows us to write statements for defining, modifying and deleting relation schemas. These are part of Data Definition Language (DDL).

We have already learned that the data are stored in relations or tables in a database. Hence, we can say that a database is a collection of tables. The Create statement is used to create a database and its tables (relations). Before creating a database, we should be clear about the number of tables the database will have, the columns (attributes) in each table along with the data type of each column, and its constraint, if any.

### **9.4.1 CREATE Database**

To create a database, we use the CREATE DATABASE statement as shown in the following syntax:

```
CREATE DATABASE databasename;
```

To create a database called StudentAttendance, we will type following command at mysql prompt.

```
mysql > CREATE DATABASE StudentAttendance;  
Query OK, 1 row affected (0.02 sec)
```

**Note:** In LINUX environment, names for database and tables are case-sensitive whereas in WINDOWS, there is no such differentiation. However, as a good practice, it is suggested to write database/table name in the same letter cases that were used at the time of their creation.

A DBMS can manage multiple databases on one computer. Therefore, we need to select the database that we want to use. To know the names of existing databases, we use the statement SHOW DATABASES. From the listed databases, we can select the database to be used. Once the database is selected, we can proceed with creating tables or querying data.

In order to use the StudentAttendance database, the following SQL statement is required.

```
mysql > USE StudentAttendance;  
Database changed
```

Initially, the created database is empty. It can be checked by using the show tables statement that lists names of all the tables within a database.

```
mysql > SHOW TABLES;  
Empty set (0.06 sec)
```

#### 9.4.2 CREATE Table

After creating a database StudentAttendance, we need to define relations in this database and specify attributes for each relation along with data type and constraint (if any) for each attribute. This is done using the CREATE TABLE statement.

*Syntax:*

```
CREATE TABLE tablename(  
attribute1 datatype constraint,  
attribute2 datatype constraint,  
:  
attributeN datatype constraint);
```

It is important to observe the following points with respect to the CREATE TABLE statement:

- The number of columns in a table defines the degree of that relation, which is denoted by N.
- Attribute name specifies the name of the column in the table.
- Datatype specifies the type of data that an attribute can hold.

#### Activity 9.3

Type the statement  
show database; Does  
it show the name of  
StudentAttendance  
database?



- Constraint indicates the restrictions imposed on the values of an attribute. By default, each attribute can take NULL values except for the primary key.

Let us identify data types of the attributes of table STUDENT along with their constraints (if any). Assuming maximum students in a class to be 100 and values of roll number in a sequence from 1 to 100, we know that 3 digits are sufficient to store values for the attribute RollNumber. Hence, data type INT is appropriate for this attribute. Total number of characters in a student name (SName) can differ. Assuming maximum characters in a name as 20, we use VARCHAR(20) for the SName column. Data type for the attribute SDateofBirth is DATE and supposing the school uses guardian's 12 digit Aadhaar number as GUID, we can declare GUID as CHAR (12) since Aadhaar number is of fixed length and we are not going to perform any mathematical operation on GUID.

Table 9.3, 9.4 and 9.5 shows the chosen data type and constraint for each attribute of the relations STUDENT, GUARDIAN and ATTENDANCE, respectively.

**Table 9.3 Data types and constraints for the attributes of relation STUDENT**

| Attribute Name | Data expected to be stored                     | Data type   | Constraint   |
|----------------|------------------------------------------------|-------------|--------------|
| RollNumber     | Numeric value consisting of maximum 3 digits   | INT         | PRI MARY KEY |
| SName          | Variant length string of maximum 20 characters | VARCHAR(20) | NOT NULL     |
| SDateofBirth   | Date value                                     | DATE        | NOT NULL     |
| GUID           | Numeric value consisting of 12 digits          | CHAR (12)   | FOREI GN KEY |

**Table 9.4 Data types and constraints for the attributes of relation GUARDIAN**

| Attribute Name | Data expected to be stored                          | Data type   | Constraint   |
|----------------|-----------------------------------------------------|-------------|--------------|
| GUID           | Numeric value consisting of 12 digit Aadhaar number | CHAR (12)   | PRI MARY KEY |
| GName          | Variant length string of maximum 20 characters      | VARCHAR(20) | NOT NULL     |
| GPhone         | Numeric value consisting of 10 digits               | CHAR(10)    | NULL UNI QUE |
| GAddress       | Variant length String of size 30 characters         | VARCHAR(30) | NOT NULL     |

**Table 9.5 Data types and constraints for the attributes of relation ATTENDANCE.**

| Attribute Name   | Data expected to be stored                   | Data type | Constraint                    |
|------------------|----------------------------------------------|-----------|-------------------------------|
| AttendanceDate   | Date value                                   | DATE      | PRI MARY KEY*                 |
| RollNumber       | Numeric value consisting of maximum 3 digits | INT       | PRI MARY KEY*<br>FOREI GN KEY |
| AttendanceStatus | 'P' for present and 'A' for absent           | CHAR(1)   | NOT NULL                      |

\*means part of composite primary key.

Once data types and constraints are identified, let us create tables without specifying constraints along with the attribute name for simplification. We will learn to incorporate constraints on attributes in Section 9.4.4.

*Example 9.1* Create table STUDENT.

```
mysql> CREATE TABLE STUDENT(
    -> RollNumber INT,
    -> SName VARCHAR(20),
    -> SDateofBirth DATE,
    -> GUID CHAR (12),
    -> PRIMARY KEY (RollNumber));
Query OK, 0 rows affected (0.91 sec)
```

**Note:** “,” is used to separate two attributes and each statement terminates with a semi-colon (;). The arrow (->) is an interactive continuation prompt. If we enter an unfinished statement, the SQL shell will wait for us to enter the rest of the statement.

### 9.4.3 Describe Table

We can view the structure of an already created table using the DESCRIBE statement or DESC statement.

*Syntax:*

```
DESCRIBE tablname;
mysql > DESCRIBE STUDENT;
```

| Field        | Type        | Null | Key | Default | Extra |
|--------------|-------------|------|-----|---------|-------|
| RollNumber   | int         | NO   | PRI | NULL    |       |
| SName        | varchar(20) | YES  |     | NULL    |       |
| SDateofBirth | date        | YES  |     | NULL    |       |
| GUID         | char(12)    | YES  |     | NULL    |       |

4 rows in set (0.06 sec)

We can use the SHOW TABLES statement to see the tables in the StudentAttendance database. So far, we have only the STUDENT table.

```
mysql > SHOW TABLES;
+-----+
| Tables_in_studentattendance |
+-----+
| student |
+-----+
1 row in set (0.00 sec)
```

### 9.4.4 ALTER Table

After creating a table, we may realise that we need to add/remove an attribute or to modify the datatype of an existing attribute or to add constraint in attribute. In

### Think and Reflect

Which datatype out of Char and Varchar will you prefer for storing contact number(mobile number)? Discuss.



### Activity 9.4

Create the other two relations GUARDIAN and ATTENDANCE as per data types given in Table 9.4 and 9.5 respectively, and view their structures. Do not add any constraint in these two tables.



all such cases, we need to change or alter the structure (schema) of the table by using the alter statement.

#### (A) Add primary key to a relation

Let us now alter the tables created in Activity 9.4. The following MySQL statement adds a primary key to the GUARDIAN relation:

```
mysql > ALTER TABLE GUARDIAN ADD PRIMARY KEY (GUID);  
Query OK, 0 rows affected (1.14 sec)  
Records: 0 Duplicates: 0 Warnings: 0
```

Now let us add the primary key to the ATTENDANCE relation. The primary key of this relation is a composite key made up of two attributes - AttendanceDate and RollNumber.

```
mysql > ALTER TABLE ATTENDANCE  
      -> ADD PRIMARY KEY(AttendanceDate,  
      RollNumber);  
Query OK, 0 rows affected (0.52 sec)  
Records: 0 Duplicates: 0 Warnings: 0
```

#### Activity 9.5

Add foreign key in the ATTENDANCE table (use Figure 9.1) to identify referencing and referenced tables.



#### Think and Reflect

Name foreign keys in table ATTENDANCE and STUDENT. Is there any foreign key in table GUARDIAN.



#### Syntax:

```
ALTER TABLE table_name ADD FOREIGN KEY(attribute  
name) REFERENCES referenced_table_name  
(attribute name);
```

Let us now add foreign key to the table STUDENT. Table 9.3 shows that attribute GUID (the referencing attribute) is a foreign key and it refers to attribute GUID (the referenced attribute) of table GUARDIAN. Hence, STUDENT is the referencing table and GUARDIAN is the referenced table as shown in Figure 8.4 in the previous chapter.

```
mysql > ALTER TABLE STUDENT  
      -> ADD FOREIGN KEY(GUID) REFERENCES  
      -> GUARDIAN(GUID);  
Query OK, 0 rows affected (0.75 sec)  
Records: 0 Duplicates: 0 Warnings: 0
```

#### (C) Add constraint **UNIQUE** to an existing attribute

In GUARDIAN table, the attribute GPhone has a constraint UNIQUE which means no two values in that column should be the same.

**Syntax:**

```
ALTER TABLE table_name ADD UNIQUE (attribute
name);
```

Let us now add the constraint UNIQUE with the attribute GPhone of the table GUARDIAN as shown at table 9.4.

```
mysql > ALTER TABLE GUARDIAN
      -> ADD UNIQUE(GPhone);
Query OK, 0 rows affected (0.44 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

**(D) Add an attribute to an existing table**

Sometimes, we may need to add an additional attribute in a table. It can be done using the ADD attribute statement as shown in the following Syntax:

```
ALTER TABLE table_name ADD attribute
name DATATYPE;
```

Suppose, the principal of the school has decided to award scholarship to some needy students for which income of the guardian must be known. But, the school has not maintained the income attribute with table GUARDIAN so far. Therefore, the database designer now needs to add a new attribute Income of data type INT in the table GUARDIAN.

```
mysql > ALTER TABLE GUARDIAN
      -> ADD income INT;
Query OK, 0 rows affected (0.47 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

**(E) Modify datatype of an attribute**

We can change data types of the existing attributes of a table using the following ALTER statement.

**Syntax:**

```
ALTER TABLE table_name MODIFY attribute DATATYPE;
```

Suppose we need to change the size of the attribute GAddress from VARCHAR(30) to VARCHAR(40) of the GUARDIAN table. The MySQL statement will be:

```
mysql > ALTER TABLE GUARDIAN
      -> MODIFY GAddress VARCHAR(40);
Query OK, 0 rows affected (0.11 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

**(F) Modify constraint of an attribute**

When we create a table, by default each attribute takes NULL value except for the attribute defined as primary key. We can change an attribute's constraint from NULL to NOT NULL using an alter statement.

### **Think and Reflect**

What are the minimum and maximum income values that can be entered in the income attribute given the data type is INT?



#### *Syntax:*

```
ALTER TABLE table_name MODIFY attribute DATATYPE  
NOT NULL;
```

**Note:** We have to specify the data type of the attribute along with constraint NOT NULL while using MODIFY.

To associate NOT NULL constraint with attribute SName of table STUDENT (table 9.3), we write the following MySQL statement:

```
mysql > ALTER TABLE STUDENT  
      -> MODIFY SName VARCHAR(20) NOT NULL;  
Query OK, 0 rows affected (0.47 sec)  
Records: 0 Duplicates: 0 Warnings: 0
```

#### **(G) Add default value to an attribute**

If we want to specify default value for an attribute, then use the following syntax:

```
ALTER TABLE table_name MODIFY attribute DATATYPE  
DEFAULT default_value;
```

To set default value of SDateofBirth of STUDENT to 15<sup>th</sup> May 2000, write the following statement:

```
mysql > ALTER TABLE STUDENT  
      -> MODIFY SDat eofBirth DATE DEFAULT '2000-05-  
          15';  
Query OK, 0 rows affected (0.08 sec)  
Records: 0 Duplicates: 0 Warnings: 0
```

**Note:** We have to specify the data type of the attribute along with DEFAULT while using MODIFY.

#### **(H) Remove an attribute**

Using ALTER, we can remove attributes from a table, as shown in the following syntax:

```
ALTER TABLE table_name DROP attribute;  
To remove the attribute income from table  
GUARDIAN (Table 9.4), write the following MySQL  
statement:
```

```
mysql > ALTER TABLE GUARDIAN DROP income;  
Query OK, 0 rows affected (0.42 sec)  
Records: 0 Duplicates: 0 Warnings: 0
```

#### **(I) Remove primary key from the table**

Sometime there may be a requirement to remove primary key constraint from the table. In that case, Alter table command can be used in the following way:

#### *Syntax:*

```
ALTER TABLE table_name DROP PRIMARY KEY;
```

To remove primary key of table GUARDIAN (Figure 9.4), write the following MySQL statement:

```
mysql > ALTER TABLE GUARDIAN DROP PRIMARY KEY;  
Query OK, 0 rows affected (0.72 sec)  
Records: 0 Duplicates: 0 Warnings: 0
```

**Note:** We have dropped the primary key from the GUARDIAN table, but each table should have a primary key to maintain uniqueness. Hence, we have to use the ADD statement with the Alter Table command to specify the primary key for the GUARDIAN table as shown in earlier examples.

#### 9.4.5 DROP Statement

Sometimes a table in a database or the database itself needs to be removed. We can use a DROP statement to remove a database or a table permanently from the system. However, one should be very cautious while using this statement as it cannot be undone.

*Syntax to drop a table:*

```
DROP TABLE table_name;
```

*Syntax to drop a database:*

```
DROP DATABASE database_name;
```

**Note:** Using the DROP statement to remove a database will ultimately remove all the tables within it.

### 9.5 SQL FOR DATA MANIPULATION

In the previous section, we created the database StudentAttendance having three relations STUDENT, GUARDIAN and ATTENDANCE. When we create a table, only its structure is created but the table has no data. To populate records in the table, INSERT statement is used. Also, table records can be deleted or updated using DELETE and UPDATE statements. These SQL statements are part of Data Manipulation Language (DML).

Data Manipulation using a database means either insertion of new data, removal of existing data or modification of existing data in the database

#### 9.5.1 INSERTION of Records

INSERT INTO statement is used to insert new records in a table. Its syntax is:

```
INSERT INTO tablename  
VALUES(value 1, value 2, ...);
```

Here, value 1 corresponds to attribute 1, value 2 corresponds to attribute 2 and so on. Note that we need not to specify attribute names in the insert statement if there are exactly the same numbers of values in the INSERT statement as the total number of attributes in the table.

**Caution:** While populating records in a table with foreign key, ensure that records in referenced tables are already populated.

Let us insert some records in the StudentAttendance database. We shall insert records in the GUARDIAN table first as it does not have any foreign key. A set of sample records for GUARDIAN table is shown in the given table (Table 9.6).

**Table 9.6 GUARDIAN Table**

| GUID         | GName           | GPhone     | GAddress                               |
|--------------|-----------------|------------|----------------------------------------|
| 444444444444 | Amit Ahuja      | 5711492685 | G-35, Ashok Vihar, Delhi               |
| 111111111111 | Baichung Bhutia | 3612967082 | Flat no. 5, Darjeeling Appt., Shimla   |
| 101010101010 | Himanshu Shah   | 4726309212 | 26/77, West Patel Nagar, Ahmedabad     |
| 333333333333 | Danny Dsouza    |            | S -13, Ashok Village, Daman            |
| 466444444666 | Sujata P.       | 3801923168 | HNO-13, B- block, Preet Vihar, Madurai |

The following insert statement adds the first record in the table:

```
mysql > INSERT INTO GUARDIAN
    -> VALUES (444444444444, 'Amit Ahuja',
    -> 5711492685, 'G-35, Ashok vihar, Del hi');
Query OK, 1 row affected (0.01 sec)
```

We can use the SQL statement `SELECT * from table_name` to view the inserted records. The `SELECT` statement will be explained in the next section.

```
mysql > SELECT * from GUARDIAN;
+-----+-----+-----+
| GUID | GName | Gphone | GAddress |
+-----+-----+-----+
| 444444444444 | Amit Ahuja | 5711492685 | G- 35, Ashok vihar, Del hi |
+-----+-----+-----+
1 row in set (0.00 sec)
```

If we want to insert values only for some of the attributes in a table (supposing other attributes having NULL or any other default value), then we shall specify the attribute names in which the values are to be inserted using the following syntax of `INSERT INTO` statement.

*Syntax:*

```
INSERT INTO tablename (column1, column2, ...)
VALUES (value1, value2, ...);
```

To insert the fourth record of Table 9.6 where `GPhone` is not given, we need to insert values in the other three fields (`GPhone` was set to `NULL` by default at the time of table creation). In this case, we have to specify the names of attributes in which we want to insert values. The values must be given in the same order in which attributes are written in `INSERT` statement.

### Activity 9.6

Write SQL statements to insert the remaining 3 rows of table 9.6 in table GUARDIAN.



```
mysql > INSERT INTO GUARDIAN(GUID, GName, GAddress)
-> VALUES (333333333333, 'Danny Dsouza',
-> 'S - 13, Ashok Village, Daman' );
Query OK, 1 row affected (0.03 sec)
```

**Note:** Text and date values must be enclosed in '' (single quotes).

```
mysql > SELECT * from GUARDIAN;
```

| GUID         | GName        | Gphone     | GAddress                     |
|--------------|--------------|------------|------------------------------|
| 333333333333 | Danny Dsouza | NULL       | S - 13, Ashok Village, Daman |
| 444444444444 | Ami t Ahuja  | 5711492685 | G- 35, Ashok vi har, Del hi  |

2 rows in set (0.00 sec)

Let us now insert the records given in Table 9.7 into the STUDENT table.

**Table 9.7 STUDENT Table**

| RollNumber | SName        | SDateofBirth | GUID         |
|------------|--------------|--------------|--------------|
| 1          | Atharv Ahuja | 2003-05-15   | 444444444444 |
| 2          | Daizy Bhutia | 2002-02-28   | 111111111111 |
| 3          | Taleem Shah  | 2002-02-28   |              |
| 4          | John Dsouza  | 2003-08-18   | 333333333333 |
| 5          | Ali Shah     | 2003-07-05   | 101010101010 |
| 6          | Manika P.    | 2002-03-10   | 466444444666 |

To insert the first record of Table 9.7, we write the following MySQL statement

```
mysql > INSERT INTO STUDENT
-> VALUES(1, 'Atharv Ahuj a', '2003-05-15',
444444444444);
Query OK, 1 row affected (0.11 sec)
OR
mysql > INSERT INTO STUDENT (Rol l Number, SName,
SDateofBi rth, GUID)
-> VALUES (1, 'Atharv Ahuj a', '2003-05-15',
444444444444);
Query OK, 1 row affected (0.02 sec)
```

#### Activity 9.7

Write SQL statements to insert the remaining 4 rows of table 9.7 in table STUDENT.



Recall that Date is stored in 'YYYY-MM-DD' format.

```
mysql > SELECT * from STUDENT;
```

| RollNumber | SName        | SDateofBirth | GUID         |
|------------|--------------|--------------|--------------|
| 1          | Atharv Ahuja | 2003-05-15   | 444444444444 |

1 row in set (0.00 sec)

Let us now insert the third record of Table 9.7 where GUID is NULL. Recall that GUID is foreign key of this table and thus can take NULL value. Hence, we can put NULL value for GUID and insert the record by using the following statement:

```

mysql> INSERT INTO STUDENT
-> VALUES(3, 'Taleem Shah', '2002-02-28', NULL);
Query OK, 1 row affected (0.05 sec)

mysql> SELECT * from STUDENT;
+-----+-----+-----+-----+
| RollNumber | SName | SDateofBirth | GUID |
+-----+-----+-----+-----+
| 1 | Atharv Ahuja | 2003-05-15 | 444444444444 |
| 3 | Taleem Shah | 2002-02-28 | NULL |
+-----+-----+-----+-----+
2 rows in set (0.00 sec)

```

We had to write NULL in the above insert statement because we are not mentioning the column names. Otherwise, we should mention the names of attributes along with the values if we need to insert data only for certain attributes, as shown in the following query:

```

mysql> INSERT INTO STUDENT (RollNumber, SName,
-> SDateofBirth) VALUES (3, 'Taleem Shah', '2002-02-
28');
Query OK, 1 row affected (0.05 sec)

```

### Think and Reflect

- Which of the two insert statement should be used when the order of data to be inserted are not known?
- Can we insert two records with the same roll number?



## 9.6 SQL FOR DATA QUERY

So far we have learnt how to create a database and how to store and manipulate data in them. We are interested in storing data in a database as it is easier to retrieve data in future from databases in whatever way we want. SQL provides efficient mechanisms to retrieve data stored in multiple tables in MySQL database (or any other RDBMS). The SQL statement SELECT is used to retrieve data from the tables in a database and is also called a query statement.

### 9.6.1 SELECT Statement

The SQL statement SELECT is used to retrieve data from the tables in a database and the output is also displayed in tabular form.

*Syntax:*

```

SELECT attribute1, attribute2, ...
FROM table_name
WHERE condition;

```

Here, attribute1, attribute2, ... are the column names of the table table\_name from which we want to retrieve data. The FROM clause is always written with SELECT clause as it specifies the name of the table from which data is to be retrieved. The WHERE clause is optional and is used to retrieve data that meet specified condition(s).

To select all the data available in a table, we use the following select statement:

```
SELECT * FROM table_name;
```

**Example 9.2** The following query retrieves the name and date of birth of student with roll number 1:

```
mysql> SELECT SName, SDateofBirth  
      -> FROM STUDENT  
      -> WHERE RollNumber = 1;  
+-----+-----+  
| SName | SDateofBirth |  
+-----+-----+  
| Atharv Ahuja | 2003-05-15 |  
+-----+  
1 row in set (0.03 sec)
```

### Think and Reflect

Think and list few examples from your daily life where storing the data in the database and querying the same can be helpful.



## 9.6.2 QUERYING using Database OFFICE

Organisations maintain databases to store data in the form of tables. Let us consider the database OFFICE of an organisation that has many related tables like EMPLOYEE, DEPARTMENT and so on. Every EMPLOYEE in the database is assigned to a DEPARTMENT and his/her Department number (DeptId) is stored as a foreign key in the table EMPLOYEE. Let us consider the relation 'EMPLOYEE' as shown in Table 9.8 and apply the SELECT statement to retrieve data:

**Table 9.8 Records to be inserted into the EMPLOYEE table**

| EmpNo | Ename    | Salary | Bonus | DeptId |
|-------|----------|--------|-------|--------|
| 101   | Aaliya   | 10000  | 234   | D02    |
| 102   | Kritika  | 60000  | 123   | D01    |
| 103   | Shabbir  | 45000  | 566   | D01    |
| 104   | Gurpreet | 19000  | 565   | D04    |
| 105   | Joseph   | 34000  | 875   | D03    |
| 106   | Sanya    | 48000  | 695   | D02    |
| 107   | Vergese  | 15000  |       | D01    |
| 108   | Nachaobi | 29000  |       | D05    |
| 109   | Daribha  | 42000  |       | D04    |
| 110   | Tanya    | 50000  | 467   | D05    |

### (A) Retrieve selected columns

The following query selects employee numbers of all the employees:

```
mysql> SELECT EmpNo FROM EMPLOYEE;
```

```
+-----+  
| EmpNo |  
+-----+  
| 101 |  
| 102 |
```

```

103
104
105
106
107
108
109
110
+-----+
10 rows in set (0.41 sec)

```

The following query selects the employee number and employee name of all the employees, we write:

```
mysql > SELECT EmpNo, Ename FROM EMPLOYEE;
```

| EmpNo | Ename    |
|-------|----------|
| 101   | Aaliya   |
| 102   | Kritika  |
| 103   | Shabbir  |
| 104   | Gurpreet |
| 105   | Joseph   |
| 106   | Sanya    |
| 107   | Vergese  |
| 108   | Nachaobi |
| 109   | Dari bha |
| 110   | Tanya    |

```
+-----+
10 rows in set (0.00 sec)
```

### (B) Renaming of columns

In case we want to rename any column while displaying the output, it can be done by using the alias 'AS'. The following query selects Employee name as Name in the output for all the employees:

```
mysql > SELECT Ename as Name FROM EMPLOYEE;
```

| Name     |
|----------|
| Aaliya   |
| Kritika  |
| Shabbir  |
| Gurpreet |
| Joseph   |
| Sanya    |
| Vergese  |
| Nachaobi |
| Dari bha |
| Tanya    |

```
+-----+
10 rows in set (0.00 sec)
```

*Example 9.3* Select names of all employees along with their annual income (calculated as Salary\*12). While displaying the query result, rename the column EName as Name

```
mysql > SELECT Ename as Name, Salary*12 FROM EMPLOYEE;
```

```
+-----+-----+
| Name      | Salary*12 |
+-----+-----+
Aaliya	120000
Kritika	720000
Shabbir	540000
Gurpreet	228000
Joseph	408000
Sanya	576000
Vergese	180000
Nachaobi	348000
Dari bha	504000
Tanya	600000
+-----+
10 rows in set (0.02 sec)
```

Observe that in the output, `Salary*12` is displayed as the column name for the Annual Income column. In the output table, we can use alias to rename that column as Annual Income as shown below:

```
mysql > SELECT Ename AS Name, Salary*12 AS 'Annual
Income'
```

```
-> FROM EMPLOYEE;
```

```
+-----+-----+
| Name      | Annual Income |
+-----+-----+
Aaliya	120000
Kritika	720000
Shabbir	540000
Gurpreet	228000
Joseph	408000
Sanya	576000
Vergese	180000
Nachaobi	348000
Dari bha	504000
Tanya	600000
+-----+
10 rows in set (0.00 sec)
```

**Note:** Annual Income will not be added as a new column in the database table. It is just for displaying the output of the query.

*If an aliased column name has space as in the case of Annual Income, it should be enclosed in quotes as 'Annual Income'*

### (C) Distinct Clause

By default, SQL shows all the data retrieved through query as output. However, there can be duplicate values. The `SELECT` statement when combined with `DISTINCT` clause, returns records without repetition (distinct records). For example, while retrieving a department number from employee relation, there can be duplicate values as many employees are assigned to the same department. To select unique department number for all the employees, we use `DISTINCT` as shown below:

```
mysql > SELECT DISTINCT DeptId FROM EMPLOYEE;
```

## NOTES

```
+-----+  
| DeptId |  
+-----+  
| D02   |  
| D01   |  
| D04   |  
| D03   |  
| D05   |  
+-----+  
5 rows in set (0.03 sec)
```

### (D) WHERE Clause

The WHERE clause is used to retrieve data that meet some specified conditions. In the OFFICE database, more than one employee can have the same salary. Following query gives distinct salaries of the employees working in the department number D01:

```
mysql > SELECT DISTINCT Salary  
      -> FROM EMPLOYEE  
      -> WHERE DeptId='D01';
```

As the column DeptId is of string type, its values are enclosed in quotes ('D01').

```
+-----+  
| Salary |  
+-----+  
| 60000 |  
| 45000 |  
| 15000 |  
+-----+  
3 rows in set (0.02 sec)
```

In the above example, = operator is used in the WHERE clause. Other relational operators (<, <=, >, >=, !=) can be used to specify such conditions. The logical operators AND, OR, and NOT are used to combine multiple conditions.

*Example 9.4* Display all the details of those employees of D04 department who earn more than 5000.

```
mysql > SELECT * FROM EMPLOYEE  
      -> WHERE Salary > 5000 AND DeptId = 'D04';  
+-----+-----+-----+-----+-----+  
| EmpNo | Ename    | Salary  | Bonus  | DeptId |  
+-----+-----+-----+-----+-----+  
| 104   | Gurpreet | 19000  | 565   | D04   |  
| 109   | Daribha  | 42000  | NULL   | D04   |  
+-----+-----+-----+-----+-----+  
2 rows in set (0.00 sec)
```

*Example 9.5* The following query selects records of all the employees except Aaliya.

```
mysql > SELECT * FROM EMPLOYEE  
      -> WHERE NOT Ename = 'Aaliya';  
+-----+-----+-----+-----+-----+
```

| EmpNo | Ename    | Salary | Bonus | DeptId |
|-------|----------|--------|-------|--------|
| 102   | Kritika  | 60000  | 123   | D01    |
| 103   | Shabbir  | 45000  | 566   | D01    |
| 104   | Gurpreet | 19000  | 565   | D04    |
| 105   | Joseph   | 34000  | 875   | D03    |
| 106   | Sanya    | 48000  | 695   | D02    |
| 107   | Vergese  | 15000  | NULL  | D01    |
| 108   | Nachaobi | 29000  | NULL  | D05    |
| 109   | Dari bha | 42000  | NULL  | D04    |
| 110   | Tanya    | 50000  | 467   | D05    |

9 rows in set (0.00 sec)

**Example 9.6** The following query selects the name and department number of all those employees who are earning salary between 20000 and 50000 (both values inclusive).

```
mysql > SELECT Ename, DeptId
-> FROM EMPLOYEE
-> WHERE Salary>=20000 AND Salary<=50000;
```

| Ename    | DeptId |
|----------|--------|
| Shabbir  | D01    |
| Joseph   | D03    |
| Sanya    | D02    |
| Nachaobi | D05    |
| Dari bha | D04    |
| Tanya    | D05    |

6 rows in set (0.00 sec)

```
SELECT * FROM EMPLOYEE
WHERE Salary > 5000 OR DeptId= 20;
```

The query in example 9.6 defines a range that can also be checked using a comparison operator BETWEEN, as shown below:

```
mysql > SELECT Ename, DeptId
-> FROM EMPLOYEE
-> WHERE Salary BETWEEN 20000 AND 50000;
```

| Ename    | DeptId |
|----------|--------|
| Shabbir  | D01    |
| Joseph   | D03    |
| Sanya    | D02    |
| Nachaobi | D05    |
| Dari bha | D04    |
| Tanya    | D05    |

6 rows in set (0.03 sec)

**Note:** The BETWEEN operator defines the range of values in which the column value must fall into, to make the condition true.

**Example 9.7** The following query selects details of all the employees who work in the departments having deptid D01, D02 or D04.

### Think and Reflect

What will happen if in the above query we write “Aaliya” as “AALIYA” or “aaliya” or “AaLIYA”? Will the query generate the same output or an error?



### Activity 9.8

Compare the output produced by the query in Example 9.6 and the output of the following query and differentiate between the OR and AND operators.



```
mysql > SELECT *
- > FROM EMPLOYEE
- > WHERE DeptId = 'D01' OR DeptId = 'D02' OR
DeptId = 'D04';
+-----+-----+-----+-----+-----+
| EmpNo | Ename   | Salary | Bonus | DeptId |
+-----+-----+-----+-----+-----+
| 101   | Aaliya  | 10000 | 234   | D02    |
| 102   | Kritika | 60000 | 123   | D01    |
| 103   | Shabbir | 45000 | 566   | D01    |
| 104   | Gurpreet | 19000 | 565   | D04    |
| 106   | Sanya   | 48000 | 695   | D02    |
| 107   | Vergese | 15000 | NULL  | D01    |
| 109   | Dari bha | 42000 | NULL  | D04    |
+-----+-----+-----+-----+-----+
7 rows in set (0.00 sec)
```

**(E) Membership operator IN**

The IN operator compares a value with a set of values and returns true if the value belongs to that set. The above query can be rewritten using IN operator as shown below:

```
mysql > SELECT * FROM EMPLOYEE
- > WHERE DeptId IN ('D01', 'D02', 'D04');
+-----+-----+-----+-----+-----+
| EmpNo | Ename   | Salary | Bonus | DeptId |
+-----+-----+-----+-----+-----+
| 101   | Aaliya  | 10000 | 234   | D02    |
| 102   | Kritika | 60000 | 123   | D01    |
| 103   | Shabbir | 45000 | 566   | D01    |
| 104   | Gurpreet | 19000 | 565   | D04    |
| 106   | Sanya   | 48000 | 695   | D02    |
| 107   | Vergese | 15000 | NULL  | D01    |
| 109   | Dari bha | 42000 | NULL  | D04    |
+-----+-----+-----+-----+-----+
7 rows in set (0.00 sec)
```

**Example 9.8** The following query selects details of all the employees except those working in department number D01 or D02.

```
mysql > SELECT * FROM EMPLOYEE
- > WHERE DeptId NOT IN('D01', 'D02');
+-----+-----+-----+-----+-----+
| EmpNo | Ename   | Salary | Bonus | DeptId |
+-----+-----+-----+-----+-----+
| 104   | Gurpreet | 19000 | 565   | D04    |
| 105   | Joseph   | 34000 | 875   | D03    |
| 108   | Nachaobi | 29000 | NULL  | D05    |
| 109   | Dari bha | 42000 | NULL  | D04    |
| 110   | Tanya   | 50000 | 467   | D05    |
+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

**Note:** Here we need to combine NOT with IN as we want to retrieve all records except with DeptId D01 and D02.

### (F) ORDER BY Clause

ORDER BY clause is used to display data in an ordered form with respect to a specified column. By default, ORDER BY displays records in ascending order of the specified column's values. To display the records in descending order, the DESC (means descending) keyword needs to be written with that column.

**Example 9.9** The following query selects details of all the employees in ascending order of their salaries.

```
mysql > SELECT * FROM EMPLOYEE
```

```
-> ORDER BY Salary;
```

| EmpNo | Ename    | Salary | Bonus | DeptId |
|-------|----------|--------|-------|--------|
| 101   | Aaliya   | 10000  | 234   | D02    |
| 107   | Vergese  | 15000  | NULL  | D01    |
| 104   | Gurpreet | 19000  | 565   | D04    |
| 108   | Nachaobi | 29000  | NULL  | D05    |
| 105   | Joseph   | 34000  | 875   | D03    |
| 109   | Dari bha | 42000  | NULL  | D04    |
| 103   | Shabbi r | 45000  | 566   | D01    |
| 106   | Sanya    | 48000  | 695   | D02    |
| 110   | Tanya    | 50000  | 467   | D05    |
| 102   | Kritika  | 60000  | 123   | D01    |

10 rows in set (0.05 sec)

**Example 9.10** Select details of all the employees in descending order of their salaries.

```
mysql > SELECT * FROM EMPLOYEE  
-> ORDER BY Salary DESC;
```

| EmpNo | Ename    | Salary | Bonus | DeptId |
|-------|----------|--------|-------|--------|
| 102   | Kritika  | 60000  | 123   | D01    |
| 110   | Tanya    | 50000  | 467   | D05    |
| 106   | Sanya    | 48000  | 695   | D02    |
| 103   | Shabbi r | 45000  | 566   | D01    |
| 109   | Dari bha | 42000  | NULL  | D04    |
| 105   | Joseph   | 34000  | 875   | D03    |
| 108   | Nachaobi | 29000  | NULL  | D05    |
| 104   | Gurpreet | 19000  | 565   | D04    |
| 107   | Vergese  | 15000  | NULL  | D01    |
| 101   | Aaliya   | 10000  | 234   | D02    |

10 rows in set (0.00 sec)

### (G) Handling NULL Values

SQL supports a special value called NULL to represent a missing or unknown value. For example, the Gphone column in the GUARDIAN table can have missing value for certain records. Hence, NULL is used to represent such unknown values. It is important to note that NULL

#### Activity 9.9

Execute the following 2 queries and find out what will happen if we specify two columns in the ORDER BY clause:

```
SELECT * FROM  
EMPLOYEE  
ORDER BY Salary,  
Bonus;
```

```
SELECT * FROM  
EMPLOYEE  
ORDER BY  
Salary, Bonus  
DESC;
```



is different from 0 (zero). Also, any arithmetic operation performed with NULL value gives NULL. For example:  $5 + \text{NULL} = \text{NULL}$  because NULL is unknown hence the result is also unknown. In order to check for NULL value in a column, we use IS NULL operator.

**Example 9.11** The following query selects details of all those employees who have not been given a bonus. This implies that the bonus column will be blank.

```
mysql > SELECT * FROM EMPLOYEE
- > WHERE Bonus IS NULL;
+-----+-----+-----+-----+-----+
| EmpNo | Ename   | Salary | Bonus | DeptId |
+-----+-----+-----+-----+-----+
| 107   | Vergese | 15000 | NULL  | D01   |
| 108   | Nachaobi| 29000 | NULL  | D05   |
| 109   | Dari bha | 42000 | NULL  | D04   |
+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

**Example 9.12** The following query selects names of all employees who have been given a bonus (i.e., Bonus is not null) and works in the department D01.

```
mysql > SELECT EName FROM EMPLOYEE
- > WHERE Bonus IS NOT NULL
- > AND DeptID = 'D01';
+-----+
| EName   |
+-----+
| Kritika |
| Shabbir |
+-----+
2 rows in set (0.00 sec)
```

#### (H) Substring pattern matching

Many a times we come across situations where we do not want to query by matching exact text or value. Rather, we are interested to find matching of only a few characters or values in column values. For example, to find out names starting with “T” or to find out pin codes starting with ‘60’. This is called substring pattern matching. We cannot match such patterns using = operator as we are not looking for an exact match. SQL provides a LIKE operator that can be used with the WHERE clause to search for a specified pattern in a column.

The LIKE operator makes use of the following two wild card characters:

- % (per cent)- used to represent zero, one, or multiple characters

- \_ (underscore)- used to represent exactly a single character

**Example 9.13** The following query selects details of all those employees whose name starts with 'K'.

```
mysql> SELECT * FROM EMPLOYEE
```

```
-> WHERE Ename like 'K%';
+-----+-----+-----+-----+
| EmpNo | Ename   | Salary | Bonus | DeptId |
+-----+-----+-----+-----+
| 102   | Kritika | 60000 | 123   | D01    |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

**Example 9.14** The following query selects details of all those employees whose name ends with 'a', and gets a salary more than 45000.

```
mysql> SELECT * FROM EMPLOYEE
-> WHERE Ename like '%a'
-> AND Salary > 45000;
```

```
+-----+-----+-----+-----+
| EmpNo | Ename   | Salary | Bonus | DeptId |
+-----+-----+-----+-----+
102	Kritika	60000	123	D01
106	Sanya	48000	695	D02
110	Tanya	50000	467	D05
+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

**Example 9.15** The following query selects details of all those employees whose name consists of exactly 5 letters and starts with any letter but has 'ANYA' after that.

```
mysql> SELECT * FROM EMPLOYEE
```

```
-> WHERE Ename like '_ANYA';
+-----+-----+-----+-----+
| EmpNo | Ename   | Salary | Bonus | DeptId |
+-----+-----+-----+-----+
| 106   | Sanya   | 48000 | 695   | D02    |
| 110   | Tanya   | 50000 | 467   | D05    |
+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

**Example 9.16** The following query selects names of all employees containing 'se' as a substring in name.

```
mysql> SELECT Ename FROM EMPLOYEE
```

```
-> WHERE Ename like '%se%';
+-----+
| Ename |
+-----+
| Joseph |
| Vergese |
+-----+
2 rows in set (0.00 sec)
```

### Think and Reflect

When we type first letter of a contact name in our contact list in our mobile phones all the names containing that character are displayed. Can you relate SQL statement with the process? List other real-life situations where you can visualise a SQL statement in operation.



**Example 9.17** The following query selects names of all employees containing 'a' as the second character.

```
mysql > SELECT EName FROM EMPLOYEE
- > WHERE Ename like '_a%';
+-----+
| EName      |
+-----+
| Aaliya     |
| Sanya      |
| Nachaobi   |
| Dari bha   |
| Tanya      |
+-----+
5 rows in set (0.00 sec)
```

## 9.7 DATA UPDATION AND DELETION

Updation and deletion of data are also part of SQL Data Manipulation Language (DML). In this section, we are going to apply these two data manipulation methods on the StudentAttendance database given in section 9.4.

### 9.7.1 Data Updation

We may need to make changes in the value(s) of one or more columns of existing records in a table. For example, we may require some changes in address, phone number or spelling of name, etc. The UPDATE statement is used to make such modifications in existing data.

*Syntax:*

```
UPDATE table_name
SET attribute1 = value1, attribute2 = value2, ...
WHERE condition;
```

STUDENT Table 9.7 has NULL value in GUID for the student with roll number 3. Suppose students with roll numbers 3 and 5 are siblings. Then, in the STUDENT table, we need to fill the GUID value for the student with roll number 3 as 101010101010. In order to update or change GUID of a particular row (record), we need to specify that record using WHERE clause, as shown below:

```
mysql > UPDATE STUDENT
- > SET GUID = 101010101010
- > WHERE RollNumber = 3;
Query OK, 1 row affected (0.06 sec) Rows matched: 1
Changed: 1 Warnings: 0
```

We can then verify the updated data using the statement SELECT \* FROM STUDENT.

**Caution :** If we miss the where clause in the UPDATE statement then the GUID of all the records will be changed to 101010101010.

We can also update values for more than one column using the UPDATE statement. Suppose, the guardian with GUID 466444444666 has requested to change Address to 'WZ - 68, Azad Avenue, Bijnour, MP' and Phone number to '4817362092'.

```
mysql > UPDATE GUARDIAN
      -> SET GAddress = 'WZ - 68, Azad Avenue,
      -> Bijnour, MP', GPhone = 9010810547
      -> WHERE GUID = 466444444666;
Query OK, 1 row affected (0.06 sec)
Rows matched: 1 Changed: 1 Warnings: 0
```

```
mysql > SELECT * FROM GUARDIAN ;
```

| GUID         | GName            | Gphone     | GAddress                              |
|--------------|------------------|------------|---------------------------------------|
| 444444444444 | Ami t Ahuja      | 5711492685 | G- 35, Ashok vi har, Delhi            |
| 111111111111 | Bai chung Bhutia | 3612967082 | Flat no. 5, Darjeeling Appt., Shi mla |
| 101010101010 | Hi manshu Shah   | 4726309212 | 26/77, West Patel Nagar, Ahmedabad    |
| 333333333333 | Danny Dsouza     | NULL       | S - 13, Ashok Village, Daman          |
| 466444444666 | Suj at a P.      | 3801923168 | WZ - 68, Azad Avenue, Bi j nour, MP   |

5 rows in set (0.00 sec)

### 9.7.2 Data Deletion

DELETE statement is used to delete/remove one or more records from a table.

*Syntax:*

```
DELETE FROM table_name
WHERE condition;
```

Suppose the student with roll number 2 has left the school. We can use the following MySQL statement to delete that record from the STUDENT table.

```
mysql > DELETE FROM STUDENT WHERE Roll Number = 2;
```

```
Query OK, 1 row affected (0.06 sec)
```

```
mysql > SELECT * FROM STUDENT ;
```

| Roll Number | SName        | SDateofBi rth | GUID         |
|-------------|--------------|---------------|--------------|
| 1           | Atharv Ahuja | 2003-05-15    | 444444444444 |
| 3           | Tal eem Shah | 2002-02-28    | 101010101010 |
| 4           | John Dsouza  | 2003-08-18    | 333333333333 |
| 5           | Ali Shah     | 2003-07-05    | 101010101010 |
| 6           | Manika P.    | 2002-03-10    | 466444444666 |

5 rows in set (0.00 sec)

**Caution:** Like UPDATE statement, we need to be careful to include the WHERE clause while using a DELETE statement to delete records in a table. Otherwise, all the records in the table will get deleted.

## 9.8 FUNCTIONS IN SQL

In this section, we will understand how to use single row functions, multiple row functions, group records based on some criteria, and working on multiple tables using SQL.

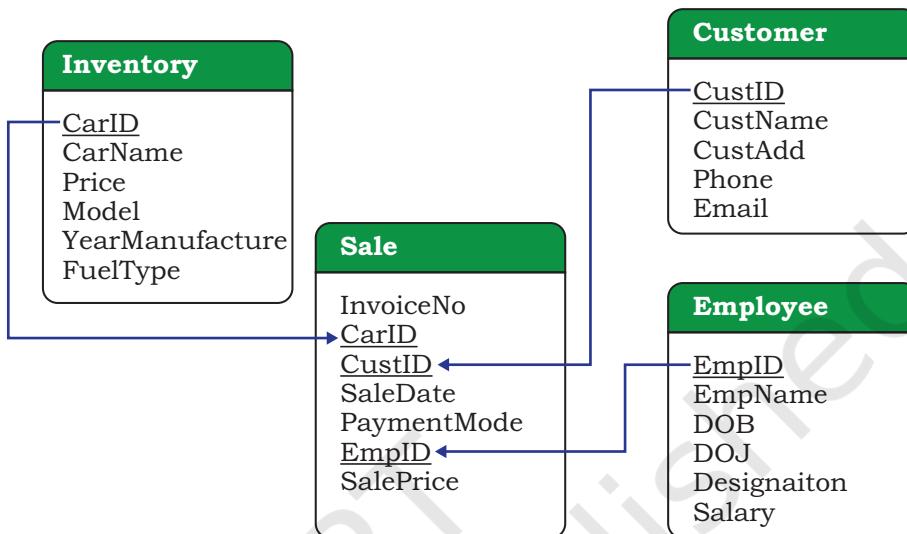


Figure 9.2: Schema diagram of database CARSHOWROOM

Let us create a database called CARSHOWROOM having the schema as shown in Figure 9.2. It has the following four relations:

- 1) INVENTORY: Stores name, price, model, year of manufacturing, and fuel type for each car in inventory of the showroom,
- 2) CUSTOMER: Stores customer id, name, address, phone number and email for each customer,
- 3) SALE: Stores the invoice number, car id, customer id, sale date, mode of payment, sales person's employee id and selling price of the car sold,
- 4) EMPLOYEE: Stores employee id, name, date of birth, date of joining, designation and salary of each employee in the showroom.

The records of the four relations are shown in Tables 9.9, 9.10, 9.11, and 9.12, respectively.

**Table 9.9 INVENTORY**

mysql > SELECT \* FROM INVENTORY;

| CarId | CarName | Price     | Model | YearManufacture | Fuel type |
|-------|---------|-----------|-------|-----------------|-----------|
| D001  | Dzire   | 582613.00 | LXI   | 2017            | Petrol    |

|      |         |            |           |  |      |        |
|------|---------|------------|-----------|--|------|--------|
| D002 | Dzi re  | 673112. 00 | VXI       |  | 2018 | Petrol |
| B001 | Bal eno | 567031. 00 | Sigma1. 2 |  | 2019 | Petrol |
| B002 | Bal eno | 647858. 00 | Delta1. 2 |  | 2018 | Petrol |
| E001 | EECO    | 355205. 00 | 5 STR STD |  | 2017 | CNG    |
| E002 | EECO    | 654914. 00 | CARE      |  | 2018 | CNG    |
| S001 | SWIFT   | 514000. 00 | LXI       |  | 2017 | Petrol |
| S002 | SWIFT   | 614000. 00 | VXI       |  | 2018 | Petrol |

8 rows in set (0.00 sec)

**Table 9.10 CUSTOMER**

mysql > SELECT \* FROM CUSTOMER;

| CustId | CustName      | CustAdd                  | Phone      | Email               |
|--------|---------------|--------------------------|------------|---------------------|
| C0001  | Amit Saha     | L- 10, Pitampura         | 4564587852 | amitsaha2@gmail.com |
| C0002  | Rehnuma       | J- 12, SAKET             | 5527688761 | rehnuma@hotmail.com |
| C0003  | Charvi Nayyar | 10/9, FF, Rohini         | 6811635425 | charvi123@yahoo.com |
| C0004  | Gurpreet      | A- 10/2, SF, Mayur Vihar | 3511056125 | gur_singh@yahoo.com |

4 rows in set (0.00 sec)

**Table 9.11 SALE**

mysql > SELECT \* FROM SALE;

| InvoiceNo | CarId | CustId | SaleDate   | PaymentMode  | EmpID | SalePrice  |
|-----------|-------|--------|------------|--------------|-------|------------|
| I00001    | D001  | C0001  | 2019-01-24 | Credit Card  | E004  | 613248. 00 |
| I00002    | S001  | C0002  | 2018-12-12 | Online       | E001  | 590321. 00 |
| I00003    | S002  | C0004  | 2019-01-25 | Cheque       | E010  | 604000. 00 |
| I00004    | D002  | C0001  | 2018-10-15 | Bank Finance | E007  | 659982. 00 |
| I00005    | E001  | C0003  | 2018-12-20 | Credit Card  | E002  | 369310. 00 |
| I00006    | S002  | C0002  | 2019-01-30 | Bank Finance | E007  | 620214. 00 |

6 rows in set (0.00 sec)

**Table 9.12 EMPLOYEE**

mysql > SELECT \* FROM EMPLOYEE;

| EmpID | EmpName   | DOB        | DOJ        | Designation  | Salary |
|-------|-----------|------------|------------|--------------|--------|
| E001  | Rushil    | 1994-07-10 | 2017-12-12 | Salesman     | 25550  |
| E002  | Sanjay    | 1990-03-12 | 2016-06-05 | Salesman     | 33100  |
| E003  | Zohar     | 1975-08-30 | 1999-01-08 | Peon         | 20000  |
| E004  | Arpit     | 1989-06-06 | 2010-12-02 | Salesman     | 39100  |
| E006  | Sanjucta  | 1985-11-03 | 2012-07-01 | Receptionist | 27350  |
| E007  | Mayank    | 1993-04-03 | 2017-01-01 | Salesman     | 27352  |
| E010  | Raj kumar | 1987-02-26 | 2013-10-23 | Salesman     | 31111  |

7 rows in set (0.00 sec)

We know that a function is used to perform some particular task and it returns zero or more values as a result. Functions are useful while writing SQL queries also. Functions can be applied to work on single or multiple records (rows) of a table. Depending on their application in one or multiple rows, SQL functions are

categorised as Single Row functions and Aggregate functions.

### 9.8.1 Single Row Functions

These are also known as Scalar functions. Single row functions are applied on a single value and return a single value. Figure 9.3 lists different single row functions under three categories — Numeric (Math), String, Date and Time.

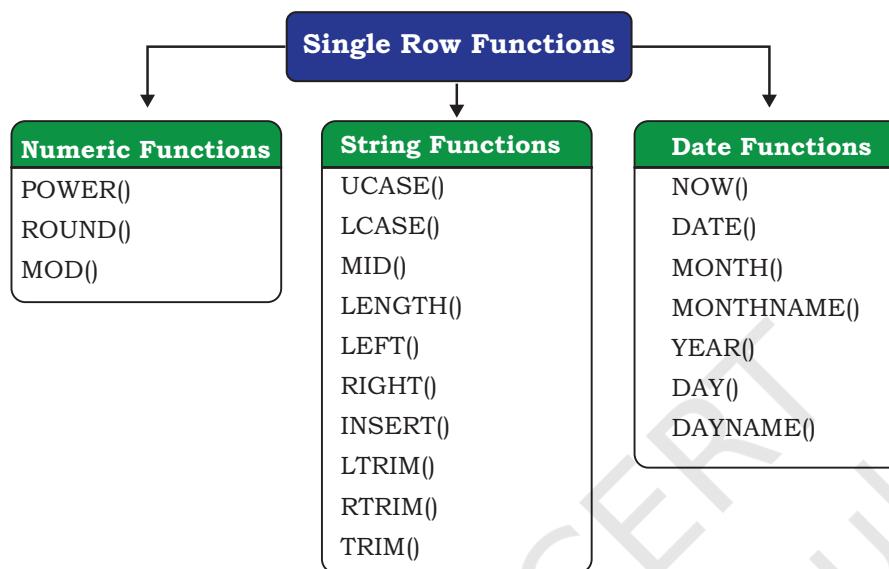


Figure 9.3: Three categories of single-row functions in SQL

Math Functions accept numeric value as input and return a numeric value as a result. String Functions accept character value as input and return either character or numeric values as output. Date and Time functions accept date and time value as input and return numeric or string or Date and Time as output.

#### (A) Math Functions

Three commonly used numeric functions are POWER(), ROUND() and MOD(). Their usage along with syntax is given in Table 9.13.

**Table 9.13 Math Functions**

| Function                                         | Description                                                                                                                      | Example with output                                                                                      |
|--------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------|
| POWER(X,Y)<br>can also be written as<br>POW(X,Y) | Calculates X to the power Y.                                                                                                     | mysql > SELECT POWER(2, 3);<br>Output:<br>8                                                              |
| ROUND(N,D)                                       | Rounds off number N to D number of decimal places.<br><u>Note:</u> If D=0, then it rounds off the number to the nearest integer. | mysql >SELECT ROUND(2912.564, 1);<br>Output:<br>2912.6<br>mysql > SELECT ROUND(283.2);<br>Output:<br>283 |
| MOD(A, B)                                        | Returns the remainder after dividing number A by number B.                                                                       | mysql > SELECT MOD(21, 2);<br>Output:<br>1                                                               |

**Example 9.18** In order to increase sales, suppose the car dealer decides to offer his customers to pay the total amount in 10 easy EMIs (equal monthly instalments). Assume that EMIs are required to be in multiples of 10000. For that, the dealer wants to list the CarID and Price along with the following data from the Inventory table:

- a) Calculate GST as 12 per cent of Price and display the result after rounding it off to one decimal place.

```
mysql > SELECT ROUND(12/100*Price, 1) "GST" FROM INVENTORY;
```

| GST     |
|---------|
| 69913.6 |
| 80773.4 |
| 68043.7 |
| 77743.0 |
| 42624.6 |
| 78589.7 |
| 61680.0 |
| 73680.0 |

8 rows in set (0.00 sec)

- b) Add a new column FinalPrice to the table inventory which will have the value as sum of Price and 12 per cent of the GST.

```
mysql > ALTER TABLE INVENTORY ADD(FinalPrice Numeric(10, 1));
Query OK, 8 rows affected (0.03 sec)
Records: 8  Duplicates: 0  Warnings: 0
```

```
mysql > UPDATE INVENTORY SET
FinalPrice=Price+Round(Price*12/100, 1);
Query OK, 8 rows affected (0.01 sec)
Rows matched: 8  Changed: 8  Warnings: 0
```

```
mysql > SELECT * FROM INVENTORY;
```

| CarId | CarName | Price     | Model     | YearManufacture | Fuel Type | Final Price |
|-------|---------|-----------|-----------|-----------------|-----------|-------------|
| D001  | Dzire   | 582613.00 | LXI       | 2017            | Petrol    | 652526.6    |
| D002  | Dzire   | 673112.00 | VXI       | 2018            | Petrol    | 753885.4    |
| B001  | Baleno  | 567031.00 | Sigma1.2  | 2019            | Petrol    | 635074.7    |
| B002  | Baleno  | 647858.00 | Delta1.2  | 2018            | Petrol    | 725601.0    |
| E001  | EECO    | 355205.00 | 5 STR STD | 2017            | CNG       | 397829.6    |
| E002  | EECO    | 654914.00 | CARE      | 2018            | CNG       | 733503.7    |
| S001  | SWIFT   | 514000.00 | LXI       | 2017            | Petrol    | 575680.0    |
| S002  | SWIFT   | 614000.00 | VXI       | 2018            | Petrol    | 687680.0    |

8 rows in set (0.00 sec)

- c) Calculate and display the amount to be paid each month (in multiples of 1000) which is to be calculated after dividing the FinalPrice of the car into 10 instalments.

### Activity 9.10

Using the table SALE of CARSHOWROOM database, write SQL queries for the following:

- a) Display the InvoiceNo and commission value rounded off to zero decimal places.
- b) Display the details of SALE where payment mode is credit card.



- d) After dividing the amount into EMIs, find out the remaining amount to be paid immediately, by performing modular division.

Following SQL query can be used to solve the above mentioned (c) and (d) problem:

```
mysql > SELECT CarId, Final Price, ROUND(Final Price - MOD(Final Price, 1000)/10, 0) "EMI",
MOD(Final Price, 10000) "Remaining Amount" FROM
INVENTORY;
```

| CarId | Final Price | EMI    | Remaining Amount |
|-------|-------------|--------|------------------|
| D001  | 652526.6    | 652474 | 2526.6           |
| D002  | 753885.4    | 753797 | 3885.4           |
| B001  | 635074.7    | 635067 | 5074.7           |
| B002  | 725601.0    | 725541 | 5601.0           |
| E001  | 397829.6    | 397747 | 7829.6           |
| E002  | 733503.7    | 733453 | 3503.7           |
| S001  | 575680.0    | 575612 | 5680.0           |
| S002  | 687680.0    | 687612 | 7680.0           |

8 rows in set (0.00 sec)

#### Example 9.19

- a) Let us now add a new column Commission to the SALE table. The column Commission should have a total length of 7 in which 2 decimal places to be there.

```
mysql > ALTER TABLE SALE ADD(Commission
Numeric(7, 2));
Query OK, 6 rows affected (0.34 sec)
Records: 6  Duplicates: 0  Warnings: 0
```

- b) Let us now calculate commission for sales agents as 12% of the SalePrice, Insert the values to the newly added column Commission and then display records of the table SALE where commission > 73000.

```
mysql > UPDATE SALE SET
Commission=12/100*SalePrice;
Query OK, 6 rows affected (0.06 sec)
Rows matched: 6  Changed: 6  Warnings: 0
```

```
mysql > SELECT * FROM SALE WHERE Commission > 73000;
```

| invoiceno | carid | custid | sal edate  | paymentmode  | empid | sale price | Commission |
|-----------|-------|--------|------------|--------------|-------|------------|------------|
| I00001    | D001  | C0001  | 2019-01-24 | Credit Card  | E004  | 613248.00  | 73589.64   |
| I00004    | D002  | C0001  | 2018-10-15 | Bank Finance | E007  | 659982.00  | 79198.84   |
| I00006    | S002  | C0002  | 2019-01-30 | Bank Finance | E007  | 620214.00  | 74425.68   |

3 rows in set (0.02 sec)

- c) Display InvoiceNo, SalePrice and Commission such that commission value is rounded off to 0.

```
mysql > SELECT InvoiceNo, SalePrice,
Round(Commission, 0) FROM SALE;
+-----+-----+-----+
| InvoiceNo | SalePrice | Round(Commission, 0) |
+-----+-----+-----+
I00001	613248.00	73590
I00002	590321.00	70839
I00003	604000.00	72480
I00004	659982.00	79198
I00005	369310.00	44317
I00006	620214.00	74426
+-----+-----+-----+
6 rows in set (0.00 sec)
```

### Activity 9.11

Using the table INVENTORY from CARSHOWROOM database, write sql queries for the following:

- Convert the CarMake to uppercase if its value starts with the letter 'B'.
- If the length of the car's model is greater than 4 then fetch the substring starting from position 3 till the end from attribute Model.



### (B) String Functions

String functions can perform various operations on alphanumeric data which are stored in a table. They can be used to change the case (uppercase to lowercase or vice-versa), extract a substring, calculate the length of a string and so on. String functions and their usage are shown in Table 9.14.

**Table 9.14 String Functions**

| Function                                                                                  | Description                                                                                                                                                                           | Example with output                                                                                                               |
|-------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------|
| UCASE(string)<br>OR<br>UPPER(string)                                                      | converts string into uppercase.                                                                                                                                                       | mysql > SELECT<br>UCASE("Informatics<br>Practices");<br>Output:<br>INFORMATICS PRACTICES                                          |
| LOWER(string)<br>OR<br>LCASE(string)                                                      | converts string into lowercase.                                                                                                                                                       | mysql > SELECT<br>LOWER("Informatics<br>Practices");<br>Output:<br>informatics practices                                          |
| MID(string, pos, n)<br>OR<br>SUBSTRING(string,<br>pos, n)<br>OR<br>SUBSTR(string, pos, n) | Returns a substring of size n starting from the specified position (pos) of the string. If n is not specified, it returns the substring from the position pos till end of the string. | mysql > SELECT<br>MID("Informatics", 3, 4);<br>Output:<br>orm<br><br>mysql > SELECT<br>MID('Informatics', 7);<br>Output:<br>atics |
| LENGTH(string)                                                                            | Return the number of characters in the specified string.                                                                                                                              | mysql > SELECT<br>LENGTH("Informatics");<br>Output:<br>11                                                                         |
| LEFT(string, N)                                                                           | Returns N number of characters from the left side of the string.                                                                                                                      | mysql > SELECT<br>LEFT("Computer", 4);<br>Output:<br>Comp                                                                         |

|                          |                                                                                                                                              |                                                                                                                                       |
|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------|
| RIGHT(string, N)         | Returns N number of characters from the right side of the string.                                                                            | <pre>mysql &gt; SELECT RIGHT("SCIENCE", 3); NCE</pre>                                                                                 |
| INSTR(string, substring) | Returns the position of the first occurrence of the substring in the given string. Returns 0, if the substring is not present in the string. | <pre>mysql &gt; SELECT INSTR("Informatics", "ma"); Output: 6</pre>                                                                    |
| LTRIM(string)            | Returns the given string after removing leading white space characters.                                                                      | <pre>mysql &gt; SELECT LENGTH("DELHI"), LENGTH(LTRIM("DELHI")); Output: +-----+-----+   7   5   +-----+ 1 row in set (0.00 sec)</pre> |
| RTRIM(string)            | Returns the given string after removing trailing white space characters.                                                                     | <pre>mysql &gt; SELECT LENGTH("PEN") LENGTH(RTRIM("PEN ")); Output: +-----+-----+   5   3   +-----+ 1 row in set (0.00 sec)</pre>     |
| TRIM(string)             | Returns the given string after removing both leading and trailing white space characters.                                                    | <pre>mysql &gt; SELECT LENGTH("MADAM ") LENGTH(TRIM("MADAM ")); Output: +-----+-----+   9   5   +-----+ 1 row in set (0.00 sec)</pre> |

### Activity 9.12

Using the table EMPLOYEE from CARSHOWROOM database, write SQL queries for the following:

- a) Display employee name and the last 2 characters of his EmpId.
- b) Display designation of employee and the position of character 'e' in designation, if present.



**Example 9.20** Let us use Customer relation shown in Table 9.10 to understand the working of string functions.

- a) Display customer name in lower case and customer email in upper case from table CUSTOMER.

```
mysql > SELECT LOWER(CustName), UPPER(Email) FROM
CUSTOMER;
```

| LOWER(CustName) | UPPER(Email)         |
|-----------------|----------------------|
| ami t saha      | AMITSAHA2@GMAIL.COM  |
| rehnuma         | REHNUMA@HOTMAIL.COM  |
| charvi nayyar   | CHARVI 123@YAHOO.COM |
| gurpreet        | GUR_SINGH@YAHOO.COM  |

4 rows in set (0.00 sec)

- b) Display the length of the email and part of the email from the email id before the character '@'. Note - Do not print '@'.

```
mysql > SELECT LENGTH(Email), LEFT(Email, INSTR(Email,
"@") - 1) FROM CUSTOMER;
```

```

+-----+-----+
| LENGTH(Email) | LEFT(Email, INSTR(Email, "@") - 1) |
+-----+-----+
|      19 | amitsaha2
|      19 | rehnuma
|      19 | charvi 123
|      19 | gur_si ngh
+-----+
4 rows in set (0.03 sec)

```

The function INSTR will return the position of “@” in the email address. So, to print email id without “@” we have to use position -1.

- c) Let us assume that four-digit area code is reflected in the mobile number starting from position number 3. For example, 1851 is the area code of mobile number 9818511338. Now, write the SQL query to display the area code of the customer living in Rohini.

```

mysql> SELECT MID(Phone, 3, 4) FROM CUSTOMER WHERE
CustAdd like '%Rohini%';
+-----+
| MID(Phone, 3, 4) |
+-----+
| 1163             |
+-----+
1 row in set (0.00 sec)

```

- d) Display emails after removing the domain name extension “.com” from emails of the customers.

```

mysql> SELECT TRIM(".com" from Email) FROM
CUSTOMER;

```

```

+-----+
| TRIM(".com" FROM Email) |
+-----+
| amitsaha2@gmai l
| rehnuma@hotmai l
| charvi 123@yahoo
| gur_si ngh@yahoo
+-----+
4 rows in set (0.00 sec)

```

- e) Display details of all the customers having yahoo emails only.

```

mysql> SELECT * FROM CUSTOMER WHERE Email LIKE "%yahoo%";

```

| CustID | CustName      | CustAdd                 | Phone      | Email                |
|--------|---------------|-------------------------|------------|----------------------|
| C0003  | Charvi Nayyar | 10/9, FF, Rohini        | 6811635425 | charvi 123@yahoo.com |
| C0004  | Gurpreet      | A-10/2, SF, Mayur Vihar | 3511056125 | gur_si ngh@yahoo.com |

2 rows in set (0.00 sec)

### (C) Date and Time Functions

There are various functions that are used to perform operations on date and time data. Some of the operations

### Activity 9.13

Using the table EMPLOYEE of CARSHOWROOM database, list the day of birth for all employees whose salary is more than 25000.



### Think and Reflect

Can we use arithmetic operators (+, -, \*, or /) on date functions?



include displaying the current date, extracting each element of a date (day, month and year), displaying day of the week and so on. Table 9.15 explains various date and time functions.

**Table 9.15 Date Functions**

| Function        | Description                                                   | Example with output                                            |
|-----------------|---------------------------------------------------------------|----------------------------------------------------------------|
| NOW()           | It returns the current system date and time.                  | mysql > SELECT NOW();<br>Output:<br>2019-07-11 19:41:17        |
| DATE()          | It returns the date part from the given date/time expression. | mysql > SELECT DATE(NOW());<br>Output:<br>2019-07-11           |
| MONTH(date)     | It returns the month in numeric form from the date.           | mysql > SELECT MONTH(NOW());<br>Output:<br>7                   |
| MONTHNAME(date) | It returns the month name from the specified date.            | mysql > SELECT MONTHNAME("2003-11-28");<br>Output:<br>November |
| YEAR(date)      | It returns the year from the date.                            | mysql > SELECT YEAR("2003-10-03");<br>Output:<br>2003          |
| DAY(date)       | It returns the day part from the date.                        | mysql > SELECT DAY("2003-03-24");<br>Output:<br>24             |
| DAYNAME(date)   | It returns the name of the day from the date.                 | mysql > SELECT DAYNAME("2019-07-11");<br>Output:<br>Thursday   |

*Example 9.21* Let us use the EMPLOYEE table of CARSHOWROOM database to illustrate the working of some of the date and time functions.

- a) Select the day, month number and year of joining of all employees.

```
mysql > SELECT DAY(DOJ), MONTH(DOJ), YEAR(DOJ)
FROM EMPLOYEE;
```

| DAY(DOJ) | MONTH(DOJ) | YEAR(DOJ) |
|----------|------------|-----------|
| 12       | 12         | 2017      |
| 5        | 6          | 2016      |
| 8        | 1          | 1999      |
| 2        | 12         | 2010      |
| 1        | 7          | 2012      |
| 1        | 1          | 2017      |
| 23       | 10         | 2013      |

7 rows in set (0.03 sec)

#### Activity 9.14

- a) Find sum of Sale Price of the cars purchased by the customer having ID C0001 from table SALE.  
  
b) Find the maximum and minimum commission from the SALE table.

- b) If the date of joining is not a Sunday, then display it in the following format "Wednesday, 26, November, 1979."

```
mysql> SELECT DAYNAME(DOJ), DAY(DOJ),
MONTHNAME(DOJ), YEAR(DOJ) FROM EMPLOYEE WHERE
DAYNAME(DOJ) != 'Sunday';
```

| DAYNAME(DOJ) | DAY(DOJ) | MONTHNAME(DOJ) | YEAR(DOJ) |
|--------------|----------|----------------|-----------|
| Tuesday      | 12       | December       | 2017      |
| Fri day      | 8        | January        | 1999      |
| Thursday     | 2        | December       | 2010      |
| Wednesday    | 23       | October        | 2013      |

4 rows in set (0.00 sec)

### 9.8.2 Aggregate Functions

Aggregate functions are also called Multiple Row functions. These functions work on a set of records as a whole and return a single value for each column of the records on which the function is applied. Table 9.16 shows the differences between single row functions and multiple row functions. Table 9.17 describes some of the aggregate functions along with their usage. Note that column must be of numeric type.

**Table 9.16 Differences between Single and Multiple Row Functions**

| Single Row Function                                                                                                                                                                                                                                                            | Multiple row function                                                                                                                                                                                                                                                                     |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ol style="list-style-type: none"> <li>It operates on a single row at a time.</li> <li>It returns one result per row.</li> <li>It can be used in Select, Where, and Order by clause.</li> <li>Math, String and Date functions are examples of single row functions.</li> </ol> | <ol style="list-style-type: none"> <li>It operates on groups of rows.</li> <li>It returns one result for a group of rows.</li> <li>It can be used in the select clause only.</li> <li>Max(), Min(), Avg(), Sum(), Count() and Count(*) are examples of multiple row functions.</li> </ol> |

**Table 9.17 Aggregate Functions in SQL**

| Function    | Description                                           | Example with output                                                         |
|-------------|-------------------------------------------------------|-----------------------------------------------------------------------------|
| MAX(column) | Returns the largest value from the specified column.  | <pre>mysql&gt; SELECT MAX(Price) FROM INVENTORY;</pre> Output:<br>673112.00 |
| MIN(column) | Returns the smallest value from the specified column. | <pre>mysql&gt; SELECT MIN(Price) FROM INVENTORY;</pre> Output:<br>355205.00 |

|               |                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                           |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| AVG(column)   | Returns the average of the values in the specified column.                                                                                                                                   | <b>mysql &gt; SELECT AVG(Price) FROM INVENTORY;</b><br>Output:<br>576091.625000                                                                                                                                                                                                                                                                                           |
| SUM(column)   | Returns the sum of the values for the specified column.                                                                                                                                      | <b>mysql &gt; SELECT SUM(Price) FROM INVENTORY;</b><br>Output:<br>4608733.00                                                                                                                                                                                                                                                                                              |
| COUNT(*)      | Returns the number of records in a table.<br><br>Note: In order to display the number of records that matches a particular criteria in the table, we have to use COUNT(*) with WHERE clause. | <b>mysql &gt; SELECT COUNT(*) from MANAGER;</b><br>+-----+<br>  count(*)  <br>+-----+<br>  4  <br>+-----+<br>1 row in set (0.00 sec)                                                                                                                                                                                                                                      |
| COUNT(column) | Returns the number of values in the specified column ignoring the NULL values.<br><br>Note:<br>In this example, let us consider a MANAGER table having two attributes and four records.      | <b>mysql &gt; SELECT * from MANAGER;</b><br>+-----+-----+<br>  MNO   MEMNAME  <br>+-----+-----+<br>  1   AMIT  <br>  2   KAVREET  <br>  3   KAVITA  <br>  4   NULL  <br>+-----+-----+<br>4 rows in set (0.00 sec)<br><br><b>mysql &gt; SELECT COUNT(MEMNAME) FROM MANAGER;</b><br>+-----+<br>  COUNT(MEMNAME)  <br>+-----+<br>  3  <br>+-----+<br>1 row in set (0.01 sec) |

### Example 9.22

- a) Display the total number of records from table INVENTORY having a model as VXI.

```
mysql > SELECT COUNT(*) FROM INVENTORY WHERE Model = "VXI";
+-----+
| COUNT(*) |
+-----+
| 2 |
+-----+
1 row in set (0.00 sec)
```

- b) Display the total number of different types of Models available from table INVENTORY.

```
mysql > SELECT COUNT(DISTINCT Model) FROM INVENTORY;
```

```

+-----+
| COUNT(DISTINCT MODEL) |
+-----+
|          6           |
+-----+
1 row in set (0.09 sec)

```

- c) Display the average price of all the cars with Model LXI from table INVENTORY.

```

mysql > SELECT AVG(Price) FROM INVENTORY WHERE
Model = "LXI";
+-----+
| AVG(Price)   |
+-----+
| 548306.500000 |
+-----+
1 row in set (0.03 sec)

```

### Activity 9.15

- a) List the total number of cars sold by each employee.
- b) List the maximum sale made by each employee.



## 9.9 GROUP BY CLAUSE IN SQL

At times we need to fetch a group of rows on the basis of common values in a column. This can be done using a group by clause. It groups the rows together that contains the same values in a specified column. We can use the aggregate functions (COUNT, MAX, MIN, AVG and SUM) to work on the grouped values. HAVING Clause in SQL is used to specify conditions on the rows with Group By clause.

Consider the SALE table from the CARSHOWROOM database:

```
mysql > SELECT * FROM SALE;
```

| Invoi ceNo | CarId | CustId | SaleDate   | PaymentMode   | EmpID | SalePrice | Commis sion |
|------------|-------|--------|------------|---------------|-------|-----------|-------------|
| I00001     | D001  | C0001  | 2019-01-24 | Credit Card   | E004  | 613248.00 | 73589.64    |
| I00002     | S001  | C0002  | 2018-12-12 | Onl i ne      | E001  | 590321.00 | 70838.52    |
| I00003     | S002  | C0004  | 2019-01-25 | Cheque        | E010  | 604000.00 | 72480.00    |
| I00004     | D002  | C0001  | 2018-10-15 | Bank Fi nance | E007  | 659982.00 | 79198.84    |
| I00005     | E001  | C0003  | 2018-12-20 | Credit Card   | E002  | 369310.00 | 44318.20    |
| I00006     | S002  | C0002  | 2019-01-30 | Bank Fi nance | E007  | 620214.00 | 74425.68    |

6 rows in set (0.11 sec)

CarID, CustID, SaleDate, PaymentMode, EmpID, SalePrice are the columns that can have rows with the same values in it. So, Group by clause can be used in these columns to find the number of records of a particular type (column), or to calculate the sum of the price of each car type.

*Example 9.23*

- a) Display the number of Cars purchased by each Customer from SALE table.

```
mysql> SELECT CustID, COUNT(*) "Number of Cars"
  FROM SALE GROUP BY CustID;
+-----+-----+
| CustID | Number of Cars |
+-----+-----+
C0001	2
C0002	2
C0003	1
C0004	1
+-----+-----+
4 rows in set (0.00 sec)
```

- b) Display the Customer Id and number of cars purchased if the customer purchased more than 1 car from SALE table.

```
mysql> SELECT CustID, COUNT(*) FROM SALE GROUP BY
CustID HAVING Count(*)>1;
+-----+-----+
| CustID | COUNT(*) |
+-----+-----+
| C0001  |      2 |
| C0002  |      2 |
+-----+-----+
2 rows in set (0.30 sec)
```

- c) Display the number of people in each category of payment mode from the table SALE.

```
mysql> SELECT PaymentMode, COUNT(PaymentMode)
  FROM SALE GROUP BY Paymentmode ORDER BY
Paymentmode;
+-----+-----+
| PaymentMode | Count(PaymentMode) |
+-----+-----+
Bank Finance	2
Cheque	1
Credit Card	2
Online	1
+-----+-----+
4 rows in set (0.00 sec)
```

- d) Display the PaymentMode and number of payments made using that mode more than once.

```
mysql> SELECT PaymentMode, Count(PaymentMode)
  FROM SALE GROUP BY Paymentmode HAVING COUNT(*)>1
ORDER BY Paymentmode;
+-----+-----+
| PaymentMode | Count(PaymentMode) |
+-----+-----+
| Bank Finance |          2 |
| Credit Card |          2 |
+-----+-----+
2 rows in set (0.00 sec)
```

## 9.10 OPERATIONS ON RELATIONS

### NOTES

We can perform certain operations on relations like Union, Intersection and Set Difference to merge the tuples of two tables. These three operations are binary operations as they work upon two tables. Note here that these operations can only be applied if both the relations have the same number of attributes and corresponding attributes in both tables have the same domain.

### 9.10.1 UNION ( $\cup$ )

This operation is used to combine the selected rows of two tables at a time. If some rows are same in both the tables, then result of the Union operation will show those rows only once. Figure 9.4 shows union of two sets.

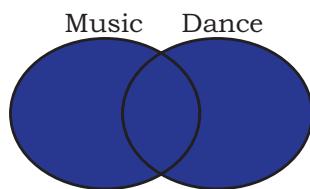


Figure 9.4: Union of two sets

Let us consider two relations DANCE and MUSIC shown in Tables 9.18 and 9.19 respectively.

**Table 9.18 DANCE**

| SNo | Name    | Class |
|-----|---------|-------|
| 1   | Aastha  | 7A    |
| 2   | Mahi ra | 6A    |
| 3   | Mohi t  | 7B    |
| 4   | Sanj ay | 7A    |

**Table 9.19 MUSIC**

| SNo | Name    | Class |
|-----|---------|-------|
| 1   | Mehak   | 8A    |
| 2   | Mahi ra | 6A    |
| 3   | Lavanya | 7A    |
| 4   | Sanj ay | 7A    |
| 5   | Abhay   | 8A    |

If we need the list of students participating in either of events, then we have to apply UNION operation (represented by symbol  $\cup$ ) on relations DANCE and MUSIC. The output of UNION operation is shown in Table 9.20.

**Table 9.20 DANCE  $\cup$  MUSIC**

| SNo | Name    | Class |
|-----|---------|-------|
| 1   | Aastha  | 7A    |
| 2   | Mahira  | 6A    |
| 3   | Mohit   | 7B    |
| 4   | Sanjay  | 7A    |
| 1   | Mehak   | 8A    |
| 3   | Lavanya | 7A    |
| 5   | Abhay   | 8A    |

**9.10.2 INTERSECT ( $\cap$ )**

Intersect operation is used to get the common tuples from two tables and is represented by symbol  $\cap$ . Figure 9.5 shows intersection of two sets.

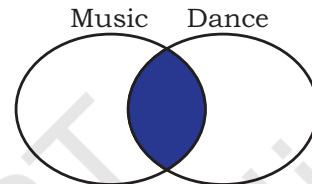


Figure 9.5: Intersection of two sets

Suppose, we have to display the list of students who are participating in both the events (DANCE and MUSIC), then intersection operation is to be applied on these two tables. The output of INTERSECT operation is shown in Table 9.21.

**Table 9.21 DANCE  $\cap$  MUSIC**

| SNo | Name   | Class |
|-----|--------|-------|
| 2   | Mahira | 6A    |
| 4   | Sanjay | 7A    |

**9.10.3 MINUS (-)**

This operation is used to get tuples/rows which are in the first table but not in the second table and the operation is represented by the symbol - (minus). Figure 9.6 shows minus operation (also called set difference) between two sets.

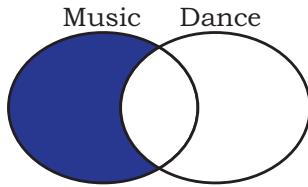


Figure 9.6: Difference of two sets

Suppose we want the list of students who are only participating in MUSIC and not in DANCE event. Then, we will use the MINUS operation, whose output is given in Table 9.22.

**Table 9.22 DANCE - MUSIC**

| SNo | Name    | Class |
|-----|---------|-------|
| 1   | Mehak   | 8A    |
| 3   | Lavanya | 7A    |
| 5   | Abhay   | 8A    |

#### 9.10.4 Cartesian Product (X)

Cartesian product operation combines tuples from two relations. It results in all pairs of rows from the two input relations, regardless of whether or not they have the same values on common attributes. It is denoted as 'X'.

The degree of the resulting relation is calculated as the sum of the degrees of both the relations under consideration. The cardinality of the resulting relation is calculated as the product of the cardinality of relations on which cartesian product is applied. Let us use the relations DANCE and MUSIC to show the output of cartesian product. Note that both relations are of degree 3. The cardinality of relations DANCE and MUSIC is 4 and 5 respectively. Applying cartesian product on these two relations will result in a relation of degree 6 and cardinality 20, as shown in Table 9.23.

**Table 9.23 DANCE X MUSIC**

| SNo | Name    | Class | SNo | Name    | Class |
|-----|---------|-------|-----|---------|-------|
| 1   | Aastha  | 7A    | 1   | Mehak   | 8A    |
| 2   | Mahi ra | 6A    | 1   | Mehak   | 8A    |
| 3   | Mohit   | 7B    | 1   | Mehak   | 8A    |
| 4   | Sanjay  | 7A    | 1   | Mehak   | 8A    |
| 1   | Aastha  | 7A    | 2   | Mahi ra | 6A    |
| 2   | Mahi ra | 6A    | 2   | Mahi ra | 6A    |

|   |         |    |   |         |    |
|---|---------|----|---|---------|----|
| 3 | Mohi t  | 7B | 2 | Mahi ra | 6A |
| 4 | Sanj ay | 7A | 2 | Mahi ra | 6A |
| 1 | Aastha  | 7A | 3 | Lavanya | 7A |
| 2 | Mahi ra | 6A | 3 | Lavanya | 7A |
| 3 | Mohi t  | 7B | 3 | Lavanya | 7A |
| 4 | Sanj ay | 7A | 3 | Lavanya | 7A |
| 1 | Aastha  | 7A | 4 | Sanj ay | 7A |
| 2 | Mahi ra | 6A | 4 | Sanj ay | 7A |
| 3 | Mohi t  | 7B | 4 | Sanj ay | 7A |
| 4 | Sanj ay | 7A | 4 | Sanj ay | 7A |
| 1 | Aastha  | 7A | 5 | Abhay   | 8A |
| 2 | Mahi ra | 6A | 5 | Abhay   | 8A |
| 3 | Mohi t  | 7B | 5 | Abhay   | 8A |
| 4 | Sanj ay | 7A | 5 | Abhay   | 8A |

20 rows in set (0.03 sec)

## 9.11 USING TWO RELATIONS IN A QUERY

Till now we have written queries in SQL using a single relation only. In this section, we will learn to write queries using two relations.

### 9.11.1 Cartesian product on two tables

From the previous section, we learnt that application of operator cartesian product on two tables results in a table having all combinations of tuples from the underlying tables. When more than one table is to be used in a query, then we must specify the table names by separating commas in the FROM clause, as shown in Example 9.24. On execution of such a query, the DBMS ( MySql ) will first apply cartesian product on specified tables to have a single table. The following query of example 9.24 applies cartesian product on the two tables DANCE and MUSIC:

#### Example 9.24

- a) Display all possible combinations of tuples of relations DANCE and MUSIC

```
mysql> SELECT * FROM DANCE, MUSIC;
```

As we are using SELECT \* in the query, the output will be the Table 9.23 having degree 6 and cardinality 20.

- b) From the all possible combinations of tuples of relations DANCE and MUSIC display only those rows such that the attribute name in both have the same value.

```
mysql> SELECT * FROM DANCE D, MUSIC M WHERE
D.Name = M.Name;
```

**Table 9.24 Tuples with same name**

| Sno | Name    | Class | Sno | Name    | class |
|-----|---------|-------|-----|---------|-------|
| 2   | Mahi ra | 6A    | 2   | Mahi ra | 6A    |
| 4   | Sanj ay | 7A    | 4   | Sanj ay | 7A    |

2 rows in set (0.00 sec)

**NOTES**

Note that in this query we have used table aliases (D for DANCE and M for MUSIC), just like column aliases (see Section 9.6.2) to refer to tables by shortened names. It is important to note that table alias is valid only for current query and the original table name cannot be used in the query if its alias is given in FROM clause.

### 9.11.2 JOIN on two tables

JOIN operation combines tuples from two tables on specified conditions. This is unlike cartesian product which make all possible combinations of tuples. While using the JOIN clause of SQL, we specify conditions on the related attributes of two tables within the FROM clause. Usually, such an attribute is the primary key in one table and foreign key in another table. Let us create two tables UNIFORM (UCode, UName, UColor) and COST (UCode, Size, Price) in the SchoolUniform database. UCode is Primary Key in table UNIFORM. UCode and Size is the Composite Key in table COST. Therefore, Ucode is a common attribute between the two tables which can be used to fetch the common data from both tables. Hence, we need to define Ucode as foreign key in the Price table while creating this table.

**Table 9.25 Uniform table**

| Ucode | Uname  | Ucolor |
|-------|--------|--------|
| 1     | Shi rt | Whi te |
| 2     | Pant   | Grey   |
| 3     | Ti e   | Blue   |

**Table 9.26 Cost table**

| Ucode | Size | Price |
|-------|------|-------|
| 1     | L    | 580   |
| 1     | M    | 500   |
| 2     | L    | 890   |
| 2     | M    | 810   |

**Example 9.25** List the UCode, UName, UColor, Size and Price of related tuples of tables UNIFORM and COST.

The given query may be written in three different ways as given below.

- a) Using condition in where clause

```
mysql > SELECT * FROM UNIFORM U, COST C WHERE
U.UCode = C.UCode;
```

**Table 9.27 Output of the query**

| UCode | UName | UCol or | Ucode | Size | Price |
|-------|-------|---------|-------|------|-------|
| 1     | Shirt | White   | 1     | L    | 580   |
| 1     | Shirt | White   | 1     | M    | 500   |
| 2     | Pant  | Grey    | 2     | L    | 890   |
| 2     | Pant  | Grey    | 2     | M    | 810   |

4 rows in set (0.08 sec)

As the attribute Ucode is in both tables, we need to use table alias to remove ambiguity. Hence, we have used qualifier with attribute UCode in SELECT and FROM clauses to indicate its scope.

- b) Explicit use of JOIN clause

```
mysql > SELECT * FROM UNIFORM U JOIN COST C ON
U.Ucode=C.Ucode;
```

The output of the query is same as shown in Table 9.26. In this query we have used JOIN clause explicitly along with condition in From clause. Hence no condition needs to be given in where clause.

- c) Explicit use of NATURAL JOIN clause

The output of queries (a) and (b) shown in Table 9.26 has a repetitive column Ucode having exactly the same values. This redundant column provides no additional information. There is an extension of JOIN operation called NATURAL JOIN which works similar to JOIN clause in SQL but removes the redundant attribute. This operator can be used to join the contents of two tables if there is one common attribute in both the tables. The above SQL query using NATURAL JOIN is shown below:

```
mysql > SELECT * FROM UNIFORM NATURAL JOIN COST;
```

| UCode | UName | UCol or | Size | Price |
|-------|-------|---------|------|-------|
| 1     | Shirt | White   | L    | 580   |
| 1     | Shirt | White   | M    | 500   |
| 2     | Pant  | Grey    | L    | 890   |

```

+---+---+-----+-----+-----+
| 2 | Pant | Grey | M   | 810 |
+---+---+-----+-----+-----+
4 rows in set (0.17 sec)

```

It is clear from the output that the result of this query is same as that of queries written in (a) and (b) except that the attribute Ucode appears only once.

Following are some of the points to be considered while applying JOIN operations on two or more relations:

- If two tables are to be joined on equality condition on the common attribute, then one may use JOIN with ON clause or NATURAL JOIN in FROM clause. If three tables are to be joined on equality condition, then two JOIN or NATURAL JOIN are required.
- In general, N-1 joins are needed to combine N tables on equality condition.
- With JOIN clause, we may use any relational operators to combine tuples of two tables.

## NOTES

### SUMMARY

- Database is a collection of related tables. MySQL is a ‘relational’ DBMS.
- DDL (Data Definition Language) includes SQL statements such as, Create table, Alter table and Drop table.
- DML (Data Manipulation Language) includes SQL statements such as, insert, select, update and delete.
- A table is a collection of rows and columns, where each row is a record and columns describe the feature of records.
- ALTER TABLE statement is used to make changes in the structure of a table like adding, removing or changing datatype of column(s).
- UPDATE statement is used to modify existing data in a table.
- WHERE clause in SQL query is used to enforce condition(s).
- DISTINCT clause is used to eliminate repetition and display the values only once.

- The BETWEEN operator defines the range of values inclusive of boundary values.
- The IN operator selects values that match any value in the given list of values.
- NULL values can be tested using IS NULL and IS NOT NULL.
- ORDER BY clause is used to display the result of a SQL query in ascending or descending order with respect to specified attribute values. By default, the order is ascending.
- LIKE operator is used for pattern matching. % and \_ are two wild card characters. The per cent (%) symbol is used to represent zero or more characters. The underscore (\_) symbol is used to represent a single character.
- A Function is used to perform a particular task and return a value as a result.
- Single Row functions work on a single row of the table and return a single value.
- Multiple Row functions work on a set of records as a whole and return a single value. Examples include COUNT, MAX, MIN, AVG and SUM.
- GROUP BY function is used to group rows of a table that contain the same values in a specified column.
- Join is an operation which is used to combine rows from two or more tables based on one or more common fields between them.



## EXERCISE

1. Answer the following questions:
  - a) Define RDBMS. Name any two RDBMS software.
  - b) What is the purpose of the following clauses in a select statement?
    - i) ORDER BY
    - ii) GROUP BY
  - c) Site any two differences between Single Row Functions and Aggregate Functions.
  - d) What do you understand by Cartesian Product?
  - e) Differentiate between the following statements:

- i) ALTER and UPDATE
  - ii) DELETE and DROP
  - f) Write the name of the functions to perform the following operations:
    - i) To display the day like “Monday”, “Tuesday”, from the date when India got independence.
    - ii) To display the specified number of characters from a particular position of the given string.
    - iii) To display the name of the month in which you were born.
    - iv) To display your name in capital letters.
2. Write the output produced by the following SQL statements:
- a) SELECT POW(2,3);
  - b) SELECT ROUND(342.9234,-1);
  - c) SELECT LENGTH("Informatics Practices");
  - d) SELECT YEAR("1979/11/26"),  
MONTH("1979/11/26"), DAY("1979/11/26"),  
MONTHNAME("1979/11/26");
  - e) SELECT LEFT("INDIA",3), RIGHT("Computer  
Science",4), MID("Informatics",3,4),  
SUBSTR("Practices",3);
3. Consider the following MOVIE table and write the SQL queries based on it.

| MovieID | MovieName     | Category  | ReleaseDate | ProductionCost | BusinessCost |
|---------|---------------|-----------|-------------|----------------|--------------|
| 001     | Hindi_Movie   | Musical   | 2018-04-23  | 124500         | 130000       |
| 002     | Tamil_Movie   | Action    | 2016-05-17  | 112000         | 118000       |
| 003     | English_Movie | Horror    | 2017-08-06  | 245000         | 360000       |
| 004     | Bengali_Movie | Adventure | 2017-01-04  | 72000          | 100000       |
| 005     | Telugu_Movie  | Action    | -           | 100000         | -            |
| 006     | Punjabi_Movie | Comedy    | -           | 30500          | -            |

- a) Display all the information from the Movie table.
- b) List business done by the movies showing only MovieID, MovieName and Total\_Earning. Total\_Earning to be calculated as the sum of ProductionCost and BusinessCost.
- c) List the different categories of movies.
- d) Find the net profit of each movie showing its MovieID, MovieName and NetProfit. Net Profit is to be calculated as the difference between Business Cost and Production Cost.

- e) List MovieID, MovieName and Cost for all movies with ProductionCost greater than 10,000 and less than 1,00,000.
- f) List details of all movies which fall in the category of comedy or action.
- g) List details of all movies which have not been released yet.
4. Suppose your school management has decided to conduct cricket matches between students of Class XI and Class XII. Students of each class are asked to join any one of the four teams – Team Titan, Team Rockers, Team Magnet and Team Hurricane. During summer vacations, various matches will be conducted between these teams. Help your sports teacher to do the following:
- Create a database “Sports”.
  - Create a table “TEAM” with following considerations:
    - It should have a column TeamID for storing an integer value between 1 to 9, which refers to unique identification of a team.
    - Each TeamID should have its associated name (TeamName), which should be a string of length not less than 10 characters.
  - Using table level constraint, make TeamID as the primary key.
  - Show the structure of the table TEAM using a SQL statement.
  - As per the preferences of the students four teams were formed as given below. Insert these four rows in TEAM table:
    - Row 1: (1, Team Titan)
    - Row 2: (2, Team Rockers)
    - Row 3: (3, Team Magnet)
    - Row 3: (4, Team Hurricane)
  - Show the contents of the table TEAM using a DML statement.
  - Now create another table MATCH\_DETAILS and insert data as shown below. Choose appropriate data types and constraints for each attribute.

**Table: MATCH DETAILS**

| MatchID | MatchDate  | FirstTeamID | SecondTeamID | FirstTeamScore | SecondTeamScore |
|---------|------------|-------------|--------------|----------------|-----------------|
| M1      | 2018-07-17 | 1           | 2            | 90             | 86              |
| M2      | 2018-07-18 | 3           | 4            | 45             | 48              |
| M3      | 2018-07-19 | 1           | 3            | 78             | 56              |
| M4      | 2018-07-19 | 2           | 4            | 56             | 67              |
| M5      | 2018-07-18 | 1           | 4            | 32             | 87              |
| M6      | 2018-07-17 | 2           | 3            | 67             | 51              |

5. Using the sports database containing two relations (TEAM, MATCH\_DETAILS) and write the queries for the following:

- a) Display the MatchID of all those matches where both the teams have scored more than 70.
- b) Display the MatchID of all those matches where FirstTeam has scored less than 70 but SecondTeam has scored more than 70.
- c) Display the MatchID and date of matches played by Team 1 and won by it.
- d) Display the MatchID of matches played by Team 2 and not won by it.
- e) Change the name of the relation TEAM to T\_DATA. Also change the attributes TeamID and TeamName to T\_ID and T\_NAME respectively.
6. A shop called Wonderful Garments who sells school uniforms maintains a database SCHOOLUNIFORM as shown below. It consisted of two relations - UNIFORM and COST. They made UniformCode as the primary key for UNIFORM relations. Further, they used UniformCode and Size to be composite keys for COSTrelation. By analysing the database schema and database state, specify SQL queries to rectify the following anomalies.
- a) M/S Wonderful Garments also keeps handkerchiefs of red colour, medium size of Rs. 100 each.
- b) `INSERT INTO COST (UCode, Size, Price) values (7, 'M',100);`
- When the above query is used to insert data, the values for the handkerchief without entering its details in the UNIFORM relation is entered. Make a provision so that the data can be entered in the COST table only if it is already there in the UNIFORM table.
- c) Further, they should be able to assign a new UCode to an item only if it has a valid UName. Write a query to add appropriate constraints to the SCHOOLUNIFORM database.
- d) Add the constraint so that the price of an item is always greater than zero.
7. Consider the following table named “Product”, showing details of products being sold in a grocery shop.

| PCode | PName          | UPrice | Manufacturer |
|-------|----------------|--------|--------------|
| P01   | Washing Powder | 120    | Surf         |
| P02   | Toothpaste     | 54     | Colgate      |
| P03   | Soap           | 25     | Lux          |
| P04   | Toothpaste     | 65     | Pepsodent    |
| P05   | Soap           | 38     | Dove         |
| P06   | Shampoo        | 245    | Dove         |

## NOTES

Write SQL queries for the following:

- a) Create the table Product with appropriate data types and constraints.
- b) Identify the primary key in Product.
- c) List the Product Code, Product name and price in descending order of their product name. If PName is the same, then display the data in ascending order of price.
- d) Add a new column Discount to the table Product.
- e) Calculate the value of the discount in the table Product as 10 per cent of the UPrice for all those products where the UPrice is more than 100, otherwise the discount will be 0.
- f) Increase the price by 12 per cent for all the products manufactured by Dove.
- g) Display the total number of products manufactured by each manufacturer.

Write the output(s) produced by executing the following queries on the basis of the information given above in the table Product:

- h) `SELECT PName, avg(UPrice) FROM Product GROUP BY Pname;`
  - i) `SELECT DISTINCT Manufacturer FROM Product;`
  - j) `SELECT COUNT (DISTINCT PName) FROM Product;`
  - k) `SELECT PName, MAX(UPrice), MIN(UPrice) FROM Product GROUP BY PName;`
8. Using the CARSHOWROOM database given in the chapter, write the SQL queries for the following:
- a) Add a new column Discount in the INVENTORY table.
  - b) Set appropriate discount values for all cars keeping in mind the following:
    - (i) No discount is available on the LXI model.
    - (ii) VXI model gives a 10 per cent discount.
    - (iii) A 12 per cent discount is given on cars other than LXI model and VXI model.
  - c) Display the name of the costliest car with fuel type "Petrol".
  - d) Calculate the average discount and total discount available on Baleno cars.
  - e) List the total number of cars having no discount.

Chapter

10

# Computer Networks



12130CH10

## In this Chapter

- » *Introduction to Computer Networks*
- » *Evolution of Networking*
- » *Types of Networks*
- » *Network Devices*
- » *Networking Topologies*
- » *Identifying Nodes in a Networked Communication*
- » *Internet, Web and the Internet of Things*
- » *Domain Name System*

*“Hoaxes use weaknesses in human behavior to ensure they are replicated and distributed. In other words, hoaxes prey on the Human Operating System.”*

— Stewart Kirkpatrick

## 10.1 INTRODUCTION TO COMPUTER NETWORKS

We are living in a connected world. Information is being produced, exchanged, and traced across the globe in real time. It's possible as almost everyone and everything in the digital world is interconnected through one way or the other.

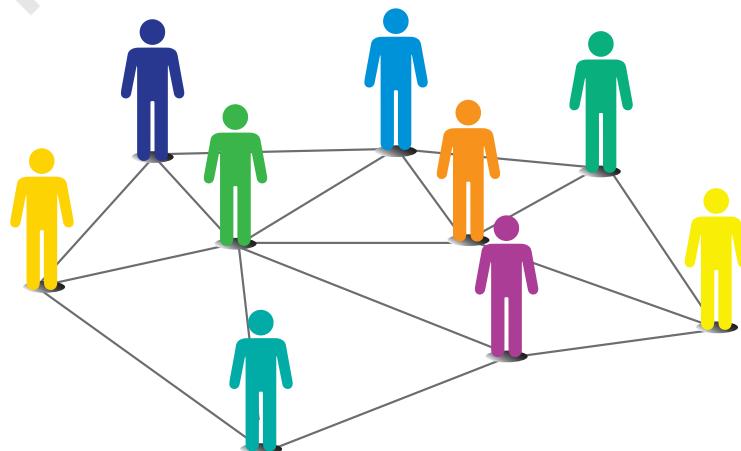


Figure 10.1: Interconnection forming a social network

### Activity 10.1

Identify some other networks in the real world.



A group of two or more similar things or people interconnected with each other is called network (Figure 10.1). Some of the examples of network in our everyday life includes:

- Social network
- Mobile network
- Network of computers
- Airlines, railway, banks, hospitals networks

A computer network (Figure 10.2) is an interconnection among two or more computers or computing devices. Such interconnection allows computers to share data and resources among each other. A basic network may connect a few computers placed in a room.

The network size may vary from small to large depending on the number of computers it connects. A computer network can include different types of hosts (also called nodes) like server, desktop, laptop, cellular phones.



Figure 10.2: A computer network

Apart from computers, networks include networking devices like switch, router, modem, etc. Networking devices are used to connect multiple computers in different settings. For communication, data in a network is divided into smaller chunks called packets. These packets are then carried over a network. Devices in a network can be connected either through wired media like cables or wireless media like air.

In a communication network, each device that is a part of a network and that can receive, create, store or send data to different network routes is called a node. In the context of data communication, a node can be a device such as a modem, hub, bridge, switch, router, digital telephone handset, a printer, a computer or a server.

Interconnectivity of computing devices in a network allows us to exchange information simultaneously with many parties through email, websites, audio/video calls, etc. Network allows sharing of resources. For example, a printer can be made available to multiple computers through a network; a networked storage can be accessed by multiple computers. People often connect their devices through hotspot, thus forming a small personal network.

#### Activity 10.2

Create a hotspot using a smartphone and connect other devices to it.



## 10.2 EVOLUTION OF NETWORKING

In the 1960s a research project was commissioned by Advanced Research Projects Agency Network (ARPANET) in the U.S. Department of Defence to connect the academic and research institutions located at different places for scientific collaborations. The first message was communicated between the University of California, Los Angeles (UCLA) and Stanford Research Institute (SRI). Slowly but gradually, more and more organisations joined the ARPANET, and many independent smaller networks were formed. Few of the milestones in the magnificent journey of evolution of computer networks is depicted in the timeline shown in Figure 10.3.

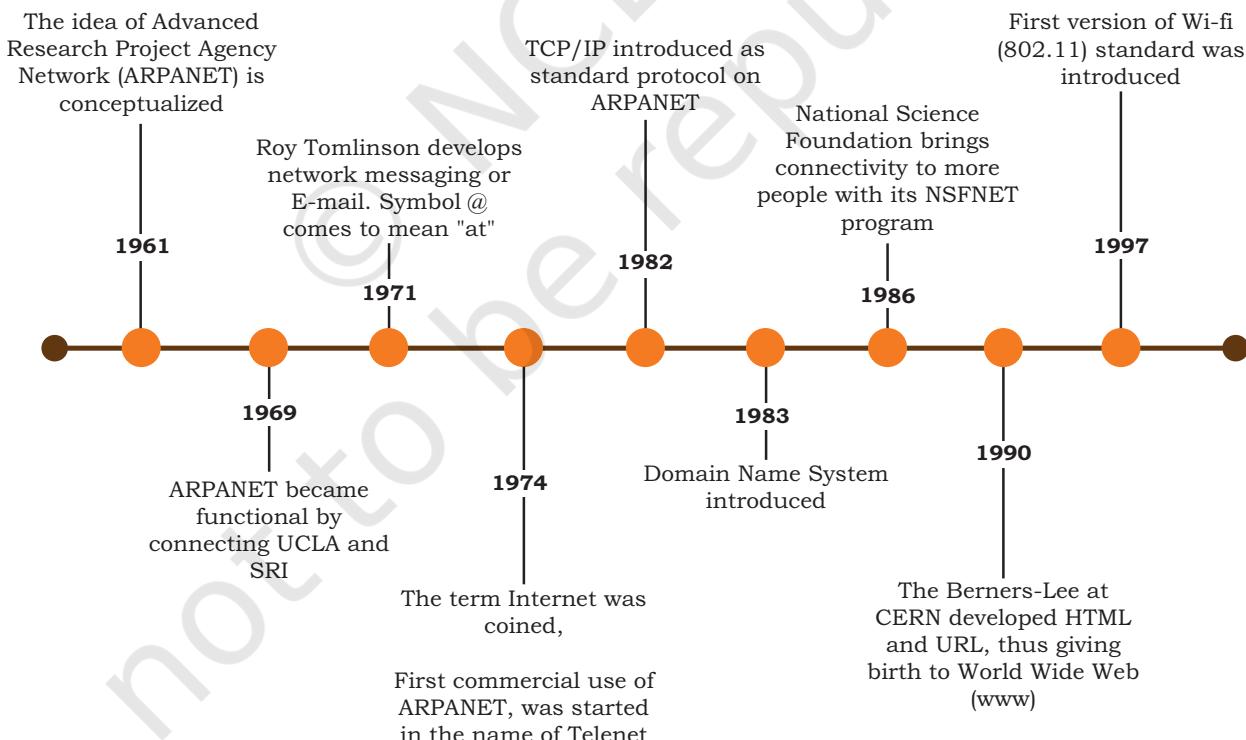


Figure 10.3: Timeline showing evolution of networking

### 10.3 TYPES OF NETWORKS

There are various types of computer networks ranging from network of handheld devices (like mobile phones or tablets) connected through Wi-Fi or Bluetooth within a single room to the millions of computers spread across the globe. Some are connected wireless while others are connected through wires.

Based on the geographical area covered and data transfer rate, computer networks are broadly categorised as:

- PAN ( Personal Area Network)
- LAN (Local Area Network)
- MAN (Metropolitan Area Network)
- WAN (Wide Area Network)

#### 10.3.1 Personal Area Network (PAN)

It is a network formed by connecting a few personal devices like computers, laptops, mobile phones, smart phones, printers etc., as shown in Figure 10.4. All these devices lie within an approximate range of 10 metres. A personal area network may be wired or wireless. For example, a mobile phone connected to the laptop through USB forms a wired PAN while two smartphones communicating with each other through Bluetooth technology form a wireless PAN or WPAN.



Figure 10.4: A Personal Area Network

### 10.3.2 Local Area Network (LAN)

It is a network that connects computers, mobile phones, tablet, mouse, printer, etc., placed at a limited distance. The geographical area covered by a LAN can range from a single room, a floor, an office having one or more buildings in the same premise, laboratory, a school, college, or university campus. The connectivity is done by means of wires, Ethernet cables, fibre optics, or Wi-Fi. A Local Area Network (LAN) is shown in Figure 10.5.

### Think and Reflect

Explore and find out the minimum internet speed required to make a video call.

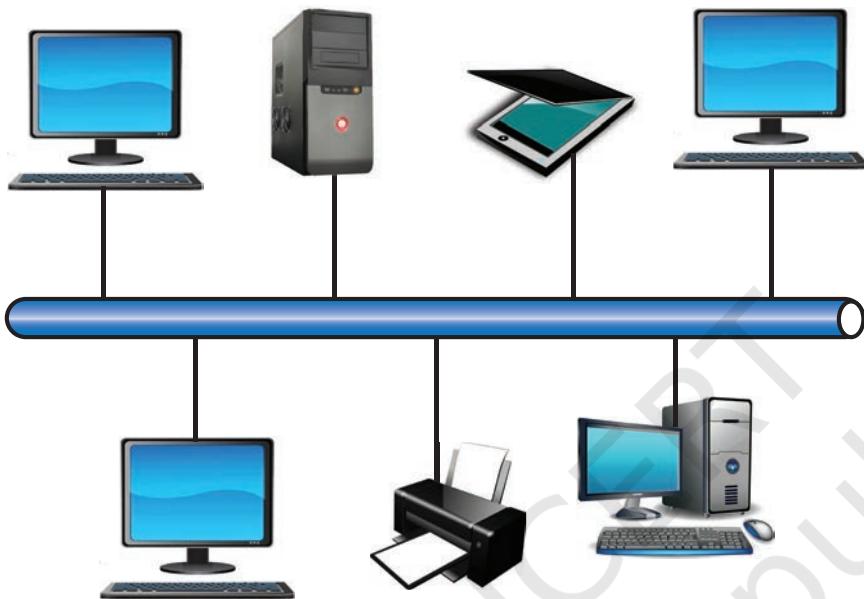


Figure 10.5: A Local Area Network

LAN is comparatively secure as only authentic users in the network can access other computers or shared resources. Users can print documents using a connected printer, upload/download documents and software to and from the local server. Such LANs provide the short range communication with the high speed data transfer rates. These types of networks can be extended up to 1 km. Data transfer in LAN is quite high, and usually varies from 10 Mbps (called Ethernet) to 1000 Mbps (called Gigabit Ethernet), where Mbps stands for Megabits per second. Ethernet is a set of rules that decides how computers and other devices connect with each other through cables in a local area network or LAN.

### 10.3.3 Metropolitan Area Network (MAN)

Metropolitan Area Network (MAN) is an extended form of LAN which covers a larger geographical area like a city or a town. Data transfer rate in MAN also ranges in Mbps,

but it is considerably less as compared to LAN. Cable TV network or cable based broadband internet services are examples of MAN. This kind of network can be extended up to 30-40 km. Sometimes, many LANs are connected together to form MAN, as shown in Figure 10.6.

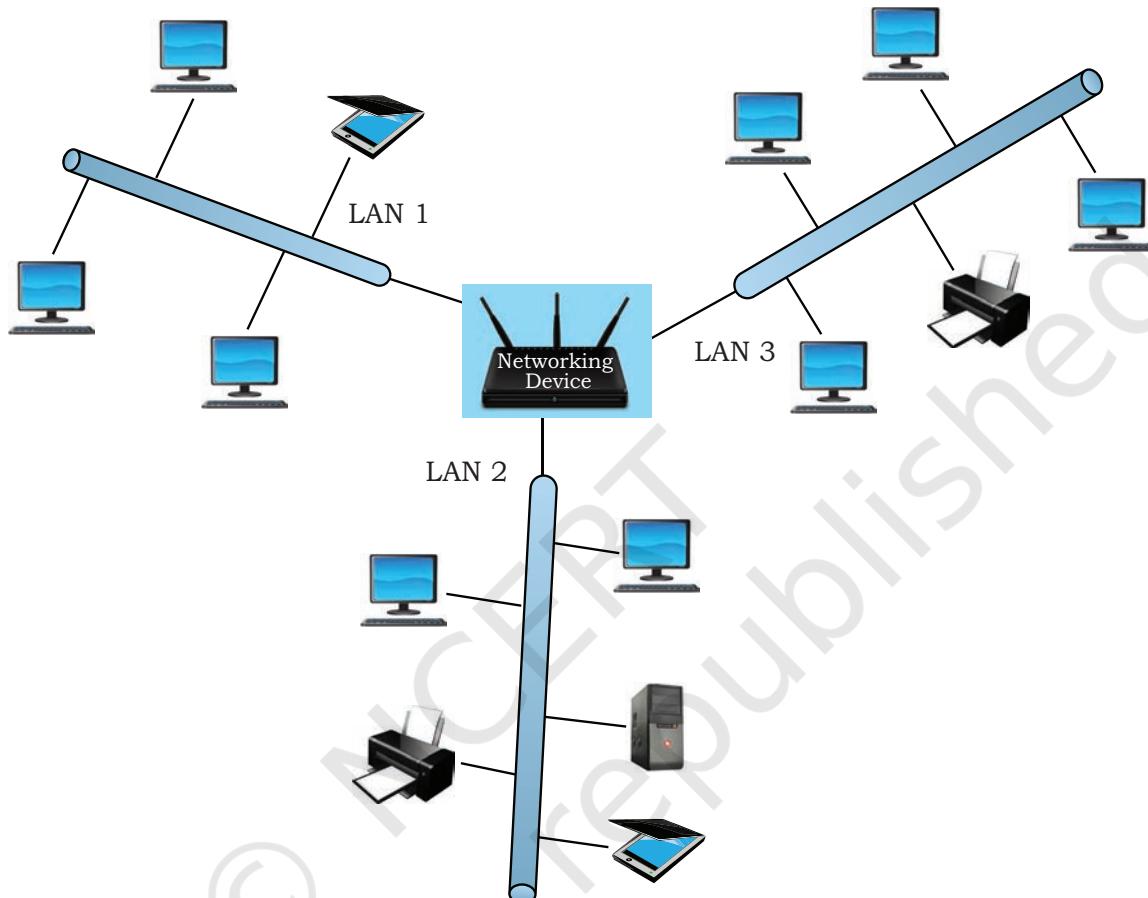


Figure 10.6: A Metropolitan Area Network

#### 10.3.4 Wide Area Network (WAN)

##### Think and Reflect

It is possible to access your bank account from any part of the world. Whether the bank's network is a LAN, MAN, WAN or any other type?

Wide Area Network connects computers and other LANs and MANs, which are spread across different geographical locations of a country or in different countries or continents. A WAN could be formed by connecting a LAN to other LANs (Figure 10.7) via wired/wireless media. Large business, educational and government organisations connect their different branches in different locations across the world through WAN. The Internet is the largest WAN that connects billions of computers, smartphones and millions of LANs from different continents.

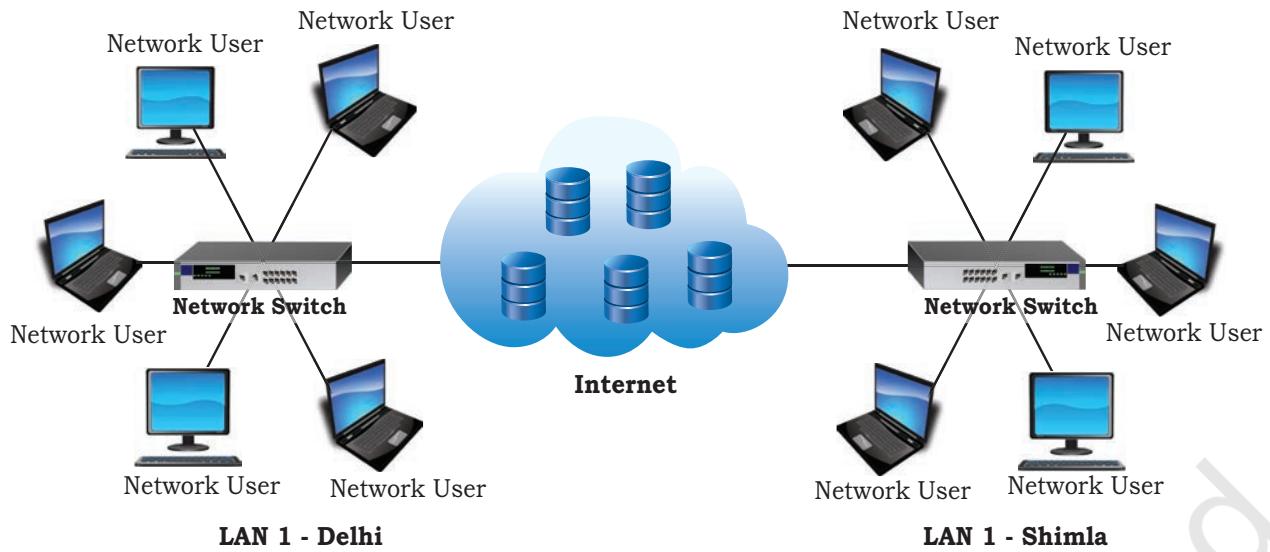


Figure 10.7: A Wide Area Network

## 10.4 NETWORK DEVICES

To communicate data through different transmission media and to configure networks with different functionality, we require different devices like Modem, Hub, Switch, Repeater, Router, Gateway, etc. Let us explore them in detail.

### 10.4.1 Modem

Modem stands for 'MOdulator DEModulator'. It refers to a device used for conversion between analog signals and digital bits. We know computers store and process data in terms of 0s and 1s. However, to transmit data from a sender to a receiver, or while browsing the internet, digital data are converted to an analog signal and the medium (be it free-space or a physical media) carries the signal to the receiver. There are modems connected to both the source and destination nodes. The modem at the sender's end acts as a modulator that converts the digital data into analog signals. The modem at the receiver's end acts as a demodulator that converts the analog signals into digital data for the destination node to understand. Figure 10.8 shows connectivity using a modem.

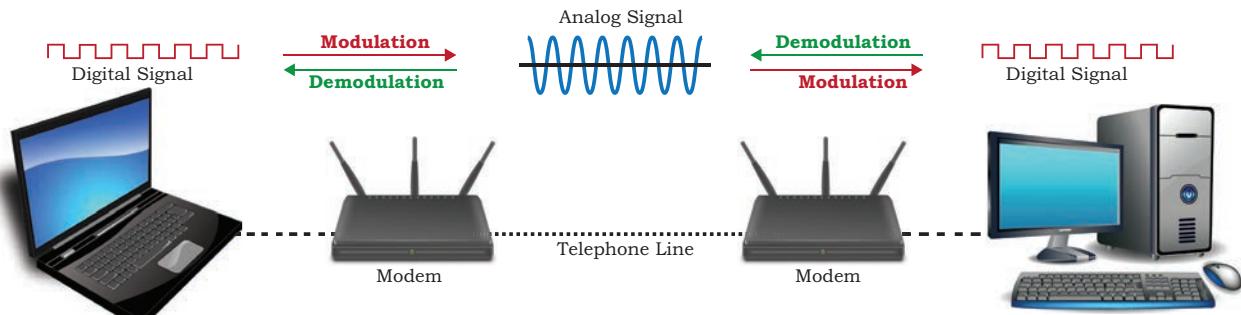


Figure 10.8: Use of modem

#### 10.4.2 Ethernet Card

Ethernet card, also known as Network Interface Card (NIC card in short) is a network adapter used to set up a wired network. It acts as an interface between computer and the network. It is a circuit board mounted on the motherboard of a computer as shown in Figure 10.9. The Ethernet cable connects the computer to the network through NIC. Ethernet cards can support data transfer between 10 Mbps and 1 Gbps (1000 Mbps). Each NIC has a MAC address, which helps in uniquely identifying the computer on the network.

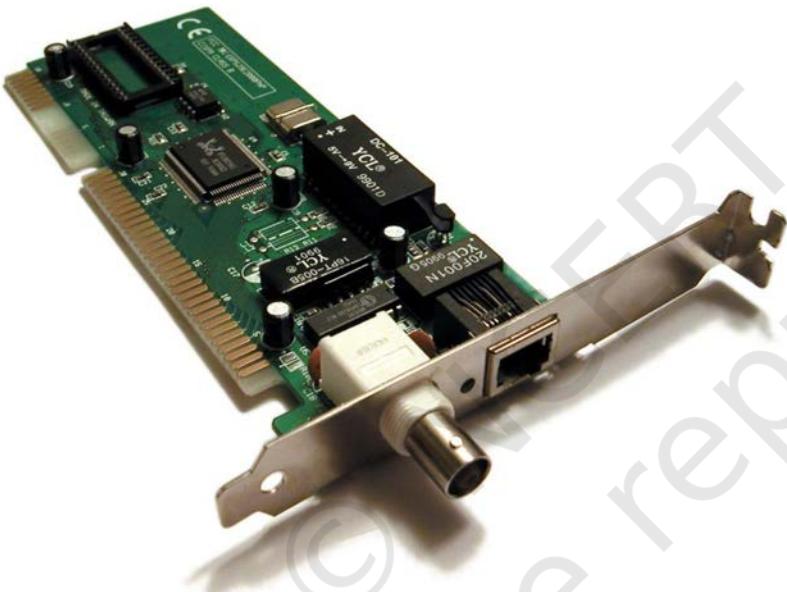


Figure 10.9: A Network Interface Card



Figure 10.10: RJ 45

#### 10.4.3 RJ45

RJ 45 or Registered Jack-45 is an eight-pin connector (Figure 10.10) that is used exclusively with Ethernet cables for networking. It is a standard networking interface that can be seen at the end of all network cables. Basically, it is a small plastic plug that fits into RJ-45 jacks of the Ethernet cards present in various computing devices.

#### 10.4.4 Repeater

Data are carried in the form of signals over the cable. These signals can travel a specified distance (usually about 100 m). Signals lose their strength beyond this limit and become weak. In such conditions, original signals need to be regenerated.

A repeater is an analog device that works with signals on the cables to which it is connected. The weakened signal appearing on the cable is regenerated and put back on the cable by a repeater.

#### 10.4.5 Hub

An Ethernet hub (Figure 10.11) is a network device used to connect different devices through wires. Data arriving on any of the lines are sent out on all the others. The limitation of Hub is that if data from two devices come at the same time, they will collide.



Figure 10.11: A network hub with 8 ports

#### 10.4.5 Switch

A switch is a networking device (Figure 10.12) that plays a central role in a Local Area Network (LAN). Like a hub, a network switch is used to connect multiple computers or communicating devices. When data arrives, the switch extracts the destination address from the data packet and looks it up in a table to see where to send the packet. Thus, it sends signals to only selected devices instead of sending to all. It can forward multiple packets at the same time. A switch does not forward the signals which are noisy or corrupted. It drops such signals and asks the sender to resend it.



Figure 10.12: Cables connected to a network switch

Ethernet switches are common in homes/offices to connect multiple devices thus creating LANs or to access the Internet.



An Internet service provider (ISP) is any organisation that provides services for accessing the Internet.



#### Activity 10.3

Find and list a few ISPs in your region.



#### 10.4.6 Router

A router (Figure 10.13) is a network device that can receive the data, analyse it and transmit it to other networks. A router connects a local area network to the internet. Compared to a hub or a switch, a router has advanced capabilities as it can analyse the data being carried over a network, decide/alter how it is packaged, and send it to another network of a different type. For example, data has been divided into packets of a certain size. Suppose these packets are to be carried over a different type of network which cannot handle bigger packets. In such a case, the data is to be repackaged as smaller packets and then sent over the network by a router.



Figure 10.13: A router

A router can be wired or wireless. A wireless router can provide Wi-Fi access to smartphones and other devices. Usually, such routers also contain some ports to provide wired Internet access. These days, home Wi-Fi routers perform the dual task of a router and a modem/switch. These routers connect to incoming broadband lines, from ISP (Internet Service Provider), and convert them to digital data for computing devices to process.

#### 10.4.7 Gateway

As the term “Gateway” suggests, it is a key access point that acts as a “gate” between an organisation's network and the outside world of the Internet (Figure 10.14). Gateway serves as the entry and exit point of a network, as all data coming in or going out of a network must first pass through the gateway in order to use routing paths. Besides routing data packets, gateways also maintain information about the host network's internal connection paths and the identified paths of other remote networks. If a node from one network wants to communicate with a node of a foreign network, it will

pass the data packet to the gateway, which then routes it to the destination using the best possible route.

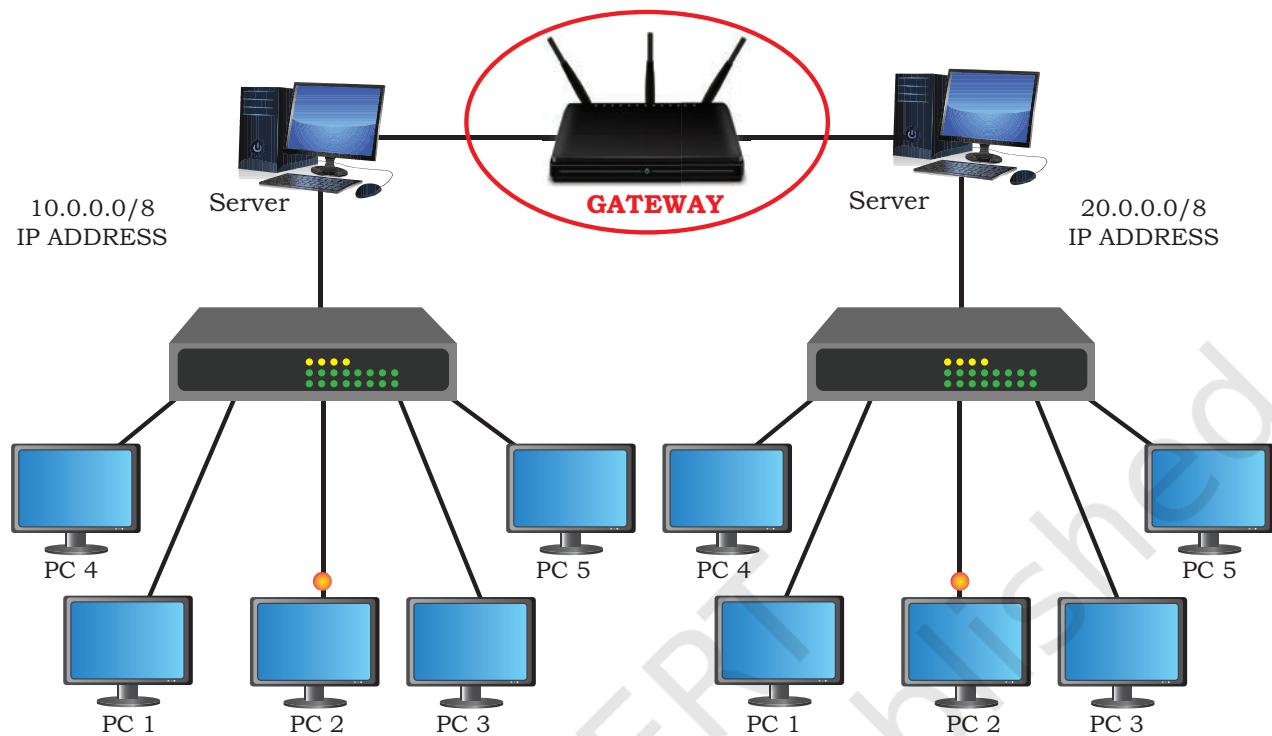


Figure 10.14: A network gateway

For simple Internet connectivity at homes, the gateway is usually the Internet Service Provider that provides access to the entire Internet. Generally, a router is configured to work as a gateway device in computer networks. But a gateway can be implemented completely in software, hardware, or a combination of both. Because a network gateway is placed at the edge of a network, the firewall is usually integrated with it.

## 10.5 NETWORKING TOPOLOGIES

We have already discussed that a number of computing devices are connected together to form a Local Area Network (LAN), and interconnections among millions of LANs forms the Internet. The arrangement of computers and other peripherals in a network is called its topology. Common network topologies are Mesh, Ring, Bus, Star and Tree.

### 10.5.1 Mesh Topology

In this networking topology, each communicating device is connected with every other device in the network as shown in Figure 10.15. Such a network can handle large amounts of traffic since multiple nodes can transmit data simultaneously. Also, such networks are more reliable in the sense that even if a node gets down, it does not cause any break in the transmission of data between other nodes. This topology is also more secure as compared to other topologies because each cable between two nodes carries different data. However, wiring is complex and cabling cost is high in creating such networks and there are many redundant or unutilised connections.

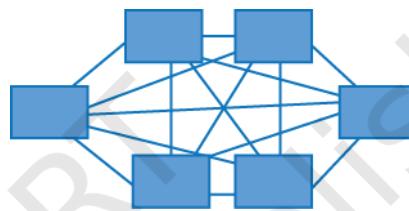


Figure 10.15: A mesh topology

To build a fully-connected mesh topology of  $n$  nodes, it requires  $n(n-1)/2$  wires.

### 10.5.2 Ring Topology

In ring topology (Figure 10.16), each node is connected to two other devices, one each on either side, as shown in Figure 10.16. The nodes connected with each other thus forms a ring. The link in a ring topology is unidirectional. Thus, data can be transmitted in one direction only (clockwise or counterclockwise).

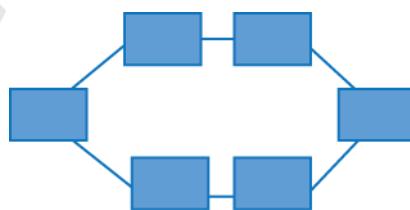


Figure 10.16: A ring topology

### 10.5.3 Bus Topology

In bus topology (Figure 10.17), each communicating device connects to a transmission medium, known as bus. Data sent from a node are passed on to the bus and hence are transmitted to the length of the bus in both directions. That means, data can be received by any of the nodes connected to the bus.

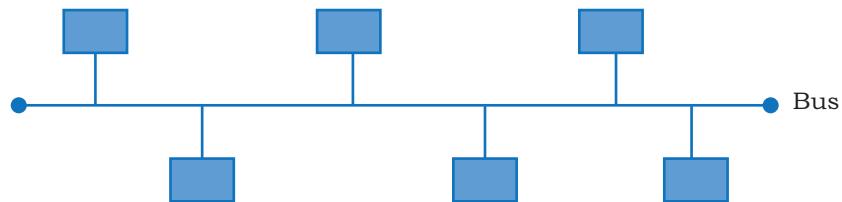


Figure 10.17:A bus topology

In this topology, a single backbone wire called bus is shared among the nodes, which makes it cheaper and easier to maintain. Both ring and bus topologies are considered to be less secure and less reliable.

#### 10.5.4 Star Topology

In star topology (Figure 10.18), each communicating device is connected to a central node, which is a networking device like a hub or a switch, as shown in Figure 10.18.

Star topology is considered very effective, efficient and fast as each device is directly connected with the central device. Although disturbance in one device will not affect the rest of the network, any failure in a central networking device may lead to the failure of complete network.

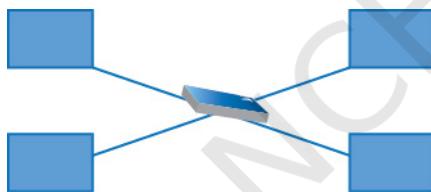


Figure 10.18:A star topology

The central node can be either a broadcasting device means data will be transmitted to all the nodes in the network, or a unicast device means the node can identify the destination and forward data to that node only.

#### 10.5.5 Tree or Hybrid Topology

It is a hierarchical topology, in which there are multiple branches and each branch can have one or more basic topologies like star, ring and bus. Such topologies are usually realised in WANs where multiple LANs are connected. Those LANs may be in the form of a ring, bus or star. In figure 10.19, a hybrid topology is shown connecting 4-star topologies in a bus.

In this type of network, data transmitted from source first reaches the centralised device and from there the data passes through every branch where each branch can have links for more nodes.

#### Think and Reflect

How will a Bus and Ring topology behave in case a Node is down?



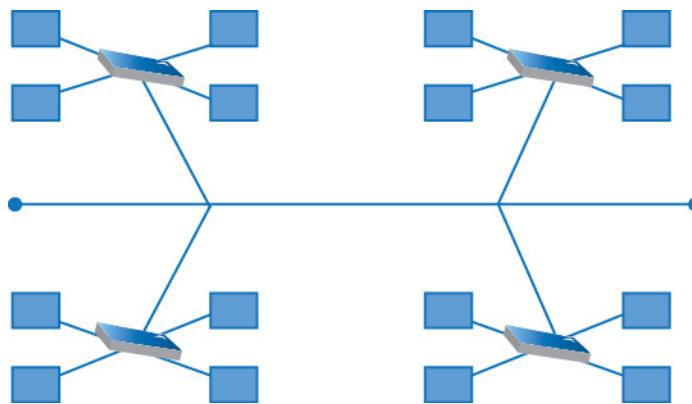


Figure 10.19: A hybrid topology

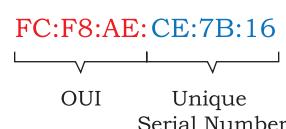
## 10.6 IDENTIFYING NODES IN A NETWORKED COMMUNICATION

Each node in a network should be uniquely identified so that a network device can identify the sender and receiver and decide a routing path to transmit data. Let us explore further and know how each node is distinguished in a network.

### 10.6.1 MAC Address

MAC stands for Media Access Control. The MAC address, also known as the physical or hardware address, is a unique value associated with a network adapter called a NIC. The MAC address is engraved on NIC at the time of manufacturing and thus it is a permanent address and cannot be changed under any circumstances. The machine on which the NIC is attached, can be physically identified on the network using its MAC address.

Each MAC address is a 12-digit hexadecimal numbers (48 bits in length), of which the first six digits (24 bits) contain the manufacturer's ID called Organisational Unique Identifier (OUI) and the later six digits (24 bits) represents the serial number assigned to the card by the manufacturer. A sample MAC address looks like:



#### Activity 10.4

Explore how can you find the MAC address of your computer system.



### 10.6.2 IP Address

IP address, also known as Internet Protocol address, is also a unique address that can be used to uniquely identify each node in a network. The IP addresses

are assigned to each node in a network that uses the Internet Protocol for communication. Thus, if we know a computer's IP address, we can communicate with that computer from anywhere in the world. However, unlike MAC address, IP address can change if a node is removed from one network and connected to another network.

The initial IP Address called version 4 (IPV4 in short), is a 32 bit numeric address, written as four numbers separated by periods, where each number is the decimal (base-10) representation for an 8-bit binary (base-2) number and each can take any value from 0 - 255. A sample IPV4 address looks like:

192 : 168 : 0 : 178

With more and more devices getting connected to the Internet, it was realised that the 32-bit IP address will not be sufficient as it offers just under 4.3 billion unique addresses. Thus, a 128 bits IP address, called IP version 6 (IPV6 in short) was proposed. An IPv6 address is represented by eight groups of hexadecimal (base-16) numbers separated by colons. A sample IPV6 address looks like:

2001 : CDBA : 0000 : 0000 : 0000 : 0000 : 3257 : 9652

## 10.7 INTERNET, WEB AND THE INTERNET OF THINGS

The Internet is the global network of computing devices including desktop, laptop, servers, tablets, mobile phones, other handheld devices, printers, scanners, routers, switches, gateways, etc. Moreover, smart electronic appliances like TV, AC, refrigerator, fan, light, etc. can also communicate through a network. The list of such smart devices is always increasing e.g., drones, vehicles, door lock, security camera. We have already studied IoT and WoT in class 11.

The Internet is evolving every day and it is difficult to visualise or describe each and every aspect of the architecture of the Internet. Computers are either connected to a modem through a cable or wirelessly (Wi-Fi). That modem, be it wired or wireless, is connected to a local Internet Service Provider (ISP) who then connects to a national network. Many such ISPs connect together forming a regional network and regional networks connect together forming a national network, and such country-wise networks form the Internet backbone.

### Think and Reflect

Do mobile phones have a MAC address?  
Is it different from the IMEI number of mobile phones?



### **Think and Reflect**

You are encouraged to take up any area of concern where you think IoT can be immensely beneficial and discuss it with your peers. An example for the same can be preventing road accidents.



The Internet today is a widespread network, and its influence is no longer limited to the technical fields of computer communications. It is being used by everyone in the society as is evident from the increasing use of online tools for education, creativity, entertainment, socialisation, and e-commerce.

#### **10.7.1 The World Wide Web (WWW)**

The World Wide Web (WWW) or web in short, is an ocean of information, stored in the form of trillions of interlinked web pages and web resources. The resources on the web can be shared or accessed through the Internet.

Earlier, to access files residing in different computers, one had to login individually to each computer through the Internet. Besides, files in different computers were sometimes in different formats, and it was difficult to understand each other's files and documents. Sir Tim Berners-Lee — a British computer scientist invented the revolutionary World Wide Web in 1990 by defining three fundamental technologies that lead to creation of web:

- HTML – HyperText Markup Language. It is a language which is used to design standardised Web Pages so that the Web contents can be read and understood from any computer. Basic structure of every webpage is designed using HTML.
- URI – Uniform Resource Identifier. It is a unique address or path for each resource located on the web. It is also known as Uniform Resource Locator (URL). Every page on the web has a unique URL. Examples are: <https://www.mhrd.gov.in>, <http://www.ncert.nic.in>, <http://www.airindia.in>, etc. URL is sometimes also called web address. However, a URL is not only the domain name. It contains other information that completes a web address, as depicted below:



- HTTP – The HyperText Transfer Protocol is a set of rules which is used to retrieve linked web pages across the web. The more secure and advanced version is HTTPS.

Many people confuse the web with the Internet. The Internet as we know is the huge global network of interconnected computers, which may or may not have any file or webpage to share with the world. The web on the other hand is the interlinking of collection of Webpages on these computers which are accessible over the Internet. WWW today gives users access to a vast collection of information created and shared by people across the world. It is today the most popular information retrieval system.

### **10.8 DOMAIN NAME SYSTEM**

The Internet is a vast ocean where information is available in the form of millions of websites. Each website is stored on a server which is connected to the Internet, which means each server has an IP address. Every device connected to the Internet has an IP address. To access a website, we need to enter its IP address on our web browser. But it is very difficult to remember the IP addresses of different websites as they are in terms of numbers or strings.

However, it is easier to remember names, and therefore, each computer server hosting a website or web resource is given a name against its IP address. These names are called the Domain names or hostnames corresponding to unique IP addresses assigned to each server. For easy understanding, it can be considered as the phonebook where instead of remembering each person's phone number, we assign names to their numbers. For example, IP addresses and domain names of some websites are as follows:

**Table 10.1 Examples of domain names and their mapped IP addresses**

| Domain Name   | IP Address     |
|---------------|----------------|
| ncert.nic.in  | 164.100.60.233 |
| cbse.nic.in   | 164.100.107.32 |
| mhrd.gov.in   | 164.100.163.45 |
| wikipedia.org | 198.35.26.96   |

#### **10.8.1 DNS Server**

Instead of remembering IP addresses, we assign a domain name to each IP. But, to access a web resource, a browser needs to find out the IP address corresponding to the domain name entered. Conversion of the domain

name of each web server to its corresponding IP address is called domain name resolution. It is done through a server called DNS server. Thus, when we enter a URL on a web browser, the HTTP protocol approaches a computer server called DNS server to obtain the IP address corresponding to that domain name. After getting the IP address, the HTTP protocol retrieves the information and loads it in our browser.

In Figure 10.20, an example is shown in which the HTTP requests a DNS server for corresponding IP address, and the server sends back an IP address.

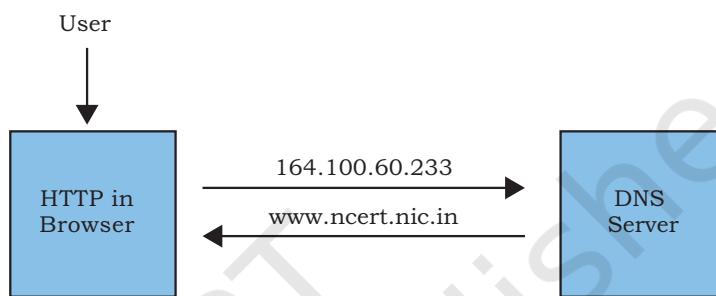


Figure 10.20: Request of IP address corresponding to domain name

A DNS server maintains a database of domain names and their corresponding IP addresses. To understand how the domain name resolution works, we have to understand how and where the DNS servers are kept. The DNS servers are placed in hierarchical order. At the top level, there are 13 servers called root servers. Then below the root servers there are other DNS servers at different levels. A DNS server may contain the IP address corresponding to a domain or it will contain the IP address of other DNS servers, where this domain entry can be searched.

### SUMMARY

- A computer network is an interconnection among two or more computers or computing devices.
- A computer network allows computers to share data and resources among each other.
- Networking devices are used to connect multiple computers in different settings.

- In a communication network, each device that is a part of a network and that can receive, create, store or send data to different network routes is called a node.
- Based on the geographical area covered and data transfer rate, computer networks are broadly categorised into LAN (Local Area Network), MAN (Metropolitan Area Network) and WAN (Wide Area Network).
- LAN is a network that connects a variety of nodes placed at a limited distance ranging from a single room, a floor, an office or a campus having one or more buildings in the same premises.
- Ethernet is a set of rules that decides how computers and other devices connect with each other through cables in a LAN.
- Metropolitan Area Network (MAN) is an extended form of LAN which covers a larger geographical area like a city or a town.
- Cable TV network or cable based broadband internet services are examples of MAN.
- Wide Area Network (WAN) connects computers and other LANs and MANs, which are spread across different geographical locations of a country or in different countries or continents.
- The Internet is the largest WAN that connects billions of computers, smartphones and millions of LANs from different continents.
- Modem stands for ‘MOdulator DEModulator’, is a device used for conversion between electric signals and digital bits.
- Ethernet card, also known as Network Interface Card (NIC card in short) is a network adaptor used to set up a wired network.
- Each NIC has a MAC address, which helps in uniquely identifying the computer on the network.
- A repeater is an analog device that regenerate the signals on the cables to which it is connected.
- A switch is a networking device used to connect multiple computers or communicating devices.
- A router is a network device that can receive the data, analyse it and transmit it to other networks.

## NOTES

- Gateway serves as the entry and exit point of a network, as all data coming in or going out of a network must first pass through the gateway in order to use routing paths.
- The arrangement of computers and other peripherals in a network is called its topology.
- Common network topologies are Mesh, Ring, Bus, Star and Tree.
- In mesh topology each communicating device is connected with every other device in the network.
- In ring topology, each node is connected to two other devices, one each on either side.
- In bus topology, a single backbone wire called bus is shared among the nodes, which makes it cheaper and easy to maintain.
- In star topology, each communicating device is connected to a central networking device like a hub or a switch.
- In tree or hybrid topology, there are multiple branches and each branch can have one or more basic topologies like star, ring and bus.
- The MAC address, also known as the physical or hardware address, is a unique permanent value associated with a network adapter called a NIC. It is used to physically identify a machine on the network.
- IP address, also known as Internet Protocol address, is a unique address that can be used to uniquely identify each node in a network.
- Unlike MAC address, IP address can change if a node is removed from one network and connected to another network.
- The Internet is the global network of computing devices.
- The World Wide Web (WWW) or web in short, is an ocean of information, stored in the form of trillions of interlinked web pages and web resources.
- Sir Tim Berners-Lee — a British computer scientist invented the revolutionary World Wide Web in 1990.
- HTML (HyperText Markup Language) is a language which is used to design standardised Web Pages so that the Web contents can be read

and understood from any computer.

- URI (Uniform Resource Identifier) or URL (Uniform Resource Locator) is a unique address or path for each resource located on the web.
- HTTP – The HyperText Transfer Protocol is a set of rules which is used to retrieve linked web pages across the web. The more secure and advanced version is HTTPS.
- Each computer server hosting a website or web resource is given a name against its IP address. These names are called the Domain names or hostnames.
- Conversion of the domain name of each web server to its corresponding IP address is called domain name resolution. It is done through a server called DNS server.



## EXERCISE

1. Expand the following:
  - a) ARPANET
  - b) MAC
  - c) ISP
  - d) URI
2. What do you understand by the term network?
3. Mention any two main advantages of using a network of computing devices.
4. Differentiate between LAN and WAN.
5. Write down the names of few commonly used networking devices.
6. Two universities in different States want to transfer information. Which type of network they need to use for this?
7. Define the term topology. What are the popular network topologies?
8. How is tree topology different from bus topology?
9. Identify the type of topology from the following:
  - a) Each node is connected with the help of a single cable.
  - b) Each node is connected with central switching through independent cables.

## NOTES

10. What do you mean by a modem? Why is it used?
11. Explain the following devices:
  - a) Switch
  - b) Repeater
  - c) Router
  - d) Gateway
  - e) NIC
12. Draw a network layout of star topology and bus topology connecting five computers.
13. What is the significance of MAC address?
14. How is IP address different from MAC address? Discuss briefly.
15. What is DNS? What is a DNS server?
16. Sahil, a class X student, has just started understanding the basics of Internet and web technologies. He is a bit confused in between the terms “World Wide Web” and “Internet”. Help him in understanding both the terms with the help of suitable examples of each.

Chapter

11

# Data Communication



12130CH11

## In this Chapter

- » Concept of Communication
- » Components of Data Communication
- » Measuring Capacity of Communication Media
- » Types of Data Communication
- » Switching Techniques
- » Transmission Media
- » Mobile Telecommunication Technologies
- » Protocol

*“People already have bionic arms and legs that work by the power of thought. And we increasingly outsource mental and communicative activities to computers. We are merging with our smartphones. Very soon, they will just be part of the body”*

— Yuval Noah Harari

## 11.1 CONCEPT OF COMMUNICATION

The term “Data Communication” comprises two words: Data and Communication. Data can be any text, image, audio, video, and multimedia files. Communication is an act of sending or receiving data. Thus, data communication refers to the exchange of data between two or more networked or connected devices. These devices must be capable of sending and receiving data over a communication medium. Examples of such devices include personal computers, mobile phones, laptops, etc. As we can see in Figure 11.1, four different types of devices — computer, printer, server and switch are connected to form the network. These devices are connected through a media to the network, which carry information from one end to other end.

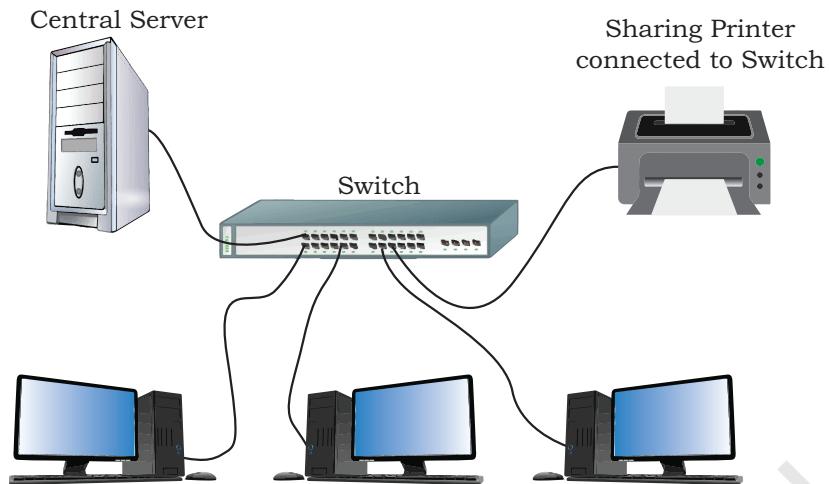
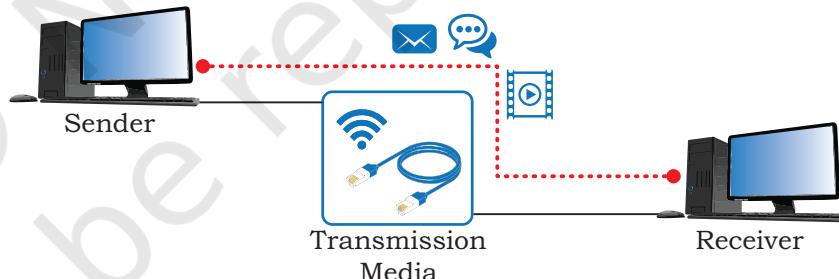


Figure 11.1: A simple network of computing devices

## 11.2 COMPONENTS OF DATA COMMUNICATION

Whenever we talk about communication between two computing devices using a network, five most important aspects come to our mind. These are sender, receiver, communication medium, the message to be communicated, and certain rules called protocols to be followed during communication. The communication media is also called transmission media. Figure 11.2 shows the role of these five components in data communication.



### Activity 11.1

List various types of senders on a network.



Figure 11.2: Components of data communication

**Sender:** A sender is a computer or any such device which is capable of sending data over a network. It can be a computer, mobile phone, smartwatch, walkie-talkie, video recording device, etc.

**Receiver:** A receiver is a computer or any such device which is capable of receiving data from the network. It can be any computer, printer, laptop, mobile phone, television, etc. In computer communication, the sender and receiver are known as nodes in a network.

**Message:** It is the data or information that needs to be exchanged between the sender and the receiver. Messages can be in the form of text, number, image, audio, video, multimedia, etc.

**Communication media:** It is the path through which the message travels between source and destination. It is also called medium or link which is either wired or wireless. For example, a television cable, telephone cable, ethernet cable, satellite link, microwaves, etc. We will study about various communication media in section 11.5.

**Protocols:** It is a set of rules that need to be followed by the communicating parties in order to have successful and reliable data communication. You have already come across protocols such as Ethernet and HTTP.

### 11.3 MEASURING CAPACITY OF COMMUNICATION MEDIA

In data communication, the transmission medium is also known as channel. The capacity of a channel is the maximum amount of signals or traffic that a channel can carry. It is measured in terms of bandwidth and data transfer rate as described below:

#### 11.3.1 Bandwidth

Bandwidth of a channel is the range of frequencies available for transmission of data through that channel. Higher the bandwidth, higher the data transfer rate. Normally, bandwidth is the difference of maximum and minimum frequency contained in the composite signals. Bandwidth is measured in Hertz (Hz).

$$1 \text{ KHz} = 1000 \text{ Hz}$$

$$1 \text{ MHz} = 1000 \text{ KHz} = 1000000 \text{ Hz}$$

#### 11.3.2 Data Transfer Rate

Data travels in the form of signals over a channel. One signal carries one or more bits over the channel. Data transfer rate is the number of bits transmitted between source and destination in one second. It is also known as bit rate. It is measured in terms of bits per second (bps). The higher units for data transfer rates are:

$$1 \text{ Kbps} = 2^{10} \text{ bps} = 1024 \text{ bps}$$

$$1 \text{ Mbps} = 2^{20} \text{ bps} = 1024 \text{ Kbps}$$

$$1 \text{ Gbps} = 2^{30} \text{ bps} = 1024 \text{ Mbps}$$

$$1 \text{ Tbps} = 2^{40} \text{ bps} = 1024 \text{ Gbps}$$

#### Activity 11.2

Find out how many hertz is 10 Megahertz.



MBps stands for Megabyte per second whereas Mbps stands for Megabit per second.

**Example 11.1** A user wants to upload a text document at the rate of 10 pages per 20 second. What will be the required data rate of the channel? (Assume that 1 page contains 1600 characters and each character is of 8 bits).

$$\begin{aligned}\textbf{Solution:} \text{ Required data rate } & \frac{(10 \times 1600 \times 8)}{20} \\ & = 6400 \text{ bps} = 6.25 \text{ Kbps}\end{aligned}$$

## 11.4 TYPES OF DATA COMMUNICATION

Data communication happens in the form of signals between two or more computing devices or nodes. The transfer of data happens over a point-to-point or multipoint communication channel. Data communication between different devices are broadly categorised into 3 types: Simplex communication, Half-duplex communication, and Full-duplex communication.

### 11.4.1 Simplex Communication

It is a one way or unidirectional communication between two devices in which one device is sender and other one is receiver. Devices use the entire capacity of the link to transmit the data. It is like a one way street where vehicles can move in only one direction. For example, data entered through a keyboard or audio sent to a speaker are one way communications.

With the advent of IoT, controlling home appliances is another example of simplex communication as shown in the Figure 11.3. One can control fans, lights, fridge, oven etc. while sitting in the office or driving a car.

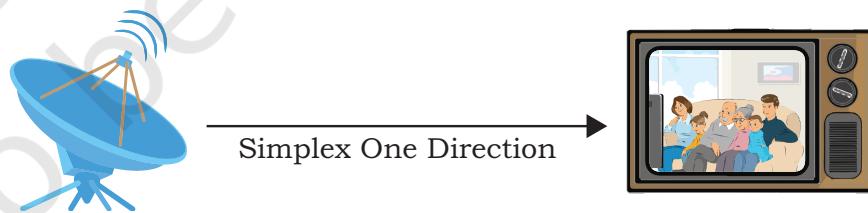
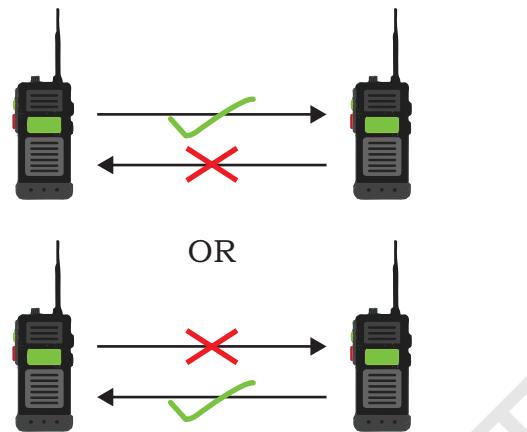


Figure 11.3: Simplex communication

### 11.4.2 Half-duplex Communication

It is two way or bidirectional communication between two devices in which both the devices can send and receive data or control signals in both directions, but not at the same time, as shown in Figure 11.4. While one device is sending data, the other one will receive and vice-versa. It is like sharing a one-way narrow bridge among vehicles

moving in both directions. Vehicles cannot pass the bridge simultaneously. Basically, it is a simplex channel where the direction of transmission can be switched. Application of such type of communication can be found in walkie-talkie where one can press the push-to-talk button and talk. This enables the transmitter and turns off the receiver in that device and others can only listen.



*Figure 11.4: Half-duplex where communication occurs in two different moments.*

#### 11.4.3 Full-duplex Communication

It is two way or bidirectional communication in which both devices can send and receive data simultaneously, as shown in Figure 11.5. It is like a two way road where vehicles can go in both directions at the same time. This type of communication channel is employed to allow simultaneous communication, for example, in our mobile phones and landline telephones. The capacity of the transmission link is shared between the signals going in both directions. This can be done either by using two physically separate simplex lines — one for sending and other for receiving, or the capacity of the single channel is shared between the signals travelling in different directions.



*Figure 11.5: Full duplex transmission of data*



VoIP is a communication methodology designed to deliver both voice and multimedia communications over Internet protocol.



Voice over Long-Term Evolution (VoLTE) is a standard for high-speed wireless communication for mobile phones, including IoT and wearables.

## 11.5 SWITCHING TECHNIQUES

In a network having multiple devices, we are interested to know how to connect the sender and receiver so that one-to-one communication is possible. One solution is to make a dedicated connection between each pair of devices (mesh topology) or between a central device and every other device (a star topology). However, we know that such methods are costly in case of large networks.

An alternative to this is switching whereby data is routed through various nodes in a network. This switching process forms a temporary route for the data to be transmitted. Two commonly used switching techniques are — Circuit Switching and Packet Switching.

### 11.5.1 Circuit Switching

In circuit switching, before a communication starts, a dedicated path is identified between the sender and the receiver. This path is a connected sequence of links between network nodes. All packets follow the same path established during the connection.

In earlier days, when we placed a telephone call, the switching equipment within the telephone system finds out a physical path or channel all the way from our telephone at home to the receiver's telephone. This is an example of circuit switching.

### 11.5.2 Packet Switching

In packet switching, each information or message to be transmitted between sender and receiver is broken down into smaller pieces, called packets. These packets are then transmitted independently through the network. Different packets of the same message may take different routes depending on availability.

Each packet has two parts — a header containing the address of the destination and other information, and the main message part. When all the packets reach the destination, they are reassembled and the complete message is received by the receiver.

Unlike circuit switching, a channel is occupied in packet switching only during the transmission of the packet. On completion of the transmission, the channel is available for transfer of packets from other communicating parties.

## 11.6 TRANSMISSION MEDIA

A transmission medium can be anything that can carry signals or data between the source (transmitter) and destination (receiver). For example, as we switch on a ceiling fan or a light bulb, the electric wire is the medium that carries electric current from switch to the fan or bulb. Two men are talking as shown in Figure 11.6. Here the medium is air.

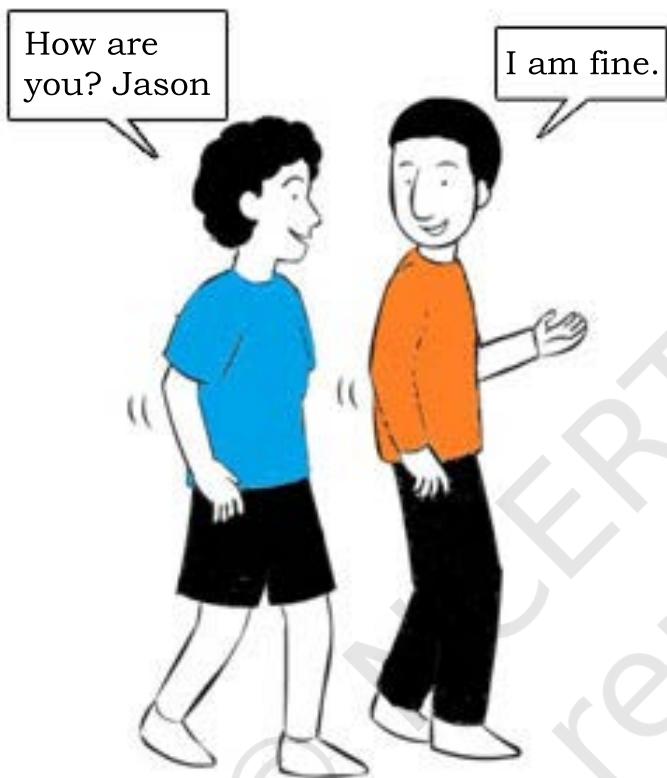


Figure 11.6: Two person communicating

In data communication, transmission media are the links that carry messages between two or more communicating devices. Transmission can be classified as guided or unguided. Figure 11.7 shows the classification of communication media.

In guided transmission, there is a physical link made of wire/cable through which data in terms of signals are propagated between the nodes. These are usually metallic cable, fiber-optic cable, etc. They are also known as wired media.

In unguided transmission, data travels in air in terms of electromagnetic waves using an antenna. They are also known as wireless media.

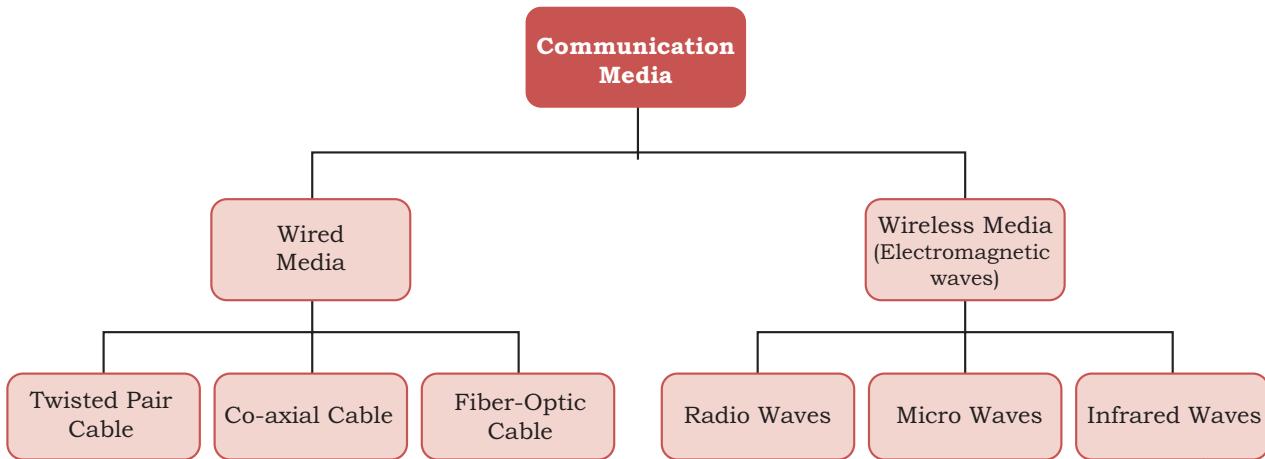


Figure 11.7: Classification of communication media

Dish-shaped antennas are used for sending and receiving data at longer distances. These antennas are mounted on taller buildings so that it would be in line-of-sight. Waves gradually become weaker and weaker after travelling a certain distance through the air. Therefore repeaters are installed to regenerate the signals of the same energy.

#### **11.6.1 Wired Transmission Media**

Any physical link that can carry data in the form of signals belongs to the category of wired transmission media. Three commonly used guided/wired media for data transmission are, twisted pair, coaxial cable, and fiber optic cable. Twisted-pair and coaxial cable carry the electric signals whereas the optical fiber cable carries the light signals.

##### **(A) Twisted Pair Cable**

A twisted-pair consists of two copper wires twisted like a DNA helical structure. Both the copper wires are insulated with plastic covers. Usually, a number of such pairs are combined together and covered with a protective outer wrapping, as shown in Figure 11.8.

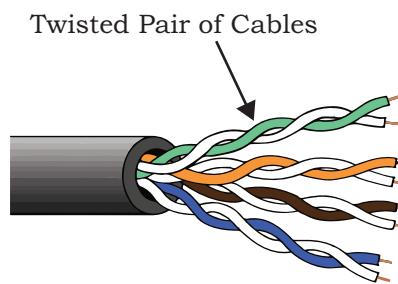


Figure 11.8: Twisted pair of cables

Each of the twisted pairs act as a single communication link. The use of twisted configuration minimises the effect of electrical interference from similar pairs close by. Twisted pairs are less expensive and most commonly used in telephone lines and LANs. These cables are of two types: Unshielded twisted-pair (UTP) and Shielded twisted-pair (STP), as shown in Figure 11.9.

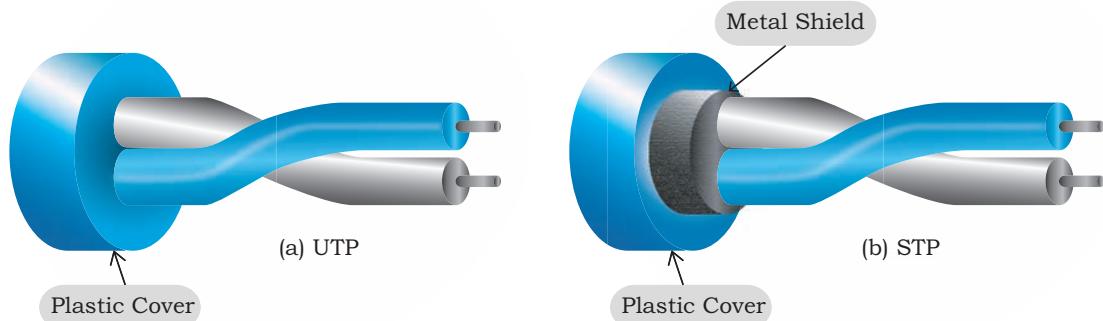


Figure 11.9: UTP Cable and STP Cable

### (B) Coaxial cable

Coaxial cable is another type of data transmission medium. It is better shielded and has more bandwidth than a twisted pair. As shown in Figure 11.10, it has a copper wire at the core of the cable which is surrounded with insulating material. The insulator is further surrounded with an outer conductor (usually a copper mesh). This outer conductor is wrapped in a plastic cover. The key to success of coaxial cable is its shielded design that allows the cable's copper core to transmit data quickly, without interference of environmental factors. These types of cables are used to carry signals of higher frequencies to a longer distance.



Backbone networks interconnect different segments of the network and provide a path to exchange information between different LANs or subnetworks..

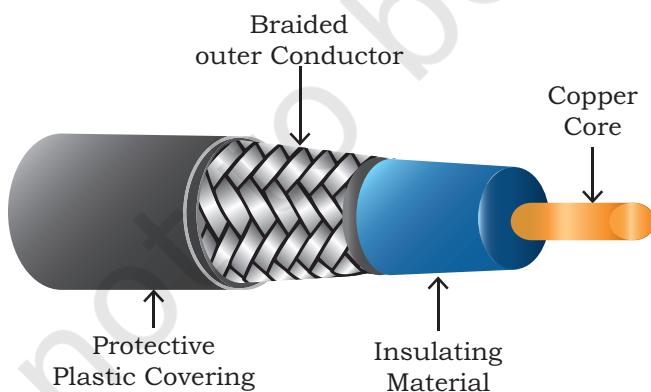


Figure 11.10: A coaxial cable



The number of oscillations a wave makes per second is called its frequency, and it is measured in Hz (Hertz).



### (C) Optical Fibre

The optical fiber cable carries data as light, which travels inside a thin fiber of glass (Figure 11.11). Optic fiber uses refraction to direct the light through the media. A thin transparent strand of glass at the centre is covered with a layer of less dense glass called cladding. This whole arrangement is covered with an outer jacket made of PVC or Teflon. Such types of cables are usually used in backbone networks. These cables are of light weight and have higher bandwidth which means higher data transfer rate. Signals can travel longer distances and electromagnetic noise cannot affect the cable. However, optic fibers are expensive and unidirectional. Two cables are required for full duplex communication.

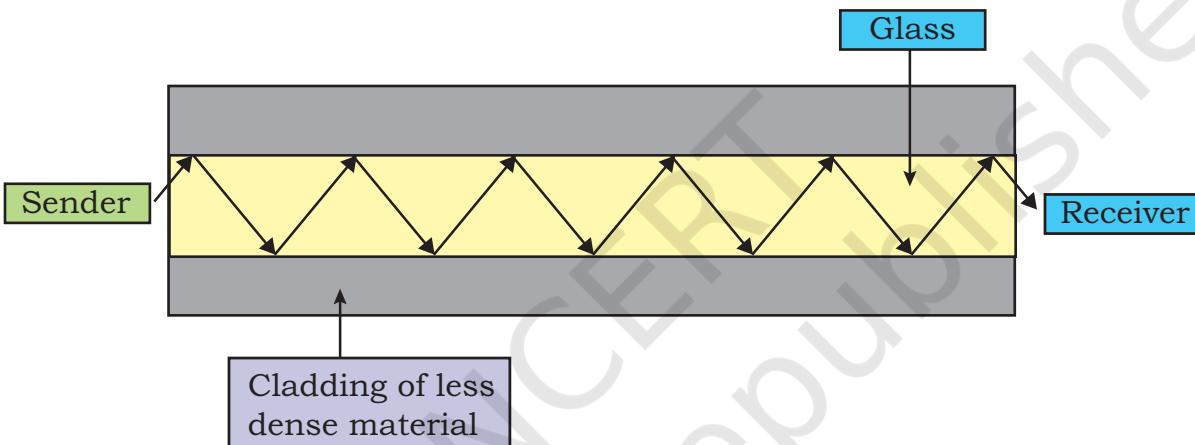


Figure 11.11: Fiber optic cable



Geostationary satellites orbiting around the Earth are used to deliver broadband Internet service, similar to the way satellite is used for television and telephone signals. These satellites use microwaves for communication between a satellite dish placed at our home and the hub of satellite internet service providers.



### 11.6.2 Wireless Transmission Media

In wireless communication technology, information travels in the form of electromagnetic signals through air. Electromagnetic spectrum of frequency ranging from 3 KHz to 900 THz is available for wireless communication (Figure 11.12). Wireless technologies allow communication between two or more devices in short to long distance without requiring any physical media. There are many types of wireless communication technologies such as Bluetooth, WiFi, WiMax etc.

The electromagnetic spectrum range (3KHz to 900THz) can be divided into 4 categories (Figure 11.12) - Radio waves, Microwaves, Infrared waves and Visible or Light waves, according to their frequency ranges. Some

of the properties of each wave are listed in Table 11.1. Of these, three are useful for wireless communication.

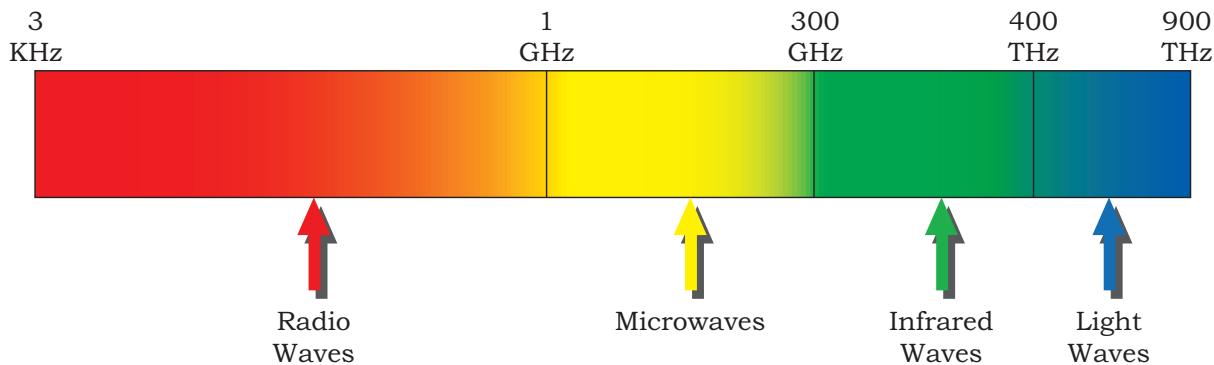


Figure 11.12: Electromagnetic waves spectrum

**Table 11.1 Classification of transmission waves and their properties**

| Transmission Waves | Properties                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Radio Waves        | <ul style="list-style-type: none"> <li>1. Waves of frequency range 3 KHz - 1 GHz</li> <li>2. Omni-directional, these waves can move in all directions</li> <li>3. Radio waves of frequency 300KHz-30MHz can travel long distance</li> <li>4. Susceptible to interference</li> <li>5. Radio waves of frequency 3-300KHz can penetrate walls</li> <li>6. These waves are used in AM and FM radio, television, cordless phones.</li> </ul>                                                                                                         |
| Microwaves         | <ul style="list-style-type: none"> <li>1. Electromagnetic waves of frequency range 1GHz - 300GHz.</li> <li>2. Unidirectional, can move in only one direction.</li> <li>3. Cannot penetrate solid objects such as walls, hills or mountains.</li> <li>4. Needs line-of-sight propagation i.e. both communicating antenna must be in the direction of each other.</li> <li>5. Used in point-to-point communication or unicast communication such as radar and satellite.</li> <li>6. Provide very large information-carrying capacity.</li> </ul> |
| Infrared waves     | <ul style="list-style-type: none"> <li>1. Electromagnetic waves of frequency range 300GHz - 400THz.</li> <li>2. Very high frequency waves.</li> <li>3. Cannot penetrate solid objects such as walls.</li> <li>4. Used for short-distance point-to-point communication such as mobile-to-mobile, mobile-to-printer, remote-control-to-TV, and Bluetooth-enabled devices to other devices like mouse, keyboards etc.</li> </ul>                                                                                                                   |

### 11.6.3 Wireless Technologies

#### (A) Bluetooth

Bluetooth is a short-range wireless technology that can be used to connect mobile-phones, mouse, headphones, keyboards, computers, etc. wirelessly over a short distance. One can print documents with bluetooth-

enabled printers without a physical connection. All these bluetooth-enabled devices have a low cost transceiver chip. This chip uses the unlicensed frequency band of 2.4 GHz to transmit and receive data. These devices can send data within a range of 10 meters with a speed of 1 - 2 Mbps.

In Bluetooth technology, the communicating devices within a range of 10 meters build a personal area network called piconet. The devices in a piconet work in a master-slave configuration. A master device can communicate with up to 7 active slave devices at the same time.

Bluetooth technology allows up to 255 devices to build a network. Out of them, 8 devices can communicate at the same time and remaining devices can be inactive, waiting for a response command from the master device.

### **(B) Wireless LAN**

This is another way of wireless communication. Wireless LAN is a local area network (LAN), and it is a popular way to connect to the Internet. The international organisation IEEE assigns numbers to each different standards of LAN. The wireless LAN is number as 802.11, and it is popularly known as Wi-Fi.

These networks consist of communicating devices such as laptops and mobile phones, as well as the network device called APs (access points) which is installed in buildings or floors (Figure 11.13). An access point is a device that is used to create a wireless local area network, by connecting to a wired router, switch, or hub. The APs are connected to a wired network, and all the devices communicate or access the Internet through an access point.

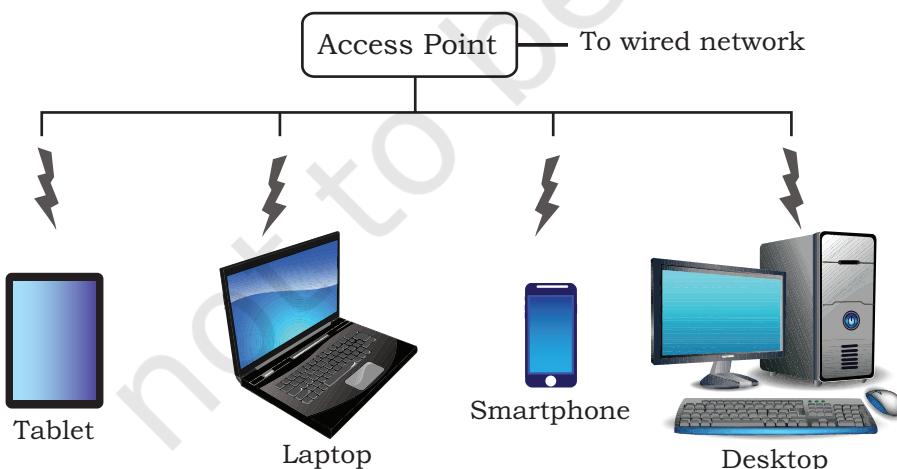


Figure 11.13: Access point creating a wireless LAN

The APs are connected to a wired network, and all the devices communicate or access the Internet through an access point.

Wi-Fi gives users the flexibility to move around within the network area while being connected to the network. Following are some of the benefits of WLAN:

- Wireless connections can be used to extend or replace an existing wired infrastructure
- Resulted in increased access for mobile devices
- Provides easy access to the Internet in public places

## 11.7 MOBILE TELECOMMUNICATION TECHNOLOGIES

Today the mobile phone network is the most used network in the world. The ability to be connected to the network on-the-go makes it very convenient to communicate with people via call or instant messages. It is also handy to access the Internet using the mobile phone network through wireless connection. Besides, the Internet of Things (IoT) is letting us control and communicate with other smart devices as well.

The architecture of the mobile network has rapidly evolved over the last few decades. The different landmark achievements in mobile communication technologies are classified as different generations. They are identified as 1G, 2G, 3G, 4G, and 5G. Let us briefly discuss the mobile telecommunication generations.

The first generation (1G) mobile network system came around 1982. It was used to transmit only voice calls. The analog signals were used to carry voices between the caller and receiver.

The second generation (2G) mobile network system came around 1991. Instead of analog signals, voice calls were transmitted in digital form thus providing improved call quality. This increased capacity allowed more people to talk simultaneously, and led to improved security as the signals could be encrypted. It also enabled an additional service to send SMS and MMS (Multimedia messages).

The third generation (3G) mobile network technology was developed during late 90s, but it was introduced commercially around 2001. It offered both digital voice and data services. 3G provided Internet access via the same radio towers that provide voice service to the mobile phone. It facilitated greater voice and data capacity. Therefore, more simultaneous calls could happen in the same frequency range and also a significantly faster data transfer speed.

Demand for faster data is always increasing and thus 4G mobile networks were developed and now



WiMax stands for Worldwide Interoperability for Microwave Access. Like Wi-Fi, it is also used for communication in wireless networks but there is a difference. Whereas Wi-Fi is used to form small wireless networks (WLANs), WiMax uses a larger spectrum to deliver connections to various devices on the network. It has a higher data transfer rate and can span over a larger area. That is why it is used in MAN applications.



### **Think and Reflect**

Explore how 5G can impact society.



5G networks have also come into being. 4G is much faster than 3G and this has revolutionised the field of telecommunication by bringing the wireless experience to a new level altogether. 4G systems support interactive multimedia, voice, video, wireless internet and other broadband services. Technologically, 4G is very different compared to 3G.

The fifth generation or 5G is currently under development. It is expected to be a milestone development for the success of IoT and Machine to Machine (M2M) communications. Machine to machine (M2M) is direct communication between devices — wired and wireless. 5G is expected to allow data transfer in Gbps, which is much faster than 4G. It is expected to be able to support all the devices of the future such as connected vehicles and the Internet of Things.

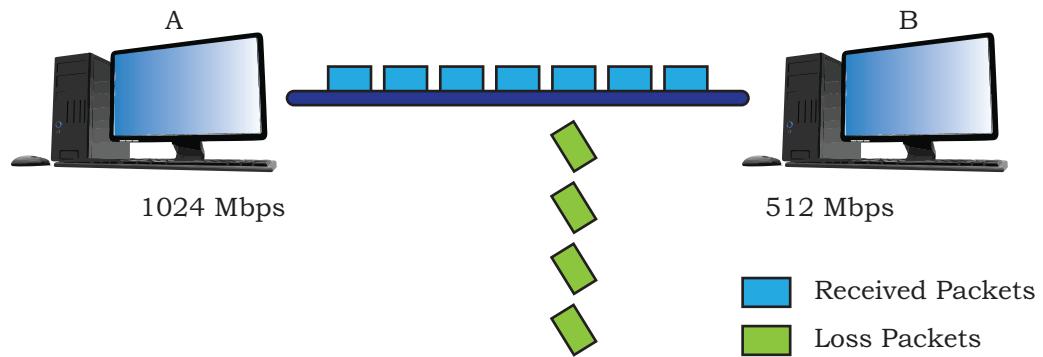
## **11.8 PROTOCOL**

In communication, Protocol is a set of standard rules that the communicating parties — the sender, the receiver, and all other intermediate devices need to follow. We know that the sender and receiver can be parts of different networks, placed at different geographic locations. Besides, the data transfer rates in different networks can vary, requiring data to be sent in different formats.

### **11.8.1 Need for Protocols**

We need protocols for different reasons such as flow control, access control, addressing, etc. Flow control is required when the sender and receiver have different speeds of sending and receiving the data. Figure 11.14 shows that Computer A is sending data at the speed of 1024 Mbps and computer B is receiving data at the speed of 512 Mbps. In this case, Computer B must be able to inform computer A about the speed mismatch so that computer A can adjust its data transmission rate. Otherwise some data will be lost, as shown in Figure 11.14.

Access control is required to decide which nodes in a communication channel will access the link shared among them at a particular instant of time. Otherwise, the transmitted data packets will collide if computers are sending data simultaneously through the same link resulting in the loss or corruption of data.



*Figure 11.14: Speed mismatch between two computers can result into loss of data*

### Protocols also define:

- how computers identify one another on a network.
- the form to which the data should be converted for transit.
- how to decide whether the data received is for that node or to be forwarded to another node.
- ensuring that all the data have reached the destination without any loss.
- how to rearrange the packets and process them at the destination.

If all the rules or protocols of a communication network are defined at one place, it becomes complex to ensure that communicating parties follow the guidelines. In this section, we will briefly talk about some of the protocols required in communication.

#### 11.8.2 HyperText Transfer Protocol (HTTP)

HTTP stands for HyperText Transfer Protocol. It is the primary protocol used to access the World Wide Web. Tim Berners-Lee led the development of HTTP at CERN in 1989 in collaboration with Internet Engineering Task Force (IETF) and the World Wide Web Consortium (W3C).

HTTP is a request-response (also called client-server) protocol that runs over TCP. The common use of HTTP is between a web browser (client) and a web server (server). HTTP facilitates access of hypertext from the World Wide Web by defining how information are formatted and transmitted, and how the Web servers and browsers should respond to various commands.



Hypertext refers to a document that contains images or text that can be linked to another document or text.



A web page is written using a markup language like HTML and is stored on a web server for access via its URL. Once a user opens a web browser and types in the URL of the intended web page, a logical communication link between the user machine (client) and the web server is created using HTTP.

For example, whenever we enter the URL `http://www.ncert.nic.in` in a browser, it sends HTTP request to the web-server where `ncert.nic.in` is hosted. The HTTP response from the web-server fetches and sends the requested Web-page, which is displayed on your browser.

#### **11.8.3 File Transfer Protocol (FTP)**

File Transfer Protocol (FTP) is the protocol used for transferring files from one machine to another. Like HTTP, FTP also works on a client-server model.

When a user requests for a file transfer with another system, FTP sets up a connection between the two nodes for accessing the file. Optionally, the user can authenticate using user ID and password. The user then specifies the file name and location of the desired file. After that, another connection sets up and the file transfer happens directly between the two machines. However, some servers provide FTP logins without authentication for accessing files.

File transfer between two systems seems simple and straightforward because FTP takes care of issues between two communicating devices, such as:

- use of different conventions while naming files.
- representation of text and data in different formats.
- having different directory structure

#### **11.8.4 Point to Point Protocol (PPP)**

PPP is a communication protocol which establishes a dedicated and direct connection between two communicating devices. This protocol defines how two devices will authenticate each other and establish a direct link between them to exchange data. For example, two routers with direct connection communicate using PPP. The Internet users who connect their home computers to the server of an Internet Service Provider (ISP) through a modem also use PPP.

The communicating devices should have duplex modes for using this protocol. This protocol maintains

data integrity ensuring that the packets arrive in order. It intimates the sender about damage or lost packets and asks to resend it.

#### **11.8.5 Simple Mail Transfer Protocol (SMTP)**

SMTP is a protocol used for email services. It uses information written on the message header (like an envelope on a letter sent by post), and is not concerned with the content of the email message. Each email header contains email addresses of recipients. The email containing header and body are entered into a queue of outgoing mails.

The SMTP sender program takes mails from the outgoing queue and transmits them to the destination(s). When the SMTP sender successfully delivers a particular mail to one or more destinations, it removes the corresponding receiver's email address from the mail's destination list. When that mail is delivered to all the recipients, it is removed from the outgoing queue. The SMTP receiver program accepts each mail that has arrived and places it in the appropriate user mailbox.

#### **11.8.6 Transmission Control Protocol (TCP)/ Internet Protocol (IP)**

TCP/IP stands for Transmission Control Protocol/Internet Protocol. It is a set of standardised rules that uses a client-server model of communication in which a user or machine (a client) requests a service by a server in the network.

The IP protocol ensures that each computer or node connected to the Internet is assigned an IP address, which is used to identify each node independently. It can be considered to be the adhesive that holds the whole Internet together. TCP ensures that the message or data is broken into smaller chunks, called IP packets. Each of these packets are routed (transmitted) through the Internet, along a path from one router to the next, until it reaches the specified destination. TCP guarantees the delivery of packets on the designated IP address. It is also responsible for ordering the packets so that they are delivered in sequence.

There are many redundant connection paths in the Internet, with backbones and ISPs connecting to each other in multiple locations. So, there are many

#### **Activity 11.3**

Find and list other Email Protocols.



possible paths between two hosts. Hence, two packets of the same message can take two different routes depending on congestion and other factors in different possible routes. When all the packets finally reach the destination machine, they are reassembled into the original message at the receiver's end.

### SUMMARY

- Data communication refers to the exchange of data between two or more networked or connected devices like laptops, PC, printers, routers etc.
- Sender, receiver, messages, channel and protocols are major components of data communication.
- In data communication, transmission media are the links that carry messages between two or more communicating devices. These are broadly classified into guided and unguided media.
- In guided transmission, there is a physical link made of wire/cable through which data in terms of signals are propagated between the nodes. These are usually metallic cable, fiber-optic cable, etc. They are also known as wired media.
- In unguided transmission, data travels in air in terms of electromagnetic waves using an antenna. They are also known as wireless media.
- The capacity of channels is measured in bandwidth. The unit of bandwidth is Hertz.
- Communication can be done in three different modes — simplex, half-duplex, and full-duplex communication.
- Switching techniques are alternative to dedicated lines whereby data is routed through various nodes in a network. It forms a temporary route for the data to be transmitted. Two commonly used switching techniques are – circuit switching and packet switching.
- Electromagnetic spectrum of frequency ranging from 3 KHz to 900 THz is available for wireless communication. This spectrum range (3KHz to 900THz) can be divided into four categories- Radio

- waves, Microwaves, Infrared waves and Visible or Light waves, according to their frequency ranges.
- Bluetooth is a short-range wireless technology that can be used to connect mobile-phones, mouse, headphones, keyboards, computers, etc. wirelessly over a short distance.
  - Based on the architecture of the mobile network, mobile communication technologies are classified into different generations identified as 1G, 2G, 3G, 4G, and 5G.
  - In communication, protocol is a set of standard rules that the communicating parties — the sender, the receiver, and all other intermediate devices need to follow. Flow control, access control, addressing, etc. are examples of protocol.
  - HTTP stands for HyperText Transfer Protocol. It is the primary protocol used to access the World Wide Web, which was developed by Tim Berners-Lee at CERN in 1989.
  - File Transfer Protocol (FTP) is the protocol used for transferring files from one machine to another. Like HTTP, FTP also works on a client-server model.
  - Point-to-Point protocol (PPP) defines how two devices will authenticate each other and establish a direct link between them to exchange data.
  - TCP/IP stands for Transmission Control Protocol/ Internet Protocol. It is a set of standardised rules that uses a client-server model of communication in which a user or machine (a client) requests a service by a server in the network.



## EXERCISE

1. What is data communication? What are the main components of data communication?
2. Which communication mode allows communication in both directions simultaneously?
3. Among LAN, MAN, and WAN, which has the highest speed and which one can cover the largest area?

## NOTES

4. What are three categories of wired media? Explain them.
5. Compare wired and wireless media.
6. Which transmission media carries signals in the form of light?
7. List out the advantages and disadvantages of optical fiber cable.
8. What is the range of frequency for radio waves?
9. 18 Gbps is equal to how many Bits per second?
10. HTTP stands for?
11. Write short note on the following:
  - a) HTTP
  - b) Bandwidth
  - c) Bluetooth
  - d) DNS
  - e) Data transfer rate
12. What is protocol in data communication? Explain with an example.
13. A composite signal contains frequencies between 500 MHz and 1GHz. What is the bandwidth of a signal?

Chapter

12

# Security Aspects



12130CH12

## In this Chapter

- » Threats and Prevention
- » Malware
- » Antivirus
- » Spam
- » HTTP vs HTTPS
- » Firewall
- » Cookies
- » Hackers and Crackers
- » Network Security Threats

*“Treat your password like your toothbrush. Don't let anybody else use it, and get a new one every six months.”*

— Clifford Stoll

### 12.1 THREATS AND PREVENTION

Being alone is the most ideal situation for an individual in terms of security. It applies to computers as well. A computer with no link to an external device or computer is free from the security threats arising otherwise. However, it is not an ideal solution for a human being or a computer to stay aloof in order to mitigate any security threats, as the world at present is on its way to become fully connected. This connectedness of various devices and computers has brought into our focus the various network threats and its prevention.

Network security is concerned with protection of our device as well as data from illegitimate access or misuse. Threats include all the ways in which one can exploit any vulnerability or weakness in a network or communication system in order to cause harm or damage one's reputation.

## 12.2 MALWARE

Malware is a short term used for MALicious softWARE. It is any software developed with an intention to damage hardware devices, steal data, or cause any other trouble to the user. Various types of malware have been created from time-to-time, and large-scale damages have been inflicted. Many of these malware programs have been identified and counter measures have been initiated. However, different types of malware keep on coming on a regular basis that compromise the security of computer systems and cause intangible damages. Besides, each year, malware incur financial damages worth billions of dollars worldwide. Viruses, Worms, Ransomware, Trojans, and Spyware are some of the kinds of malware.

### 12.2.1 Virus

The term computer virus was coined by Fred Cohen in 1985 and has been borrowed from biological science with almost similar meaning and behavior, the only difference is that the victim is a computer system and the virus is a malicious software. A virus is a piece of software code created to perform malicious activities and hamper resources of a computer system like CPU time, memory, personal files, or sensitive information.

Mimicking the behaviour of a biological virus, the computer virus spreads on contact with another system, i.e. a computer virus infects other computer systems that it comes into contact with by copying or inserting its code into the computer programs or software (executable files). A virus remains dormant on a system and is activated as soon as the infected file is opened (executed) by a user.

Viruses behave differently, depending upon the reason or motivation behind their creation. Some of the most common intentions or motives behind viruses include stealing passwords or data, corrupting files, spamming the user's email contacts, and even taking control of the user's machine. Some well-known viruses include CryptoLocker, ILOVEYOU, MyDoom, Sasser and Netsky, Slammer, Stuxnet, etc.

### 12.2.2 Worms

The Worm is also a malware that incurs unexpected or damaging behaviour on an infected computer system. The major difference between a worm and a virus is that

unlike a virus, a worm does not need a host program or software to insert its code into. Worms are standalone programs that are capable of working on its own. Also, a virus needs human triggering for replication (i.e. when a user opens/executes the infected file), while a worm replicates on its own and can spread to other computers through the network. Some prominent examples of worms include Storm Worm, Sobig, MSBlast, Code Red, Nimda, Morris Worm, etc.

### 12.2.3 Ransomware

It is a type of malware that targets user data. It either blocks the user from accessing their own data or threatens to publish the personal data online and demands ransom payment against the same. Some ransomware simply block the access to the data while others encrypt data making it very difficult to access. In May 2017, a ransomware WannaCry infected almost 200,000 computers across 150 countries. It worked by encrypting data and demanding ransom payments in the Bitcoin cryptocurrency. It literally made its victims “cry” and hence the name.



Figure 12.1: A ransomware

### 12.2.4 Trojan

Since the ancient Greeks could not infiltrate the city of Troy using traditional warfare methods, they gifted the king of Troy with a big wooden horse with hidden soldiers inside and eventually defeated them. Borrowing

the concept, a Trojan is a malware, that looks like a legitimate software and once it tricks a user into installing it, it acts pretty much like a virus or worm. However, a Trojan does not self-replicate or infect other files, it spreads through user interaction such as opening an email attachment or downloading and executing a file from the Internet. Some Trojans create backdoors to give malicious users access to the system.

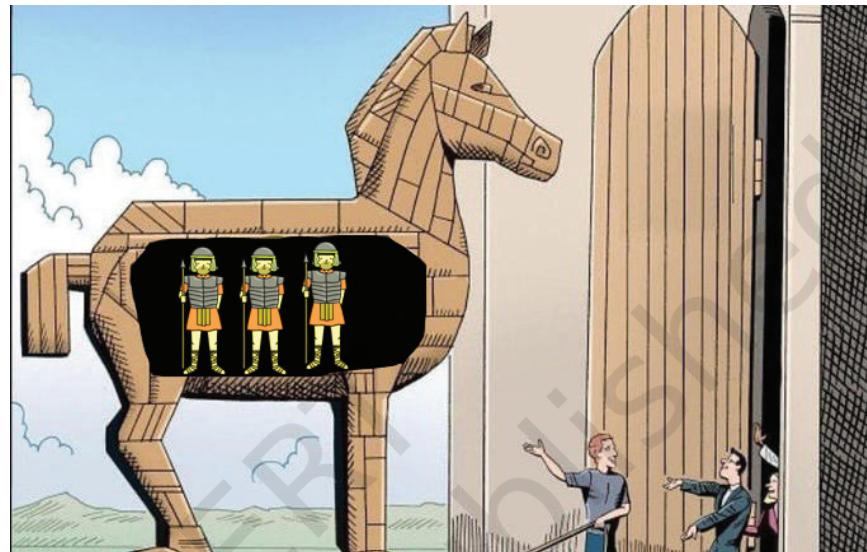


Figure 12.2: A trojan horse

#### **12.2.5 Spyware**

It is a type of malware that spies on a person or an organisation by gathering information about them, without the knowledge of the user. It records and sends the collected information to an external entity without consent or knowledge of the user.

Spyware usually tracks internet usage data and sells them to advertisers. They can also be used to track and capture credit card or bank account information, login and password information or user's personal identity.

#### **12.2.6 Adware**

An Adware is a malware that is created to generate revenue for its developer. An adware displays online advertisements using pop-ups, web pages, or installation screens. Once an adware has infected a substantial number of computer systems, it generates revenue either by displaying advertisements or using “pay per click” mechanism to charge its clients against the number of clicks on their displayed ads. Adware

is usually annoying, but harmless. However, it often paves way for other malware by displaying unsafe links as advertisements.

### 12.2.7 Keyloggers

A keylogger can either be malware or hardware. The main purpose of this malware is to record the keys pressed by a user on the keyboard. A keylogger makes logs of daily keyboard usage and may send it to an external entity as well. In this way, very sensitive and personal information like passwords, emails, private conversations, etc. can be revealed to an external entity without the knowledge of the user. One strategy to avoid the threat of password leaks by keyloggers is to use a virtual keyboard while signing into your online accounts from an unknown computer.



To implement a keylogger in hardware, a thin transparent keyboard is placed atop the actual keyboard or input pad of the intended machine, which then records the keystrokes pressed by the user.



#### (A) Online Virtual Keyboard Vs On-Screen Keyboard

The names “on-screen” and “virtual” keyboard refer to any software-based keyboard and are sometimes used interchangeably. But, there exists a notable difference between “on-screen” and “online virtual” keyboards. Both types of keyboards may look the same, but the difference is in terms of the layout or ordering of the keys. The on-screen keyboard of an operating system uses a fixed QWERTY key layout (Figure 12.3), which can be exploited by sophisticated keylogger software. However, an online virtual keyboard randomises the key layout every time it is used (Figure 12.4), thereby making it very difficult for a keylogger software to know or record the key(s) pressed by the user.

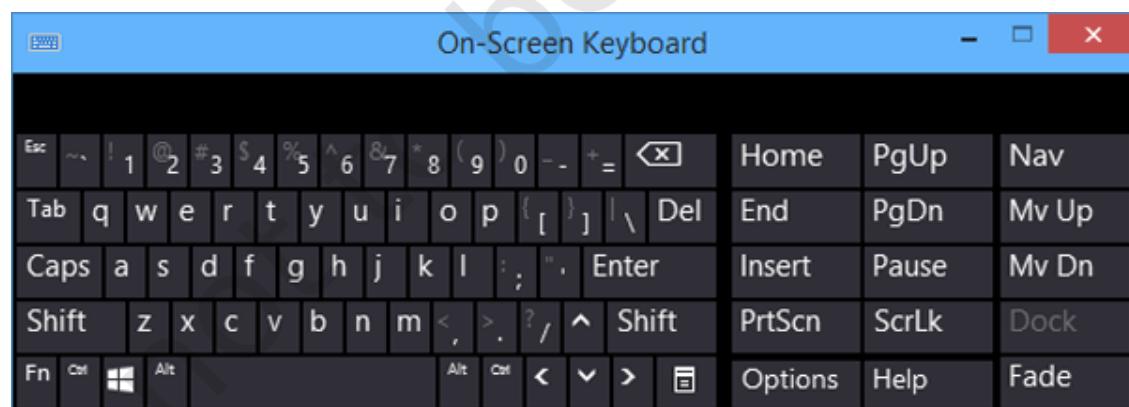


Figure 12.3: A QWERTY keyboard layout

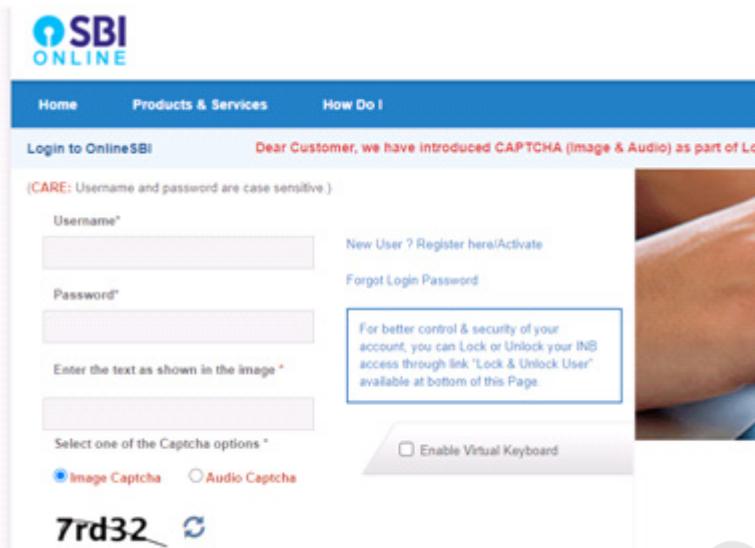


Figure 12.4: Online virtual keyboard

### 12.2.8 Modes of Malware distribution

A malware once designed, can take many routes to reach your computer. Some of the common distribution channels for malware are:

- **Downloaded from the Internet:** Most of the time, malware is unintentionally downloaded into the hard drive of a computer by the user. Of course, the malware designers are smart enough to disguise their malware, but we should be very careful while downloading files from the Internet (especially those highlighted as free stuff).
- **Spam Email:** We often receive an unsolicited email with embedded hyperlinks or attachment files. These links or attached files can be malware.
- **Removable Storage Devices:** Often, the replicating malware targets the removable storage media like pen drives, SSD cards, music players, mobile phones, etc. and infect them with malware that gets transferred to other systems that they are plugged into.
- **Network Propagation:** Some malware like Worms have the ability to propagate from one computer to another through a network connection.

### 12.2.9 Combating Malware

Common signs of some malware infection include the following:

- frequent pop-up windows prompting you to visit some website and/or download some software;
- changes to the default homepage of your web browser;
- mass emails being sent from your email account;
- unusually slow computer with frequent crashes;
- unknown programs startup as you turn on your computer;
- programs opening and closing automatically;
- sudden lack of storage space, random messages, sounds, or music start to appear;
- programs or files appear or disappear without your knowledge.

Malware exists and continues to evolve, and so is the mechanism to combat them. As the saying goes that prevention is better than cure, we list some preventive measures against the malware discussed earlier.

- ✓ Using antivirus, anti-malware, and other related software and updating them on a regular basis.
- ✓ Configure your browser security settings
- ✓ Always check for a lock button in the address bar while making payments.
- ✓ Never use pirated or unlicensed software. Instead go for Free and Open Source Software (FOSS).
- ✓ Applying software updates and patches released by its manufacturers.
- ✓ Taking a regular backup of important data.
- ✓ Enforcing firewall protection in the network.
- ✓ Avoid entering sensitive (passwords, pins) or personal information on unknown or public computers.
- ✓ Avoid entering sensitive data on an unknown network (like Wi-Fi in a public place) using your own computer also.
- ✓ Avoid clicking on links or downloading attachments from unsolicited emails.
- ✓ Scan any removable storage device with an antivirus software before transferring data to and from it.
- ✓ Never share your online account or banking password/pins with anyone.
- ✓ Remove all the programs that you don't recognise from your system.

- ✓ Do not install an anti-spyware or antivirus program presented to you in a pop-up or ad.
- ✓ Use the pop-up window's 'X' icon located on the top-right of the popup to close the ad instead of clicking on the 'close' button in the pop-up. If you notice an installation has been started, cancel immediately to avoid further damage.

## 12.3 ANTIVIRUS

Antivirus is a software, also known as anti-malware. Initially, antivirus software was developed to detect and remove viruses only and hence the name anti-virus. However, with time it has evolved and now comes bundled with the prevention, detection, and removal of a wide range of malware.

### 12.3.1 Methods of Malware Identification used by Antivirus

#### *(A) Signature-based detection*

In this method, an antivirus works with the help of a signature database known as "Virus Definition File (VDF)". This file consists of virus signatures and is updated continuously on a real-time basis. This makes the regular update of the antivirus software a must. If there is an antivirus software with an outdated VDF, it is as good as having no antivirus software installed, as the new malware will infect the system without getting detected. This method also fails to detect malware that has an ability to change its signature (polymorphic) and the malware that has some portion of its code encrypted.

#### *(B) Sandbox detection*

In this method, a new application or file is executed in a virtual environment (sandbox) and its behavioural fingerprint is observed for a possible malware. Depending on its behaviour, the antivirus engine determines if it is a potential threat or not and proceeds accordingly. Although this method is a little slow, it is very safe as the new unknown application is not given access to actual resources of the system.

#### *(C) Data mining techniques*

This method employs various data mining and machine learning techniques to classify the behaviour of a file as either benign or malicious.



#### **Virus Signature**

A virus signature is a consecutive sequence of bytes that is commonly found in a certain malware sample. That means it's contained within the malware or the infected file and not in unaffected files.



#### **(D) Heuristics**

Often, a malware infection follows a certain pattern. Here, the source code of a suspected program is compared to viruses that are already known and are in the heuristic database. If the majority of the source code matches with any code in the heuristic database, the code is flagged as a possible threat.

#### **(E) Real-time protection**

Some malware remains dormant or gets activated after some time. Such malware needs to be checked on a real-time basis. In this technique, the anti-malware software keeps running in the background and observes the behavior of an application or file for any suspicious activity while it is being executed i.e. when it resides in the active (main) memory of the computer system.

### **12.4 SPAM**

Spam is a broad term and applies to various digital platforms like messaging, forums, chatting, emailing, advertisement, etc. However, the widely recognised form is email spam. Depending on their requirements, organisations or individuals buy or create a mailing list (list of email addresses) and repeatedly send advertisement links and invitation emails to a large number of users. This creates unnecessary junk in the inbox of the receiver's email and often tricks a user into buying something or downloading a paid software or malware.

Nowadays, email services like Gmail, Hotmail, etc. have an automatic spam detection algorithm that filters emails and makes things easier for the end users. A user can also mark an undetected unsolicited email as "spam", thereby ensuring that such type of email is not delivered into the inbox as normal email in future.

### **12.5 HTTP vs HTTPS**

Both the HTTP (Hyper Text Transfer Protocol) and its variant HTTPS (Hyper Text Transfer Protocol Secure) are a set of rules (protocol) that govern how data can be transmitted over the WWW (World Wide Web). In other words, they provide rules for the client web browser and servers to communicate.

HTTP sends information over the network as it is. It does not scramble the data to be transmitted, leaving



Always look for the "https:///" at the beginning of the address (URL) of the websites while entering your banking, personal, or other sensitive information.



it vulnerable to attacks from hackers. Hence, HTTP is sufficient for websites with public information sharing like news portals, blogs, etc. However, when it comes to dealing with personal information, banking credentials and passwords, we need to communicate data more securely over the network using HTTPS. HTTPS encrypts the data before transmission. At the receiver end, it decrypts to recover the original data. The HTTPS based websites require SSL Digital Certificate.

### Activity 12.1

Ask your teacher to show you how to enable and disable firewall on your computer.



## 12.6 FIREWALL

Computer firewall is a network security system designed to protect a trusted private network from unauthorised access or traffic originating from an untrusted outside network (e.g., the Internet or different sections of the same network) to which it is connected (Figure 12.5). Firewall can be implemented in software, hardware or both. As discussed earlier, a malware like worm has the capability to move across the networks and infect other computers. The firewall acts as the first barrier against malware.

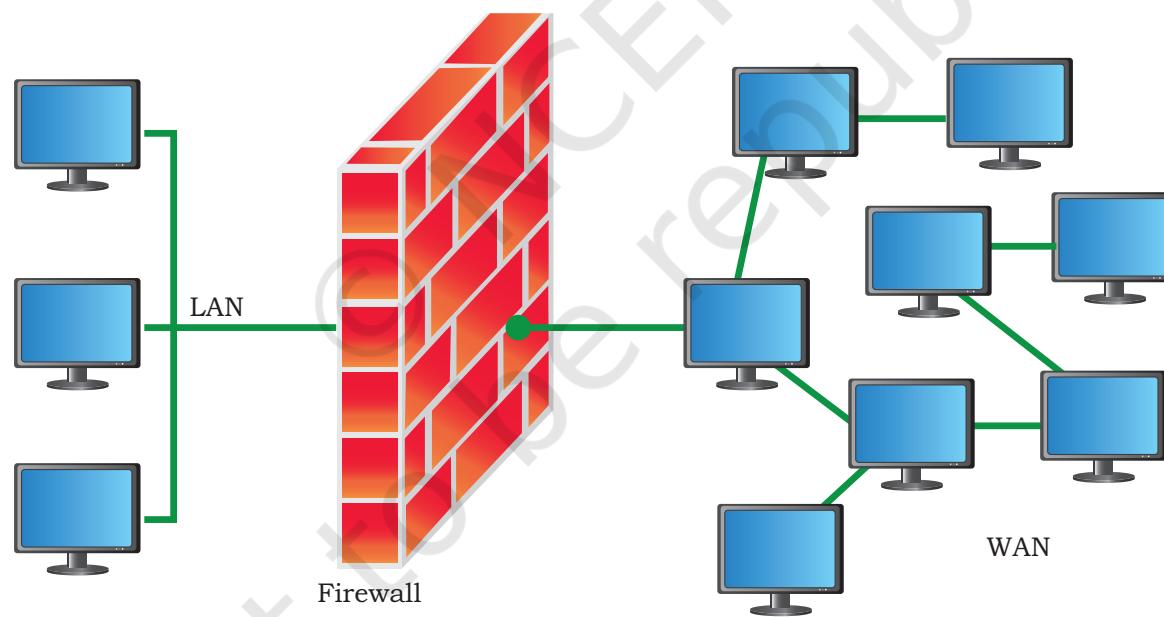


Figure 12.5: A firewall between two networks

A firewall acts as a network filter and based on the predefined security rules, it continuously monitors and controls the incoming and outgoing traffic. As an example, a rule can be set in the firewall of a school LAN, that a student cannot access data from the finance

server, while the school accountant can access the finance server.

### 12.6.1 Types of Firewall

- Network Firewall: If the firewall is placed between two or more networks and monitors the network traffic between different networks, it is termed as Network Firewall.
- Host-based Firewall: If the firewall is placed on a computer and monitors the network traffic to and from that computer, it is called a host-based firewall.

## 12.7 COOKIES

The term "cookie" was derived from the term "magic cookie" used by Unix programmers to indicate a packet of data that a program receives and sends it back unchanged. A computer cookie is a small file or data packet, which is stored by a website on the client's computer. A cookie is edited only by the website that created it, the client's computer acts as a host to store the cookie. Cookies are used by the websites to store browsing information of the user. For example, while going through an e-commerce website, when a user adds items to cart, the website usually uses cookies to record the items in the cart. A cookie can also be used to store other user-centric information like login credentials, language preference, search queries, recently viewed web pages, music choice, favorite cuisine, etc., that helps in enhancing the user experience and making browsing time more productive.

Depending upon their task, there are different types of cookies. Session cookies keep track of the current session and even terminate the session when there is a time-out (banking website). So, if you accidentally left your e-banking page open, it will automatically close after the time-out. Similarly, authentication cookies are used by a website to check if the user is previously logged in (authenticated) or not. This way, you don't need to login again and again while visiting different web pages or links of the same website. You might have also noticed that certain information like your Name, Address, Contact, D.O.B, etc. automatically fills up while filling an online form. This auto-fill feature is also implemented by websites using cookies.

### Think and Reflect

Assume students in a class are to finish their project. For this, the access to the Internet has also been given. To ensure maximum output i.e. timely completion, can you utilise Firewall to prevent distraction while surfing the net?



### Activity 12.2

Open your internet browser and check the settings for cookies. Also, try to locate some cookie files on your computer system.



### **12.7.1 Threats due to Cookies**

Usually, cookies are used for enhancing the user's browsing experience and do not infect your computer with malware. However, some malware might disguise as cookies e.g. "supercookies". There is another type of cookie known as "Zombie cookie" that gets recreated after being deleted. Some third-party cookies might share user data without the consent of the user for advertising or tracking purposes. As a common example, if you search for a particular item using your search engine, a third-party cookie will display advertisements showing similar items on other websites that you visit later. So, one should be careful while granting permission to any websites to create and store cookies on the user computer.

## **12.8 HACKERS AND CRACKERS**

Hackers and crackers are people having a thorough knowledge of the computer systems, system software (operating system), computer networks, and programming. They use this knowledge to find loopholes and vulnerabilities in computer systems or computer networks and gain access to unauthorised information. In simple terms, a hacker is a person that is skilled enough to hack or take control of a computer system. Depending on the intent, there are different types of hackers.



A hacktivist is a hacker with an aim to bring about political and social change.



### **12.8.1 White Hats: Ethical Hacker**

If a hacker uses its knowledge to find and help in fixing the security flaws in the system, it is termed as White Hat hacker. These are the hackers with good intentions. They are actually security experts. Organisations hire ethical or white hat hackers to check and fix their systems for potential security threats and loopholes. Technically, white hats work against black hats.

### **12.8.2 Black Hats: Crackers**

If hackers use their knowledge unethically to break the law and disrupt security by exploiting the flaws and loopholes in a system, then they are called black hat hackers.

### **12.8.3 Grey Hats**

The distinction between different hackers is not always clear. There exists a grey area in between, which

represents the class of hackers that are neutral, they hack systems by exploiting its vulnerabilities, but they don't do so for monetary or political gains. The grey hats take system security as a challenge and just hack systems for the fun of it.

## 12.9 NETWORK SECURITY THREATS

### 12.9.1 Denial of Service

Denial of Service (DoS) is a scenario, wherein an attacker (Hacker) limits or stops an authorised user to access a service, device, or any such resource by overloading that resource with illegitimate requests. The DoS attack floods the victim resource with traffic, making the resource appear busy. If attackers carry out a DoS attack on a website, they will flood it with a very large number of network packets by using different IP addresses. This way, the web server would be overloaded and will not be able to provide service to a legitimate user. The users will think that the website is not working, causing damage to the victim's organisation. Same way, DoS attacks can be done on resources like email servers, network storage, disrupting connection between two machines or disrupting the state of information (resetting of sessions).

If a DoS attack makes a server crash, the server or resource can be restarted to recover from the attack. However, a flooding attack is difficult to recover from, as there can be some genuine legitimate requests in it as well.

A variant of DoS, known as Distributed Denial of Service (DDoS) is an attack, where the flooded requests come from compromised computer (Zombies) systems distributed across the globe or over a very large area. The attacker installs a malicious software known as Bot on the Zombie machines, which gives it control over these machines. Depending upon the requirement and availability, the attacker activates a network of these Zombie computers known as Bot-Net to carry out the DDoS attack. While as a simple DoS attack may be countered by blocking requests or network packets from a single source, DDoS is very difficult to resolve, as the attack is carried from multiple distributed locations.

## 12.9.2 Intrusion Problems

Network Intrusion refers to any unauthorised activity on a computer network. These activities may involve unauthorised use of network resources (DoS) or threatening the security of the network and the data. Network intrusion is a very serious problem and the network administrator needs to devise strategy and implement various security measures to protect the network. We have already discussed some of the intrusion attacks such as DoS, Trojans, and Worms. The remaining attacks are briefly discussed below.

### (A) Asymmetric Routing

The attacker tends to avoid detection by sending the intrusion packets through multiple paths, thereby bypassing the network intrusion sensors.

### (B) Buffer Overflow Attacks

In this attack, the attacker overwrites certain memory areas of the computers within the network with code (set of commands) that will be executed later when the buffer overflow (programming error) occurs. Once the malicious code is executed, an attacker can initiate a DoS attack or gain access to the network.

### (C) Traffic Flooding

It is one of the most trivial methods of network intrusion. It involves flooding the network intrusion detection system with message packets. This huge load leaves the network detection system incapable of monitoring the packets adequately. The hacker takes advantage of this congested and chaotic network environment to sneak into the system undetected.

## 12.9.3 Snooping

Snooping means secretly listening to a conversation. In the context of networking, it refers to the process of secret capture and analysis of network traffic. It is a computer program or utility that has a network traffic monitoring capability. In this attack, the hacker taps or listens to a channel of communication by picking all of the traffic passing through it. Once the network packets are analysed by the snooping device or software, it reproduces the exact traffic packets and places them back in the channel, as if nothing has happened. So, if the data that is being sent over the network is not encrypted, it is vulnerable to snooping and eventually



### URL Snooping

It is a software package that downloads and stores a web stream as a file, that can be viewed or used later. The common online video downloaders use the same techniques to download videos from the Web.



may cause serious damage, depending upon the type of information leak. However, snooping is not always an attack, at times it is also used by network administrators for troubleshooting various network issues. Snooping is also known as Sniffing.

Various snooping software exist that act as network traffic analyser. Besides, various network hubs and switches have a SPAN (Sniffer Port Analyser) port function for snooping.

#### **12.9.4 Eavesdropping**

The term eavesdropping has been derived from the literal practice of secretly listening to the conversations of people by standing under the eaves of a house. Unlike snooping, where the network traffic can be stored for later analysis, eavesdropping is an unauthorised real-time interception or monitoring of private communication between two entities over a network. Also, the targets



Figure 12.6: Eavesdropping

are usually the private communication channels like phone calls (VoIP), instant messages, video conference, fax transmission, etc. In older days, eavesdropping was performed on the conventional telephone line and was known as wiretapping. Digital devices like laptops and cell phones that have a built-in microphone or camera can be easily hacked and eavesdropped using rootkit malware.

Eavesdropping is different from Snooping. While the former happens in real time, the latter does not. As an

example, in eavesdropping, imagine someone listening to your private conversation with the help of a hidden microphone in your room or by physically standing near the window of your room. However, in snooping, that person may make a copy of a letter that is addressed to your friend and keep the copy with himself and send the original letter to the intended address.

## SUMMARY

- Malware is a software developed with an intention to damage computer hardware, software, steal data, or cause any other trouble to a user.
- A virus is a piece of software code created to perform malicious activities and hamper resources of a computer system.
- The Worm is also a malware that incurs unexpected or damaging behaviour on an infected computer system.
- Worms are standalone programs that are capable of working on its own.
- Ransomware is a type of malware that targets user data.
- Ransomware either blocks the user from accessing their own data or threatens to publish their personal data online and demands ransom payment against the same.
- Trojan is a malware, that looks like a legitimate software and once it tricks a user into installing it, it acts pretty much like a virus or a worm.
- Spyware records and sends the collected information to an external entity without the consent or knowledge of a user.
- An adware displays unwanted online advertisements using pop-ups, web pages, or installation screens.
- A keylogger makes logs of daily keyboard usage and may send it to an external entity as well.
- The on-screen keyboard is an application software that uses a fixed QWERTY key layout.
- Online virtual keyboard is a web-based or a standalone software with a randomised key layout every time it is used.
- A malware can take many routes to reach your computer, which include: Downloaded from the

Internet, Spam Email, using infected Removable Storage Devices, and network propagation.

- An antivirus software is used to detect and remove viruses and hence the name anti-virus.
- Antiviruses now come bundled with the prevention, detection, and removal of a wide range of malware.
- Some of the prominent methods of malware identification used by an antivirus include: Signature-based detection, Sandbox detection, Heuristics.
- Any unwanted data, information, email, advertisement, etc. is called Spam.
- HTTP (Hyper Text Transfer Protocol) and HTTPS (Hyper Text Transfer Protocol Secure) are a set of rules or protocol that govern how data can be transmitted over the World Wide Web.
- Firewall is a network security system designed to protect a trusted private network from unauthorised access or traffic originating from an untrusted external network.
- There are two basic types of firewalls — Network Firewall and Host-based Firewall.
- A computer cookie is a small file or data packet, which is stored by a website on the client's computer.
- Cookies are used by the websites to store browsing information of the user.
- Hackers/Crackers find loopholes and vulnerabilities in computer systems or computer networks and gain access to unauthorised information.
- If a hacker uses its knowledge to find and help in fixing the security flaws in the system, it's termed as White Hat hacker.
- If hackers use their knowledge unethically to break the law and disrupt security by exploiting the flaws and loopholes in a system, then they are called black hat hackers.
- The grey hats take system security as a challenge and just hack systems for the fun of it.
- The Denial of Service (DoS) attack floods the victim resource with traffic, making the resource appear busy.
- Distributed Denial of Service (DDoS) is an attack, where the flooded requests come from

compromised computer (Zombies) systems distributed across the globe or over a very large area.

- Network Intrusion refers to any unauthorised activity on a computer network.
- Snooping is the process of secret capture and analysis of network traffic by malicious users.
- Eavesdropping is an unauthorised real-time interception or monitoring of private communication between two entities over a network.



## EXERCISE

1. Why is a computer considered to be safe if it is not connected to a network or Internet?
2. What is a computer virus? Name some computer viruses that were popular in recent years.
3. How is a computer worm different from a virus?
4. How is Ransomware used to extract money from users?
5. How did a Trojan get its name?
6. How does an adware generate revenue for its creator?
7. Briefly explain two threats that may arise due to a keylogger installed on a computer.
8. How is a Virtual Keyboard safer than On Screen Keyboard?
9. List and briefly explain different modes of malware distribution.
10. List some common signs of malware infection.
11. List some preventive measures against malware infection.
12. Write a short note on different methods of malware identification used by antivirus software.
13. What are the risks associated with HTTP? How can we resolve these risks by using HTTPS?
14. List one advantage and disadvantage of using Cookies.
15. Write a short note on White, Black, and Grey Hat Hackers.
16. Differentiate between DoS and DDoS attack.
17. How is Snooping different from Eavesdropping?

Chapter

13

# Project Based Learning



12130CH13



## In this Chapter

- » *Introduction*
- » *Approaches for Solving Projects*
- » *Teamwork*
- » *Project Descriptions*

*“An idea that is developed and put into action is more important than idea that exists only as an idea.”*

— Gautam Buddha

### 13.1 INTRODUCTION

Project based learning gives a thorough practical exposure to students regarding a problem upon which the project is based. Through project based learning, students learn to organise their project and use their time effectively for successful completion of the project. Projects are developed generally in groups where students can learn various skills such as working together, problem solving, decision making, and investigating activities. Project based learning involves the steps such as analysing the problem, formulating the problem into small modules, applying the mechanism or method to solve each module and then integrating the solution of all the modules to arrive at the complete solution of the problem. To solve a problem, it is required that those who work on it gather the relevant data and process it by applying a particular method. Data may

be collected as per the requirement of the project in a particular format. All the team members should be associated to accomplish the task. After collecting data, it should be processed to solve the problem. The results should be reported in a predetermined format.

### 13.2 APPROACHES FOR SOLVING PROJECTS

The approach followed for the development and completion of a project plays a pivotal role in project based learning. There are several approaches to execute a project such as modular approach, top down approach and bottom up approach. A structured or a modular approach to a project means that a project is divided into various manageable modules and each of the modules has a well-defined task to be performed with a set of inputs. This would lead to a set of outputs which when integrated leads to the desired outcome.

Different steps involved in project based learning (Figure 13.1) are :

- (1) **Identification of a project:** The project idea may come through any real-life situation. For example, one could think of doing a project for organising a seminar. One needs to understand the usefulness of the project and its impact. Students must be encouraged to undertake interdisciplinary projects.
- (2) **Defining a plan:** Normally for any kind of project, there are several project members involved in it. One project leader has to be identified. The roles of project leader and each project member have to be clearly defined. Students who are performing a project must be assigned with specific activities. The various tools for executing these activities must be known. To obtain a better solution, one should always think of the extreme situations.
- (3) **Fixing of a time frame and processing:** Every project is a time relevance project. A student must understand the importance of time frame for completion of the project. All the activities which are performed in the projects require a certain amount of time. Every project must be well structured and at the same time it must be flexible in its time frame.

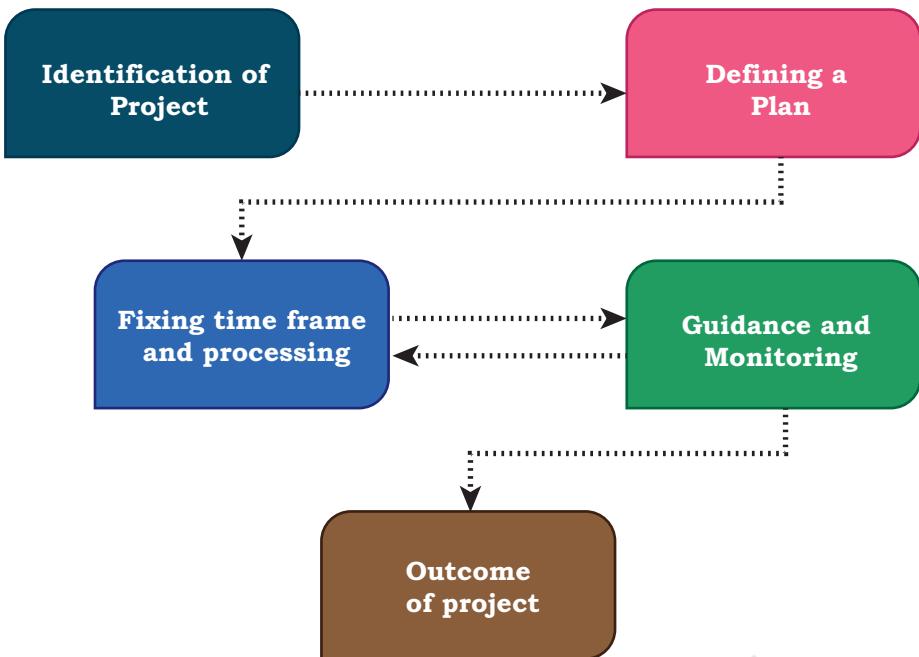


Figure 13.1: Steps in project based learning

#### (4) Providing guidance and monitoring a project:

Many times, the participants in the project get stuck up with a particular process and it becomes impossible to proceed further. In such a case, they need guidance, which can be obtained from various resources such as books, websites and experts in the field. While it is essential that the project leader should ensure monitoring of the project, the guide teacher also helps in monitoring the project.

#### (5) Outcome of a project:

One needs to understand thoroughly the outcome of a project. The outcome can be single, or it can be multiple. The output of a project can be peer reviewed and can be modified as per the feedback from the guide teacher or other users.

### 13.3 TEAMWORK

Many real-life tasks are very complex and require a lot of individuals to contribute in achieving them. Efforts made by individuals collectively to accomplish a task is called teamwork.

For example, in many sports, there is a team of players. These players play together to win a match. Take an example of a cricket team. We find that even if a bowler bowls a good ball but if the fielder cannot take

a catch then the wicket cannot be taken. So, in order to take a catch, efforts of a bowler as well as of fielders are needed. To win a cricket match, contributions from all the team members in all the three areas batting, bowling and fielding are required.

### **13.3.1 Components of Teamwork**

Apart from technical proficiency, a wide variety of other components make a successful teamwork. It comprises skilled team members with specific roles to achieve the goal.

#### **(A) Communicate with Others**

When a group of individuals perform one job, it is necessary to have effective communication between the members of the team. Such communication can be done via e-mails, telephones or by arranging group meetings. This helps the team members to understand each other and sort out their problems to achieve the goal effectively.

#### **(B) Listen to Others**

It is necessary to understand the ideas of others while executing a job together. This can be achieved when the team members listen to each other in group meetings and follow steps that are agreed upon.

#### **(C) Share with Others**

Ideas, images and tools need to be shared with each other in order to perform a job. Sharing is an important component of teamwork. Any member of the team who is well versed in a certain area should share the expertise and experience with others to effectively achieve the goal within the time frame.

#### **(D) Respect for Others**

Every member of the team must be treated respectfully. All the thoughts and ideas that are put forth in the group meetings may be respected and duly considered. Not respecting the views of a particular member may cause problems and that particular team member may not give his best.

#### **(E) Help Others**

A helping hand from every member is a key to success. Sometimes help from people who are not a part of the team is also obtained in order to accomplish a job.

**(F) Participate**

All the team members must be encouraged by each other to participate in completing the project and also in discussions in group meetings. Also, every member should take an active participation so that they feel their importance in the team.

### **13.4 PROJECT DESCRIPTIONS**

In this section, some examples of project works are given, which can be taken up in groups under project based learning. However, a group may choose any other project in consultation with the guide teacher.

#### **Project Title 1: Automation of Order Processing in a Restaurant**

##### **Description**

A new restaurant “Stay Healthy” is coming up in your locality. The owner/management of the restaurant wants to use a computer to generate bills and maintain other records of the restaurant. Your team is asked to develop an application software to automate the order placing and associated processes.

##### **Specifications**

Make a group of students to undertake a project on automating the order processing of the restaurant ‘Stay Healthy’. The owner of the restaurant wants the following specific functionalities to be made available in the developed application:

- There should be two types of Login options — one for the manager of the joint and other for the customer.
- Kiosk(s) running the software for customers will be placed at reception for placing the order. On the opening screen, menu for placing orders will be displayed.
- To place orders, customers will enter Item Code(s) and quantity desired.
- After placing an order, a soft copy of the bill will be displayed on the kiosk, having an Order Number.
- Every bill will have a unique identification (such as combination of date, and order number of the day) and should be saved in the data file/database.
- Order Number starts from 1 every day.

- For Manager login—provision for entry/change of Menu, deletion of Order (on demand) and generation of following report is desired.
  - ✓ A Report giving Summary of the Sales made on a Day. Program should accept the date for which the Summary is required.
- Add at least one more relevant report of your choice to the program.

### **Project Title 2 : Development of a Puzzle**

#### **Description**

Implement a puzzle solving game in Python. The game presents a grid board composed of cells to the player, in which some cells have Bomb. Player is required to clear the board (of the bomb), without detonating any one of them with the help of clue(s) provided on the board.

#### **Specifications**

For clearing the board, the player will click a cell on the board, if the cell contains a bomb, the game finishes. If the cell does not contain a bomb, then the cell reveals a number giving a clue about the number of bombs hidden in adjacent cells.

Before you start coding the game, play any Minesweeper game five times. This will help you in proper understanding of your project. To reduce the complexity of the program you can fix the grid size to 6x6 and number of bombs to 6.

**Note:** Do ensure to handle various exception(s) which may occur while playing the game, in your code.

### **Project Title 3 : Development of an Educational Game**

#### **Description**

You are a member of the ICT club of your school. As a club member, you are given the responsibility of identifying ways to improve mathematical skills of kids, in the age group of 5-7 years. One of the club members suggested developing an Edutainment Game named “Match the Sum” for it. Match the Sum will hone summing skills of student(s), by allowing them to form number 10 by adding 2/3 digits.

## Specifications

Following are the details of provisions required for program:

- Display a list of 15 cells on screen, where each cell can hold a digit (1 to 9)
- Randomly generate a digit at a time and place it in the list from the right end. Program will keep on generating digits at equal intervals of time and place it in the rightmost cell. (Already existing digits, will be shifted left, by one cell, with every new addition of digits' in the list)
- For playing the game, students' will be allowed to type 2/3 digits (one at a time) currently displayed in the list of cells.
- If the sum of those digits is 10, then those digits should get removed from the list of cells.
- Game will continue till there is an empty cell to insert a digit in the list of cells.

**Note:** Do take care of the situation when digits displayed in a list of cells do not add up to 10.

## NOTES

## NOTES

---

not to be republished © NCERT