

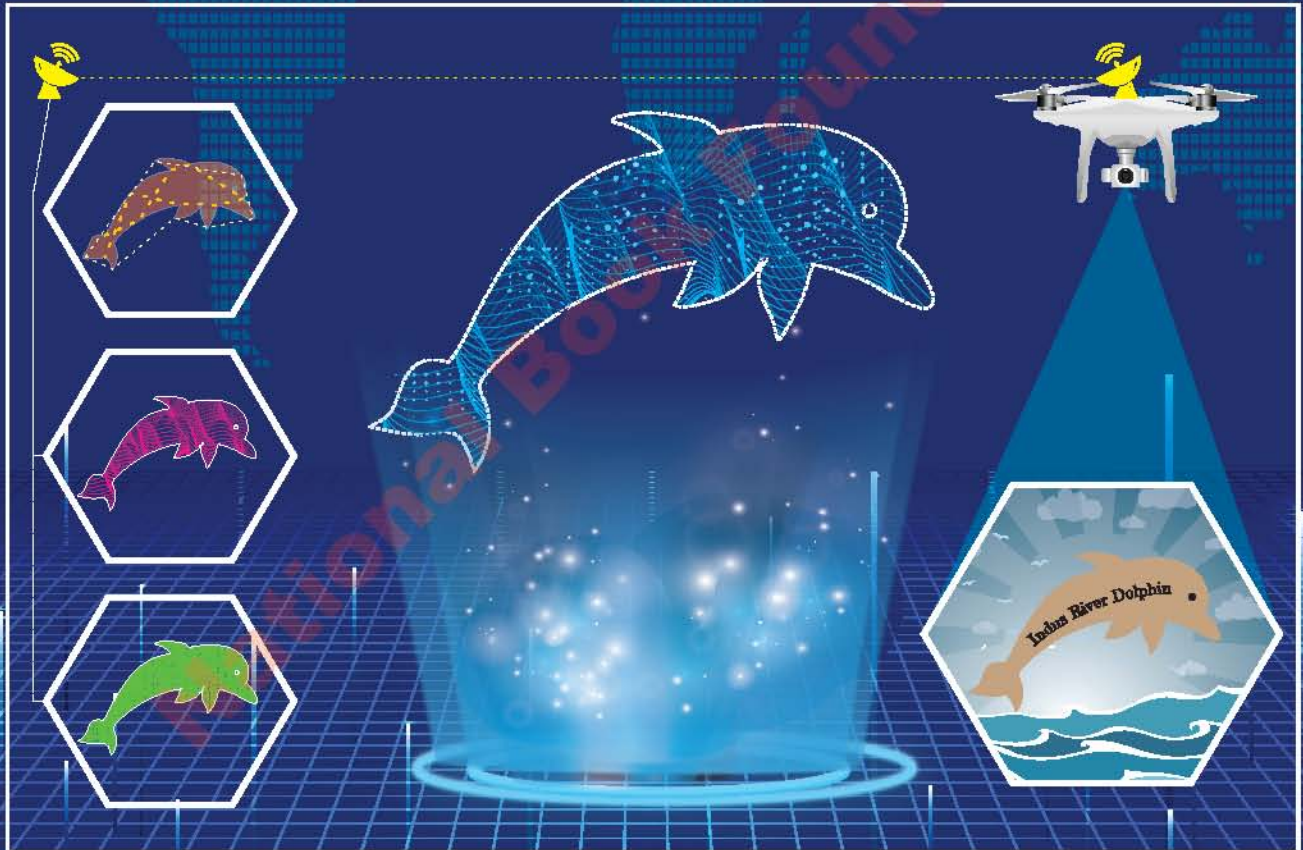
Based on National Curriculum of Pakistan 2022-23

MODEL TEXTBOOK OF

11

GRADE

# COMPUTER SCIENCE



NATIONAL BOOK FOUNDATION  
AS  
FEDERAL TEXTBOOK BOARD, ISLAMABAD



**National Book Foundation**



Based on National Curriculum of Pakistan 2022-23

# **Model Textbook of**

# **Computer Science**

# **Grade**

# **11**

**National Curriculum Council**

Ministry of Federal Education and Professional Training



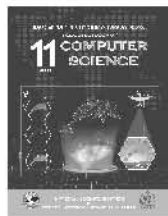
National Book Foundation  
as  
Federal Textbook Board  
Islamabad



© 2024 National Book Foundation as Federal Textbook Board, Islamabad

All rights reserved. This volume may not be reproduced in whole  
or in part in any form (abridged, photo copy, electronic etc.)  
without prior written permission from National Book Foundation

Model Textbook of **Computer Science**  
for Grade 11



**Authors**

Dr. Faraz Ahsan, Mr. Mohammad Khalid, Dr. Saleem Iqbal, Ms. Lubna Kousar,  
Mr. Mohammad Sajjad, Mr. Shah Islam, Dr. Hamid Hassan.

**Supervision**

**Dr. Mariam Chughtai**

Director, National Curriculum Council

Ministry of Federal Education and Professional Training, Islamabad

**IRC Members**

Nadeem Mahmood Shaikh, Fazaia Teacher Training Institute Islamabad, Sana Talha, Federal Government Educational Institutions (FGEI) (C/G) Regional Office Chaklala / Cantt, Muhammad Nabeel, Naval Education Headquarter Islamabad, Ms Lubna Zakia, Federal Government Educational Institutions (FGEI) (C/G) Regional Office Chaklala / Cantt, Ms Asma Haroon, Naval Education Headquarter Islamabad  
Sadia Mujtaba, Federal Directorate Of Education, Usman Adeel, Fazia Schools And Collages, Ayaz Abdullah Khubab Hamza, Fazia Schools and Collages, Sadaf Zehra Kazmi, Federal Directorate of Education

**IPCW-1 Members**

Mr. Muhammad Hashim, School Edu Department, Bolochistan, Sufain Matloob, Directorate Of Curriculum and Teacher Education Khyber Pakhtunkhwa Abbottabad, Mudassar Manzoor Qureshi, Azad Government Of Jammu & Kashmir Directorate of Curriculum Research & Development, Sadaf Zehra Kazmi, Federal Directorate Of Education, Lubna Zakia, Federal Government Educational Institutions (FGEI) (C/G) Regional Office Chaklala / Cantt, Mr. Jahanzaib Khan, Punjab Curriculum & Textbook Board, Qurban Karim,  
Mr. Mushtaq Ahmed, Government Of Sindh, School Education & Literacy Department

**Desk Officer:** Zehra Khushal

**Management:** National Book Foundation

**First Edition - First Impression:** March 2024 | **Pages:** 226 | **Quantity:** 39500

**Price:** PKR 380/-

**Code:** STE-696, **ISBN:** 978-969-37-1609-2

**Printer:** M. Arshad Salman & Bilal Printers, Lahore

**Note:** All the pictures, paintings and sketches used in this book are only  
for educational and promotional purpose in public interest.

for Information about other publications of National Book Foundation,  
visit our Web Site: [www.nbf.org.pk](http://www.nbf.org.pk) or Phone: 051-9261125  
or E-mail: [books@nbf.org.pk](mailto:books@nbf.org.pk)

to share feedback or correction, please send us an email to [nbftextbooks@gmail.com](mailto:nbftextbooks@gmail.com)

**TEST  
EDITION**

# PREFACE

This Model Textbook has been developed by NBF according to the National Curriculum of Pakistan 2022- 2023. The aim of this textbook is to enhance learning abilities through inculcation of logical thinking in learners, and to develop higher order thinking processes by systematically building upon the foundation of learning from the previous grades. A key emphasis of the present textbook is on creating real life linkages of the concepts and methods introduced. This approach was devised with the intent of enabling students to solve daily life problems as they go up the learning curve and for them to fully grasp the conceptual basis that will be built upon in subsequent grades.

After amalgamation of the efforts of experts and experienced authors, this book was reviewed and finalized after extensive reviews by professional educationists. Efforts were made to make the contents student friendly and to develop the concepts in interesting ways.

The National Book Foundation is always striving for improvement in the quality of its books. The present book features an improved design, better illustration and interesting activities relating to real life to make it attractive for young learners. However, there is always room for improvement and the suggestions and feedback of students, teachers and the community are most welcome for further enriching the subsequent editions of this book.

May Allah guide and help us (Ameen).

**Dr. Raja Mazhar Hameed**

Managing Director



**National Book Foundation**

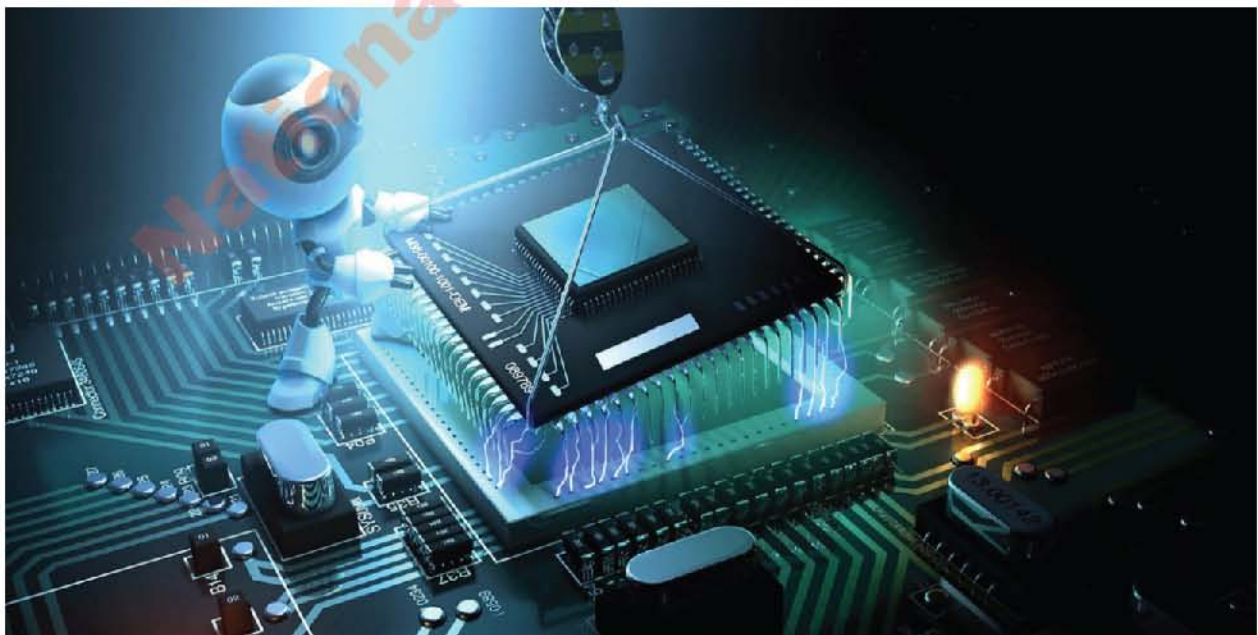
# Unit 01

## COMPUTER SYSTEMS



After completing this lesson, you will be able to

- understand and apply logic gates in digital systems, define and create truth tables using Boolean operators like AND, OR, NOT, NAND, XOR) and logic diagrams.
- understand and evaluate stages of the systems design, e.g. software development life cycle (analysis, design, coding, and testing etc.), and software development methodologies.
- understand and explain the scalability and reliability of networking systems via network topology.
- understand and explain the need for cybersecurity and contrast different methods of encryption to transmit data.



## UNIT INTRODUCTION

A computer system is a fundamental and permeating part of modern life. It has revolutionized the way we work, communicate, learn, and entertain ourselves. A computer system is not just a single device but a sophisticated combination of hardware and software components that work together to process information, solve problems, and execute a multitude of tasks.

A computer is fundamentally a digital machine because it operates based on digital data and binary logic. A computer processes data digitally by converting input information into binary code (0s and 1s), storing it in memory, and utilizing its central processing unit (CPU) to execute a series of binary-based instructions. The CPU performs arithmetic, logical, and control operations on this binary data, with the help of specialized units like the Arithmetic and Logic Unit (ALU) and control unit. Conditional branching and looping allow the computer to make decisions and repeat tasks as needed. Once processing is complete, the results are converted back to human-readable form and presented to the user through output devices. This digital approach enables computers to perform a wide range of tasks with incredible speed and accuracy.

### 1.1 Data Representation in a Digital Computer

A Digital Computer is an electronic machine as it consists of millions of electronic switches. An electronic switch is very similar to an electric switch that is used to turn ON/OFF light. The main difference is that it works on very low voltage. If the electronic switch is closed, electricity flows and if it is open, electricity does not flow. Thus, switch has only two states. Binary number system used for counting also consists of two digits, 0 and 1. Therefore, binary digits, 0 and 1, are used in computer to represent the two states of a switch.

When current flows in a switch, it is **ON** and represents binary 1. When there is no current in a switch, it is **OFF** and represents binary 0. A binary digit, 0 or 1 is referred to as a bit. Groups of 1s and 0s are used in computer to represent data and this is known as binary code. For example, the binary code 1000100, may represent the letter D.

A computer must be able to recognize codes that represent numbers, letters and special characters. These codes are known as alphanumeric codes. Alphanumeric codes include the following characters.

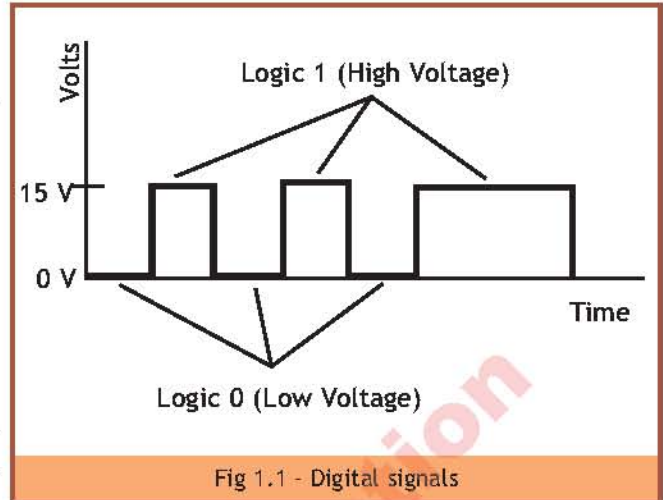
- Lower-case letters, a to z
- Upper-case letters, A to Z
- Numeric digits, 0 to 9
- Special characters including punctuation marks and special symbols such as %, \$, &, #, +, etc.

The American Standard Code for Information Interchange (ASCII) is the most commonly used alphanumeric code. It uses 7 bits or 8-bits to represent a character and the total number of characters it represents is 128

The circuitry of a digital computer works on DC voltage. Such voltage can have only two possible

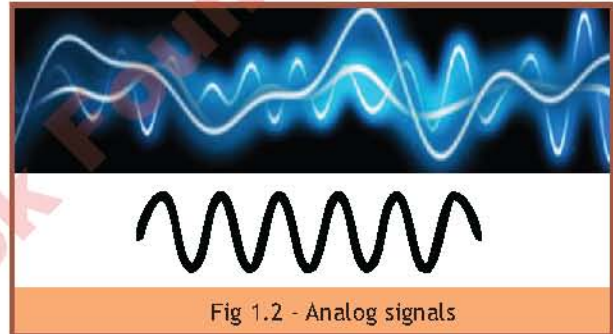


values at any time. Fig. 1.1 shows a digital pulse that takes the values 0 Volt and +5 Volts. These voltage levels are assigned the binary values 0 and 1. These 0 and 1 representations are called logic 0 and logic 1, respectively. Although the logic levels in Fig. 1-1 are shown as exactly 0V and +5V, in practical systems each logic level will represent a range of voltages. For example, logic 0 might be any voltage between 0V and +0.8V, and logic 1 might range from +2V to +5V. The circuits that operate on these signals are called digital logic circuits.



## 1.2 Analog and Digital Signals

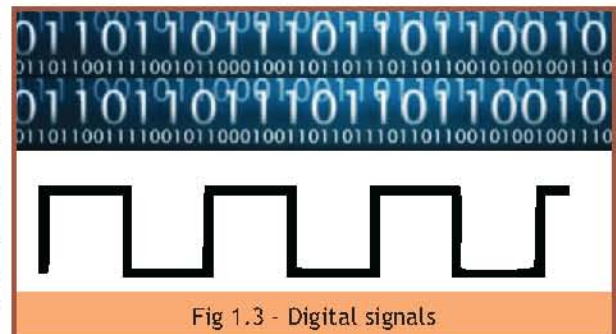
**Analog Signals:** Analog signals are continuous signals that vary smoothly over time. These signals can take any value within a given range and are represented by a continuous waveform. Analog signals are often used in traditional audio and video transmission, as well as in various physical phenomena. Analog signals shown in Fig 1.2.



Examples of Analog Signals:

- Analog audio signals (e.g., , analog audio cassette tapes)
- Analog video signals (e.g., VHS tapes)
- Human speech
- Analog temperature readings
- Analog voltage signals

**Digital Signals:** Digital signals are discrete, binary signals that represent information using a series of discrete values. These values are typically represented by 0s and 1s, where 0 usually represents the absence of a signal or a low voltage, and 1 represents the presence of a signal or a high voltage. Digital signals are commonly used in modern electronic devices and communication systems. Digital signals shown in Fig 1.3.



Examples of Digital Signals:

- Binary code (0s and 1s)
- Digital audio signals (e.g., MP3 files)
- Digital images (e.g., JPEG files)
- Digital video signals (e.g., MP4 files)
- Digital data in computers

Differences between Analog and Digital Signals:

Feature	Analog Signals	Digital Signals
Representation	Continuous waveform	Discrete values (0s and 1s)
Signal Nature	Infinite possibilities of values within a range	Limited discrete values
Noise Susceptibility	Susceptible to noise and interference	More resistant to noise and interference
Transmission	Degrades over long distances	Can be transmitted over long distances with less degradation
Storage and Reproduction	Prone to quality loss during copying or storage	Can be copied and reproduced without quality loss
Scalability	Not easily scalable	Easily scalable
Complexity of Processing	Analog processing is often more complex	Digital processing is typically more straightforward
Examples	Analog audio signals, analog temperature readings	Digital audio signals, digital images

## 1.3 Digital Logic and Logic Gates

**Digital logic** is fundamental in creating electronic devices such as calculator, computer, digital watches, etc. It is used to create digital circuits which consist of large number of logic gates.

**Logic gates** are building blocks of digital circuits. A logic gate performs a particular logical function. Logic gate has two or more logic inputs (LOW or HIGH) and produces a single output which may be LOW (0) or HIGH (1), determined by the logic levels present at the inputs.

### 1.3.1 Logic Gates and their Truth Tables

The commonly used logic gates are AND, OR, NAND, NOR, NOT, Exclusive-OR and Exclusive-NOR which are explained below.



## AND Gate

AND gate operates such that the output will be at level 1 (HIGH) only when all inputs are 1 (HIGH) as shown in Fig.1.4. The mathematical expression for the two-input AND gate is written as  $F=xy$ . For a three-input gate, it would be  $F=xyz$ , and so on for more inputs. The truth table for AND gate for two variables is shown in Fig.1.4.


	x	y	F
	0	0	0
	0	1	0
	1	0	0
	1	1	1

Fig 1.4 - Symbol, expression and truth table of AND gate

## OR Gate

OR gate produces a 1 output when any input is 1 as shown in Fig.1-5. Its mathematical expression is  $F=x+y$ , where the + stands for the OR operation and not normal addition. For a three-input OR gate, it would be  $F=x+y+z$ , and so on. The truth table for OR gate for two variables is shown in Fig.1.5.


	x	y	F
	0	0	0
	0	1	1
	1	0	1
	1	1	1

Fig 1.5 - Symbol, expression and truth table of OR gate

## NAND Gate

NAND gate combines the AND and NOT gates, such that the output will be 0 only when all the inputs are 1 as shown in Fig.1.6. Its logic expression is  $F=\overline{xy}$  which indicates that inputs x and y are first ANDed and then the result is inverted. Inversion is indicated by a bar. Thus, an AND gate always produces an output that is the inverse (opposite) of an AND gate. The truth table for NAND gate for two variables is shown in Fig.1.6.

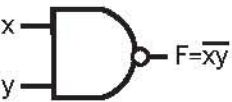
	x	y	F
	0	0	1
	0	1	1
	1	0	1
	1	1	0

Fig 1.6 - Symbol, expression and truth table of NAND gate

## NOR Gate

NOR gate combines the OR and NOT gates, such that the output will be 0 when any input is 1 as shown in Fig.1.7. Its logic expression is  $F=\overline{x+y}$ , which indicates that x and y are first ORed and then the result inverted. Inversion is indicated by a bar. A NOR gate always gives an output that is the inverse of an OR gate. The truth table for NOR gate for two variables is shown in Fig.1.7.

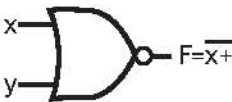
	x	y	F
	0	0	1
	0	1	0
	1	0	0
	1	1	0

Fig 1.7 - Symbol, expression and truth table of NOR gate

## NOT Gate

NOT gate is a single input gate. It converts LOW to HIGH and vice versa as shown in the truth table of Fig.1.8. Therefore, it is commonly known as an inverter. Its logic expression is  $F=\bar{x}$ . The bar in the expression indicates the inversion operation. In the output of graphical symbol, the small circle indicates inversion. The truth table for NOT gate is shown in Fig.1.8.

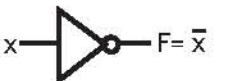
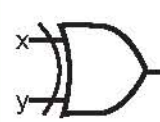
	x	F
	0	1
	1	0

Fig 1.8 - Symbol, expression and truth table of NOT gate



## Exclusive-OR Gate

Exclusive-OR (XOR) gate has a graphic symbol similar to that of OR gate, except for the additional curved line on the input side. It produces a 1 output only when the two inputs are at different logic levels. Its output expression is  $F = x\bar{y} + \bar{x}y$ . The truth table for XOR gate for two variables is shown in Fig. 1.9.



x	y	F
0	0	0
0	1	1
1	0	1
1	1	0

Fig 1.9 - Symbol, expression and truth table of XOR gate

### 1.3.2 Truth Table

A truth table represents a Boolean function or digital logic circuit in table form. It shows how a logic circuit's output or Boolean expression responds to all the possible combinations of the inputs using logic '1' for true and logic '0' for false.

It has the following properties.

- Truth table consists of rows and columns.
- It shows relationship between inputs and output from a Boolean function or digital logic circuit.
- It shows output for all the possible combinations of inputs using 0 for LOW and 1 for HIGH.
- All the combinations of inputs are listed in columns on the left, working in the middle and output is shown in the right most column.
- The input columns are constructed in the order of binary counting with a number of bits equal to the number of inputs.

Example: Truth Table for the Boolean function is shown in Table 1.1:

$$F = (x \cdot y \cdot \bar{z}) + (x \cdot y \cdot z) + (\bar{x} \cdot y)$$

Table 1.1 - Truth Table

Inputs			Working					Outputs
x	y	z	$\bar{x}$	$\bar{z}$	$(x.y.\bar{z})$	$(x.y.z)$	$(\bar{x}.y)$	$F=(x.y.\bar{z}) + (x.y.z) + (\bar{x}.y)$
0	0	0	1	1	0	0	0	0
0	0	1	1	0	0	0	0	0
0	1	0	1	1	0	0	1	1
0	1	1	1	0	0	0	1	1
1	0	0	0	1	0	0	0	0
1	0	1	0	0	0	0	0	0
1	1	0	0	1	1	0	0	1
1	1	1	0	0	0	1	0	1

### 1.3.3 Boolean Identities

Boolean identities are mathematical expressions or equations that are always true, regardless of the values of the variables involved. In the context of Boolean algebra, which deals with binary variables and logic operations (AND, OR, NOT), these identities are fundamental for simplifying and analyzing logical expressions. These Boolean identities are essential for simplifying Boolean expressions and analyzing digital circuits. Table 1.2 shows some commonly used Boolean identities.

Table 1.2 - Boolean identities

Identity Type	Boolean Identity
Identity Law (OR)	$A + 0 = A$
Identity Law (AND)	$A \cdot 1 = A$
Domination Law (OR)	$A + 1 = 1$
Domination Law (AND)	$A \cdot 0 = 0$
Complement Law (OR)	$A + \bar{A} = 1$
Complement Law (AND)	$A \cdot \bar{A} = 0$
Double Negation Law	$\bar{\bar{A}} = A$
Idempotent Law (OR)	$A + A = A$
Idempotent Law (AND)	$A \cdot A = A$
Associative Law (OR)	$(A + B) + C = A + (B + C)$
Associative Law (AND)	$(A \cdot B) \cdot C = A \cdot (B \cdot C)$
Distributive Law (AND over OR)	$A \cdot (B + C) = (A \cdot B) + (A \cdot C)$
Distributive Law (AND over OR)	$A + (B \cdot C) = (A + B) \cdot (A + C)$
Absorption Law (OR)	$A + (A \cdot B) = A$
Absorption Law (AND)	$A \cdot (A + B) = A$
Negation Law (De Morgan's Theorem)	$\overline{A + B} = \bar{A} \cdot \bar{B}$
Negation Law (De Morgan's Theorem)	$\overline{A \cdot B} = \bar{A} + \bar{B}$

### 1.3.4 Boolean Function and its Conversion to Logic Circuit

A Boolean function is an expression formed with binary variables, the logical operators (OR, AND and NOT), parenthesis and equal sign. A binary variable can take the value of 0 or 1. For a given value of the variables, the function can be either 0 or 1.

As an example, consider the following Boolean function.

$$F = x + y$$

The function F is equal to 0, if  $x=0$  and  $y=0$ . For all the other combinations of x and y, the function will be equal to 1.

A Boolean function can be transformed from an algebraic expression into a logic circuit composed

#### Improve your skills!

You can use digital circuit simulation tools (like Logisim) to implement and test Boolean circuits. Logisim is an open-source program that helps you make and simulate logic circuits.

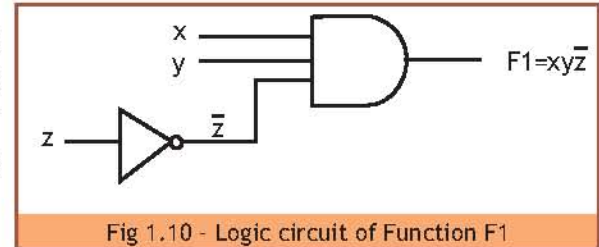
Download Link:

<https://sourceforge.net/projects/circuit/>

of AND, OR and NOT gates. This is explained by the following four examples.

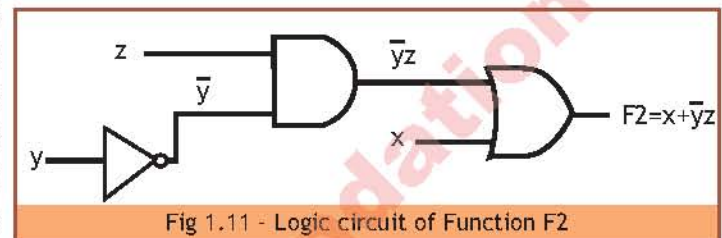
**Example 1:**

Conversion of Boolean function  $F1=xy\bar{z}$  to logic circuit To convert this function to logic circuit, a single AND gate is required for the term  $xy\bar{z}$ . A NOT gate is also required to convert  $z$  to  $\bar{z}$ , before it is input to the AND gate as shown in Fig.1.10.



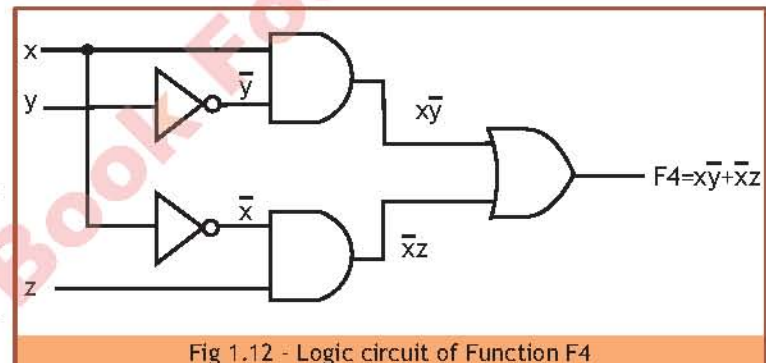
**Example 2:**

Conversion of Boolean function  $F2=x+\bar{y}z$  to logic circuit To create the logic circuit of this function, one AND gates is required for the term  $\bar{y}z$ , one NOT gate to convert  $y$  to  $\bar{y}$  and an OR gate to perform OR operation on the terms  $x$  and  $\bar{y}z$ . The logic circuit of this function is shown in Fig.1.11.



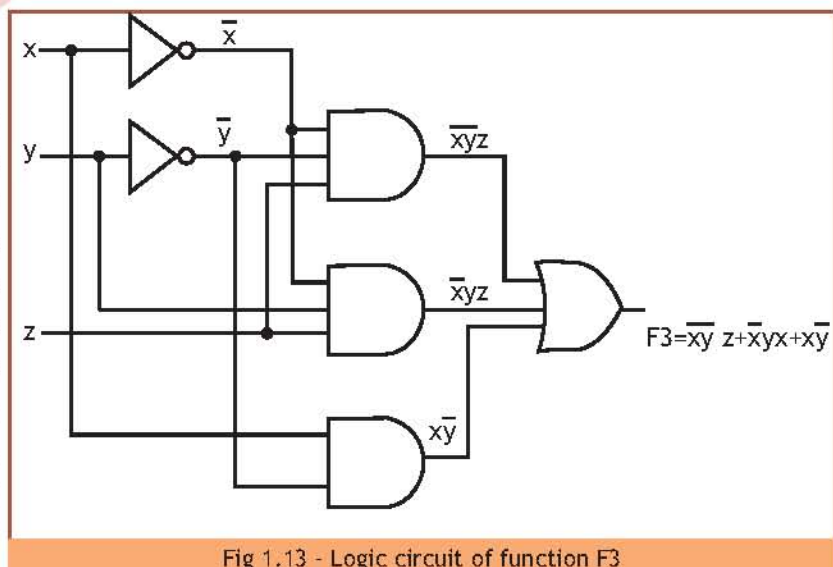
**Example 3:**

Conversion of Boolean function  $F4=x\bar{y}+\bar{x}z$  to logic circuit To create the logic circuit of this function, two AND gates are required for the terms  $x\bar{y}$  and  $\bar{x}z$ , two NOT gates to convert  $x$  to  $\bar{x}$  and  $y$  to  $\bar{y}$  and an OR gate to perform the OR operation on the outputs of two AND gates as shown in Fig.1.12.



**Example 4:**

Conversion of Boolean function  $F3=\bar{x}\bar{y}z + \bar{x}yz + x\bar{y}$  to logic circuit. This function has three terms. Therefore, three AND gates are required for these terms. Two NOT gates are required to obtain  $\bar{x}$  and  $\bar{y}$ . The output of AND gates is to be input into an OR gate to perform the OR operation between all the three terms. The logic circuit of this function is shown in Fig.1.13.





### 1.3.5 Simplification of Boolean Function using Karnaugh Map (K-Map)

#### Karnaugh Map (K - Map)

Karnaugh Map (K-Map) was introduced by Maurice Karnaugh in 1953. It provides a simple method for simplifying Boolean functions. When a simplified Boolean function is converted into a logic circuit, it requires less number of gates and hence costs less.

K-map is a pictorial form of a truth table. It consists of square boxes called cells. All the possible combinations of variables involved in a Boolean function are written inside the cells in their respective positions. A two-variable K-map contains  $2^2=4$  cells, three-variable  $2^3=8$  cells and so forth.

#### Simplification of Two-Variable Boolean Function Using K - Map

A two-variable K-map for variables A and B is shown in Fig.1.14. It consists of four cells having two rows and two columns. Suppose the two variables are A and B. Each row and column is labelled with a variable and its complement. Complement of a variable is also called prime. Each cell contains a product term of variables A and B in its respected cell. For example, the term  $\bar{A}B$  is placed in cell that is in row  $\bar{A}$  and column B.

Simplification of a two-variable Boolean function will be explained with the following example.

	$\bar{B}$	B
$\bar{A}$	$\bar{A}\bar{B}$	$\bar{A}B$
A	$A\bar{B}$	$AB$

Fig 1.14 - Two-variable K-map

#### Example 1:

Simplify the Boolean function  $F1 = \bar{A}B + A\bar{B} + AB$  using K-map.

- The first step to simplify the Boolean function is to plot the terms of the function on the Karnaugh map. This function has three terms, for each term, a 1 will be placed in the corresponding cell. This is shown in Fig.1.15.
- The next step is grouping cells for simplification. Grouping means combining cells in adjacent cells. The K-map contains a pair of 1s that is horizontally adjacent and another pair of 1s that is vertically adjacent as shown in boxes in Fig.1.16.

	$\bar{B}$	B
$\bar{A}$		1
A	1	1

Fig 1.15 - Placement of 1s in corresponding cells

- Combine two terms by eliminating the variable that is in both normal and complemented form. In the horizontal group, B appears in both normal and complemented form. Therefore, B will be eliminated in this group and only A is left. Similarly, in the vertical group, A appears in both normal and complemented form. Therefore, A will be eliminated in this group and only B will be left.
- Finally, the result is written as the sum of variables as:

$$F1 = A + B$$

	$\bar{B}$	B
$\bar{A}$		1
A	1	1

Fig 1.16 - Grouping of 1s in two-variable K-map

The following are the rules for simplifying a two-variable Boolean function.

- For each term of the function, place 1 in the corresponding cell in Karnaugh map.
- Make groups of two cells that contain 1. Groups may be horizontal or vertical but not diagonal.
- Groups may overlap.
- Eliminate the variable that is in normal and complemented form in the group.
- Write the simplified function in the form of sum of variables that were not eliminated in groups.
- If a K-map contains two 1s in diagonal cells then group cannot be formed which means the function cannot be simplified.

### Simplification of Three-Variable Boolean Function using K - Map

A three-variable K-map for variables A, B and C is shown in Fig.1.17. It consists of eight cells having two rows and four columns. Rows are labelled with the complement and normal form of the variable A. Each column is labelled with two variables, B and C, in their normal or complemented form. Each cell contains a product term of variables A, B and C in its respected cell. For example, the term  $\bar{A}\bar{B}\bar{C}$  is placed in cell that is in row  $\bar{A}$  and column  $\bar{B}\bar{C}$ .

	$\bar{B}$	$\bar{B}$	B	B
$\bar{A}$	$\bar{A}\bar{B}\bar{C}$	$\bar{A}\bar{B}C$	$\bar{A}B\bar{C}$	$\bar{A}BC$
A	$A\bar{B}\bar{C}$	$A\bar{B}C$	$AB\bar{C}$	$ABC$
	$\bar{C}$	C	C	$\bar{C}$

Fig 1.17 - Three-variable K-map

The following are the rules for simplifying a three-variable K-map.

For each term of the function, place 1 in the corresponding cell in Karnaugh map.

- Form groups of four if possible otherwise groups of two.
- Groups can contain only 1s.
- Groups can be horizontal or vertical
- Groups can overlap and wrap around the side of the K-map.
- If possible include each 1 in at least one group.
- Eliminate the variables that are in normal and complemented form in a group and create a term for each group.
- Write the simplified function in the form of sum of terms. If a cell containing a 1 cannot be included in any group then write the full term with three variables.

#### Example 2: Simplify the Boolean function

$$F1 = \bar{A}B\bar{C} + \bar{A}B\bar{C} + A\bar{B}\bar{C} + A\bar{B}C$$

- The first step to simplify the Boolean function is to plot all the terms of the function on the three variable Karnaugh map. This function has four terms, for each term, a 1 will be placed in the corresponding cell. This is shown in Fig.1.18.



	$\bar{B}$	$\bar{B}$	B	B		$\bar{B}$	$\bar{B}$	B	B
$\bar{A}$	$\bar{A}\bar{B}\bar{C}$	$\bar{A}\bar{B}C$	$\bar{A}BC$	$\bar{A}B\bar{C}$	$\bar{A}$			1	1
A	$A\bar{B}\bar{C}$	$A\bar{B}C$	$ABC$	$AB\bar{C}$	A	1	1		
	$\bar{C}$	C	C	$\bar{C}$		$\bar{C}$	C	C	$\bar{C}$

Fig 1.18 - Placement of 1s in corresponding cells

- Make two horizontal groups of two 1s as shown in Fig.1.19.

- Combine two terms by eliminating the variable that is in both normal and complemented form in a group. In the group that is on the top, the variable C appears in both normal and complemented form. Therefore, C will be eliminated from this group and the combined term becomes  $\bar{A}B$ . Similarly, in the bottom group also, C appears in both normal and complemented form. Therefore, C will be eliminated from this group also and the combined term becomes  $A\bar{B}$ .

	$\bar{B}$	$\bar{B}$	B	B
$\bar{A}$			1	1
A	1	1		
	$\bar{C}$	C	C	$\bar{C}$

Fig 1.19 - Grouping of 1s in four-variable K-map

- The simplified function can be written as the sum of the resulting terms after eliminating the variable C from both groups as given.  $F2 = \bar{A}B + A\bar{B}$ .

### Example 3:

Simplify the following Boolean function

$$F3 = \bar{A}BC + A\bar{B}\bar{C} + ABC + AB\bar{C}$$

The map for this function is shown in Fig.1.20. There are four squares marked with 1s, one for each term of the function. Two adjacent squares are combined in the third column to give two variable term BC because A is dropped. The remaining two squares with 1s are also adjacent. The map is considered to wrap around the sides to form adjacent squares. Therefore, these two squares are combined to give two variable term AC because here B is dropped. Thus the simplified function becomes:

$$F3 = BC + AC$$

	$\bar{B}$	$\bar{B}$	B	B
$\bar{A}$			1	
A	1		1	1
	$\bar{C}$	C	C	$\bar{C}$

Fig 1.20 - K-map of function F3.

### Example 4:

Simplify the following Boolean function

$$F4 = \bar{A}\bar{B}C + \bar{A}BC + \bar{A}B\bar{C} + A\bar{B}C + ABC$$

The map to simplify this function is shown in Fig.1.21. The function in this example has five



terms, as indicated by the five squares marked with 1s. It is simplified by combining four squares in the center to give the single term  $C$  because  $A$  and  $B$  are eliminated. The remaining single square marked with a 1 is combined with an adjacent square that has already been used once. This is allowed because the combination of the two squares gives the simplified two variable term  $\bar{A}B$ . The simplified function is:

$$F4 = C + \bar{A}B$$

#### Example 5:

Simplify the Boolean function

$$F5 = \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + A\bar{B}\bar{C} + A\bar{B}C + AB\bar{C}$$

The K-map of this function is shown in Fig.1-22. It is simplified by combining four squares in the right and left columns to give the single variable term  $\bar{C}$  because  $A$  and  $B$  are eliminated. The remaining single square marked with a 1 is combined with the adjacent square on the left that has already been used once. The combination of the two squares gives the term  $A\bar{B}$ . From the map we obtain the simplified function:  $F5 = \bar{C} + A\bar{B}$ .

	$\bar{B}$	$\bar{B}$	$B$	$B$
$\bar{A}$		1	1	1
$A$		1	1	
	$\bar{C}$	$C$	$C$	$\bar{C}$

Fig 1.21 - K-map of function F4

	$\bar{B}$	$\bar{B}$	$B$	$B$
$\bar{A}$	1			1
$A$	1	1		1
	$\bar{C}$	$C$	$C$	$\bar{C}$

Fig 1.22 - K-map of function F5

### 1.3.7 Principle of Duality In Boolean Algebra

The Principle of Duality in Boolean algebra states that for any given Boolean expression/function, a dual expression can be obtained by interchanging the AND (.) and OR (+) operators while complementing (negating) the variables.

Duality in Boolean algebra refers to a fundamental property where certain operations and rules can be interchanged while still preserving the logic and truth of expressions. In other words, it highlights the symmetry between two pairs of operations: AND (conjunction) and OR (disjunction), as well as 0 (false) and 1 (true).

Some examples to illustrate the Principle of Duality are shown Table 1.3.

Table 1.3 - Some examples to illustrate the Principle of Duality

Expression	Dual
$1 = 0$	$0 = 1$
$0 = 1$	$1 = 0$
$1.0 = 0$	$0 + 1 = 1$
$A.0 = 0$	$A + 1 = 1$
$0.A = 0$	$1 + A = 1$
$A.1 = 0$	$A + 0 = 1$
$1.A = 0$	$1 + A = 1$
$A.A = 0$	$A + A = 1$
$A.B = B.A$	$A + B = B + A$
$X.(Y.Z) = (X.Y).Z$	$X + (Y + Z) = (X + Y) + Z$
$A.(A + B) = A$	$A + A.B = A$
$XY + Y + ZXY = 0$	$(X + Y).Y.(Z + X + Y) = 1$

### 1.3.8 Uses of Logic Gates

Logic gates are essential components in digital electronics and computing. They are used in numerous applications in various fields.

The following are some important usages of logic gates.

**Memory Circuits:** Flip-flops and latches, which are built using logic gates, are used to store binary data in memory circuits. A **flip-flop** is a digital circuit that stores binary information and is widely used in digital electronics for building memory elements and sequential logic circuits. A **latch** is another type of digital circuit that stores binary information. Latches are often used in memory storage elements and data path circuits.

**Clock Synchronization:** Logic gates help in clock synchronization and signal processing in digital systems.

**Data Encoding and Decoding:** Logic gates are used to encode and decode data for transmission and reception in communication systems.

**Multiplication and Division:** Complex mathematical operations like multiplication and division can be performed using a combination of logic gates.

**Digital Signal Processing (DSP):** Logic gates are used in DSP circuits for filtering, modulation, and demodulation.

**Data Encryption and Decryption:** Cryptographic algorithms use logic gates for data encryption and decryption.

**Calculator Circuits:** Basic calculators use logic gates to perform arithmetic calculations.

**Traffic Light Control:** Logic gates are used in traffic light control systems to manage traffic flow.

**Robotics:** Logic gates play a crucial role in controlling the movement and decision-making of robots.

**Security Systems:** Logic gates are used in security systems to control access, alarms, and surveillance.

**Automotive Electronics:** In vehicles, logic gates are used for engine control, airbag deployment, and anti-lock brake systems (ABS).

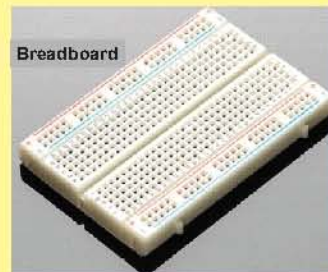
**Home Automation:** Logic gates are employed in smart home systems to automate tasks like lighting and temperature control.

**Medical Devices:** Medical equipment uses logic gates for monitoring and controlling various functions.

**Aerospace Applications:** Logic gates are used in navigation systems, autopilots, and guidance systems for aircraft and spacecraft.

#### Improve your Skills! (Advanced)

Now that you have acquired a foundational understanding of logic gates, you are well-equipped to engage in practical digital electronics experiments. Utilize this knowledge to conduct hands-on experiments with logic gates using a Breadboard and various electronic components. A sample experiment has been prepared to guide you through the process. A breadboard (sometimes called a plug-block) is used for building temporary circuits. It is useful to designers because it allows components to be removed and replaced easily.





## Digital Electronics Experiment

### AND Gate Practical on Bread-Board

Suggested Youtube link:

[https://www.youtube.com/watch?v=aMVgrSU2PLc&list=PLe\\_7x5eaUqtVgVnAccCemHekNNzVbHq\\_&index=5](https://www.youtube.com/watch?v=aMVgrSU2PLc&list=PLe_7x5eaUqtVgVnAccCemHekNNzVbHq_&index=5)

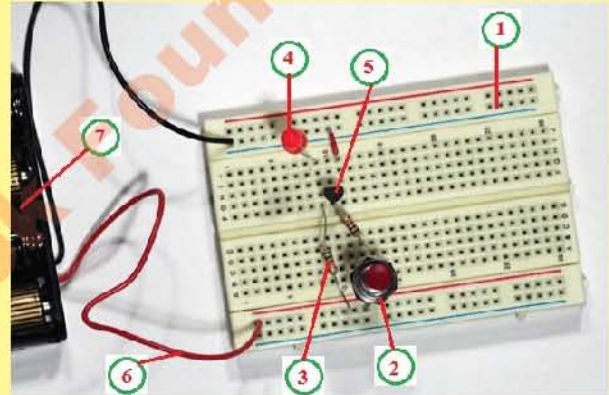
How to use Breadboard: <https://www.youtube.com/shorts/05ZrbtwUzMk>

#### Requirements:

1. Breadboard: Provides a platform for circuit assembly.
2. Input Switch (Push Button or Toggle Switch): Represents the binary input state (0 or 1).
3. Resistor (e.g., 10k $\Omega$ ): Used for current limiting and pull-up resistor configurations.
4. LED: Indicates the output state.
5. NPN Transistor (e.g., 2N3904): Acts as an electronic switch in the circuit.
6. Connecting Wires: Establish electrical connections between components.
7. Power Supply (e.g., 5V): Provides the necessary voltage for the circuit.

#### Procedure:

1. Place Components on the Breadboard:
  - Position the breadboard on a stable surface.
  - Insert the input switch, resistor, LED, and NPN transistor into the breadboard.
2. Connect Power Supply:
  - Attach the power supply to the positive and negative rails of the breadboard.
3. Connect Resistor to Switch:
  - Connect one leg of the resistor to one terminal of the input switch.
  - Connect the other leg of the resistor to the positive rail.
4. Connect Switch to Ground:
  - Connect the other terminal of the input switch to the ground rail.
5. Connect Transistor Base:
  - Connect the junction between the resistor and the input switch to the base (B) of the NPN transistor.
6. Connect Transistor Emitter:
  - Connect the emitter (E) of the NPN transistor to the ground rail.
7. Connect Transistor Collector:
  - Connect the collector (C) of the NPN transistor to the positive rail through a current-limiting resistor.
8. Connect LED:
  - Connect the cathode (shorter leg) of the LED to the collector (C) of the NPN transistor.
  - Connect the anode (longer leg) of the LED to the positive rail.
9. Test the NOT Gate:
  - Press the input switch. The LED should light up when the switch is not pressed (logic 0), and vice versa, simulating the NOT gate behavior.

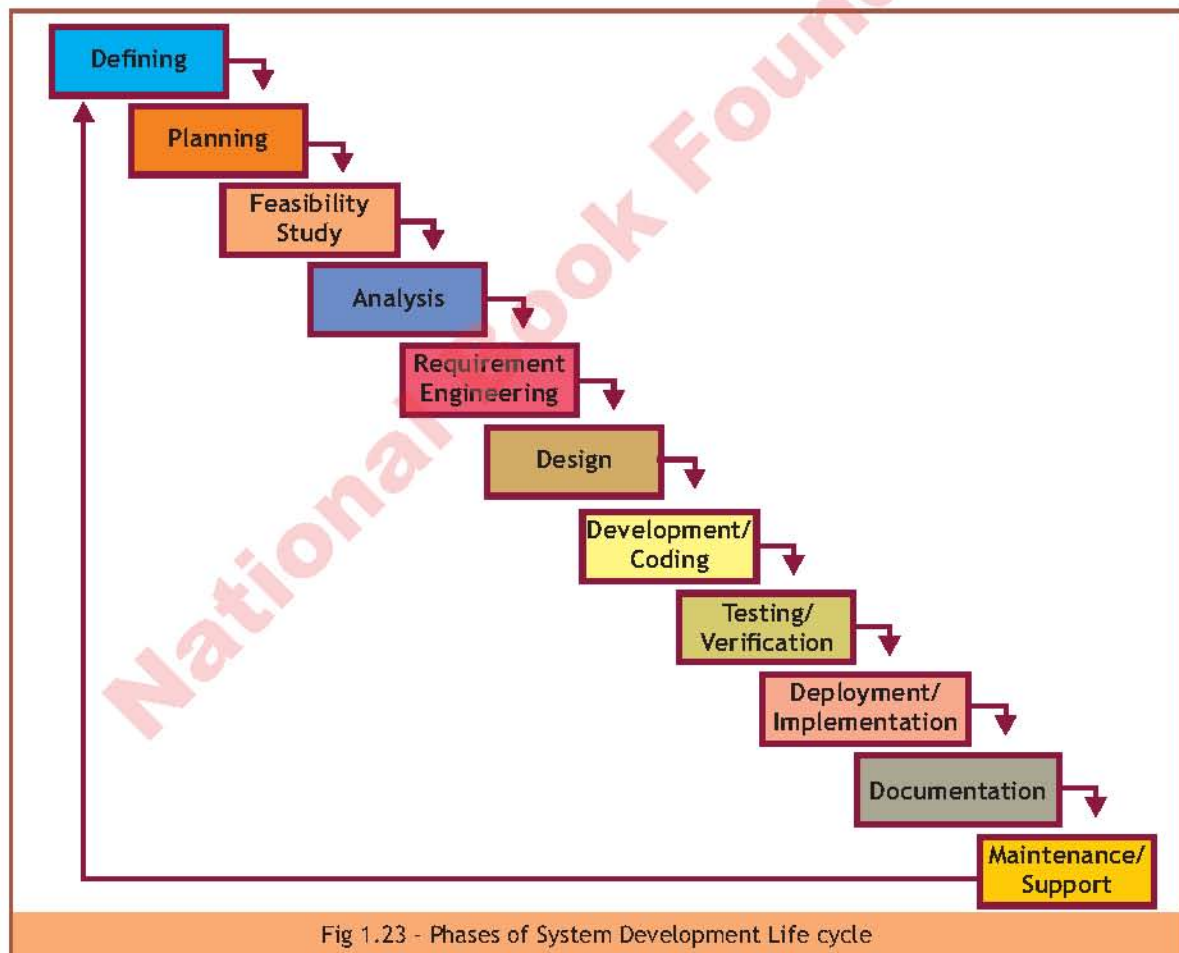




## 1.4 Software Development Life Cycle (SDLC)

Software Development Life Cycle (SDLC) is the process of creating a new software or system. In other words these are the models and methodologies that experts use to develop the system. In **software engineering** the SDLC concept reinforces many kinds of software development techniques. These techniques process the framework for planning and controlling the creation of software. The software development life cycle (SDLC) is the process of planning, writing, modifying, and maintaining software. Developers use the methodology as they design and write modern software for computers, cloud deployment, mobile phones, video games, etc. In IT, the term "life cycle" was first used in the 1950s to describe the stages involved in developing a new computer system, but it is now commonly used to refer to all the stages in the production of any type of software.

### 1.4.1 Different Phases of SDLC



The following are phases/steps in SDLC.

- Defining the Problem Phase
- Planning Phase
- Feasibility Study Phase
- Analysis Phase
- Requirement Engineering Phase
- Design Phase
- Development/Coding Phase
- Testing/Verification Phase
- Deployment/Implementation Phase
- Documentation Phase
- Maintenance/Support Phase

### 1. DEFINING THE PROBLEM PHASE

In this phase the problem to be solved or system to be developed is clearly defined. All the requirements are documented and approved from the customer or the company which consists of all the product requirements to be designed and developed during the development life cycle.

**Example:** Students' Examination System Development

**Defining the problem:** A Students' Examination System is needed to be developed that covers all the aspects from Examination taking to the Students' results generations.

### 2. PLANNING PHASE

In the project planning phase, the project's goal is identified, and the necessary requirements for product development are assessed. A thorough evaluation of resources, including personnel and costs, is conducted, accompanied by the conceptualization of the new product. The gathered information undergoes analysis to explore potential alternative solutions. If no feasible alternatives are found, the data is organized into a comprehensive project plan, which is then presented to management for approval.

**Example:** In the Students' Examination System Development project planning will be made to set the ultimate goals and an estimate of resources, such as personnel and costs, is prepared.

### 3. FEASIBILITY STUDY PHASE

Feasibility study is an essential aspect of project planning and decision-making in the Software Development Life Cycle (SDLC). It involves evaluating various dimensions to determine the viability and practicality of developing a proposed system.

Feasibility study is the analysis and evaluation of a proposed system, to determine, whether it is technically, financially/economically, legally and operationally feasible within the estimated cost and time. Different feasibility studies are explained as follows.

**Technical Feasibility:** Technical feasibility assesses the practicality of implementing a proposed project from a technological standpoint. It involves evaluating whether the necessary technology (hardware, software), tools and resources are available or can be developed to support the system.

**Example:** Consider the **Students' Examination System Development** project

**Technical Feasibility Considerations:** The developing company will evaluate whether the existing infrastructure (hardware/software) can support the proposed system.

**Economic Feasibility:** Economic feasibility evaluates the financial viability of a proposed system by comparing its costs and benefits.

**Operational Feasibility:** Operational feasibility assesses the extent to which a proposed system aligns with the organization's operational processes and goals.

**Legal Feasibility:** Legal feasibility evaluates whether a proposed system complies with applicable laws, regulations, and standards.

**Schedule Feasibility:** Schedule feasibility assesses whether a system can be completed within a specified timeframe.

#### 4. ANALYSIS PHASE

During the analysis phase the project team determines the end-user requirements. Often this is done with the assistance of client focus groups, which provide an explanation of their needs and what their expectations are for the new system and how it will perform.

In this phase, the in-charge of the project team must decide whether the project should go ahead with the available resources or not. Analysis is also looking at the existing system to see what and how it is doing its job. The project team asks the following questions during the analysis.

- Can the proposed software system be developed with the available resources and budget?
- Will this system significantly improve the organization?
- Does the existing system even need to be replaced etc?

**Example:** The **Students' Examination System Development** project is analyzed for development. The project team will visit the College to study the existing system and will suggest the possible improvements.

#### 5. REQUIREMENT ENGINEERING PHASE

Requirement Engineering is a crucial phase in the Software Development Life Cycle (SDLC) that focuses on gathering, analyzing, documenting, and managing requirements for the development of the proposed system. This phase lays the foundation for the subsequent stages of development and ensures that the software meets the needs and expectations of stakeholders (e.g. End users). Requirement engineering consists of the following steps.

- Requirement gathering
- Requirement validation
- Requirements management



## Requirement Gathering

Requirements gathering is a pivotal stage in the Software Development Life Cycle (SDLC), aiming to identify and document the needs and expectations of stakeholders. Various techniques are employed for this purpose, and they can be broadly classified into different types. Below are some typical types of requirement gathering types with examples.

**i. Interviews:** It involves direct conversations with stakeholders to gather information about their needs, expectations, and preferences.

**Example:** In the **Students' Examination System Development** project the business analyst conducts interviews with key users and managers to understand their requirements for a new customer relationship management (CRM) system.

**ii. Surveys and Questionnaires:** This method involves distributing surveys or questionnaires to collect information from a large number of stakeholders.

**Example:** In the context of developing a **Students' Examination System** for a college, surveys or questionnaires could be distributed to gather information from students, faculty, and administrators. For example, the IT team might send out a survey to students and faculty members to collect feedback on the user interface and functionality of the proposed examination system.

**iii. Observation:** This technique involves observing users in their natural work environment to understand how they currently perform tasks and identify areas for improvement.

**Example:** In the context of developing a **Students' Examination System** for a college, the observation method involves actively watching and understanding how students, faculty, and administrators currently handle examination-related tasks in their work environments. For instance, designers could observe students during examination times, noting how they submit papers, how faculty members manage grading, and how administrators oversee the overall process. This direct observation allows the project team to identify weak areas, inefficiencies, or areas for improvement in the existing examination system.

**iv. Document Analysis:** This approach includes reviewing existing documentation, reports, and manuals to extract relevant information about the current system or processes.

**Example:** For the **Students' Examination System Development** project, the development team could analyze academic policies, grading criteria, and any previous reports on examination processes to extract relevant information.

## Requirements Validation

Requirement validation focuses on scrutinizing the gathered requirements to ensure they align with the stakeholders' intentions. This process distinguishes itself from verification, which takes place after requirements have been accepted. During requirements validation, a thorough review is conducted to verify the completeness and accuracy of the elicited requirements.

**Example:** In the context of the **Students' Examination System development** project, once the

initial requirements for the examination system are compiled, the project team engages in requirement validation by carefully examining each specification. This ensures that the proposed features and functionalities accurately reflect the expectations of students, faculty, and administrators involved in the examination process.

### Requirements Management

Requirements management is a continual process aimed at guaranteeing that the software consistently fulfills the expectations of both the acquirer and users. It involves the collection of new requirements that may emerge due to evolving expectations, changing regulations, or other sources of modification.

**Example:** In the context of the **Students' Examination System development project**, as the Students' Examination System evolves, requirements management becomes essential. This involves actively seeking and incorporating new requirements that may arise from feedback sessions with users, changes in academic policies, or advancements in examination methodologies. Continuous requirements management ensures that the examination system adapts to the dynamic needs of the educational environment.

## 6. DESIGN PHASE

The design phase in the Software Development Life Cycle (SDLC) is a crucial step where the system architecture is planned and detailed specifications are created based on the requirements gathered during the analysis phase. **Unified Modeling Language (UML)** and various Design patterns play significant roles in this phase. Unified Modeling Language (UML) is a standardized visual modeling language widely used in software engineering and system design. It plays a crucial role in the Software Development Life Cycle (SDLC) by providing a common notation that allows developers, analysts, and stakeholders to communicate and visualize the different aspects of a system. **Design patterns** play a pivotal role in the SDLC by offering reusable solutions, promoting best practices, enhancing communication among team members, and contributing to the creation of maintainable and scalable software systems. They are valuable tools for software engineers striving to build high-quality, robust, and efficient software solutions.

The design phase normally consists of two simple structures. These are:

- Algorithms
- Flow chart

### I. Algorithms

Algorithms are precise and systematic procedures designed to guide the step-by-step solution of a problem. They provide a structured and detailed set of instructions for solving a particular problem or performing a specific task.

**Example:** In the **Students' Examination System Development project** the following algorithms will find the result of a student using percentage marks.



Algorithm to find the Percentage Marks	Algorithm to find the Student's Result
1. Start	1. Start
2. Read Total Marks, TM	2. Read Percentage Marks, PM
3. Read Obtained Marks, OM	3. If PM >=40 Then Print " Pass" Else Print "Fail"
4. Percentage Marks(PM) = OM / TM X 100	
5. Print "Percentage Marks", PM	4. End
6. End	

Fig 1.24 - Algorithms

## ii. Flowcharts

A flowchart is a diagrammatic representation used to illustrate an algorithm or a process. It visually presents the sequence of steps in the algorithm through special shapes (symbols) and connects them with arrows to depict their sequence. This graphical representation offers a systematic, step-by-step solution to a specified problem.

Some most commonly used flowchart symbols are shown in Fig 1.25.







Symbol	Name	Function
Oval 	Terminal	Indicates the starting or ending of the program, process, or interrupt program
Parallelogram 	Input/output	Used for any Input/Output(I/O) operation, Indicates that the computer is to obtain data or output results
Diamond 	Decision	Used to ask a question that can be answered in a binary format (Yes/No, True/False)
Rectangle 	Process	Used for Arithmetic operations and data manipulations
Arrows 	Flow Lines	Shows direction of flow
Circle 	Connector	Used to connect one part of the flowchart to the other

Fig 1.25 - Flowchart

**Example:**

In the Students' Examination System Development project, the Flowchart for the above algorithms will be as follows.



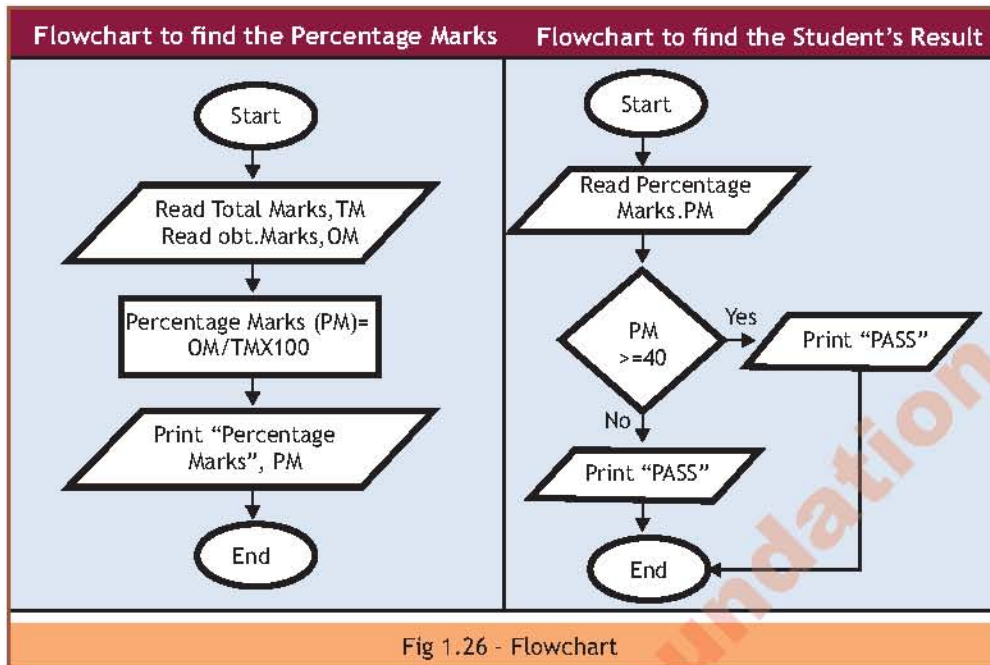


Fig 1.26 - Flowchart

## 7. DEVELOPMENT/CODING PHASE

In the development/coding phase, developers translate the plans formulated in the design phase into actions. They develop the database structures, design codes for the data flow processes, and design the tangible user interface screens. This stage involves the preparation and iterative processing of test data to enhance the precision and efficiency of the written code. The actual coding is carried out using programming languages, a process commonly referred to as Computer programming.

**Example:** In the **Students' Examination System development project**, developers create the database structure, specifying how the system will store and retrieve exam-related data. They write code for the flow of information within the system, ensuring seamless processes from question generation to result compilation. Additionally, user interface screens are precisely designed to provide a user-friendly environment for students, faculty, and administrators interacting with the examination system. Throughout this phase, rigorous testing using prepared test data

Fig 1.27 - Development / Coding Phase

```

1. #include<iostream>
2. int main()
3. {
4.     int cs;
5.     float total, per;
6.     cout << "Please Enter subject marks of Computer Science: \n";
7.     cin >>cs; // taking values from user
8.     total = 50;
9.     per = cs/total*100; // calculating percentage
10.    cout << "\nTotal Marks = " << total;
11.    cout << "\nMarks in Percentage = " << per<<"%";
12.    return 0;
13. }
  
```

Please Enter subject marks of Computer Science: 34  
Total Marks = 50  
Marks in Percentage = 68%

Output

is performed to identify and rectify any issues, contributing to the creation of a robust and efficient examination system.

The Figure 1.27 displays example of programming code with output, created in the C++ programming language to calculate students' percentage marks in a subject for the **Students' Examination System Development project**.

## 8. TESTING/VERIFICATION PHASE

In the testing/verification phase, all aspects of the system are tested for functionality and performance. Testing involves the execution of programming modules to detect errors, commonly known as bugs. The primary goal of testing is to assess specific attributes or capabilities of a program or system, ensuring that they are aligned with the required specifications. It involves checking items for consistency by comparing results against predetermined requirements. Software testers employ various techniques, including black box and white box testing, to identify and eliminate all identified bugs or errors before the final product is released. The process of error identification and removal is referred to as debugging.

**Black Box Testing:** Black Box Testing is a software testing method where the internal workings or logic of a system are not known to the tester. The focus is on evaluating the system's outputs based on specified inputs without considering its internal code structure.

Example: Login Functionality of a Web Application

**Scenario:** Testing the login functionality of a web application.

**Tester's Perspective:** The tester does not have access to the source code or knowledge of the internal implementation details.

**Test Cases:**

- Input: Valid username and password, Expected Output: Successful login.
- Input: Incorrect password, Expected Output: Login failure.
- Input: Invalid username, Expected Output: Login failure.
- Input: Empty username and password, Expected Output: Login failure.

**White Box Testing:** White Box Testing is a testing approach where the tester has knowledge of the internal code, logic, and architecture of the system being tested. It involves evaluating the system's internal structures, code paths, and overall code quality.

**Example:** In the Students' Examination System Development project the above programming module is tested for errors using White box testing.

**Scenario:** Testing a specific function within a software application. Here the students' correct percentage marks and the result is being tested.

**Tester's Perspective:** The tester has access to the source code and understands the internal logic of the function being tested.



### Test Cases:

- Test each statement within the program code.
- Verify that variables are correctly initialized and updated.
- Check for the errors and debug, if needed.
- Evaluate the code's performance under various conditions by providing values.

The test result of the program code is given as follows.

Test of Code to find the Percentage Marks	Test of Code to find the Student's Result
Test 1: Enter Total marks of the student=1100 Enter Obt. marks of the student=990 Percentage Marks = 90.00  Test 2: Enter Total marks of the student=1100 Enter Obt. marks of the student=340 Percentage Marks = 30.90	Test 1: Enter Percentage of the student 90.00 Pass  Test 2: Enter Percentage of the student 30.90 Fail

Fig 1.28 - TESTING/VERIFICATION PHASE

## 9. DEPLOYMENT / IMPLEMENTATION PHASE

The Deployment/Implementation Phase involves a series of activities aimed at making the software/system accessible for use. During this phase, the finalized software is handed over to users, making the transition from development to utilization.

The main activities that are involved during deployment/implementation phase are:

- Installation and activation of the hardware and software.
- In some cases the users and the computer operation personals are trained on the developed software system.
- Conversion: The process of changing from the old system to the new one is called conversion.

### Deployment/Implementation Methods

The deployment or implementation of a system can be executed through various methods, as illustrated in Figure 1.29.

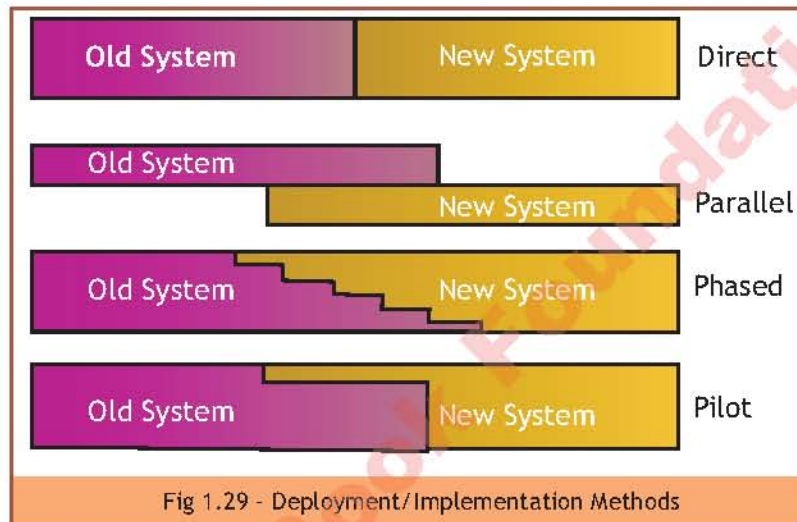
**i. Direct:** In this method, the old system is entirely replaced by the new system simultaneously. The transition is abrupt, and once implemented, the old system becomes obsolete.



**ii.Parallel:** The parallel method involves running both the old and new systems concurrently for a certain period. This approach allows for the identification and rectification of major issues with the new system without risking data loss.

**iii.Phased:** The phased implementation method facilitates a gradual transition from the old system to the new one. It involves the incremental introduction of the new system while progressively phasing out the old system.

**iv.Pilot:** In the Pilot method, the new system is initially deployed for a small user group. These users engage with, assess, and provide feedback on the new system. Once the system is deemed satisfactory, it is then rolled out for use by all users.



## 10. DOCUMENTATION PHASE

In SDLC, developing good user documentation is an important part of the implementation process. Documentation serves as a method of communication among the people responsible for developing, implementing, and maintaining the newly developed system. Installing and operating a newly designed system or modifying an established application requires a detailed record of that system's design. Documentation is extremely important in diagnosing errors and making changes, especially if the systems analysts who developed a system are no longer with the organization.

## 11. MAINTENANCE/SUPPORT PHASE

In SDLC, the system maintenance is an ongoing process. The system is monitored continually for performance in accordance with user requirements and needed system modifications are incorporated. When modifications are identified, the system may reenter the planning phase. This process continues until a complete solution is provided to the customer. Maintenance can be either be repairing or modification or some enhancement in the existing system.



## Case Study: Development of a Library Management System (LMS)

- 1. Defining the Problem Phase:** The library of XYZ Elementary School faces challenges in efficiently managing its book inventory, tracking borrowed books, and providing an organized system for library staff. There is a need for a comprehensive Library Management System (LMS) to address these issues and enhance the overall library experience.
- 2. Planning Phase:** A project team is formed, including representatives from the school administration, librarians, and IT professionals. The team establishes project goals, scope, budget, and a timeline for the development of the LMS. Key milestones and deliverables are identified.
- 3. Feasibility Study Phase:** A feasibility study is conducted to assess the technical, operational, and economic viability of implementing the LMS. The study concludes that the benefits of the system, including improved efficiency and user satisfaction, outweigh the costs.
- 4. Analysis Phase:** Detailed analysis is performed to understand the current library processes, identify user needs, and document system requirements. The team conducts interviews with librarians and observes day-to-day operations to gather essential information.
- 5. Requirement Engineering Phase:** Based on the analysis, a detailed list of functional and non-functional requirements for the LMS is compiled. This includes features such as book cataloging, user management, check-in/check-out functionality, and reporting capabilities.
- 6. Design Phase:** The system architecture and database design are created. User interface prototypes are developed to visualize how the LMS will look and function. The design phase also includes planning for data security, system scalability, and user accessibility.
- 7. Development/Coding Phase:** The actual coding of the LMS begins, adhering to the design specifications. The development team uses appropriate programming languages and tools to implement the various modules of the system, ensuring that it meets the specified requirements.
- 8. Testing/Verification Phase:** Comprehensive testing is conducted to identify and rectify any bugs or issues. The LMS undergoes unit testing, integration testing, and system testing to ensure its functionality, reliability, and performance meet the desired standards.
- 9. Deployment/Implementation Phase:** The LMS is deployed and integrated into the school's library infrastructure. Training sessions are conducted for library staff to familiarize them with the new system. Data migration from the old system, if applicable, is executed seamlessly.
- 10. Documentation Phase:** Comprehensive documentation, including user manuals, system architecture, and code documentation, is created. This documentation serves as a reference for users and future developers, ensuring the system's maintainability.
- 11. Maintenance/Support Phase:** Post-implementation support and maintenance are provided to address any issues that may arise. Regular updates and improvements are made to the system based on user feedback and changing requirements, ensuring its continued effectiveness.



## 1.4.2 Software Development Models

Software Development Models refer to the fundamental activities and approaches used in software development to plan, design, build, test, implement, and maintain software systems. These processes provide a structured framework for managing software projects. Many software development models are available in the market. Each has its own advantages and disadvantages. The following are the two common models.

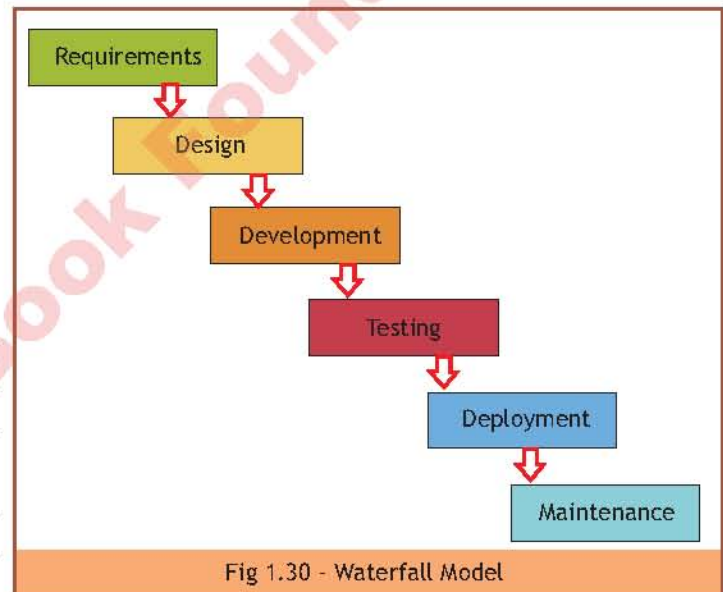
- Waterfall Model
- Agile Model

### Waterfall model

The waterfall model is a sequential or linear development approach, in which development is seen as flowing steadily downwards (like a waterfall) through several phases. It is the most popular process models in software development. Developers use this approach when the requirements for a product are well-defined and resources are available. This model has five phases:

- Requirements
- Design
- Development
- Testing
- Deployment
- Maintenance

These steps always follow in this order and do not overlap. The developer must complete every phase before the next phase begins. This model is named "Waterfall Model", because its diagrammatic representation resembles a flow of waterfalls as shown in Fig 1.30.



#### 1. Requirements:

In this phase the development team works on the feasibility of developing a software product. The aim of this phase is to understand the exact requirements of the customer and to document them properly. Both the customer and the software developer work together so as to document all the functions, performance, and interfacing requirement of the software. This takes place in the two following steps:

- Requirements Gathering and Analysis: The development team collects the essential requirements from all the possible stakeholders and analyzes them. During the analysis, they eliminate incomplete and inconsistent requirements.
- Requirements Specification: After requirements analysis, the development team documents



them in a Software Requirement Specification (SRS) document. It contains both functional and non-functional requirements of the software.

## **2. Design:**

In this step the development team transforms requirements into a format that can be easily converted into the code in chosen programming language. Further, it involves creating design documents specifying the different modules/components of the system, their interfacing, data flow, etc. All the data collected is stored in a software Design Document (SDD) document. This document helps to establish the software's architecture.

## **3. Development:**

The Development phase involves the actual coding of the software based on the design specifications. During this phase, design is implemented. If the SDD is complete, the coding phase proceeds smoothly, because all the information needed by software developers is contained in the SDD. Developers use the detailed design documentation to write the necessary code, turning the conceptualized system into a functioning software product.

## **4. Testing:**

During testing, the code is thoroughly tested for bugs and debugged or modified if required. After this the Implementation (also called deployment) phase involves making the software in use in the real environment after it undergoes multiple rigorous tests. This phase is highly important as the quality of the end product is determined by the effectiveness of the testing carried out. The better output will lead to satisfied customers, lower maintenance costs, and accurate results.

## **5. Deployment:**

The Deployment phase involves making the software available in the intended environment for end-users. This includes activities such as installation, configuration, and data migration to ensure a smooth transition from the development environment to live production. It marks the point at which the software is ready for use.

## **6. Maintenance phase:**

Over a period of time, a software product may require some updates to remain functional in the real-world environment. The maintenance phase takes care of this activity by timely tuning the software as per the requirement.

It is very important phase of waterfall development model. Even if the software is live and end users use it, it is essential to update it regularly.

There are two main types of maintenances:

- **Corrective Maintenance:** It involves correcting errors left undiscovered during the development and testing stages.
- **Perfective Maintenance:** This entails enhancing the functionality of the software product as and when required keeping in mind the future trends and customers demand.

### **Advantages of the Waterfall Model:**

- It is perfectly suitable for projects where all the requirements are predefined and understood

clearly. Requirements do not change throughout development.

- It is easy to understand and implement.
- All the activities to be performed in each phase are clearly defined.
- The release date and the final cost are already estimated before the development begins.
- It is easier to assign tasks to different team members.
- All processes, actions, and results are well documented.

#### Disadvantages of the Waterfall Model:

- Inflexibility to changes is a significant drawback, making mid-project adjustments challenging.
- Long delivery times are inherent as the entire project must be completed before delivery.
- High risk of project failure due to late client visibility and potential dissatisfaction.
- Late defect detection is a concern as testing is performed toward the end of development.

#### Agile Model

The **Agile model** is a software development process that is based on the iterative and incremental approach, emphasizing flexibility, collaboration, and customer feedback. The Agile model is a type of incremental model where software is developed in a rapid incremental cycle as shown in Fig. 1.31. This method breaks tasks into smaller iterations, or parts and does not directly involve long term planning. The project scope and requirements are laid down at the beginning of the development process. This model has six phases:

- Requirements gathering
- Planning
- Design
- Implementation
- Testing
- Deployment
- Maintenance

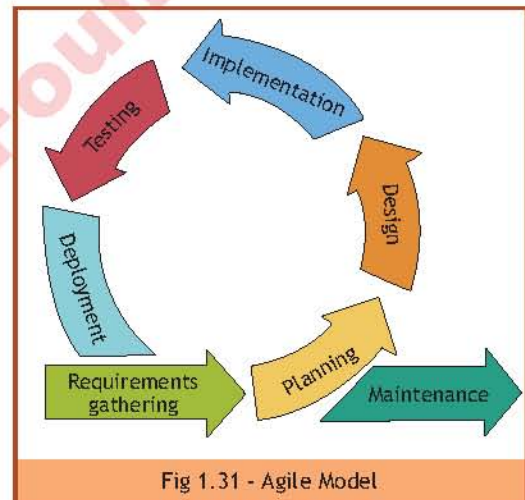


Fig 1.31 - Agile Model

#### Requirements Gathering:

In the Agile software development model, the "Requirements Gathering" phase is characterized by continuous collaboration and dynamic responsiveness. Rather than relying on a comprehensive upfront documentation, Agile teams engage in ongoing discussions with stakeholders. Through workshops, meetings, and iterative feedback sessions, the team identifies and prioritizes user stories, capturing requirements in a flexible and evolving manner. The emphasis is on adaptability, allowing for adjustments and refinements as the project progresses.



### **Planning:**

Agile planning is iterative and done in short cycles called sprints. The team, including developers and stakeholders, collaboratively plans the work for the upcoming sprint. Priorities are set, and tasks are pulled from the product backlog into the sprint backlog based on their importance and feasibility. Planning sessions are dynamic, allowing adjustments based on changing requirements.

### **Design:**

The design phase in Agile Model is a collaborative process that adapts to changing requirements and emerging insights. Unlike traditional approaches, Agile design activities are ongoing and occur at various levels throughout the development lifecycle. Cross-functional teams collaborate to design and refine features incrementally, ensuring that design decisions align with evolving project needs. This iterative approach allows for flexibility, facilitating the incorporation of feedback from stakeholders and team members.

### **Implementation:**

The implementation phase in Agile Model involves developing the features identified in the design phase. Agile development occurs incrementally in short cycles known as sprints. During the "Development" phase, cross-functional teams implement prioritized user stories based on the sprint backlog. The focus is on delivering small, functional increments regularly, promoting adaptability to changing requirements. Collaboration within the team is crucial, fostering communication and shared understanding. Agile development principles prioritize customer satisfaction through the continuous delivery of tangible and valuable software.

### **Testing:**

In the Agile context, testing is an integral and continuous activity that happens in parallel with development. The "Testing" phase involves both automated and manual testing, ensuring that each increment is thoroughly validated. Testers collaborate closely with developers to identify and address issues promptly. The iterative nature of Agile allows for the early detection and resolution of defects, contributing to the overall quality of the product.

### **Deployment:**

The Deployment phase in Agile enables incremental and continuous delivery of product increments. Completed features are deployed/implemented to a staging environment for validation before being released to production. The emphasis is on ensuring a smooth and continuous delivery process that aligns with customer needs.

### **Maintenance:**

Maintenance in Agile involves addressing any issues identified after deployment. Regular review and feedback sessions characterize the "Maintenance" phase in Agile. At the end of each iteration, a sprint review is conducted to showcase completed features to stakeholders. This session provides an opportunity to gather feedback, assess the achieved goals, and discuss potential improvements. The iterative feedback loop ensures that the product evolves based on real-world usage and stakeholder insights, contributing to continuous improvement.



### Advantages of the Agile Model:

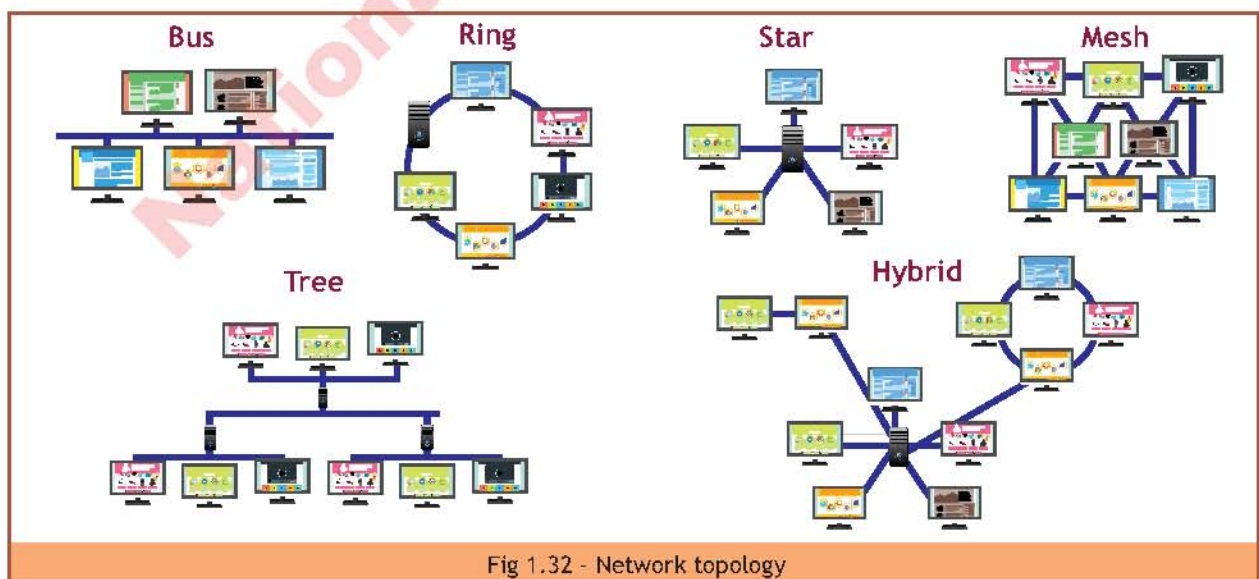
- Agile model offers flexibility and adaptability, allowing adjustments for evolving requirements.
- In this model continuous client involvement ensures higher satisfaction.
- It enables early and incremental delivery of a functional product.
- Continuous improvement is facilitated through regular demonstrations.
- Improved communication is emphasized, fostering collaboration.

### Disadvantages of the Agile Model:

- Dependency on customer availability poses a challenge.
- The potential for scope creep exists with the flexibility to accommodate changes.
- Coordination challenges emerge with larger team sizes.
- Limited emphasis on documentation can be a drawback.
- Not ideal for projects with stable and well-defined requirements.

## 1.5 Network Topology

Network topology is a systematic arrangement of computers and other devices in a network. It is an important concept when building or managing a computer network. It is the backbone of any networking application. It defines how computers, servers, switches, routers, storage devices, and other network-related devices are connected with each other and set up to effectively share data and resources across the system in an organized manner. Network devices are called “Nodes”. In network topology all nodes on a network are physically or logically arranged in relation to each other. There are several ways to arrange a network. Each has advantages and disadvantages and depending on the needs of the company or organization. Some common topologies are shown in the Fig. 1.32 as follows.



There are two approaches to network topology: **Physical** and **Logical**. Physical network topology, as the name suggests, refers to the physical connections and interconnections between nodes and the network—the wires, cables, and so forth. Logical network topology is a little more abstract and strategic, referring to the conceptual understanding of how and why the network is arranged the way it is, and how data moves through it.

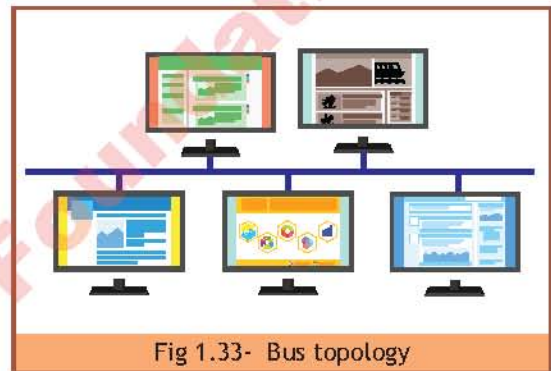
### 1.5.1 Types of Network Topologies

Various network topologies are available for configuring a network. The most commonly used topologies include bus, star, ring, mesh, tree and hybrid. Each topology comes with its own set of advantages and disadvantages. The selection of the most suitable topology for a particular organization or company depends on its specific needs and requirements.

#### Bus Topology

Bus topology is a network topology in which devices are connected to one cable or line running through the entire network. It is one of the most cost-effective and efficient ways to connect computers, printers, and other devices. With bus topology, all the data travels along a single cable, making it easy to install and maintain.

**Ethernet** is commonly used in bus topologies. It defines the hardware (such as cables and connectors) and communication protocols for devices in the network. Ethernet operates using the **CSMA/CD** (Carrier Sense Multiple Access with Collision Detection) protocol to manage access to the shared communication medium.



#### Advantages of Bus Topology:

- The main advantage of bus topology is that it is easy to install and maintain. All the nodes are connected to one cable, eliminating the need for complex wiring, and this makes it a great choice for smaller networks.
- Bus topology is very cost-effective because all the nodes are connected to one cable. Users don't have to buy multiple cables or network switches, making it a great choice for those on a budget.
- Bus topology has a fast data transmission rate because all the nodes are connected to one cable, making it ideal for applications requiring high-speed data transfer.
- Bus topology is relatively easy to expand by simply connecting additional nodes to the existing cable. This makes it a great choice for larger networks that require scalability.

#### Disadvantages of Bus Topology:

- In bus topology all nodes share the same bandwidth. This can lead to network congestion and slow down transmission speeds.



- In this topology if one node goes down, it can cause a network failure because there is only one cable. In addition, the entire network will be affected if the main cable is damaged or cut.

### Applications of Bus Topology:

- Bus topology is a great choice for small office networks because of its cost-effectiveness and ease of installation.
- Bus topology can also be used in home networks due to its low cost and ease of use.

### Star Topology

In a star topology, every device is linked to a central hub or switch. This configuration is widely employed and popular due to its flexibility, scalability, and ease of installation.

In a star topology, devices are typically categorized as **clients** or end-user devices. Clients include computers, laptops, smartphones, and other devices that connect to the network to access services or resources offered by a **Server**. A server is a powerful computer or device that provides services, resources, or data to other devices in the network. Servers are often connected to the central hub in a star topology, allowing multiple clients to access their services.

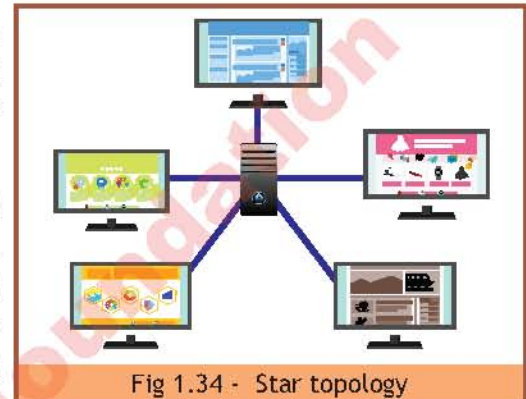


Fig 1.34 - Star topology

### Advantages of Star Topology:

- Star topology is relatively easy to install, as all nodes are connected directly to the hub or switch. This eliminates the need for complex wiring and makes it an excellent choice for more extensive networks.
- Star topology is very reliable because it has no single point of failure. If one node goes down, the rest of the network will still be operational.
- Star topology has a high-performance rate due to its dedicated links between nodes and the hub, making it ideal for applications requiring fast data transfer.

### Disadvantages of Star Topology:

- Star topology can be more expensive than other types of networks because of the additional cost of hubs or switches required to connect all devices together.
- Star topology is more prone to failure than other types of networks, as all nodes are connected to one hub or switch. If the hub or switch fails, the entire network will be affected.

### Applications of Star Topology:

Star topology is often used in large office buildings and campus networks due to its scalability and reliability. It is also a great choice for home networks as it is easy to install and maintain.



## Mesh Topology

Mesh topology is a network configuration where every node establishes connections with all other nodes within the network. This type of topology is particularly advantageous for larger networks, offering a significant level of redundancy and reliability.

### Advantages of Mesh Topology:

- Mesh topology provides a high level of redundancy as each node is connected to all other nodes in the network. This ensures continuous data transmission even if one or more nodes experience failure.
- Mesh topology is very reliable due to its redundant connections. The redundant connections in mesh topology contribute to its exceptional reliability. If one node experiences a failure, the remaining network nodes continue to operate seamlessly.
- Mesh topology demonstrates high flexibility and can be easily scaled to accommodate the requirements of larger networks.

### Disadvantages of Mesh Topology:

- The implementation of mesh topology can be costly due to the requirement for multiple links between nodes, increasing expenses related to hardware and installation.
- Setting up a mesh topology network can be complex due to the multiple connections required between nodes.

### Applications of Mesh Topology:

Mesh topology is often used in corporate networks and military applications due to its high redundancy and reliability. It is a great choice for wireless applications as well due to its flexible and scalable network structure.

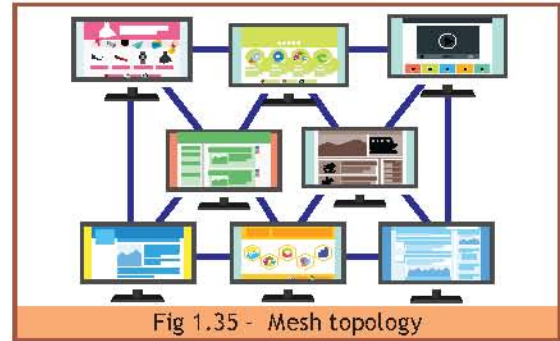


Fig 1.35 - Mesh topology

## Tree Topology

Tree topology (or hierarchical topology) is a network topology that utilizes a hierarchical or tree-like layout of interconnected nodes. It is similar to the star topology but with multiple hubs or switches instead of one, allowing for more efficient data transmission and scalability. This structure exhibits a hierarchy of nodes, with a central or top-level node commonly referred to as the "Root node".

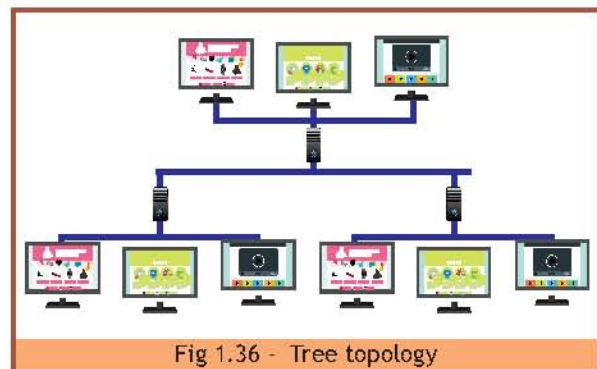


Fig 1.36 - Tree topology

### Advantages of Tree Topology:

- Tree topology is more scalable than other types of networks as it allows for the addition and

removal of nodes without disrupting the entire network. This makes it ideal for those networks that require frequent changes or expansion.

- Tree topology provides a high level of reliability due to its redundant connections between nodes. If one node fails, the rest of the network will still be operational.
- Tree topology is cost-effective as it eliminates the need for complex wiring and allows for more efficient data transmission.

#### Disadvantages of Tree Topology:

- Installing a tree topology network can be complex due to the multiple connections required between nodes.
- Troubleshooting a tree topology network can be difficult as there are multiple paths for data transmission.

#### Applications of Tree Topology:

Tree topology is well-suited for large corporate environments where departments or divisions can be organized in a hierarchical fashion, and the network can be easily scaled to meet evolving connectivity requirements.

#### Ring Topology

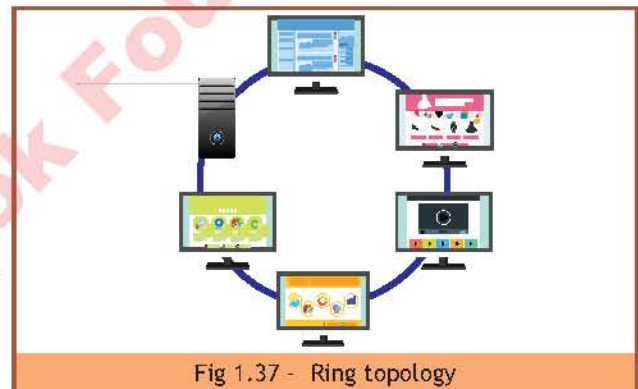
Ring topology is a network topology in which all devices are connected to one another in a circular loop. Data travels around the ring in one direction, passing through each device until it reaches its destination. Ring topology is often used Fiber Distributed Data Interface (FDDI) networks, also called dual ring network.

In a Ring Topology, the key term associated with the communication protocol is "**Token Ring**". Token Ring is a network protocol that controls access to the network by passing a special data packet, known as a "**token**," around the network. Only the device holding the token is allowed to transmit data. The token circulates continuously around the ring.

**Dual Ring topology:** Ring topology often use Fiber Distributed Data Interface (**FDDI**) networks, also called dual ring network. In a dual ring topology, two separate rings are connected to form one large loop. This provides greater security and efficiency as data can travel in both directions around the loop in a **clockwise** or **counterclockwise** direction.

#### Advantages of Ring Topology:

- Ring topology provides a high level of reliability as each device has an alternate route in case of failure.
- Ring topology is cost-effective as it eliminates the need for complex wiring and allows for more efficient data transmission.





### Disadvantages of Ring Topology:

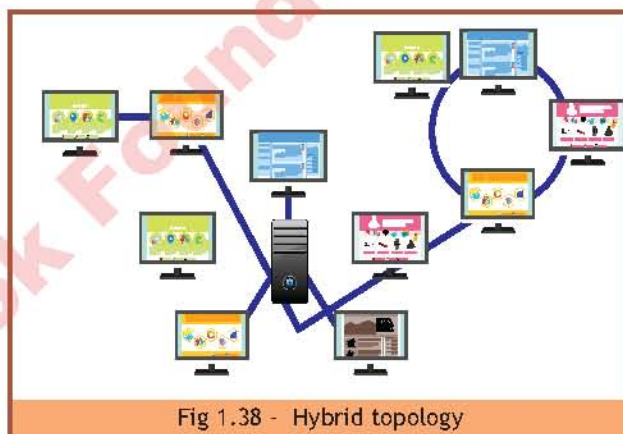
- Ring topology faces the challenge of limited bandwidth since each device on the ring must share the same communication path. This sharing can result in bottlenecks and affect the data transmission speed.
- The ring topology rely on a continuous circular connection between all nodes. If this connection is disrupted or broken at any point, the entire network may become unavailable.
- The troubleshooting process for a ring topology network is difficult and can be challenging.

### Applications of Ring Topology:

Ring topology is often used in Local Area Networks (LANs) due to its scalability and reliability. Dual ring topology is also used in fiber-optic networks as it provides more efficient data transmission than other topologies.

### Hybrid Topology

Hybrid Topology integrates different network configurations to establish a more efficient and dependable network. Due to its scalability and reliability, it is often used in large-scale networks such as corporate networks, military applications, and wireless networks. Hybrid topology combines elements from star, tree, ring, and bus topologies to create an interconnected network with multiple pathways for data transmission.



### Advantages of Hybrid Topology:

- Hybrid topology is flexible and easily adaptable to changes in the network.
- With multiple pathways for data transmission, hybrid topology offers higher reliability.
- The configuration of hybrid topology facilitates high scalability, accommodating modifications or expansions in the network.
- By merging two or more basic networks, hybrid topology reduces the overall cost of a network by reducing the amount of hardware and wiring needed.
- Hybrid topology enhances security by providing multiple layers of protection, making unauthorized access to the network more challenging.

### Disadvantages of Hybrid Topology:

- Hybrid topology can be more complex to set up and maintain, as it requires the merging of different types of topologies.
- Troubleshooting a hybrid topology network is more challenging as there are multiple pathways for data transmission.

### Applications of Hybrid Topology:

Hybrid topology is often used in large-scale applications and industrial applications such as process control systems, as it provides a reliable connection between nodes in complex environments.

## 1.5.2 Scalability and Reliability of Network Topologies

Scalability and reliability are two important aspects of networking systems. These can be influenced by the choice of network topology. Network topology refers to the physical or logical layout of devices and connections within a network. Different topologies have different implications for scalability and reliability.

### Scalability and Reliability in Star Topology

- **Scalability:** Star topology is moderately scalable. Adding more devices is possible by connecting them to the central hub or switch. However, the capacity of the central hub can become a bottleneck as the network grows.
- **Reliability:** Star topology is relatively reliable. If a device or cable fails, it typically only affects that specific device's connectivity, not the entire network.

Example: In a corporate office network using a star topology, as the company expands, new employees' computers can be added by connecting them to the central switch. However, if the central switch fails, all connected devices lose connectivity.

### Scalability and Reliability in Bus Topology

- **Scalability:** Bus topology is not highly scalable. Adding more devices can lead to signal degradation, as electrical signals must travel the entire length of the bus. Extending the bus often requires signal repeaters or switches.
- **Reliability:** Bus topology is less reliable because a single break in the main cable can disrupt the entire network.

Example: Older Ethernet networks used bus topology. If there is a cable break in the main bus, it can disable communication for all devices on the network segment. An **Ethernet network** is a widely used local area network (LAN) technology that enables devices within a specific geographical area, such as a home, office, or campus, to communicate with each other. It employs a protocol known as Ethernet for data transmission over a wired connection.

### Scalability and Reliability in Ring Topology

- **Scalability:** Ring topology is moderately scalable. Adding more devices to the ring is possible, but it can become complex to manage as the number of devices increases.
- **Reliability:** Ring topology can be highly reliable. Data can travel in both directions, reducing the risk of a single point of failure. However, if one device or cable segment fails, it can disrupt the entire ring.

Example: A token ring network, used in some legacy LANs, forms a ring topology. If one device



fails, it does not disrupt the entire network, but the failed device cannot participate in communication.

### Scalability and Reliability in Mesh Topology

- **Scalability:** Mesh topology offers high scalability. Devices can be added with ease, and each device has multiple paths to communicate with others, reducing congestion.
- **Reliability:** Mesh topology is highly reliable. The redundancy of multiple paths ensures that if one path or device fails, data can take an alternative route, minimizing downtime.

Example: Large data centers and cloud networks often use mesh topologies to ensure scalability and redundancy. Adding servers to such networks does not require a significant overhaul.

### Scalability and Reliability in Hybrid Topology

- **Scalability:** Hybrid topologies can be highly scalable, depending on the combination used. It allows for combining the strengths of multiple topologies.
- **Reliability:** Reliability in hybrid topologies varies based on the components used. Redundancy from mesh components can enhance reliability.

Example: A large enterprise network may use a hybrid topology, combining a star topology for ease of management in office branches with a mesh topology for redundancy in data centers.

## 1.5.3 Testing the Scalability and Reliability of a Network system

Testing the scalability and reliability of a network system is essential to ensure that it can handle increasing workloads and maintain high availability under various conditions.

### Testing Scalability

- **Load Testing:** Conduct load testing to evaluate the network's performance under heavy traffic conditions. Tools like Apache, JMeter, LoadRunner, or locust.io can simulate a large number of users or devices accessing the network simultaneously.
- **Stress Testing:** Stress testing involves pushing the network beyond its capacity to identify breaking points. Gradually increase the load or traffic until the network performance starts to experience significant latency or fail.
- **Scalability Testing:** Perform scalability testing by adding resources (e.g., servers, switches, or routers) to the network and measuring its ability to handle increased demand. Test horizontal scalability (adding more instances) and vertical scalability (upgrading existing instances).
- **Benchmarking:** Compare network performance against industry standards or competitors to assess how well it scales.
- **Realistic Scenarios:** Use real-world usage scenarios to test scalability, considering peak traffic times and growth projections.
- **Performance Monitoring:** Implement performance monitoring tools to continuously monitor the network's scalability in production environments.

## Testing Reliability

- **Availability Testing:** Test the network's availability by simulating various failure scenarios, including hardware failures, network congestion, and server crashes. Measure the time it takes to recover from failures.
- **Redundancy Testing:** Evaluate the effectiveness of redundancy mechanisms, such as load balancing, failover, and backup systems. Simulate the failure of redundant components to ensure seamless failover.
- **Disaster Recovery Testing:** Test disaster recovery plans to ensure data and services can be restored in the event of disastrous failures or data breaches.
- **Fault Tolerance Testing:** Verify the network's ability to continue functioning even when individual components fail. Intentionally inject faults or errors into the network and observe how it responds.
- **Security Testing:** Security is a key aspect of reliability. Perform penetration testing and vulnerability assessments to identify and address security weaknesses.
- **Load Balancing Testing:** Verify that load balancing mechanisms evenly distribute traffic across servers or resources to prevent overloading and maintain reliability.
- **Continuous Monitoring:** Implement continuous monitoring tools to detect anomalies and potential issues in real-time, allowing for preemptive maintenance and improvements.
- **Documentation and Reporting:** Maintain thorough documentation of tests performed and their results. Provide detailed reports to stakeholders.

### 1.5.4 Cloud Computing

Cloud computing is a technology model that provides access to computing resources and services over the internet, rather than owning and maintaining physical hardware and infrastructure. In a cloud computing environment, users can access and use computing resources such as servers, storage, databases, networking, software, and analytics through the internet.

Cloud computing has become a fundamental technology for businesses and individuals, offering scalability, cost-effectiveness, and the ability to innovate rapidly without the burden of managing physical infrastructure.

Key characteristics of cloud computing include:

- **On-Demand Self-Service:** Users can provision and manage computing resources as needed, without requiring human intervention from service providers.
- **Broad Network Access:** Services and resources are accessible over the network and can be accessed by various devices with internet connectivity.

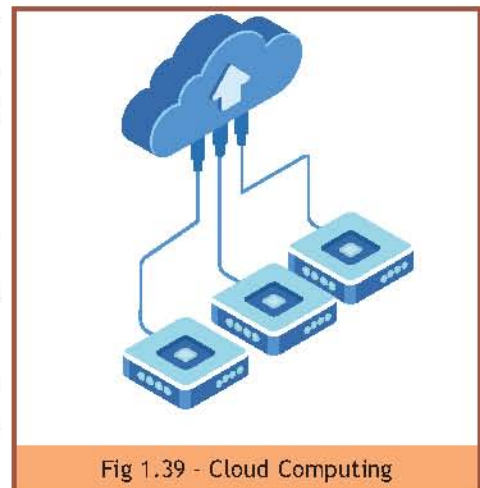


Fig 1.39 - Cloud Computing



- **Resource Pooling:** Computing resources are shared and pooled to serve multiple customers. Resources dynamically adjust to meet demand.
- **Rapid Elasticity:** Resources can be quickly scaled up or down to accommodate changes in demand, providing flexibility and cost efficiency.
- **Measured Service:** Usage of computing resources is monitored, controlled, and billed based on the actual usage.
- **Security:** Cloud providers implement robust security measures to protect data and resources. This includes encryption, identity and access management, and compliance with industry standards and regulations.

Cloud computing is typically categorized into three service models and four deployment models:

### Service Models:

- Infrastructure as a Service (IaaS):** This model delivers virtualized computing resources, including virtual machines, storage, and networks, via the internet.
- Platform as a Service (PaaS):** It offers a platform that includes tools, services, and frameworks for application development, without the complexity of managing underlying infrastructure.
- Software as a Service (SaaS):** SaaS delivers software applications through the internet on a subscription basis, eliminating the need for users to locally install, maintain, or update software.

### Deployment Models:

- Public Cloud:** In this model cloud resources are owned and operated by a third-party cloud service provider and are made available to the general public.
- Private Cloud:** In this model cloud resources are used exclusively by a single organization, providing more control and customization over the infrastructure.
- Hybrid Cloud:** It combines elements of both public and private clouds, allowing data and applications to be shared between them.
- Community Cloud:** In this the cloud infrastructure is shared by several organizations with common interests, such as regulatory requirements or industry standards.

### Examples of Cloud Computing Services:

- **Amazon Web Services (AWS):** Offers a wide range of cloud services, including computing power, storage, databases, machine learning, and more.
- **Microsoft Azure:** Provides cloud services for computing, analytics, storage, and networking, along with tools for building, deploying, and managing applications.
- **Google Cloud Platform (GCP):** Offers cloud services for computing, storage, machine learning, and data analytics, along with various development tools.
- **IBM Cloud:** Provides a range of cloud computing services, including IaaS, PaaS, SaaS, and

hybrid cloud solutions.

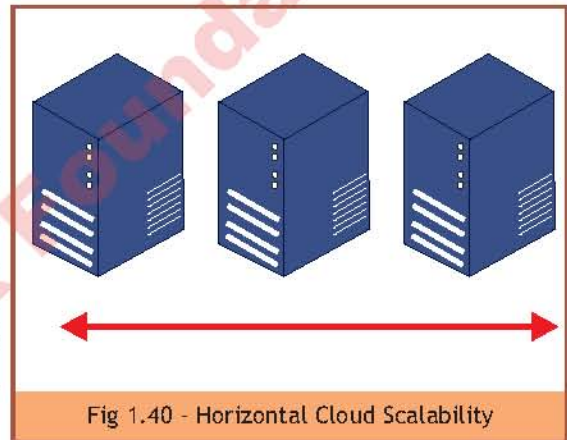
### 1.5.6 Scalability and Reliability in Cloud computing

Scalability and reliability are two important characteristics of cloud computing, each serving a distinct but interconnected purpose in ensuring that cloud services meet the demands of users and applications effectively.

#### Scalability in Cloud Computing

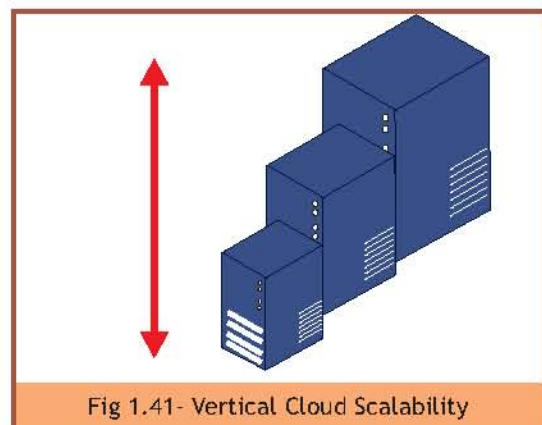
Scalability in cloud computing refers to the ability of a cloud system or service to handle increasing workloads, either by expanding resources (scaling out) or upgrading existing ones (scaling up). Scalability ensures that as the demand for computing resources grows, the cloud infrastructure can accommodate it without causing a drop in performance or service quality. There are two main types of scalability in cloud computing:

- **Horizontal Scalability (Scaling Out):** Horizontal scaling means increasing the number of servers that run the application, and distributing the workload among them. This can enhance the scalability and availability of the application to users, as it can handle more requests and tolerate failures or outages of individual servers. However, horizontal scaling also has some challenges and trade-offs. It needs to design the application to support distributed architecture, and use techniques such as load balancing, replication, sharing, caching, or micro services to manage the communication and coordination among multiple servers. Also it needs to deal with the complexity and overhead of managing more servers, and ensure the consistency and security of the data and code.



Horizontal scaling is primarily useful for organizations that need high availability and near-zero downtime for their online services. Compared to vertical scaling, horizontal scaling is quicker and easier to accomplish.

- **Vertical Scalability (Scaling Up):** Vertical scaling means increasing the capacity of a single server by adding more resources, such as CPU, RAM, disk space, or network bandwidth. This can improve the performance and reliability of the application, as it can handle more requests and process them faster. However, vertical scaling also has some limitations and drawbacks. First, there is a physical limit to how much a single server can be





upgraded, and eventually more powerful machines need to be replaced. Second, vertical scaling can be costly and time-consuming, and deals with potential downtime and compatibility issues.

### Reliability in Cloud Computing

Reliability in cloud computing relates to the ability of a cloud service or infrastructure to consistently deliver its intended functionality and maintain uptime, often referred to as "high availability." Reliability ensures that applications and services hosted in the cloud are accessible and perform as expected, minimizing downtime and disruptions.

Key components of reliability in cloud computing include redundancy, fault tolerance, disaster recovery, and failover mechanisms. Cloud providers invest in robust infrastructure, data centers, and networking to maintain high levels of reliability.

**Example of Reliability in Cloud Computing:** Consider a financial institution that uses cloud-based storage for critical customer data. To ensure data reliability and availability, the cloud provider may employ redundancy by storing data in multiple data centers across different geographical regions. If one data center experiences an outage due to unforeseen circumstances, the data is still accessible from other locations, ensuring that customer transactions can continue without significant interruptions.

## 1.6 Cybersecurity

Cybersecurity is the protection of internet-connected systems such as computers, servers, mobile devices, electronic systems, networks, and data from malicious attacks. It is also known as information technology security or electronic information security. The practice is used by individuals and enterprises to protect against unauthorized access to data centers and other computerized systems.

### 1.6.1 Importance of Cybersecurity

Cybersecurity is vital in any organization, no matter how big or small the organization is. This is due to increasing technology and increasing software needs across various areas like government, education, hospitals, etc. Attackers target small and large companies and obtain their essential documents and information. Cybersecurity is becoming increasingly important in today's interconnected world. As more and more data is stored and transmitted electronically, the risk of cyber-attacks has also increased.

The following points further elaborate the need for cybersecurity.



Fig 1.42 - Cybersecurity

### **a) Protecting Sensitive Data**

With the increase in digitalization, data is becoming more and more valuable. Cybersecurity helps protect sensitive data such as personal information, financial data, and intellectual property from unauthorized access and theft.

### **b) Prevention of Cyber Attacks**

Cyber-attacks, such as Malware infections, Ransomware, Phishing, and Distributed Denial of Service (DDoS) attacks, can cause significant disruptions to businesses and individuals. Effective cybersecurity measures help prevent these attacks, reducing the risk of data breaches, financial losses, and operational disruptions.

### **c) Safeguarding Critical Infrastructure**

Critical infrastructure, including power grids, transportation systems, healthcare systems, and communication networks, heavily relies on interconnected computer systems. Protecting these systems from cyber threats is crucial to ensure the smooth functioning of essential services and prevent potential disruptions that could impact public safety and national security.

### **d) Maintaining Business Continuity**

Cyber-attacks can cause significant disruption to businesses, resulting in lost revenue, damage to reputation, and in some cases, even shutting down the business. Cybersecurity helps ensure business continuity by preventing or minimizing the impact of cyber-attacks.

### **e) Compliance with Regulations**

Many industries are subject to strict regulations that require organizations to protect sensitive data. Failure to comply with these regulations can result in significant fines and legal action. Cybersecurity helps ensure compliance with these regulations.

### **f) Protecting National Security**

Cyber-attacks can be used to compromise national security by targeting critical infrastructure, government systems, and military installations. Cybersecurity is critical for protecting national security and preventing cyber warfare.

### **g) Preserving Privacy**

In an era where personal information is increasingly collected, stored, and shared digitally, cybersecurity is crucial for preserving privacy. Protecting personal data from unauthorized access, surveillance, and misuse helps maintain individuals' privacy rights and promotes trust in digital services.



### 1.6.2 Cybersecurity Threats

Cybersecurity threats are harmful acts performed by individuals or groups with destructive intent that aims to gain unauthorized access, damage, disrupt, or steal an information technology asset, computer network, intellectual property, or any other form of sensitive data.

The following are some common types of cybersecurity threats.



Fig 1.43 - Cybersecurity threats

#### Malware (Malicious Software)

Malware is a broad category of software specifically designed to harm or exploit computer systems, steal data, or gain unauthorized access. Malware can take many forms and often disguises itself as legitimate software. Some common types of malware includes:

- **Viruses:** Malicious code that attaches itself to legitimate programs and spreads when the infected program is executed.
- **Worms:** Self-replicating malware that spreads across networks without user intervention.
- **Trojans:** Software that appears legitimate but hides malicious functions, such as remote control or data theft.
- **Ransomware:** Encrypts files or entire systems and demands a ransom for decryption keys.
- **Spyware:** Collects user information without their knowledge, often for advertising or surveillance purposes.

#### Phishing

Phishing is a social engineering attack where cybercriminals impersonate trusted entities to deceive users into revealing sensitive information, such as login credentials, credit card details, or personal information.

- **Spear Phishing:** Targeted phishing attacks directed at specific individuals or organizations, often using personalized information.
- **Email Phishing:** Cybercriminals send deceptive emails that appear to be from legitimate sources, encouraging recipients to click on malicious links or download infected attachments.

#### Denial of Service (DoS) and Distributed Denial of Service (DDoS) Attacks

DoS and DDoS attacks aim to overwhelm a target system or network with an excessive amount of traffic, rendering it inaccessible to legitimate users.

- **DoS:** A single attacker floods the target with traffic, often using multiple devices.
- **DDoS:** Multiple compromised devices coordinate to flood the target, making it more challenging to mitigate.

## Ransomware

Ransomware encrypts a victim's data and demands a ransom for the decryption key. Payment does not guarantee data recovery, and victims may lose access to critical information. Ransomware can lead to data loss, financial losses (including ransom payments), and operational disruptions.



Fig 1.44 - Ransomware

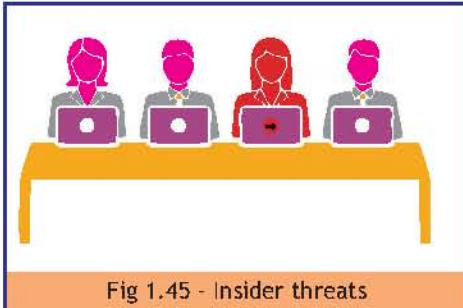


Fig 1.45 - Insider threats

## Insider Threats

Insider threats involve individuals within an organization (employees, contractors, or business partners) who misuse their access to systems and data for malicious purposes, for example sharing sensitive information with external parties. This can lead to data breaches, financial losses, and reputational damage.

## Cloud security threats

Cloud security threats are potential risks and vulnerabilities that can compromise the security of data, applications, and infrastructure hosted in cloud environments. As organizations increasingly adopt cloud computing services, it is important to be aware of these threats and take measures to shrink them. With businesses moving to cloud resources daily, many environments are growing more complex. This is particularly true in the case of hybrid and multi-cloud environments, which require extensive monitoring and integration.



Fig 1.46 - Cloud security threats

### 1.6.3 Protection against Cyber-Threats

Protecting computer systems against cyberattacks is crucial in today's digital age. Various methods and techniques are used to safeguard systems from different types of cyber threats.

The following are different methods and techniques for the protection of systems against cyberattacks.

#### Use strong passwords

Strong password security is an important aspect of cybersecurity. A strong password helps protect users' accounts and sensitive information from unauthorized access.



Fig 1.47 - Strong password



## Characteristics of a Strong Password:

A strong password typically possesses the following characteristics.

- **Length:** A strong password should be long, usually at least 12 characters. Longer passwords are harder to crack.
- **Complexity:** It should contain a mix of uppercase and lowercase letters, numbers, and special characters (e.g., !, @, #, \$, %).
- **Unpredictability:** Avoid using easily guessable information like names, birthdays, or common phrases.
- **Uniqueness:** Use different passwords for different accounts. Reusing passwords increases the risk if one account is compromised.

**Examples of Strong Passwords:** These password includes a mix of uppercase letters, lowercase letters, numbers, and special characters, making it complex and unpredictable.

P@ssw0rd\$Secur3!, Gr33nH0lid@y\$R0ck!, Op!f@8@2002aug, 8@Qn&\$v%KpLw3!@z

## Keep your software up to date

“Keeping the software up to date” is a fundamental practice in cybersecurity. It involves regularly updating software, including operating systems, applications, and security tools, to patch known vulnerabilities and protect against security threats. Software updates may also introduce new security features and improvements to better defend against emerging threats. Outdated software can be an easy target for malware. Updates help safeguard your system against malware attacks, such as ransomware and viruses.



Fig 1.48 - Keeping the software up to date



Fig 1.49 - Two-Factor Authentication

## 2FA (Two-Factor Authentication)

Authentication is the process of verifying the identity of a user or system trying to access a resource. It ensures that the entity is who it claims to be. **2FA** is an authentication method that requires users to provide two different forms of verification before granting access. Typically, it involves something the user knows (password) and something they have e.g. an OTP (one-time password) from a mobile app.

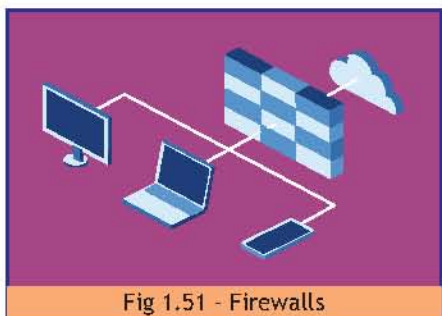
Example: Logging into an online banking account with a password and a unique code, OTP (one-time password) sent to the user's smartphone.

## Be wary of suspicious emails

Be cautious of unsolicited emails, particularly those that ask for personal or financial information or contain suspicious links or attachments.

## Educate yourself

Stay informed about the latest cybersecurity threats and best practices by reading cybersecurity blogs and attending cybersecurity training programs.



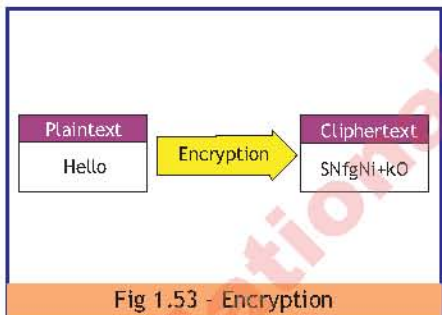
## Firewalls

Firewalls are network security devices or software that monitor and control incoming and outgoing network traffic. They establish a barrier between a trusted internal network and untrusted external networks (e.g., the internet) to filter and block malicious traffic.

Example: Blocking incoming requests from unknown or suspicious IP addresses to protect a corporate network.

## Antivirus and Anti-malware Software

Antivirus and anti-malware software are designed to detect, quarantine, and remove malicious software, such as viruses, Trojans, and spyware, from a system. These software provide real-time scanning of files and applications, ensuring that known malware is not allowed to execute on the system.



## Encryption

Encryption transforms data (plaintext) into a coded format (cipher text) that can only be deciphered with the appropriate decryption key. It ensures the confidentiality and integrity of sensitive data. Encryption protects data when it is transmitted over networks, making it unreadable to unauthorized people even if they gain access to it.

## Backup and Disaster Recovery

Regular data backups and disaster recovery plans ensure that in case of a cyberattack or data breach, critical data can be restored, minimizing downtime and data loss. Backup and recovery strategies help organizations recover from cyber incidents and ransomware attacks without paying ransoms.

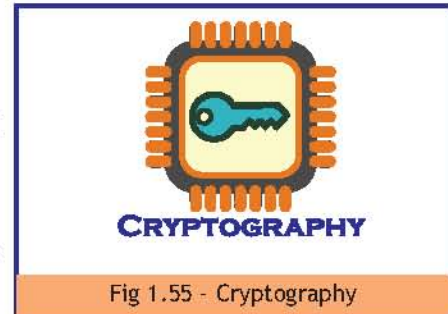




## Cryptography

Cryptography is a scientific approach of securing information by transforming it into an unreadable format using mathematical algorithms. It ensures data confidentiality, integrity, and authentication.

Example: Encrypting a message with a key to protect it from unauthorized access during transmission.

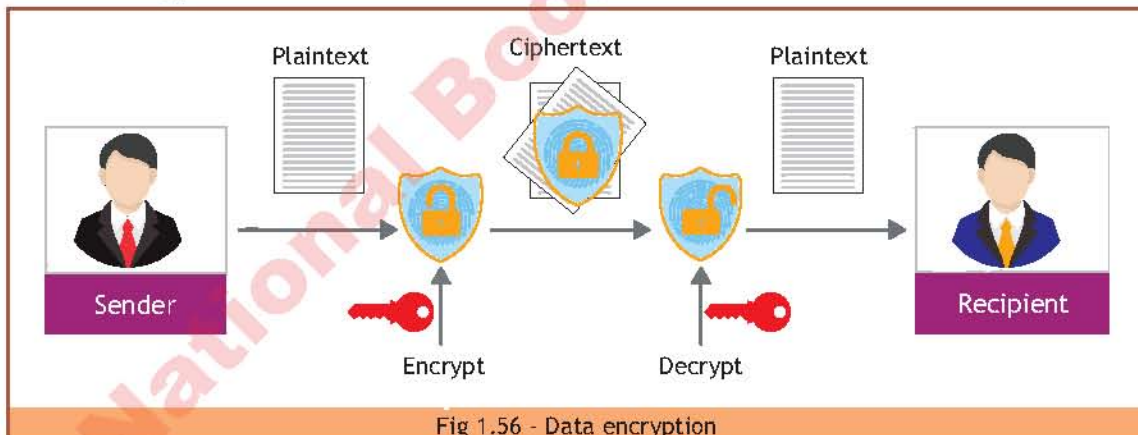


### 1.6.4 Encryption

Data encryption is a process of converting **plaintext** data into an unreadable format called **ciphertext** using encryption algorithms and keys. The primary purpose of data encryption is to protect sensitive information from unauthorized access, ensuring that even if someone gains access to the encrypted data, they cannot decipher it without the appropriate decryption key. Encryption primarily aims to achieve data confidentiality, ensuring that only authorized parties with the decryption key can read the protected data.

Data encryption plays a fundamental role in modern cybersecurity, and its importance continues to grow as digital threats and privacy concerns evolve. It provides a strong layer of defense against unauthorized access and data breaches, helping individuals and organizations keep their information confidential and secure.

#### How data encryption works:



- **Plaintext:** This is the original, human-readable form of the data that you want to protect. It can be any type of digital information, such as text, files, or communication messages.
- **Encryption Algorithm:** An encryption algorithm is a mathematical formula or process that transforms plaintext into cipher text. Different encryption algorithms have varying levels of complexity and security.
- **Encryption Key:** A key is a piece of information used by the encryption algorithm to control the transformation of plaintext into cipher text and vice versa. The key can be a numeric value, a passphrase, or a combination of characters. The security of the encryption depends on the

strength of the key.

The encryption process involves applying the encryption algorithm to the plaintext using the encryption key to produce the cipher text. This cipher text is a scrambled and unreadable version of the original data. To decrypt the data back into its original plaintext form, the recipient needs to possess the decryption key, which is used with the decryption algorithm to reverse the process. The whole encryption process is shown in the following Figure.

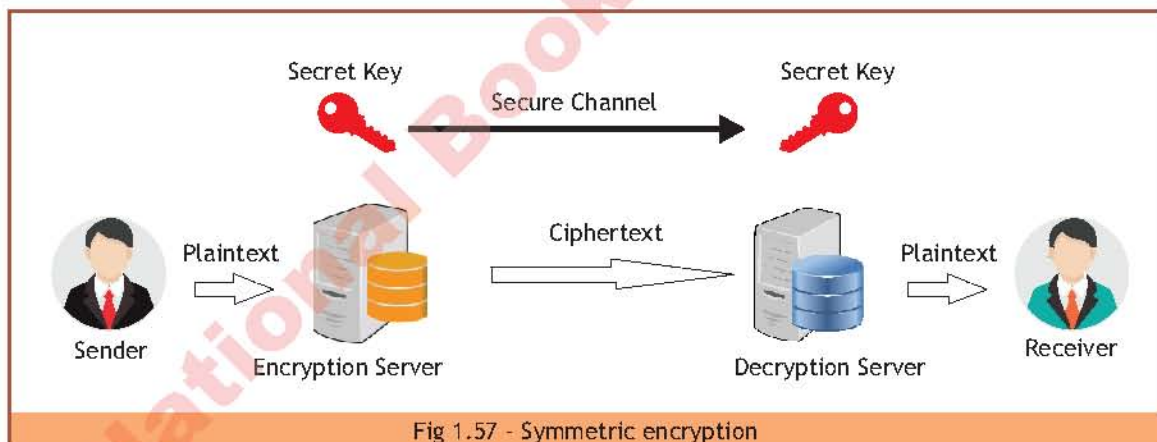
**Cryptography and Encryption** are related concepts in the field of information security, but they are not synonymous. Cryptography is the science of concealing messages with a secret code. Encryption is the way to encrypt and decrypt data. The first is about studying methods to keep a message secret between two parties (like symmetric and asymmetric keys), and the second is about the process itself.

There are two types of encryption algorithms:

- Symmetric
- Asymmetric

### Symmetric Encryption

Symmetric encryption is a fundamental data protection technique, relying on a single cryptographic key for both encrypting plaintext and decrypting cipher text. The process is shown in the following Figure.



Symmetric encryption is one of the most widely used encryption techniques. The objective of symmetric encryption is to secure sensitive information. It is used in many major industries, including defense, aerospace, banking, health care, and other industries in which securing a person's, business', or organization's sensitive data is of the utmost importance.

### Advantages of Symmetric Encryption:

- **Speed and Efficiency:** Symmetric encryption is generally faster and more computationally efficient than asymmetric encryption. It is well-suited for encrypting large amounts of data in real-time, such as data transmission or disk encryption.



- **Strong Security:** When used with strong, randomly generated keys, symmetric encryption provides a high level of security. The encryption and decryption processes are highly secure as long as the key remains secret.
- **Simplicity:** Symmetric encryption is simpler to implement and faster to execute because it involves a single key for both encryption and decryption.

### Disadvantages of Symmetric Encryption:

- **Key Distribution:** One of the major challenges with symmetric encryption is securely distributing and managing the secret keys. If a key is compromised, all encrypted data becomes vulnerable.
- **Limited Use for Secure Communication:** Symmetric encryption isn't suitable for secure communication between parties who have never met before or don't share a pre-established key.

### Asymmetric Encryption

Asymmetric encryption uses two separate keys: a **public key** and a **private key**. Often a public key is used to encrypt the data while a private key is required to decrypt the data. The private key is only given to users with authorized access. As a result, asymmetric encryption can be more effective, but it is more costly. The process is shown in the following Figure.

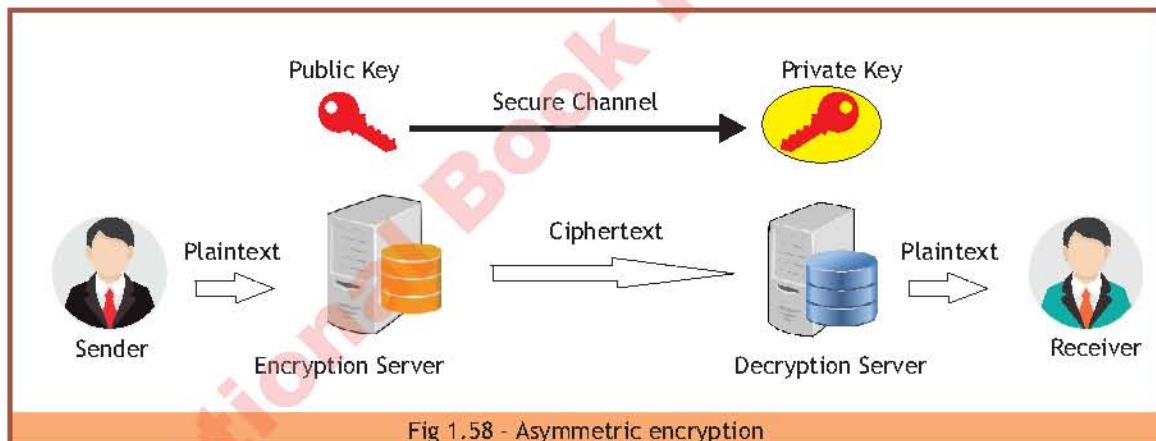


Fig 1.58 - Asymmetric encryption

### Advantages of Asymmetric Encryption:

- **Key Distribution:** The use of asymmetric encryption eliminates the necessity for a secure key distribution mechanism. Each user or entity possesses a unique pair of keys, public and private, where public keys can be openly shared. This approach simplifies the complexities associated with key management.
- **Non-refutation:** Asymmetric encryption provides non-refutation, which means the sender of a message cannot deny sending it because only their private key can decrypt the message.
- **Secure Communication with Untrusted Parties:** Asymmetric encryption is ideal for secure communication between parties who have never met before or don't share a secret key. It

enables secure key exchange for subsequent symmetric encryption.

### Disadvantages of Asymmetric Encryption:

- **Computational Complexity:** Asymmetric encryption algorithms are computationally intensive compared to symmetric encryption, which can lead to slower performance, especially when encrypting or decrypting large volumes of data.
- **Key Length:** Longer key lengths are needed to achieve the same level of security as shorter symmetric keys, which can increase the size of data and messages.

### Contrast between Symmetric and Asymmetric data transmissions

The contrast highlights the key differences between symmetric and asymmetric encryption, including their mechanisms, use cases, key distribution challenges, and computational characteristics.

Symmetric Encryption	Asymmetric Encryption
1. It uses a single shared key for both encryption and decryption.	1. It uses a pair of public and private keys for encryption and decryption.
2. In this, the same key is used by both the sender and the receiver.	2. In this, Public key is used for encryption, and private key is used for decryption.
3. It is well-suited for encrypting large amounts of data.	3. It is generally slower, and suitable for smaller amounts of data or secure key exchange.
4. It requires a safe channel to transmit the secret key for key distribution.	4. It enables secure key exchange over insecure channels without requiring a pre-established secure channel.
5. It is computationally less complex compared to asymmetric encryption.	5. It Involves more complicated mathematical operations, making it computationally more complex.
6. It provides better performance in terms of speed and efficiency.	6. It is slower due to the complexity of mathematical operations involved.
7. It is suitable for scenarios where a secure key distribution channel is established.	7. It is more suited for scenarios where secure key exchange over insecure channels is required.
8. It is often used in situations where performance and efficiency are critical.	8. It is commonly used for secure data transmission, digital signatures, and securing communication channels.



## Summary

### A Computer System Overview:

- A computer system is a fundamental and pervasive part of modern life, revolutionizing work, communication, learning, and entertainment.
- It encompasses a combination of hardware and software components working together for various tasks.

### Digital Operation of Computers:

- Computers are digital machines operating on binary data and binary logic.
- Data is processed digitally by converting input information into binary code (0s and 1s), storage in memory, and execution by the CPU.

### CPU and Binary Logic:

- The CPU performs arithmetic, logical, and control operations on binary data.
- Specialized units like the Arithmetic and Logic Unit (ALU) and control unit assist in processing.
- Conditional branching and looping enable decision-making and task repetition.

### Input and Output in Digital Computers:

- Results are converted back to human-readable form and presented through output devices.
- Digital approach enables computers to perform tasks with high speed and accuracy.

### Data Representation in Digital Computers:

- Binary digits (0 and 1) and alphanumeric codes are used for data representation.
- The American Standard Code for Information Interchange (ASCII) is a common alphanumeric code.
- ASCII uses 7 bits to represent 128 characters, including lowercase and uppercase letters, numeric digits, and special characters.

### Digital Logic and Logic Gates:

- Digital logic is fundamental for creating electronic devices like calculators, computers, and digital watches.
- Logic gates are building blocks of digital circuits, with inputs and outputs represented as LOW (0) or HIGH (1).
- Basic logic gates include AND, OR, and NOT gates, each with specific logic operations.
- Digital circuits operate using logic 0 and logic 1 to represent specific voltage levels.

### **Karnaugh Maps :**

- Karnaugh Maps (K-Maps) are used to simplify Boolean functions and reduce logic circuit complexity.
- K-Maps help group ones (1s) in a truth table, facilitating simplification.

### **Principle of Duality in Boolean Algebra:**

- The principle of duality allows interchange of AND and OR operators while complementing variables, maintaining logical equivalence.

### **Uses of Logic Gates:**

- Logic gates have numerous applications in memory circuits, clock synchronization, data encoding/decoding, mathematical operations, digital signal processing, encryption/decryption, calculators, traffic light control, robotics, security systems, automotive electronics, home automation, medical devices, and aerospace applications, etc.

### **Software Development Life Cycle (SDLC):**

- SDLC: is the process of creating new software products and involves various phases/steps.
- Defining the Problem Phase: Clearly defining the problem or system to be developed, documenting requirements.
- Planning Phase: Determining project objectives, estimating resources, and preparing project plans.
- Feasibility Study Phase: Assessing technical, economic, operational, legal, and schedule feasibility.
- Analysis Phase: Determining end-user requirements and evaluating the need for system replacement.
- Requirement Engineering Phase: Identifying and validating user expectations, including functional and non-functional requirements.
- Design Phase: Creating architectural and user interface designs, often using algorithms and flowcharts.
- Development/Coding Phase: Writing and coding the software based on design specifications.
- Testing/Verification Phase: Evaluating the software's functionality and performance, identifying and fixing defects.
- Deployment/Implementation Phase: Making the software available for use through



installation and training.

- Documentation Phase: Creating detailed documentation for communication and system maintenance.
- Maintenance/Support Phase: Continuously monitoring, maintaining, and enhancing the software.

#### **Software Development Models:**

- Waterfall Model: A sequential design process with phases that do not overlap, including Requirements Analysis and Specification, Design, Implementation and Unit Testing, Integration and System Testing, and Operation and Maintenance.
- Agile Model: An iterative development approach with phases including Requirements Gathering, Design, Development, Testing, Deployment, and Review. It emphasizes flexibility and adaptation.

#### **Network Topology:**

- Network topology refers to the arrangement of devices in a computer network.
- Common network topologies include bus, star, ring, mesh, tree, and hybrid.
- Each topology has its own advantages and disadvantages.
- Bus topology is cost-effective but can lead to congestion.
- Star topology is easy to install and reliable.
- Ring topology provides redundancy but has limited bandwidth.
- Mesh topology is highly redundant but can be expensive.
- Tree topology combines hierarchy and scalability.
- Hybrid topology combines multiple topologies for flexibility.

#### **Scalability and Reliability Testing:**

- Scalability testing assesses a network's ability to handle increasing workloads.
- Load testing, stress testing, and benchmarking are used for scalability testing.
- Reliability testing ensures consistent performance and uptime.
- Availability, redundancy, fault tolerance, and security testing are part of reliability testing.
- Continuous monitoring and documentation are essential for both aspects.

#### **Cloud Computing:**

- Scalability in cloud computing involves expanding resources horizontally (scaling out) or vertically (scaling up) to meet demand.

- Horizontal scalability adds more identical instances to distribute workloads.
- Vertical scalability involves upgrading existing resources within instances.
- Reliability in cloud computing ensures high availability and minimal downtime.
- Redundancy, fault tolerance, disaster recovery, and failover mechanisms enhance reliability.
- Cloud providers invest in robust infrastructure and data centers.

#### **Cybersecurity:**

- Cybersecurity involves protecting internet-connected systems, networks, and data from malicious attacks.
- It is essential for individuals and organizations to safeguard against unauthorized access to computerized systems.

#### **Importance of Cybersecurity:**

- Cybersecurity is crucial for organizations of all sizes due to increasing technology usage and software needs.
- It protects sensitive data, prevents cyber attacks, safeguards critical infrastructure, ensures business continuity, and ensures compliance with regulations.
- It also plays a vital role in protecting national security, preserving privacy, and maintaining trust in digital services.

#### **Cybersecurity Threats:**

- Cybersecurity threats include various types of malicious activities, such as malware (viruses, worms, Trojans, ransomware, spyware), phishing attacks, denial of service (DoS) and distributed denial of service (DDoS) attacks, insider threats, and cloud security threats.

#### **Protection Against Cyber Threats:**

- Strong Passwords: Use long, complex, and unique passwords to enhance security.
- Keep Software Updated: Regularly update software to patch vulnerabilities.
- Two-Factor Authentication (2FA): Require users to provide two forms of verification for access.
- Be Wary of Suspicious Emails: Avoid clicking on suspicious links or downloading attachments from unsolicited emails.
- Educate Yourself: Stay informed about cybersecurity threats and best practices.
- Firewalls: Use firewalls to filter and block malicious network traffic.
- Antivirus and Anti-malware Software: Employ software to detect and remove malicious



software.

- Encryption: Protect data confidentiality by transforming it into unreadable cipher text.
- Backup and Disaster Recovery: Regularly back up data and have recovery plans in place.
- Cryptography: Use mathematical algorithms to secure information.
- Differentiate between Cryptography and Encryption: Cryptography is the science of securing messages, while encryption is the process of encrypting and decrypting data.

#### Encryption:

- Encryption involves converting plaintext data into unreadable cipher text using encryption algorithms and keys.
- Encryption provides data confidentiality and requires decryption keys for data recovery.
- There are two types of encryption algorithms: symmetric (uses a single secret key) and asymmetric (uses a public-private key pair).
- Symmetric encryption is faster and simpler but requires secure key distribution.
- Asymmetric encryption uses a public-private key pair, simplifying key distribution and enabling secure communication with untrusted parties.



#### Teacher's Guide

Implement diverse assessment strategies, including practical exercise, quizzes and projects to judge students' understanding of cybersecurity concepts.



#### Activity 1: Encryption and Decryption

**Objective:** To demonstrate the principles of encryption and decryption.

- Explain the difference between symmetric and asymmetric encryption.
- Provide sample plaintext and encryption algorithms (e.g., Caesar cipher) for students to practice encrypting and decrypting messages.
- Create a challenge where students must decrypt a message without knowing the encryption method or key.
- Demonstrate the use of encryption software or tools for secure communication.
- Discuss real-world applications of encryption, such as secure email and data protection.

## Exercise



Select the best answer for the following Multiple-Choice Questions (MCQs).

1. What are the two fundamental digits used in binary code?
  - a) 0 and 2
  - b) 1 and 3
  - c) 0 and 1
  - d) A and B
2. How many bits does the ASCII code use to represent a character?
  - a) 8 bits
  - b) 16 bits
  - c) 32 bits
  - d) 64 bits
3. In a digital computer, what do logic levels "0" and "1" represent in terms of voltage?
  - a) 0V for "0" and 5V for "1"
  - b) 0V for "1" and 5V for "0"
  - c) 0V for both "0" and "1"
  - d) 5V for both "0" and "1"
4. Which of the following logic gates produces a HIGH (1) output only when all of its inputs are HIGH (1)?
  - a) AND gate
  - b) OR gate
  - c) NOT gate
  - d) XOR gate
5. Which principle in Boolean algebra involves interchanging AND and OR operators while complementing variables?
  - a) Absorption Law
  - b) Identity Law
  - c) Principle of Duality
  - d) Distributive Law
6. Which principle of Boolean algebra allows interchanging the AND and OR operators while negating the variables?
  - a) Complement principle
  - b) Dual principle
  - c) Inversion principle
  - d) Identity principle
7. Which phase of SDLC involves assessing whether the proposed software/system is feasible in terms of resources and budget?
  - a) Design Phase
  - b) Analysis Phase
  - c) Feasibility Study Phase
  - d) Coding Phase
8. Which phase of SDLC focuses on understanding user expectations and identifying software requirements?
  - a) Maintenance/Support Phase
  - b) Deployment/Implementation Phase
  - c) Requirement Engineering Phase
  - d) Testing/Verification Phase



9. Which Agile model phase involves evaluating the software to identify and fix defects?
- a) Design
  - b) Testing
  - c) Deployment
  - d) Requirements Gathering
10. Which software development model is characterized by iterative development and frequent updates throughout the project's lifecycle?
- a) Waterfall Model
  - b) Agile Model
  - c) Feasibility Model
  - d) Design Model
11. Which type of topology is known for its high redundancy and reliability?
- a) Bus Topology
  - b) Star Topology
  - c) Mesh Topology
  - d) Ring Topology
12. What is the main disadvantage of Ring Topology?
- a) Limited bandwidth
  - b) High cost
  - c) Complex installation
  - d) Prone to failure
13. What type of topology combines elements from multiple topologies to create a more efficient network?
- a) Star Topology
  - b) Bus Topology
  - c) Mesh Topology
  - d) Hybrid Topology
14. What type of scalability involves adding more identical virtual machines or instances in cloud computing?
- a) Horizontal Scalability
  - b) Vertical Scalability
  - c) Linear Scalability
  - d) Elastic Scalability
15. Which of the following is NOT a common type of malware?
- a) Viruses
  - b) Worms
  - c) Spyware
  - d) Firewall
16. What is the main advantage of asymmetric encryption over symmetric encryption?
- a) Faster encryption and decryption
  - b) Simplified key distribution
  - c) Non-repudiation
  - d) Lower computational complexity
17. What type of attack involves cybercriminals impersonating trusted entities to deceive users into revealing sensitive information?
- a) Malware attack
  - b) Denial of Service attack
  - c) Phishing attack
  - d) Insider threat
18. What type of encryption uses a single, secret key for both encryption and decryption?
- a) Asymmetric encryption
  - b) Public-key encryption
  - c) Symmetric encryption
  - d) Triple DES encryption

19. What does 2FA (Two-Factor Authentication) require for user access?
- a) Two different passwords
  - b) Two different encryption keys
  - c) Two different authentication methods
  - d) Two different user accounts methods
20. What is the primary purpose of a firewall in cybersecurity?
- a) Encrypting data
  - b) Monitoring network traffic
  - c) Blocking malicious traffic
  - d) Generating strong passwords



**Give short answers to the following Short Response Questions (SRQs).**

1. What is the principle of duality in Boolean algebra, and why is it important in digital logic?
2. How memory circuits use logic gates? Give their significance in digital systems.
3. Provide two examples of data encoding and decoding applications that involve logic gates.
4. Give three uses of logic gates.
5. What is the primary purpose of the Software Development Life Cycle (SDLC)?
6. Name the different phases of SDLC.
7. Why feasibility study is important in the SDLC? Give three reasons.
8. How does the design phase contribute to the development of a software system?
9. What is the significance of testing/verification in SDLC?
10. Give three advantages and 2 disadvantages of Bus Topology in networking?
11. How does Mesh Topology provide redundancy in network communication?
12. Compare and contrast Horizontal Scalability and Vertical Scalability in cloud computing.
13. Define cybersecurity. Also give its significance in today's interconnected world.
14. Name three common types of cybersecurity threats.
15. What is the role of encryption in cybersecurity, and how does it protect sensitive data?
16. Differentiate between symmetric and asymmetric encryption methods.
17. Why is it essential for individuals and organizations to keep their software up to date in terms of cybersecurity?
18. What is 2FA (Two-Factor Authentication)? Give its importance in securing user accounts.
19. What is the primary purpose of a firewall in network security, and how does it work?
20. What are the characteristics of a strong password? Give two examples.





**Give long answers to the following Extended Response Questions (ERQs).**

1. Design logic circuits for the following Boolean functions.
  - i)  $E1 = (\overline{A}+B) \cdot (\overline{A}+\overline{B})$
  - ii)  $E2 = (A.\overline{B}) + (A+B) \cdot (\overline{C})$
  - iii)  $E3 = (\overline{A}.B+C) + \overline{A} \cdot (C+B)$
  - iv)  $E4 = (\overline{A}+\overline{B}) \cdot \overline{B} + (\overline{A}+C)$
  - v)  $E5 = \overline{x}\overline{y}\overline{z} + \overline{x}\overline{y}z + \overline{x}yz + xy\overline{z}$
  - vi)  $E6 = x\overline{z} + \overline{x}\overline{y}$
2. Draw Truth tables for the Boolean functions in Q1.
3. Simplify the following Boolean functions using K-Maps method.
  - i)  $E1 = (\overline{A}.B) + (A.B) + (A.\overline{B})$
  - ii)  $E2 = (\overline{A}.B.C) + (A.B.C) + (A.\overline{B}.C) + (A.B.\overline{C})$
  - iii)  $E3 = (\overline{A}.B.\overline{C}) + (A.B.C) + (A.\overline{B}.C) + (A.\overline{B}.\overline{C})$
  - iv)  $E4 = (\overline{A}.B.\overline{C}) + (A.B.C) + (A.\overline{B}.C) + (A.\overline{B}.\overline{C}) + (\overline{A}.\overline{B}.C)$
  - v)  $E5 = \overline{x}\overline{y}\overline{z} + \overline{x}\overline{y}z + \overline{x}yz + xy\overline{z}$
  - vi)  $E6 = x\overline{z} + \overline{x}\overline{y}$
4. Compare and contrast the Waterfall model and Agile model in software development. Which one do you think is more suitable for modern software development, and why?
5. Discuss the role of requirements engineering in SDLC. What are the challenges and benefits of gathering and managing requirements effectively?
6. Outline the various methods of system deployment/implementation mentioned in the text (Direct, Parallel, Phased, Pilot). Provide real-world scenarios where each deployment method would be most suitable.
7. Explain Bus, Star and Ring network topologies. Give their advantages and disadvantages.
8. In the context of cloud computing, elaborate on the concepts of scalability and reliability. How do these concepts contribute to the effectiveness of cloud services? Provide a real-world example.
9. Explain Symmetric and Asymmetric encryption methods in the context of cybersecurity.
10. Imagine you are responsible for the cybersecurity of a large organization. Describe a comprehensive cybersecurity strategy that includes multiple layers of defense against various threats.



## Activity 2: Logic Gates

### 1: Logic Gates Simulation:

- Use digital logic simulator software (e.g., Logisim <https://sourceforge.net/projects/circuit>, Digital Works <https://download.freedownloadmanager.org/Windows-PC/Digital-Works/FREE-3.0.4.38.html>) to create logic gate circuits.
- Assign students various logic gate combinations and ask them to simulate and observe the outputs for different input combinations.
- Challenge them to design specific logic gate circuits to perform basic functions like addition, subtraction, or even a simple calculator.

### 2: Truth Table Construction:

- Provide students with Boolean expressions and ask them to construct truth tables for those expressions.
- Include expressions involving AND, OR, NOT, NAND, and NOR gates to cover a variety of scenarios.

### 3: Karnaugh Map Practice:

- Give students Boolean functions and ask them to simplify them using Karnaugh Maps.
- Provide various examples with different numbers of variables to challenge their simplification skills.

### 4: Logic Gate Applications:

- Present real-world scenarios where logic gates are used, such as traffic light control or elevator systems.
- Ask students to design logic circuits to simulate these scenarios using digital logic simulator software.

### 5: Hands-On Hardware Activity:

- If available, conduct a hands-on activity where students build simple logic gate circuits using breadboards, LEDs, and logic gate ICs.
- This will give them a tangible understanding of how logic gates work in hardware.



## Activity 3: SDLC

Start with a presentation or discussion on what SDLC is, its importance in software development, and its various phases.

Use visual aids and diagrams to illustrate the different phases of SDLC.

### 1: Phases of SDLC:

- Create a worksheet or handout with a list of SDLC phases and brief descriptions.
- Ask students to match each phase with its description.
- Discuss the correct answers as a class.

### 2: Case Study Analysis:

- Provide a real-world case study or scenario related to software development.
- Ask students to identify which phase of the SDLC each part of the scenario corresponds to.



- Encourage students to discuss their answers and reasoning.

### 3: Feasibility Study Exercise:

Divide students into groups.

- Provide each group with a different software project idea.
- Ask them to conduct a feasibility study for their assigned project, considering technical, economic, operational, legal, and schedule feasibility.
- Have each group present their findings to the class.

### 4: Requirement Gathering Practice:

- Give students a simple problem or task (e.g., creating a mobile app for a specific purpose).
- In pairs or small groups, have them brainstorm and document functional and non-functional requirements for the software.
- Share and discuss their requirements with the class.

### 5: Coding Practice:

- Depending on the programming languages students are familiar with, assign coding exercises related to a specific phase of the SDLC.
- For example, you can provide a coding task related to the development phase, where students write code to solve a problem.

### 6: Testing and Debugging:

- Present a buggy code snippet to the class.
- Ask students to work individually or in pairs to identify and fix the errors (bugs) in the code.
- Discuss common debugging techniques and strategies.

### 7: Deployment Strategies:

- Explain the four deployment/implementation methods mentioned in the text (Direct Implementation, Parallel, Phased, and Pilot).
- Discuss scenarios in which each method might be appropriate.
- Have students analyze and choose the most suitable deployment method for a given project scenario.

### 8: Documentation and Maintenance:

- Provide students with a sample software documentation template.
- Ask them to create documentation for a hypothetical software project, including user manuals, system architecture diagrams, and maintenance plans.
- Encourage them to discuss the importance of documentation in the maintenance phase.

### 9: Comparison of SDLC Models:

- Discuss the two common SDLC models mentioned in the text: Waterfall and Agile.
- Divide students into two groups, with one group focusing on Waterfall and the other on Agile.
- Have each group prepare a presentation comparing the advantages and disadvantages of their assigned model.



#### Activity 4: Network Topology Exploration

**Objective:** To introduce students to different network topologies and their advantages and disadvantages.

- Provide students with the text on network topology.
- Divide students into groups and assign each group one type of network topology (e.g., bus, star, ring, mesh, tree, hybrid).
- Have each group create a presentation or poster that includes:
  - An explanation of their assigned topology.
  - Advantages and disadvantages of the topology.
  - Real-world applications of the topology.
- Each group presents their findings to the class.
- Engage in a class discussion about which topology might be suitable for different scenarios (e.g., a small office, a large corporation, a data center).



#### Activity 5: Testing Network Scalability and Reliability

**Objective:** To demonstrate how to test the scalability and reliability of a network system.

- Introduce the concept of network testing as discussed in the text.
- Provide students with a simulated network environment (virtual machines, switches, routers, etc.) using network simulation software.
- Instruct students to perform the following tests on the simulated network:
  - Load testing: Simulate heavy traffic to assess network performance.
  - Stress testing: Push the network beyond its capacity to identify limitations.
  - Scalability testing: Add resources to the network and observe its ability to handle increased demand.
  - Redundancy testing: Test failover mechanisms and redundancy.
- Have students document the results and any issues encountered during testing.
- Discuss the importance of these tests in real-world network maintenance and troubleshooting.



#### Activity 6: Phishing Awareness

**Objective:** To raise awareness about phishing attacks and teach students how to identify phishing attempts.

- Show examples of phishing emails to students and discuss the common elements and red flags.
- Create a quiz or scenario-based exercise where students must identify phishing emails from legitimate ones.
- Provide tips on how to verify the authenticity of emails and websites.



- Demonstrate how to check email headers to identify phishing attempts.
- Discuss the consequences of falling for phishing scams and how to report them.



### Activity 7: Firewall Configuration

**Objective:** To introduce students to firewall concepts and how to configure a software firewall.

- Install a software firewall on the lab computers (if not already installed).
- Explain the basics of how firewalls work, including inbound and outbound traffic filtering.
- Guide students through the process of configuring firewall rules to allow or block specific applications or ports.
- Provide scenarios where students must create firewall rules based on security requirements.
- Discuss the importance of keeping firewalls up to date and monitoring firewall logs for suspicious activity.



### Activity 8: Cybersecurity Threat Analysis

**Objective:** To explore common cybersecurity threats and discuss preventive measures.

- Present various cybersecurity threat scenarios, such as malware infections, phishing attacks, and DDoS attacks.
- Ask students to analyze each scenario, identify potential vulnerabilities, and propose preventive measures.
- Provide case studies of real cyberattacks and discuss lessons learned.
- Organize group discussions or debates on cybersecurity topics, allowing students to present their findings and solutions.
- Encourage students to stay updated on cybersecurity news and trends and share relevant articles or reports in class.

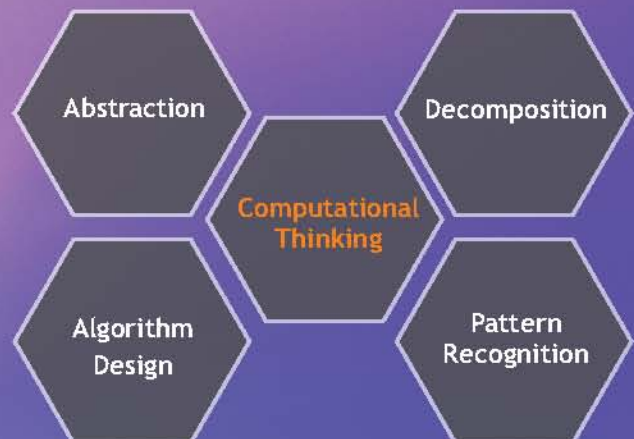
# Unit 02

## Computational Thinking and Algorithms



After completing this lesson, you will be able to:

- plan, develop, systematically test, and refine computational artifacts for problem solving.
- apply common search, and sort algorithms.





## UNIT INTRODUCTION

Computer systems are used to solve real word problems relevant to any topic or aspect of our life. For example education, healthcare, Transportation, economics. A problem is a challenge or situation for which a solution is required. The process of analyzing that situation and accordingly handling it is referred as problem solving.

Computational thinking is a thought process that helps us to understand the problem and solve it in a way that computers do. Computational thinking equires understanding the capabilities of computers. For example, computers process the input in a faster and reliable way and it works by following the input-process-output model.

To solve problems in any field including computer science, we can apply computational thinking. The solution of a problem where computational thinking is applied uses the following properties:

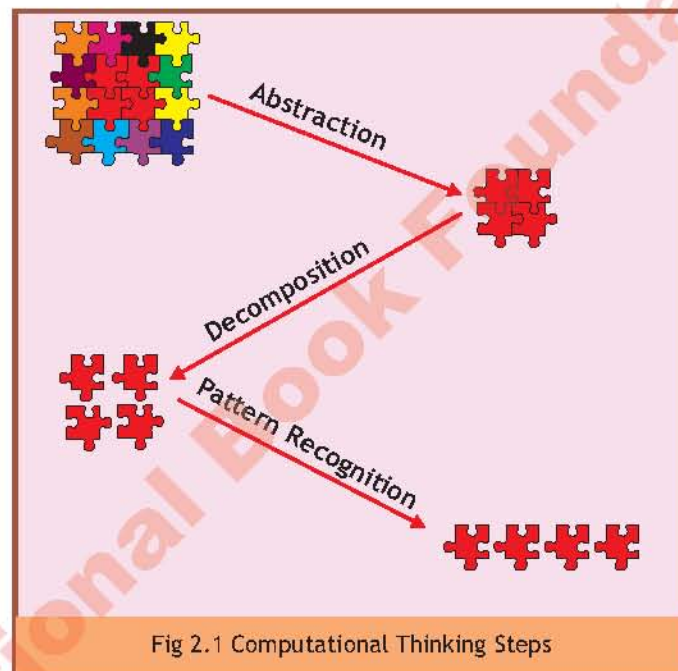


Fig 2.1 Computational Thinking Steps

### **Abstraction:**

Simplifying a problem or system by focusing on the most relevant aspects while ignoring unnecessary details. It is about looking at the big picture without getting lost in the tiny details.

### **Consider the task of Creating a website**

Abstraction is to focus on the main design and functionality without getting into the coding details. Think about the layout, colour scheme, and user interaction, rather than every line of code.

### **Decomposition:**

Breaking down a complex problem into smaller, more manageable sub-problems or tasks.

### Consider the task of Developing a mobile app

Decomposition is to break the task into smaller parts like designing the user interface, coding specific features (e.g., login, navigation), and handling data storage. Each part becomes a manageable sub-task.

### Pattern Recognition:

Identifying patterns or trends within data, problems, or solutions.

### Consider the task where we need to analysing user behaviour on a website.

Pattern Recognition is to identify trends, such as which pages users visit the most or common paths they take. This helps in making improvements based on observed patterns.

### Algorithmic Design:

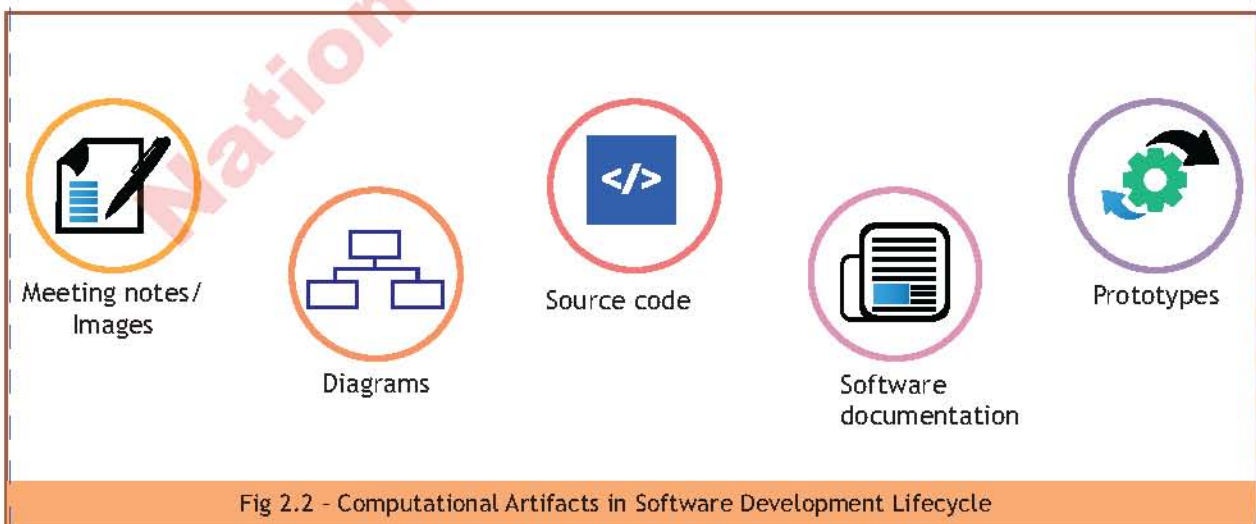
Developing a step-by-step solution to solve a problem.

## 2.1 Computational Artifacts

Computational artifacts refer to human made objects and systems using computational thinking. An artifact is a byproduct of software. Particularly in computer science, the computational artifacts could include programs, websites, videos, simulations, databases, digital animations, software systems, e-commerce platforms, and mobile applications.

### 2.1.1 Computational Artifacts in Software Development

In software development, the artifact could be an object that helps to describe the architecture, design, and function of software. There are a range of different artifacts generated during the lifecycle of software development as shown in the Fig 2.2. The artifacts are considered important because they make the process of developing software easy.





The artifacts generated during computational thinking in the software development process are the objects to be used before starting the coding phase. Artifacts define the behavior of a software with the help of some control sequences. By using these artifacts, software developers can easily understand how software works without requiring going into coding complexities.

### 2.1.2 Computational Solution Design

During the computational thinking, following are the commonly used computational artifacts.

#### a. Design Methods

- Algorithms
- Flowcharts
- Pseudocode

Algorithms, pseudocodes, and flowcharts serve as a structured and systematic representation of a computational process.

**Algorithms:** While solving a problem using a computer, the first step is to think about the algorithm. An algorithm is a step-by-step set of instructions that defines how a specific problem should be solved. It is a high-level description of the solution without worrying about the implementation details such as specific programming language. Algorithms are normally written in natural language.

#### Example

Let's consider a simpler example and write algorithm to add two numbers.

#### Algorithm: Add Two Numbers

1. Input two numbers, let's call them num1 and num2.
2. Add num1 and num2 together.
3. Store the result in a variable, let's call it sum.
4. Output the value of sum.

**Flowchart:** Flowchart is another tool used in the process of algorithm design. This provides a visual representation of flow and logic using defined symbols. Flowcharts are mostly helpful in explaining complex loops and decision-making processes.

#### Example

#### Flowchart: Add Two Numbers



**Pseudocode:** Once you have devised an algorithm, now it's time to create pseudocode. Pseudocode is an intermediate step between the algorithm and actual programming code. It's a more structured and semi-formal way of representing the algorithm using a combination of natural language and simplified programming constructs. Pseudocode helps in explaining the logic and structure of the algorithm before starting programming. It can be thought of as a detailed plan of how the algorithm should be implemented.

### Example

#### Pseudocode: Add Two Numbers

```
// Step 1: Input two numbers
    Input "Enter the first number: " into num1
    Input "Enter the second number: " into num2
// Step 2: Add the two numbers and store result in sum
    sum <- num1 + num2
// Step 3: Output the result
    Output "The sum of two numbers is: " sum
```

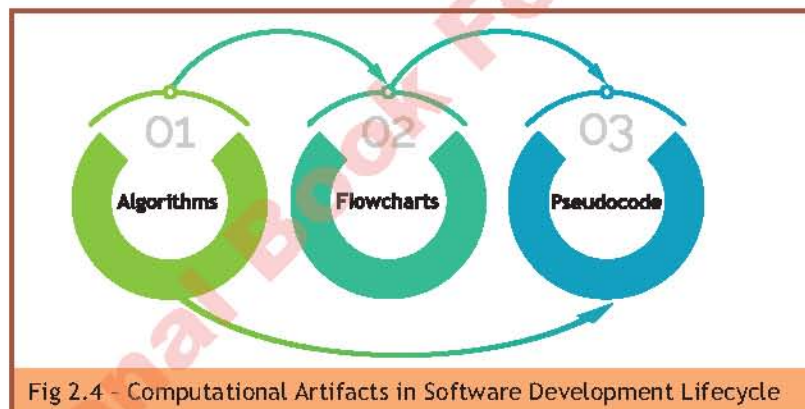


Fig 2.4 - Computational Artifacts in Software Development Lifecycle

However, it is important to note that the above sequence can vary depending on the complexity of the problem and your personal preference. Some programmers may skip flowcharts and go directly from algorithms to pseudocode or even from algorithms to writing code; especially for simpler tasks.

Let's consider another example to check whether a given number is even or odd and write algorithm, pseudocode and also draw its flowchart.

#### Algorithm: Check Even or Odd

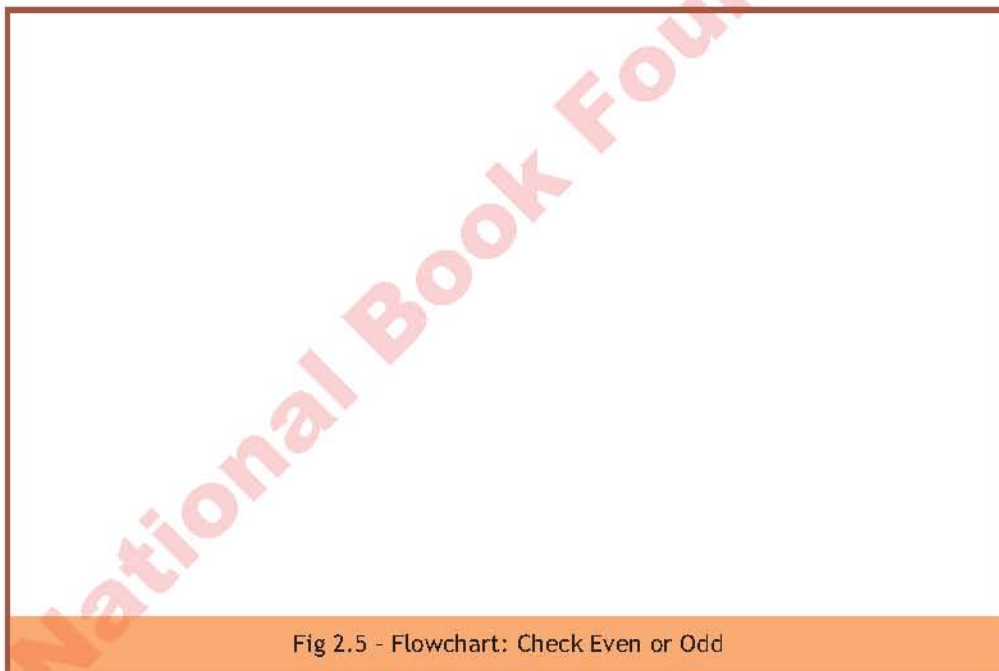
1. Input a number, let's call it num.
2. Check if num is divisible by 2.
3. If the remainder is 0, output "The number is even."
4. If the remainder is not 0, output "The number is odd."



### Pseudocode: Check Even or Odd

```
// Step 1: Input a number
    Input "Enter a number: " into num
// Step 2: Check if the number is even or odd
    if num % 2 = 0 then
// Step 3: Output result for even number
        Output " number is even."
    else
// Step 4: Output result for odd number
        Output " number is odd."
    end if
```

### Flowchart: Check Even or Odd



### b. Algorithm Design

The primary goal of pseudocode is to convey the algorithm's logic in a clear and understandable way, irrespective of specific programming language conventions.

Pseudocode writing is not standardized and there isn't a universal set of rules for its formatting. However, certain conventions are commonly used. Below is a set of guidelines that are often followed, but keep in mind that these are not strict rules, and you can adapt them based on your preferences or the requirements of a specific solution:

### Font, Size, Style:

Font: It is typically written in a plain, easy-to-read font.

Size: Use a standard font size that is easy to read, often around 10-12 points.

Style: Italicizing or bolding keywords is sometimes done for emphasis, but it's not a strict rule.

### Indentation:

Use consistent indentation to visually represent the structure of your code. Commonly, 2-4 spaces or a tab character is used for each level of indentation.

### Case:

While pseudocode is not case-sensitive, using uppercase for keywords and lowercase for variables can enhance readability. For example, `FOR i FROM 1 TO n` or `if condition then`.

### Line Numbers:

Including line numbers is optional and depends on personal preference or the requirements of a specific project. Line numbers can aid in discussions about the algorithm.

### Comments:

Comments are essential for explaining the logic. They are often denoted by symbols like `//` for single-line comments or `/**/` for multiline comments.

### Data Type Keywords:

Some pseudocode styles use data type keywords like `INTEGER`, `STRING`, etc., to indicate the type of a variable. However, this is optional.

### Variable Assignments & Declarations:

Variables can be declared using `DECLARE` or simply by mentioning them. Assignments are often denoted using `or :=`.

### Common Operators:

Use standard operators such as `+`, `-`, `*`, `/`, `=`, `<`, `>`, `<=`, `>=`, `!=` to represent common operations.

### Key Commands:

Key commands like `INPUT`, `OUTPUT`, `FOR`, `IF`, `WHILE` can be used to represent specific actions or control structures.

## 2.1.3 Planning and Developing a Computational Artifact

The following steps are used for planning and development of an artifact.

- a. **Define the problem:** State the problem you are trying to solve in clear and concise terms.
- b. **List the inputs** (information needed to solve the problem) and the expected outputs (what the algorithm will produce as a result)
- c. **Plan:**
  - i) **Outline the logic:** For this purpose, breakdown the problem into smaller problems and



consider how to solve each one.

ii) **Choose Data Structures:** A data structure is a storage to save and organize the data. We have different types of data structures such as array, stack, queue, etc. The choice of data structures is determined by considering the merits and drawbacks associated with each specific data structure.

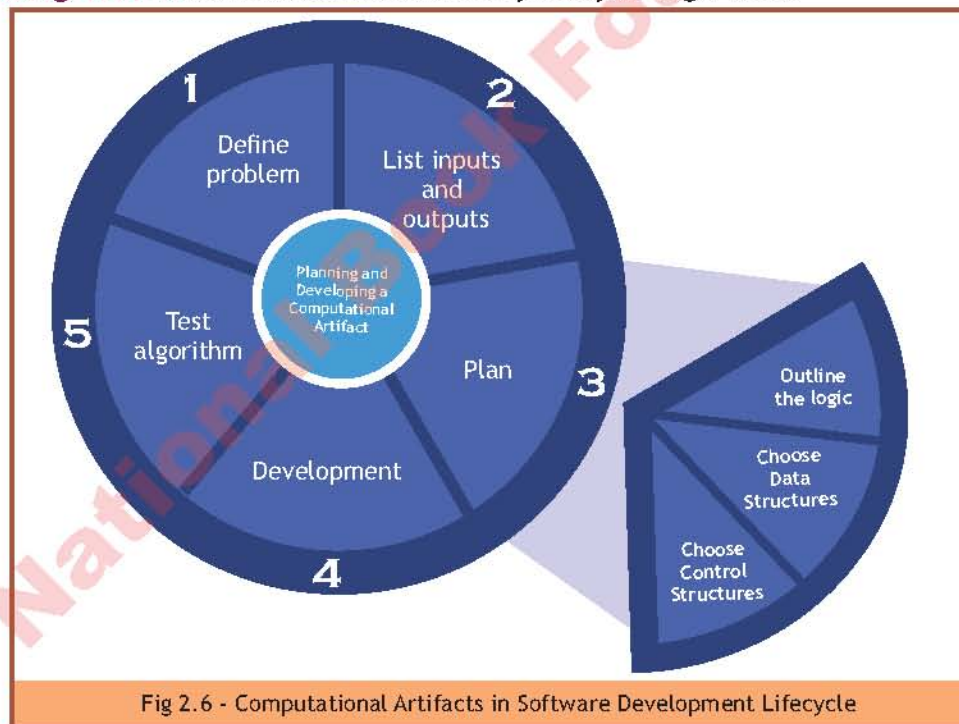
iii) **Choose Control Structures:** enables to determine the order of program execution. The sequential order of execution is disturbed in two aspects:

- **skip some program statements.** This is usually done with the help of conditional statements.
- **repeat one or more program statements.** This is done by using loops.

Those control structures are selected that are most appropriate for over solution.

d. **Development:** Describe the steps needed to convert or manipulate the inputs to produce the outputs. Start at a high level first and keep refining the steps until they are effectively computable operations. Keep the process simplified and remember that the number of steps should be finite.

e. **Test the algorithm:** choose data sets and verify that your algorithm.



### Example

To solve this example, we need to have some domain knowledge first about the problem.

Jeroo is a Kangaroo like animal. A Jeroo can pick flowers from a location and can plant them to some other. The Jeroo moves by hopping in the four directions e.g. forward, backward, right and left. As the Jeroo hop; it must avoid water, nets, and other of it kind.

## Problem Definition

Consider the given diagram, where the field is divided into 5x5 matrix and a Jerroo is available at position (0, 0) facing East. The flower is located at (3, 0). Write an algorithm where the Jerroo picks the flower and plant it to another location (3, 2). After its plantation, the Jerroo should move one hop East and stop there. Assume there are no other Jerroos, flowers or nets available in the field. **Top left corner of matrix is (0,0) values of x increases horizontally and y downwards.**



## List of inputs & Outputs

### Inputs:

Jerroo starts location (0, 0)  
flower current location (3, 0)

### Outputs

flower new location (3, 2)  
Jerroo should stop one hop East

## Plan:

Breakdown the problem:

Pick flower  
Put flower  
Move East

## Development:

Pick Flower

Hop 3 times  
take the flower

Put Flower

Turn right  
Hop 2 times



Plant the flower

Move East

Turn left

Hop 1 time

After the development of the algorithm, try to refine it by asking yourself the following questions:

- Does the developed algorithm solve some specific problem or could it may be generalized.

For example:

The algorithm that finds the area of a circle having radius of 3 meters (formula  $A=\pi 3^2$ ) solves a specific problem, but it could be made generalized by using the formula  $A=\pi r^2$

- Can the developed algorithm be made simplified?

For Example:

The formula for calculating perimeter of square is:

Perimeter of square = side + side + side + side

A simple formula would be:

Perimeter of square =  $4 \times \text{side}$

### 2.1.4 Testing Computational Artifacts

The computational artifacts should be tested by considering potential logical errors. For example, you should think what would happen if a user enters invalid or a certain out of range input. Testing and accordingly refinement of solution is the iterative process of a computational artifact.

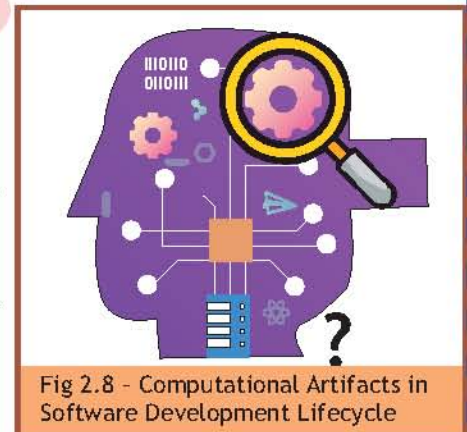


Fig 2.8 - Computational Artifacts in Software Development Lifecycle

#### a) Tracing an Algorithm

Tracing of algorithm is also known as a "desk check or dry-run". The programmers convert the algorithms to code that is ultimately handed over to computer for compilation and execution. But before going to write code for an algorithm, we manually verify that the algorithm works correctly. For this purpose, trace method is used that hand simulate the execution of your algorithm, keep change track of variable values and flow of logic control.

Tracing of Algorithm involves the following:

- Understand the Algorithm:** Before tracing is started, thoroughly understand the algorithm's logic flow.
- Choose Test Input:** To trace the algorithm it requires input; a complete set of input is therefore chosen at this point.

- c. **Initialization:** Initialize the variables and data structures. This helps in setting up the algorithm in its initial state, where the algorithm has never run before.
- d. **Trace Each Step:** hand execute the algorithm step-by-step, tracking the flow of control. For each step of algorithm, do the following:
  - Track record Inputs: write the input values and variable values relevant to that step.
  - Perform processing: Do computation / perform the operation specified in that step of algorithm.
  - Update Variables: keep an eye on the changes in the variable values and data structures in response to performing operation.
  - Go by Control Flow: Normally, the steps of algorithms are followed sequentially but sometime this sequence is broken by loops, conditional statements, and function calls. Therefore, we need to track how an algorithm decides which step to execute next.
- e. **Repeat Until Completion:** Tracing the algorithm continues step by step until we reach the end of algorithm.
- f. **Verify Output:** At the end, evaluate the final output and ensure that it matches our expectations based on the given set of inputs.

#### Example:

Let's take example of Jeroo that we solved in previous topic and apply algorithm tracing.

- a. **Understand the Algorithm:** What we need to do, from where we need to pick flowers and to which location, we need to plant flower.
- b. **Choose test Input:** In the example flower pickup location e.g. (3,0) serves as input
- c. **Initialization:** Initial position of Jeroo and Initial position of flower
- d. **Trace each step:**
  - Track record input: Jeroo's location, flower's location are the input variables.
  - Processing: Taking moves, taking turns, picking flower, putting flower are the action that require processing.
  - Update variables: Jeroo's location and flower's location will be updated after each step movement of jeroo and flower pick / put action respectively.
  - Go by control flow: follow the algorithm e.g. sequentially.
- e. **Repeat until Completion:** take moves and turns until the jeroo is reached its destination and flower is placed at the designated location.
- f. **Verify output:** verify whether the completion done in above step is same as desired in the problem statement.



**Trace Tables:** The dry run of an algorithm can be done with a technique known as Trace Tables. These tables show the variables change at each stage in the algorithm. The trace tables generate the output on a given set of input data to test if algorithm is giving expected output. It also helps to check that our logic is correct and find logical errors, if any.

**Example:** Consider the following the pseudocode and accordingly its trace table.

**Pseudocode**

1. `number = 3`
2. `PRINT number`
3. `FOR i from 1 to 3:`
4. `number = number +5`
5. `PRINT number`
6. `PRINT "?"`

Trace Table			
Pseudocode steps	Number	i	Output
1	3		
2			3
3		1	
4	8		
5			8
3		2	
4	13		
5			13
3		3	
4	18		
5			18
6			?

**Example:**  
Consider the same example, we discussed in the chapter above.

**Pseudocode**

1. Start
2. Set `c=1`
3. Read `n`
4. `m=n*c`
5. print `m`
6. `c=c+1`
7. Repeat steps 4 to 6 while (`c<=10`)
8. End



### Teacher's Guide

"CS Unplugged: Computational Thinking Activities"

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

This resource provides a series of activities designed to familiarize students with computational thinking principles such as problem decomposition, algorithms, and testing. These activities serve as an excellent introduction to the fundamental aspects of creating computational artifacts, all without the requirement of computers.

Table 2.1- Trace table for the given pseudocode

Pseudocode Step	c	Input n	m	Output m	Algorithm Step	c	Input n	m	Output m
1	-	-	-	-	7	6	2	10	-
2	1	-	-	-	4	6	2	12	-
3	1	2			5	6	2	12	12
4	1	2	2		6	7	2	12	-
5	1	2	2	2	7	7	2	12	-
6	2	2	2	-	4	7	2	14	-
7	2	2	2	-	5	7	2	14	14
4	2	2	4	-	6	8	2	14	-
5	2	2	4	4	7	8	2	14	-
6	3	2	4	-	4	8	2	16	
7	3	2	4	-	5	8	2	16	16
4	3	2	6	-	6	9	2	16	-
5	3	2	6	6	7	9	2	16	-
6	4	2	6	-	4	9	2	18	-
7	4	2	6	-	5	9	2	18	18
4	4	2	8	-	6	10	2	18	-
5	4	2	8	8	7	10	2	18	-
6	5	2	8	-	4	10	2	20	-
7	5	2	8	-	5	10	2	20	20
4	5	2	10		6	11	2	20	-
5	5	2	10	10	7	11	2	20	-
6	6	2	10	-	8	11	2	20	

(a)

(b)

### b) Algorithms Evaluation Parameters

Algorithms are evaluated using a variety of criteria and methods to assess their performance, efficiency, and correctness for specific tasks. The evaluation process determines whether an algorithm meets its intended objectives. The following are some common ways to evaluate algorithms, however at advanced level there are many other ways also.

#### Correctness:

Correctness is considered the most fundamental evaluation parameter. An algorithm must produce the desired output for all possible valid inputs. Therefore, various inputs are tried during evaluation of algorithm.

#### Efficiency:

Efficiency of an algorithm deals with time and space (amount of memory) complexity. The time complexity refers to the amount of time taken by an algorithm to run and space complexity refers to the amount of memory required to run that algorithm.

#### Clarity:

This refers to how easily the algorithm's logic and steps can be understood by humans involved in



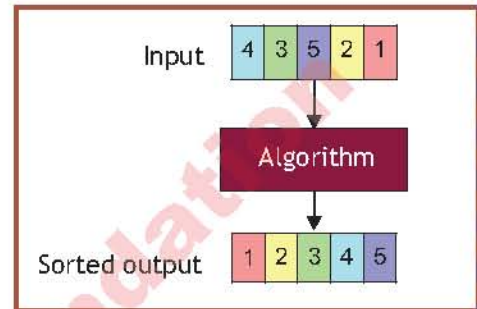
the software development lifecycle.

### Reliability:

Reliability refers to the ability to always produce correct and accurate results when the algorithm is given a set of inputs. A reliable algorithm should always perform the required task with a high degree of accuracy and consistency.

## 2.2 Common Computing Algorithms

Different problems require different types of algorithmic techniques to solve the problems. Similarly, different programmers prefer to use different algorithms to solve the same problem. There are many algorithms that have been pre-developed because they are considered fundamental and commonly used in solving daily life problems. Few of them are discussed in the below sections.



### 2.2.1 Sorting Algorithms:

The sorting algorithms are used to arrange data in a useful manner (Such as ascending or descending order). Various methods of sorting are used for example Insertion sort algorithms, Bubble sort algorithms, etc.

#### a) Insertion sort algorithm

In Insertion sort, we compare adjacent elements and sort them if they are not in correct order. While comparison, the smallest element is selected and swapped with the element that is placed at first location. This is the first iteration, and we will continue until all elements are sorted. At the end, we have an array of sorted elements.

##### i. Working of insertion sort algorithm

Consider the following array of unsorted elements which need to be sorted in ascending order.

For ease let's use four colors to represent four different things in the given array

- Yellow color for unsorted array.
- Blue color for elements to be compared in the unsorted array.
- Green color for sorted array.
- Red color for elements that are not correct in the sorted part of array.

At the start, all elements are colored yellow

12	31	25	8	32	17
----	----	----	---	----	----

Assuming first element (12) is already sorted.

12	31	25	8	32	17
----	----	----	---	----	----

We pick the second element (31) and compare with the adjacent element with sorted array.

12	31	25	8	32	17
----	----	----	---	----	----

Because 31 is larger than 12. It means that first element is already in correct order.

12	31	25	8	32	17
----	----	----	---	----	----

Now, move to the next element (25) and compare it with adjacent element in the sorted array (25 with 31).

12	31	25	8	32	17
----	----	----	---	----	----

Here, 25 is smaller as compared to 31. It means, 31 is not at its correct position, therefore, swap both the elements 25 with 31.

12	25	31	8	32	17
----	----	----	---	----	----

Along with swapping, we also need to check and compare it with all elements in the sorted array. Till now, the sorted array contains one element (12). So, 25 will be compared with 12. Here, 25 is greater than 12. So, the sorted array remains same.

We now move forward to next element that is 8 and compare it with its adjacent element in the sorted array (31).

12	25	31	8	32	17
----	----	----	---	----	----

Here, 31 is greater than 8, so swap them.

12	25	31	8	32	17
----	----	----	---	----	----

After the above swapping, you will note that elements 25 and 8 are not in its correct order.

12	25	8	31	32	17
----	----	---	----	----	----

So, accordingly swap them.

12	8	25	31	32	17
----	---	----	----	----	----

The elements 12 and 8 are also not in its correct order.

12	8	25	31	32	17
----	---	----	----	----	----

So, accordingly swap them.

8	12	25	31	32	17
---	----	----	----	----	----

Now we move to the next elements that is 32 and compare it with adjacent element in sorted array that is 31.

similarly 32 will be compared with previous element in the sorted array there will be no change because they are already sorted.

8	12	25	31	32	17
---	----	----	----	----	----

No swapping is required because they are already sorted.

8	12	25	31	32	17
---	----	----	----	----	----



Here we will take the next element that is 17 and compare it with its adjacent element in sorted array that is 32.

8	12	25	31	32	17
---	----	----	----	----	----

32 is greater than 17. So, swap them

8	12	25	31	32	17
---	----	----	----	----	----

This swapping will make 31 and 17 unsorted.

8	12	25	31	17	32
---	----	----	----	----	----

So, swap them too.

8	12	25	17	31	32
---	----	----	----	----	----

Now, this will make 25 and 17 unsorted.

8	12	25	17	31	32
---	----	----	----	----	----

So, swapping them again.

8	12	25	17	31	32
---	----	----	----	----	----

Now, the sorting process is complete as the array is in correct order.

## ii) Insertion Sort in context of Computational Thinking

Let's solve the Insertion sort by applying the computational thinking properties.

### Abstraction:

Organize a collection of items by repeatedly picking up each item and inserting it into its correct position relative to the items already sorted. This will gradually build a sorted collection by inserting items in the right order.

### Decomposition:

#### Sub-Processes:

Pick and Insert: For each item, pick it up and insert it into the sorted collection.

Correct Position Logic: Determine the correct position for each item in the sorted collection.

### Pattern Recognition:

Repetition: The process involves repeating the steps for each item.

Building Order: The algorithm focuses on building a sorted collection incrementally.

Ordered Comparison: It employs comparisons to determine the correct position of each item.

### Algorithmic Design

1. Start with the second element:

Begin with the second element of the array (assuming the first element is already "sorted").

2. Pick the current element:

Pick the current element and remember it.

3. Compare with sorted elements:

Compare the current element with the elements in the "sorted" part of the array.

4. Shift larger elements to the right:

If the current element is smaller than the elements in the "sorted" part, shift those larger elements to the right to create space.

5. Insert the current element:

Insert the current element into its correct position in the "sorted" part.

6. Repeat for each element:

Repeat these steps for each element in the array, gradually expanding the "sorted" part.

## b) Bubble sort algorithm

In Bubble Sort algorithm, we repeatedly compare and swap the adjacent elements if they are not in correct order.

If we need to sort the array in ascending order, then bubble sort algorithm starts by taking first element of the array and compare it with the second element first iteration. Suppose if the first element is greater than the second then it will swap these elements, and then move forward to compare the second element with the third element, and it continues until the largest element reaches the last place in the array.

We need to repeat this process (iteration) until the complete list is sorted.

### 1. Working of Bubble Sort Algorithm

Consider the following array of elements:

13	32	26	35	10
----	----	----	----	----

#### First Iteration

Take the first element and compare it with its adjacent element

13	32	26	35	10
----	----	----	----	----

Here, the first element (13) is small than the second element (32), so it is already sorted.

Now, the second element and compare it with next adjacent element (e.g. 32 with 26).

13	32	26	35	10
----	----	----	----	----

Here, 32 is greater than 26. So, we need to swap them both. After the swapping, array will look like this:

13	26	32	35	10
----	----	----	----	----



Now, take third element and compare it with its next adjacent (e.g. 32 with 35). No swapping is required because 35 is greater than 32

13	26	32	35	10
----	----	----	----	----

Now, we will compare 35 with 10.

13	26	32	35	10
----	----	----	----	----

Here, 35 is greater than 10 and need swapping.

The element 35 has reached at the last place of the array. At the end of first iteration, the array will look like this:

13	26	32	10	35
----	----	----	----	----

Now, its time for second iteration.

### Second Iteration

In second iteration, we will follow the same process.

13	26	32	10	35
13	26	32	10	35
13	26	32	10	35

Here, 32 is greater than 10. So, perform swapping operation.

13	26	10	32	35
13	26	10	32	35

Similarly, next iteration will be done.

### Third Iteration

In third iteration, we will follow the same process again.

13	26	10	32	35
13	26	10	32	35

Here, 26 is greater than 10. So, perform the swapping again.

13	10	26	32	35
13	10	26	32	35
13	10	26	32	35

Now, move to next iteration.

### Fourth Iteration

Again follow the same process the array is completely sorted now.

10	13	26	32	35
----	----	----	----	----

## ii. Bubble Sort in context of Computational Thinking

Let's solve the Bubble sort by applying the computational thinking properties.

### Abstraction:

Organize a collection of items by repeatedly comparing and swapping adjacent items until the entire collection is sorted. It will perform iterative comparison and swapping to achieve order.

### Decomposition:

#### Sub-Processes:

1. Adjacent Comparison: Compare each pair of adjacent items.
2. Swap Logic: If a pair is out of order, swap them.
3. Iteration: Repeat the process until the collection is sorted.

### Pattern Recognition:

Pairwise Comparison: The algorithm focuses on comparing and adjusting adjacent pairs.

Iterative Process: The sorting process is achieved through multiple iterations.

Repeating Patterns: The comparison and swapping steps are repeated until the entire collection is in order.

### Algorithmic Design

1. Start from the first element of the array.
2. Compare the first element with the adjacent element.
  - If the first element is greater than the adjacent element, swap them.
  - If the first element is smaller or equal, do nothing and move to the next pair of elements.
3. Move to the next pair of elements and repeat the comparison and swapping process.
  - Continue this process until you reach the end of the array for the first iteration.
4. After the first iteration, the largest element is guaranteed to be at the end of the array.
  - You don't need to consider this element in the subsequent passes since it is already in its correct position.
5. Repeat the above steps for the remaining elements (excluding the ones that are already sorted).
  - In each iteration, the next largest unsorted element will be placed in its correct position.
6. Continue this process until the entire array is sorted.
  - The number of iterations required is one less than the total number of elements in the array.

Both Insertion Sort and Bubble Sort are basic sorting algorithms. The choice of sorting algorithm depends on the specific requirements of your problem. Following are some generic guidelines



that could be helpful while selecting any sorting algorithm. However, there are some advanced sorting algorithms available that you would probably study in your higher class.

Table 2.2- Insertion Vs. Bubble Sort

	Insertion Sort	Bubble Sort
When to use	Use for a small number of items	Use for a small number of items
	nearly sorted data	nearly sorted data
	Practical applications	Educational purposes Educational purposes

### Example

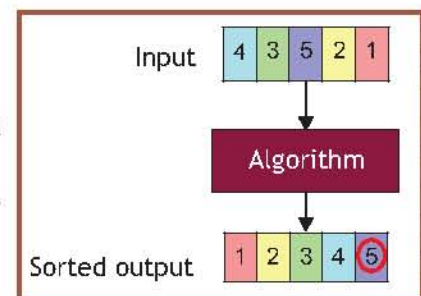
**Sorting a Shelf of Books:** Imagine you have a shelf of books that you want to arrange in alphabetical order. Compare different methods:

- **Insertion Sort:** Pick up each book one by one. Insert each book into its correct position on the shelf relative to the books you're already holding.
- **Bubble Sort:** Compare each pair of adjacent books from left to right. If a pair is out of order, swap their positions. Continue this process, moving from left to right, until the entire shelf is sorted.

**Algorithm Evaluation -> Efficiency:** The time it takes and the number of movements represent the efficiency of the sorting algorithm.

### 2.2.2 Searching Algorithms:

The searching algorithms are used to search desired element that is available in some group of elements. Like sorting, various searching methods are available for example Binary Search algorithms and Linear Search algorithms.



#### a) Binary Search algorithm

This search method works on sorted lists. It follows the divide and conquer approach in which the sorted list is divided into two parts. The item to be searched is then compared with the middle element of the list. If it matches then, the location of the middle element is returned. Otherwise, we search into one of the divided parts depending upon the result produced in the last comparison.

#### i. Working of Binary Search

Consider the following sorted array of nine elements and we need to find a number 56.

0	1	2	3	4	5	6	7	8
10	12	24	29	39	40	51	56	69

Divide the array in to two parts, the middle element is 39.

0	1	2	3	4	5	6	7	8
10	12	24	29	39	40	51	56	69



The item to be searched (56) is compared with the middle value (e.g. 39). we came to know that 56 is greater than 39, therefore the item is surely in the right side of the array (because the array is sorted).

Here we get right part of the array.

5	6	7	8
40	51	56	69

It will again be divided into two parts.

5	6	7	8
40	51	56	69



Considering 51 the middle element, we will compare the item (56) with this, because 56 is greater than 51 therefore, we will again take right side of the array.

7	8
56	69

Divide the array again into two parts.

7	8
56	69



Considering 56 as the middle element, we will compare it with the item to be searched (e.g. 56). Now, the required item is found, So we will provide output of the algorithm that is location 7. If the required item is not found the algorithm returns NULL.

## ii) Binary Search in context of Computational Thinking

Let's solve the Binary Search by applying the computational thinking properties.

### Abstraction:

Search for a specific item in a sorted collection by repeatedly dividing the search space in half until the target item is found or the search space is empty. It efficiently narrow down the search space by half in each iteration.

### Decomposition:

#### Sub-Processes:

1. Middle Item Check: Examine the middle item to determine if it matches the target.
2. Search Space Division: Divide the search space based on the comparison result.
3. Recursive Process: Repeat the process on the narrowed down search space.



### Pattern Recognition:

Halving the Search Space: The core concept is repeatedly reducing the search space by half.

### Algorithmic Design

// Assume target value is the element that needs to be searched

1. Start with the entire sorted array.
2. Check the middle element of the array.
  - If it's equal to the target value, you're done.
  - If it's greater than the target value, ignore the right half.
  - If it's less than the target value, ignore the left half.
3. Repeat the process on the remaining half.
4. Continue until you find the target value, or the array becomes empty.

### b) Linear Search algorithm

This is a simple search algorithm; we go through the complete list and match the elements one-by-one until the required item is found. If the item is found, then the location of that element is returned, otherwise a NULL is returned. This search is also called as sequential search algorithm. However, there is no compulsion that the array should be sorted.

#### 1) Working of Linear Search

Consider the given unsorted array and we need to find location of item 41.

0	1	2	3	4	5	6	7	8
70	40	30	11	57	41	25	14	56

We will start from first element.

0	1	2	3	4	5	6	7	8
70	40	30	11	57	41	25	14	56



The required item (41) is compared with the first element. Both don't match, we will go to next element.

0	1	2	3	4	5	6	7	8
70	40	30	11	57	41	25	14	56



We will again compare the second element (e.g. 40) with required item (e.g. 41), both don't match. So will continue the process.

0	1	2	3	4	5	6	7	8
70	40	30	11	57	41	25	14	56



0	1	2	3	4	5	6	7	8
70	40	30	11	57	41	25	14	56

↑

0	1	2	3	4	5	6	7	8
70	40	30	11	57	41	25	14	56

↑

0	1	2	3	4	5	6	7	8
70	40	30	11	57	41	25	14	56

↑

We found the element at location 5 so output of this algorithm would be 5. In these type of algorithms, if the required item is found we provide the output or if item is not found and we reach at the end of array then a NULL is given in output.

Let's solve the Linear search by applying the computational thinking properties.

### Abstraction:

Search for a specific item in a collection by checking each item in order until the target item is found or the end of the collection is reached. It sequentially inspects each item until a match is found.

### Decomposition:

#### Sub-Processes:

1. Starting Point: Begin searching from the first item.
2. Item Checking: Inspect each item to see if it matches the target.
3. Stopping Condition: Stop when the target is found or the end of the collection is reached.

### Pattern Recognition:

- Sequential Nature: The process involves moving through the collection one item at a time.
- Linear Progression: The search progresses linearly through the collection.
- Termination Condition: The search stops when the target is found or the entire collection is checked.

### Algorithmic Design

// Assume target value is the element that needs to be searched

1. Start from the beginning of the array.
2. Check if the current element is equal to the target value.
  - If it is, you've found the target value.
  - If it's not, move to the next element in the sequence.
3. Repeat step 2 until the target value is found or you reach the end of the array.
4. If you reach the end of the array without finding the target value, the target value is not in the array.



The choice of searching algorithm depends on the specific requirements of your problem. However, following are some generic guidelines that could be helpful while selecting any searching algorithm:

Table 2.3- Linear Vs Binary Search		
When to use	Linear Search	Binary Search
	When the list is not sorted	When the list is sorted
	Suitable for a smaller number of elements	Ideal for larger number of elements

### Example

**Movie Streaming:** Choosing a movie to watch from a list:

- Linear Search: Start from the top and scroll down until you find the movie.
- Binary Search: Divide the list in half, quickly narrowing down the options.

**Algorithm Evaluation -> Efficiency:** The time it takes to find a movie reflects the efficiency of the search algorithm.



### Teacher's Guide

"CodeCombat: Learn to Code by Playing a Game"

<https://codecombat.com/>

This resource transforms the learning experience into a game, where students can grasp programming concepts through an entertaining role-playing adventure. The game effectively introduces fundamental coding logic, which can be applied to comprehend search and sort algorithms.

### Summary

- **Abstraction:** Identifying essential information and removal of the unnecessary details to solution.
- **Algorithm** is a step-by-step set of instructions that define how a specific problem should be solved.
- **Algorithm Design:** This is actual designing of solution. This involves creating step-by-step plan of the problem solution.
- **Algorithm Evaluation** assess the algorithm performance, efficiency, and correctness for specific tasks.
- **Algorithm Tracing** is used to hand simulate the execution of your algorithm, keep change track of variable values and flow of logic control.
- **Binary Search:** Sorted list is divided into two parts. The item to be searched is then compared with the middle element of the list. If it matches then, the location of the middle element is returned. Otherwise, we search into one of the divided parts depending upon the result produced in the last comparison.
- **Bubble Sort:** we repeatedly compare and swap the adjacent elements if they are not in correct order. Bubble sort algorithm starts by taking first element of the array and compare it with the second element. Suppose if the first element is greater than the second then it will swap these elements, and then move forward to compare the second element with the third element, and it continues until the largest element reaches the last place in the array.
- **Computational Artifacts** refer to human made objects and systems using computational thinking.
- **Computational Thinking** is a thought process that helps us to understand the problem and solve it in a way that computers do.
- **Decomposition:** Breaking down the larger problems into smaller/ manageable ones and working on them one by one. These smaller problems are referred to as sub-problems. This way we simplify the problem and solve it easily.
- **Development** describes the steps needed to convert or manipulate the inputs to produce the outputs.
- **Flowchart** provides a visual representation of flow and logic using defined symbols.
- **Insertion sort,** we compare adjacent elements and sort them if they are not in correct order. While comparison, the smallest element is selected and swapped with the element that is placed at first location.
- **Linear Search:** we go through the complete list and match the elements one-by-one until the required item is found. If the item is found, then the location of that element is



returned, otherwise a NULL is returned. This search is also called as sequential search algorithm. However, there is no compulsion that the array should be sorted.

- **Pattern Recognition:** Examine the problem for a pattern or similarities between previously solved problems.
- **Pseudocode** is an intermediate step between the algorithm and actual programming code.
- **Searching algorithms** are used to search desired element that is available in some group of elements.
- **Sorting Algorithms** are used to arrange data in a useful manner (Such as ascending or descending order).
- **Trace Tables:** The dry run of an algorithm can be done with a technique known as Trace Tables.

National Book Foundation

## Exercise



Select the best answer for the following Multiple-Choice Questions (MCQs).

1. A complexity of algorithm depends upon \_\_\_\_\_
  - a) Time Only
  - b) Space Only
  - c) Both Time and Space
  - d) None of above
2. An algorithm: can be represented through \_\_\_\_\_
  - a) Flow charts
  - b) Pseudo-codes
  - c) Instructions in common language
  - d) All of the mentioned above
3. There are two algorithms suppose A takes 1.41 milliseconds while B takes 0.9 milliseconds, which one of them is better considering all other things the same?
  - a) A is better than B
  - b) B is better than A
  - c) Both are equally good
  - d) None of the mentioned
4. Considering an array has 10 elements and the searching element is at array index 6. A starting element is present at index zero. How many comparisons are required to search and element using linear search?
  - a) 5
  - b) 6
  - c) 7
  - d) 8
5. In computer science, the computational artifacts could include \_\_\_\_\_
  - a) Programs
  - b) Simulations
  - c) Videos
  - d) Programs, Simulations and videos
6. In the planning phase of computational artifact development following is included in
  - a) Logic
  - b) Data Structures
  - c) Control Structures
  - d) Logic, Data and Control Structures
7. trace method is used to \_\_\_\_\_
  - a) Take input
  - b) Hand simulate the execution algorithm.
  - c) Take a print
  - d) Align margins
8. Trace Tables are used to \_\_\_\_\_
  - a) dry run algorithm
  - b) test if algorithm is giving expected output
  - c) show the variables change
  - d) a, b and c



9. Algorithms are evaluated using \_\_\_\_\_
- a) Correctness
  - b) Efficiency
  - c) a and b
  - d) None of above



**Give short answers to the following Short Response Questions (SRQs).**

1. Differentiate between
  - i Clarity vs. Efficiency
  - ii Abstraction vs. Pattern Recognition
  - iii Pseudocode vs. Flowcharts
  - iv Data Structures vs. Control Structures
  - v Algorithm vs. Pseudocode
2. Write a note on working of Bubble Sort.
3. Write names of commonly used computational artifacts made during computational thinking.
4. Where we prefer to use binary search algorithm rather than linear search algorithm?
5. What are the advantages of using flowcharts?



**Give long answers to the following Extended Response Questions (ERQs).**

1. Determine the properties involved in computational thinking.
2. Sketch an algorithm that inputs length in inches and prints it in centimetres.
3. Implement an algorithm to print multiplication table of a number in reverse order.
4. Examine the uses of Flowcharts.
5. A newly developed Algorithm needs to be tested. Argue about the reasons.



**Activity 1: Complete the given trace table using the following algorithm snippet.**

1. Start
2.  $x = 2$
3.  $total = 0$
4. REPEAT Step 5 Three TIMES
5.  $total = total + x$
6. Print (total)
- End

Line No.	x	total	output



### Activity 2: Insertion sort algorithm

Sort the following array using insertion sort

10	4	5	8	6	9	2
----	---	---	---	---	---	---



### Activity 3: Bubble sort algorithm

Sort the following array using bubble sort

3	1	7	8	2	5	6	4	0
---	---	---	---	---	---	---	---	---



### Activity 4: Binary Search algorithm

Consider the following sorted array of seven elements and we need to find a number 9.

2	4	5	6	8	9	10
---	---	---	---	---	---	----



### Activity 5: Linear Search algorithm

Consider the following array of eight elements and we need to find a number 9.

58	25	39	78	12	9	79	80
----	----	----	----	----	---	----	----



# Unit 03

## PROGRAMMING FUNDAMENTALS



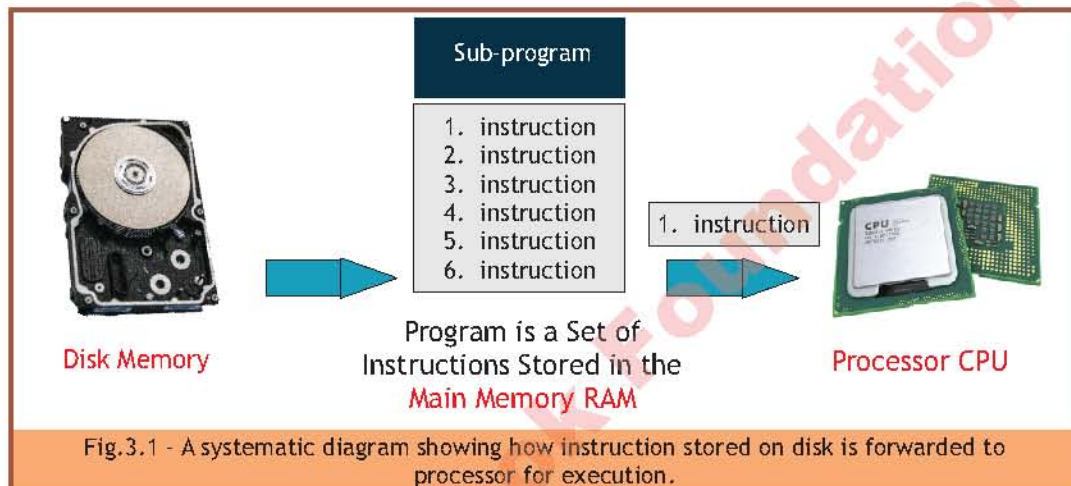
After completing this lesson, you will be able to:

- understand the importance of computer programming and applications
- write and execute simple programs in Python.
- draw shapes using Turtle Graphics functions in Python
- understand the need for libraries and learn the use of some simple libraries in Python.
- translate simple algorithms that use sequence and repetition in Python.
- decompose a problem into sub-problems and implement those sub-problems using functions in Python
- determine ways of debugging their code in Python



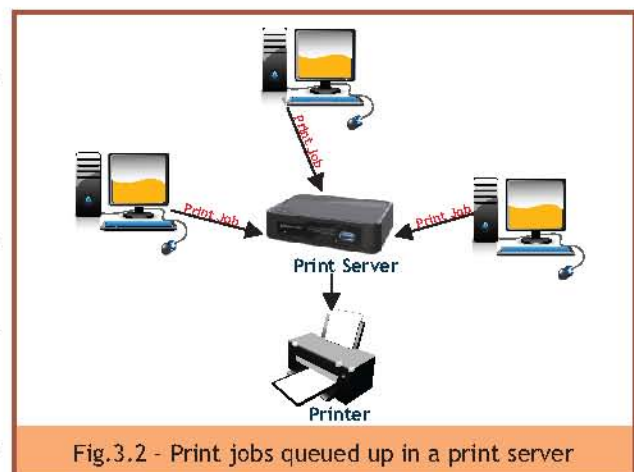
### 3.1 Computer Program

A set of well-defined instructions which a computer can execute to solve a problem is called a computer program. The program interacts with the user for some input, then executes one instruction at a time from set of instructions (called code) and delivers the result. In its simplest form, the program requires one algorithm to build but large programs can contain multiple algorithms for various tasks in a single program. When the program is executed, the program is loaded into main memory from where one instruction at a time is fetched and provided to the processor which executes it, as shown in Fig 3.1. Keep in mind that the program itself and the data it operates upon both are kept in the main memory for faster execution



Generally, computer programs are categorized into interactive and batch programs. The former, accepts some input which is either from a user or some other program, like a web browser. Whereas the latter can be executed by user or an interactive program, where multiple commands are bunched together which are processed sequentially and then halts. For example print jobs on a printer lines up in a queue, processed one after the other and halts when no more jobs are in queue, as shown in Fig 3.2.

Computer Programs are developed in programming languages which provide the development environment via set of rules and keywords to use. With the help of keywords, you can create set of instructions (called code) which need to follow the rules of syntax of the said programming language. Thereafter, when you compile your code the compiler checks whether the syntax is as per the rules and converts your code into an object file, which comprises of binary data and is called machine





language, as shown in Fig 3.3. The machine language is understandable to the processor and executes your code. Examples of such programming languages which use compiler are C++, Java, Assembly Language, Fortran etc. Apart from compiler, languages like Python, JavaScript and Ruby use interpreter which are more flexible in a sense that the code written in high level language is converted into machine language line-by-line, as the code executes.

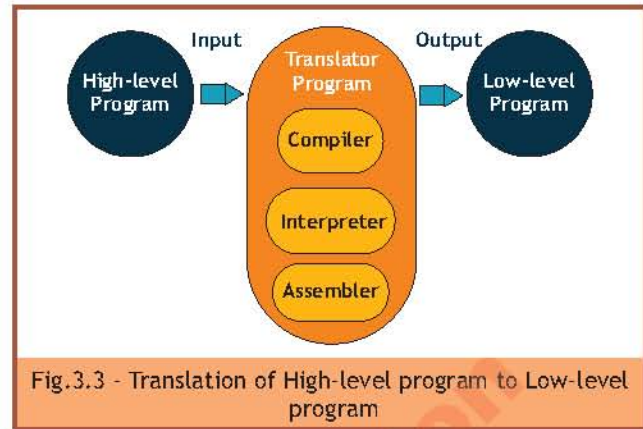


Fig.3.3 - Translation of High-level program to Low-level program

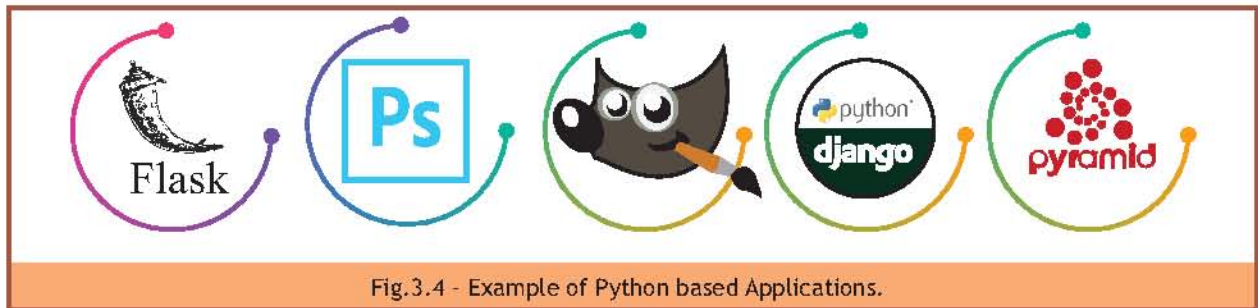
## 3.2 Python: An Overview

One of the most popular and widely used programming languages today is Python. Python is used for data analytics, machine learning, and software development. Python is a high-level, object-oriented (data-driven), and human-friendly (easier to grasp) computer language. It was first introduced in 1992 and is designed to be reasonably simple to write and comprehend. As a result, it is the perfect coding language for individuals looking for rapid development. Python's widespread use is due to a variety of factors, including:

- Python may be a great starting point for individuals who are unfamiliar with the art of programming, due to its simplicity.
- Python's syntax is quite similar to that of English, making it very simple to read and comprehend. You can determine what each line of code does just by looking at it.
- Python is an open-source programming language, meaning anybody can use it and help is available on different sites and platforms, as developers around the world share their experiences.

Python allows you to program a wide range of applications. Data scientists frequently use Python, which has a large number of machine learning and artificial intelligence tools and packages. Python programming is used in the quickly evolving field of data analytics. People who can gather, manage, and arrange information are needed at a time when we are producing higher quantities of data like never before. Python is excellent for web development, mainly because there are several Python website development frameworks available, like Django, Pyramid, and Flask. Python may be used to create apps for graphic design as well, as shown in Fig 3.4.

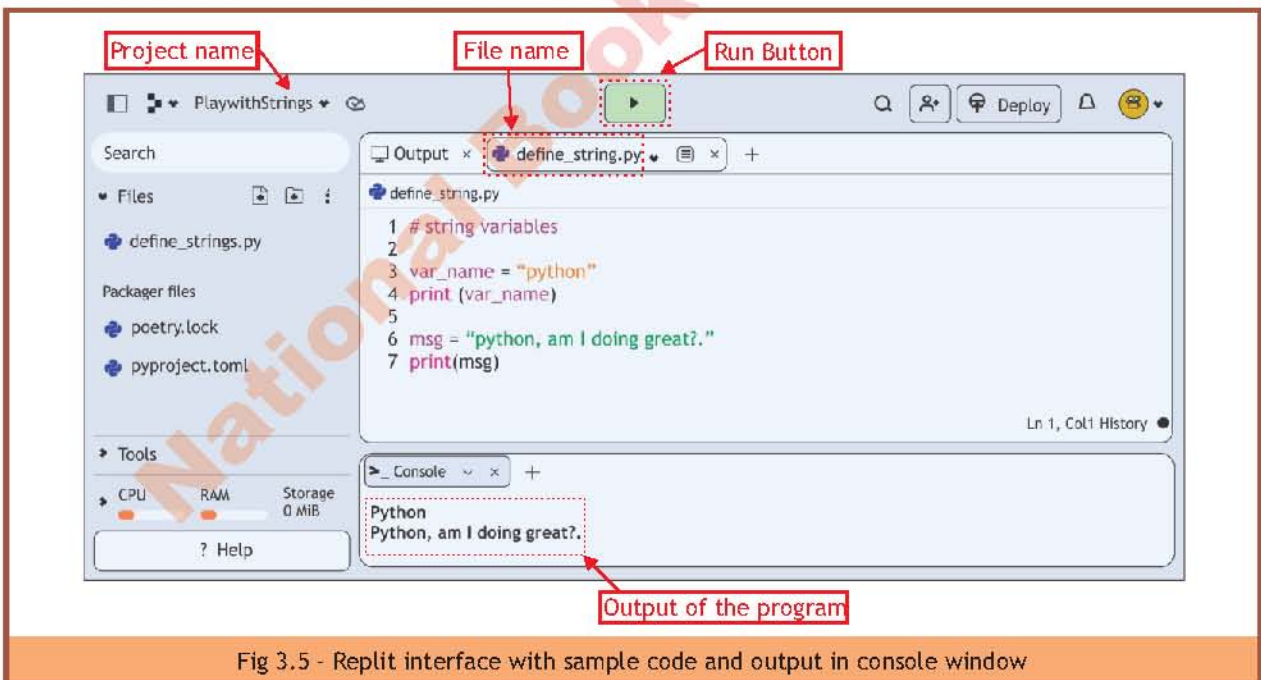
Python is also employed extensively in the creation of online educational materials and programs. For novices, learning this programming language is simple since it has an English-like syntax. It makes the learning process easier by providing a beginner with a standard library and a number of resources to comprehend the language. This is probably the main reason that Python is considered by novices as first choice to enter the programming paradigm. Last but not the least, Python programming language may be used to make small games, making it a helpful tool for quickly creating a prototype.



### 3.3 Python Integrated Development Environments

Python Integrated Development Environment (IDE) can be downloaded as IDLE from <https://www.python.org/downloads>. It has a simple command line interface with an interactive console. This means you can give commands on the console prompt as well as write your whole code in the form of a block, save in a file and run it.

Besides this, there are online IDEs available through which you can write and run your code, without installing it on your computer. One of them is Replit, which also provides an interactive console and easy to use environment, which are feasible for small blocks of code, especially for beginners. You just create your login or you may use your Gmail account to log in. Create a REPL (Read-Eval-Print-Loop) by pressing the 'Create Repl' button. Create a new project and assign a name to it, else Replit will provide one by default.



You can write code in the **code area** and then press the green button which runs the program. The corresponding output is displayed in **console area**.

A sample code of two variables are assigned string data type values and printed in Replit out, as



shown in Fig 3.5. The filename is 'define\_strings.py' and respective project is titled 'PlaywithStrings'. Assigning such meaningful names to files, folders and projects is a good practice. This way, even without opening the file, you will recall what the code in the file is all about.

### 3.4 TURTLE GRAPHICS

The turtle graphics is installed with Python and we have just to invoke it and use it. No need to install anything to use it. It provides functions which can be easily used even from the shell prompt. Through these functions we can design basic drawings and two dimensional geometrical shapes. It is easy to use and provide interactive session and motivation for the novice. Nevertheless, turtle provides advanced features which are quite useful for the development of animations, simulations and computer games.

On the Shell, first statement needs to be “Import turtle”. This command will invoke the turtle library and all the functions of the said library can now be used. The very next line as shown in figure is “turtle.Screen()”, which loads the canvas of the turtle graphics. As soon as, a white canvas loads, you will be able to see a '>' which is the **pen** that will draw our lines and shapes. This pen is called **turtle** and correspondingly the name turtle graphics arose.

As line 3 highlights that we can resize the canvas and it's color. Keep in mind that all measurements that are used in turtle graphics are in terms of **pixel**. To emphasize this, we print a dot of blue color of size 10 pixels in line 4. Another important point to remember that every instruction is executed from the point where the pen is currently located. Line 5, reflects that by default the pen color is “Black”, as the **circle()** function is used and the circumference of the circle is drawn in black. Additionally, the **circle()** function takes radius as input and draws the resultant circle. To distinguish, we next change the pen color to “Blue” and draw another circle, whose resulting circumference is of blue color, as shown in Fig 3.6.

```
1. Import turtle
2. turtle.Screen()
3. turtle.screensize(200,150,"coral")
4. turtle.dot(10,"Blue")
5. turtle.circle(50)
6. turtle.pencolor("Blue")
7. turtle.circle(-50)
```

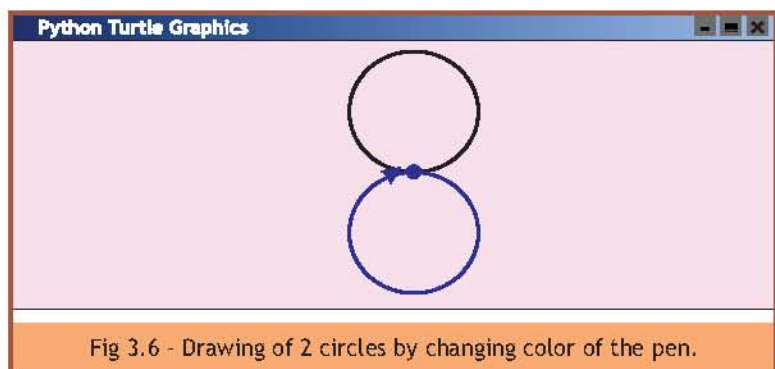


Fig 3.6 - Drawing of 2 circles by changing color of the pen.

To draw a triangle, you need to play with angles. The drawing of first line with **forward()** function is easy, as earlier. Thereafter, depending upon what type of triangle you want to draw, you will set the angle and length of line accordingly. This way, you will be thankful to your school teachers who taught you all the trigonometric formulae and the resultant shape of the triangle is visible on screen, as shown in Fig 3.7.

1. `turtle.reset()`
2. `turtle.forward(140)`
3. `turtle.left(135)`
4. `turtle.forward(100)`
5. `turtle.left(90)`
6. `turtle.forward(100)`
7. `turtle.left(45)`
8. `turtle.left(90)`

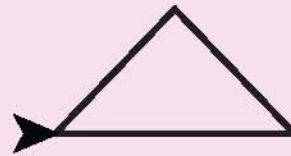


Fig 3.7 - Example of another linear shape drawing

To clear up the canvas, “`turtle.reset()`” is used and now you may have your next drawing. This time, let's use the most basic line drawing functions of `forward()` and `back()`. These functions take distance in pixels that how far you want to draw a line and in which direction. By now, it should be obvious enough that `forward()` will draw in the direction of the pen and `back()` will draw in the reverse direction of the pen. So, as the code shows, the `forward()` function with 100 pixels is called which draws a line and thereafter, `left()` function is called which changes the direction of pen. The functions `left()` and `right()` take input in terms of angles. Though, the standard of angles is radian, but since degrees are the more frequent in use, therefore Python by default uses degrees, but also provides the provision for radians. Hence, moving forward and rotating at 90 degree angles, we have created a basic layout of a car. To provide the car with tyres, let's use the above example scenario and adjust the size of circle accordingly. After, the first tyre is drawn, let's use the `penup()` function and relocate your pen. Next, `pendown()` function is called and the pen is again ready to draw the second circle, as shown in Fig 3.8.

1. <code>turtle.forward(100)</code>	18. <code>turtle.left(90)</code>
2. <code>turtle.left(90)</code>	19. <code>turtle.forward(25)</code>
3. <code>turtle.forward(25)</code>	20. <code>turtle.pendown()</code>
4. <code>turtle.left(90)</code>	21. <code>turtle.circle(-10)</code>
5. <code>turtle.forward(25)</code>	22. <code>turtle.penup()</code>
6. <code>turtle.right(90)</code>	23. <code>turtle.right(90)</code>
7. <code>turtle.forward(25)</code>	24. <code>turtle.forward(10)</code>
8. <code>turtle.left(90)</code>	25. <code>turtle.dot(5, "Blue")</code>
9. <code>turtle.forward(50)</code>	26. <code>turtle.back(10)</code>
10. <code>turtle.left(90)</code>	27. <code>turtle.left(90)</code>
11. <code>turtle.forward(25)</code>	28. <code>turtle.forward(50)</code>
12. <code>turtle.right(90)</code>	29. <code>turtle.pendown()</code>
13. <code>turtle.forward(25)</code>	30. <code>turtle.circle(-10)</code>
14. <code>turtle.left(90)</code>	31. <code>turtle.penup()</code>
15. <code>turtle.forward(25)</code>	32. <code>turtle.right(90)</code>
16. <code>turtle.pencolor("Blue")</code>	33. <code>turtle.forward(10)</code>
17. <code>turtle.penup()</code>	34. <code>turtle.dot(5, "Blue")</code>

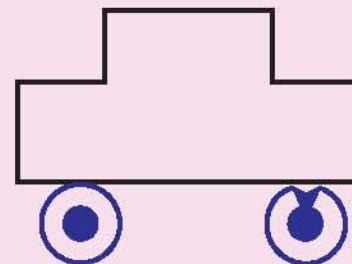


Fig 3.8 - Drawing a shape, like a car, by lifting the pen in between.

Turtle graphics is easy to learn and simple to use. Later in the chapter, when we will explore Selection and Iterations, using which you will be able to use better and complex shapes with comparatively lesser number of lines of code.



## 3.5 Libraries

A set of code which are meant for a particular task more proficient, is called **library**. We can produce result, with just one line of code via using a library, rather than writing our own tens of lines of code. For example, as discussed earlier, the Turtle library is already available in Python, which provides already written and tested functions that we can use. We simply have to import library and use from the available functions, with the right set of arguments. For any additional library, it can be downloaded and imported similarly.

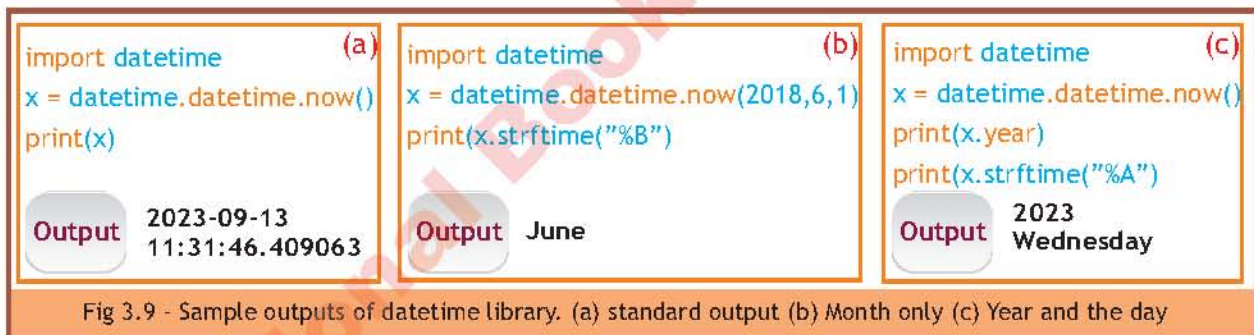
By default date is not the type of data supported by Python, but 'datetime' library can be downloaded and imported. This way, date and time in different formats can be accommodated.

Rather than manually installing a package and trace for missing dependencies by going through all the required files one by one and later install them as well, package managers are quite useful. Pip is a package manager for installing Python packages which is simple to use and assures that all dependencies of the said package are also installed.

For example, if you want to install "datetime" package to avail various options to display date in Python, simply type in command prompt:

```
>pip install datetime
```

So, let's consider the first example which displays the current date and time. The date returns year, month, day of the week along with the time till millisecond as shown in Fig 3.9 (a).



Second example in Fig 3.9 (b) shows how to extract year from the date, in line 3. Line 4 is of special consideration, which uses '`strftime()`' function and formats according to the argument. The `strftime()` function takes the previous example's output and treats it as a string. This way, based on the argument, the function is able to extract the particular output. In the Fig 3.9 (c) argument '`%A`' returns the day. Similarly, '`%B`' argument returns the name of the month, only. The most common arguments used for date-time format, are listed in Table 3.1.

Table -3.1- Sample list of arguments available for use in the library		
Argument	Description	Example
%A	Weekday, full version	Friday
%B	Month name, full version	August
%C	Century	19
%d	Day of month 01-31	21
%H	Hour 00-23	18
%I	Hour 00-12	03
%S	Second 00-59	48
%Y	Year, full version	2023

## 3.6 Python Variables

A variable stores some information which when allocated in the program, can be used later on. For example, rather than solving the same equation which is used multiple times in a program, we just store the resultant in a variable, and use it without recalculating the formula. A good programming practice is to keep meaningful variable name.

The name of a variable:

- can be alphanumeric i.e. combination of alphabets and numbers, but cannot start with a number.
- are Case Sensitive, i.e. 'Num' and 'num' are treated as two separate variables.
- cannot contain spaces, so a good technique is to use underscore ("\_"), like 'Temp\_var' which reflects a meaningful message that the value assigned to this variable is temporary.

Generally, instructions are written in the order they are supposed to execute, in all programming language. This way, it is easier to apply the logic which is meant to be implemented via those instructions. So, if for some reason, the sequence of instructions changes, it may result in undesirable results. And when you look again at your code the syntax of the code seems all right but the instructions are not behaving according to your expectations. Such a scenario is called **Logical Bug** or **Bug**. Bug is hard to detect just by looking at the code and this capability comes with experience. But the simplest method is to Dry-Run your code.

Let's write a code to swap the values of two variables, namely 'x' and 'y', with the help of a temporary variable 'temp', as shown in Fig 3.10. For some reason, if the sequence changes like as follows, then the corresponding output will change too. Hence, both x and y will have same values. In other words, if the sequence of instructions are not carefully structured and placed, the desired output is not likely to happen and data may be lost.

A variable needs to be defined as what type of data it can hold and thus programming language needs to define support for various data types.

Python also provides support for different data types by default. Numbers are stored in numeric data types, like int, float. Sequence of characters are handled by string data type. To manoeuvre a series of similar values, list can be used represented by "[\_,\_]". Tuple is identical to list, but are immutable, i.e. once defined the values in a tuple cannot be changed. A tuple holds the values in "(\_,\_)".



### Activity 1

In context of Fig 3.10, rewrite code to swap the values of 2 variables without using temporary variable.

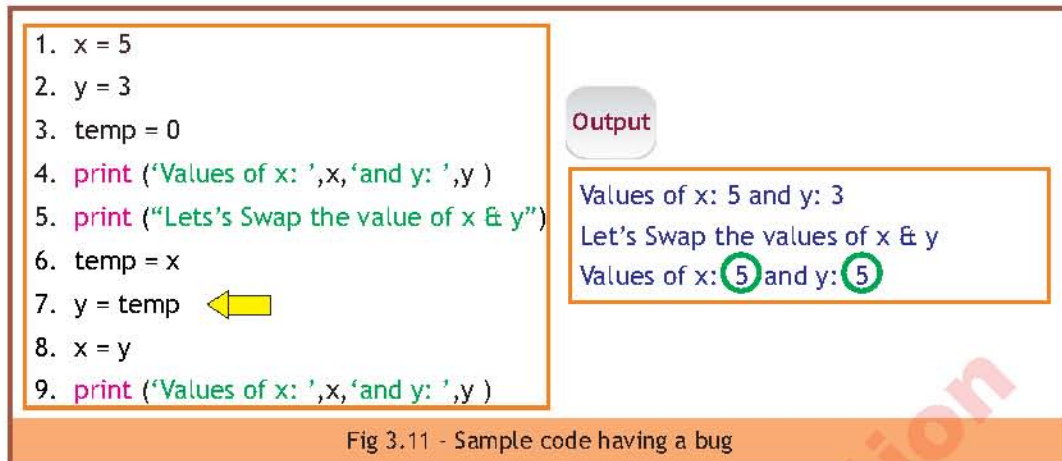
```
1. x = 5
2. y = 3
3. temp = 0
4. print ('Values of x: ',x,'and y: ',y)
5. print ("Let's Swap the value of x & y")
6. temp = x
7. x = y
8. y = temp
9. print ('Values of x: ',x,'and y: ',y)
```

Output

```
Values of x: 5 and y: 3
Let's Swap the values of x & y
Values of x: 3 and y: 5
```

Fig 3.10 - Code to swap the values of 2 variables





So, let's try to fix the code of Fig 3.11 by Dry-Run, as shown in Table 3.2. Dry-Run is where you execute your program on paper, and keep track of the variables. Initially, we break down our code. First 3 lines are variable initialization. Line 4 and 5 are print commands and so is the last line. Thus, we are left with lines 6-8 which need to be focused on and examined.

Table 3.2 -Table showing the values of variables during a Dry-Run

Line	Variables		
	x	y	temp
Before 6	5	3	0
6	5	3	5
7	5	5	5
8			

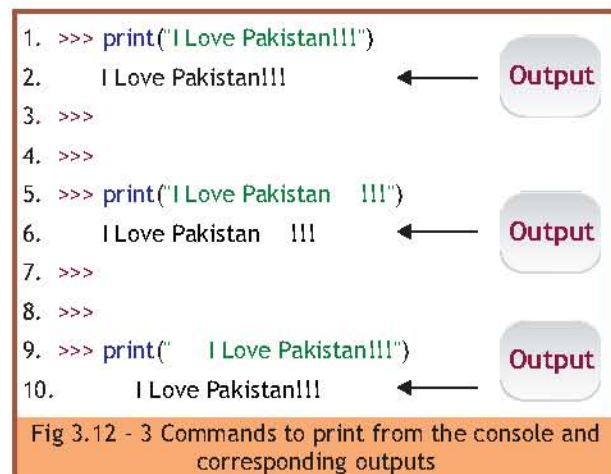
In view of that before executing line 6, the values of the three variables are the same as initialized. In line 6, value of temp gets updated to 5. As soon as, we check line 7, we are able to highlight that value of y updates to 5, which though is the desired one but all other variables have the same values, as well. Hence, value of '3' is lost and cannot be assigned to x.

As a result, we now know where the problem lies, and we can fix it.

### 3.7 Python Input/Output (I/O)

The Python Shell is interactive and accepts commands. So, result of arithmetic operations and desired output using the print function can be used. Parenthesis followed by print, i.e. print() function instructs Python to output whatever is inside the parenthesis, within quotes. Following are 3 outputs shown to emphasize that even if a space is entered before or after the sentence within the quotes, it will be displayed as it is; as shown in Fig 3.12.

Computer Programs are meant to be processed while taking some input and generating



corresponding output. User can enter data through `input()` function, via which data will be stored in a variable. This way, the same input does not need to enter again and again by the user and whenever the data is required, the corresponding variable is used. In the following example, a variable named 'country' takes the input in the form of character-string and the same variable is displayed using the `print()` function, in the next line. Steps are show in Fig 3.13.

```
1. country = input ( 'Enter the name of your country: ' )  
2. print ( 'I love ' , country)
```

#### Output

```
Enter the name of your country : Pakistan  
I love Pakistan
```

Fig 3.13 - Command to take input from the console and respective output

Python provides basic variable types as char, integer, float, boolean, etc. The `input()` function treats the entered data as a set of characters. Hence, if a numeral is entered, it needs to be converted to number, explicitly using the `eval()` function and can be used in conjunction with `input()`. Fig 3.14 highlights the importance of `eval()` function while dealing with numbers. In Python, `eval()` function evaluates expressions which are generally mathematical in nature. However, it also assesses strings/ set of characters and responds with an integral value, as highlighted in Fig 3.14 where with `eval()` the input is treated as integer. Note that the first output shows that when two variables are added, while the values are characters the add function joins the values rather than adding them up. The respective terminology is 'concatenation' of two strings or characters to merge them up.

```
1. num = input ("Enter a Number: ")  
2. print ( "Double of the said number is",num+num)  
3. num=eval (input("Enter a number:"))  
4. print ( "Double of the said number is",num+num)
```

#### Output

```
Enter a Number: 123  
Double of the said number is: 123123
```

```
Enter a Number: 123  
Double of the said number is: 246
```

Fig 3.14 - Code highlighting how Python treats input



### Activity 2

Take three inputs as day, month & year, e.g. your date of birth, store in three variables and print in the form of:

- 13-09-2023
- 09-13-2023
- 2023-09-13

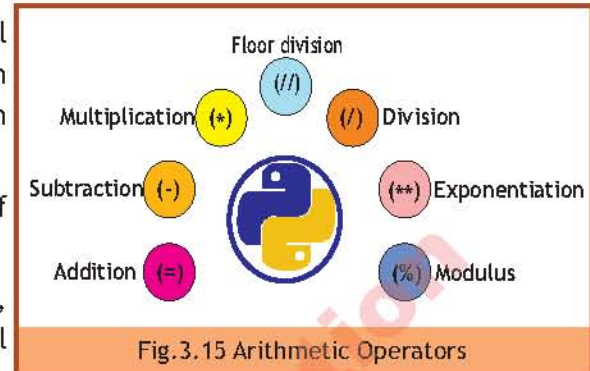


## 3.8 Operators in Python

### 3.8.1 Arithmetic Operators

Other than the assignment operator (=) and normal arithmetic operators i.e. addition (+), subtraction (-), multiplication(\*) and division (/); as shown in Fig 3.15, Python provides the option of:

- Modulo operator (%) which returns remainder of a division. e.g.  $5/4=1.25$  but result of  $5\%4=1$ .
- Integer division (//), e.g.  $5//4$  will return 1, only integral value and will discard the decimal portion.
- `***`, which is used for exponentiation.



Python considers operator precedence in the sequence of: \*,/,//,%,+,-. Operators having same precedence are evaluated from left to right. However, parentheses have the highest precedence and will be evaluated first.

The assignment operator can be used in Python with arithmetic operators, as well. e.g.  $i=i+1$  can also be written as  $i+=1$ ,  $i=i-2$  as  $i-=2$ ,  $i=i*3$  as  $i*=3$ , etc.

### 3.8.2 Bitwise Operators

Python provides operators for bitwise operations, which are used for comparison purposes of binary numbers. An overview of these is already discussed in chapter 1, under the heading of 'Basic Logic Gates'.

- AND (&) - Bit is fixed to 1 if both comparing bits are found as 1.
- OR (|) - Bit is fixed to 1 if at least one of the comparing bits is found as 1.
- XOR (^) - Bit is fixed to 1 if only one of the comparing bits is found as 1.
- NOT (~) - Flips all the bits, i.e. 0s are flipped to 1s and vice versa. It is a single operand operator.
- Left Shift (<<) - Shifts all the bits one place to the left, while the left most bit is discarded and right most bit is set to a 0.
- Right Shift (>>) - Shifts all the bits one place to the right, while the right most bit is discarded and left most bit is set to a 0.

### 3.8.3 Membership Operators

Membership operators in Python are used to check whether a specified value either exists in an Object or not. The operators are 'in' and 'not in' and return true if a value is present and not present, respectively.

### 3.8.4 Comparison Operators

Consider the scenario, where you need to log in to your system/ cell phone and you enter password. If the password matches, you will be logged in else it will ask you to re-enter the password. Similar is the case with the programs, we develop. In a normal execution of the program, the instructions are executed one-by-one in a top-down manner and the program should terminate properly. But, there may be scenarios where the program has to make a selection based on some conditions. For someone reason, if the program is unable to execute the next instruction, the program will reach state of 'hang' and will not terminate properly.

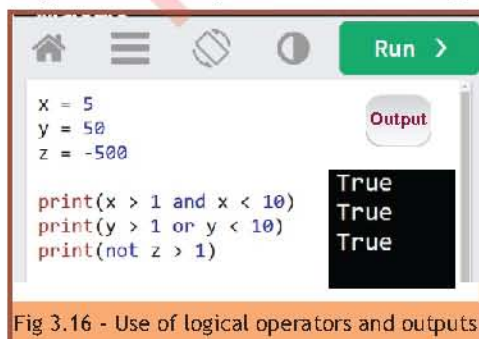
In the system log in scenario, if you have setup the password as 1234 but while logging in a typing error occurs and the password you entered is 1123. So, what should the log-in program do? Obviously it should compare the typed-in password with that of saved one, if it does not matches it should ask the user to re-enter the password. Alternatively, it should allow the user to log-in anyways. What do you think?

### 3.8.5 Logical Operators

The flow of the program is controlled via conditional statements which rely mainly on the comparison statements, the result of which either positive or negative will glide the program in respective direction. However, there are scenarios where more than one comparisons needs to be placed in a single conditional statement. For example, checking of a value for lower and upper limits, like mathematical constraint of  $(1 < x < 10)$  can be incorporated in a program code like  $(x > 1)$  and  $(x < 10)$ . So, to incorporate such multiple constraints in a single conditional statement, logical operators are used. Python provides the support of 3 types of logical operators, as:

- and - all conditions needs to be met, for a true outcome.
- or - any one condition needs to be met, for a true outcome.
- not - negates the outcome, i.e. opposite of the outcome will be the result.

The code in Fig 3.16 initializes three variables and in the following lines checks each of the variables whether their respective conditions are met in terms of the three logical operators. This way, the use of operator and corresponding outputs are highlighted for better understanding.



```
x = 5
y = 50
z = -500

print(x > 1 and x < 10)
print(y > 1 or y < 10)
print(not z > 1)
```

Output

```
True
True
True
```

Fig 3.16 - Use of logical operators and outputs

Table 3.3 - List of Comparison Operators

Operator	Name	Example
==	is equal	$x==y$
!=	is not equal	$x!=y$
>	Greater than	$x>y$
<	Less than	$x<y$
>=	Greater than or equal to	$x>=y$
<=	Less than or equal	$x<=y$



#### Teacher's Guide

For quick reference or checking output of a code-block, online Python interpreters can be used, like:

- [www.online-python.com](http://www.online-python.com)
- [www.onecompiler.com/python](http://www.onecompiler.com/python)
- [www.pythononlinecompiler.com](http://www.pythononlinecompiler.com)



### 3.8.6 Conditional Statements

Let us consider another example, where we take a number, input from the user and we have to determine whether the entered number is even or odd. So, we apply comparison with the help of 'if' statement and based on the result of the condition/ comparison, we will display our result to the user. For such comparisons, Python provides us with the six comparison operators, as shown in table 3-3.

Now, we can write a program as shown in Fig 3.17, to determine whether a number is even or odd.

```
1. num=eval (input("Enter a Number: "))
2. if(num%2==0):
3:   print("Entered Number Is Even. ")
4. if(num%2==1):
5:   print("Entered Number Is Odd. ")
```

Run-1

Output

```
Enter a Number : 7
Entered a Number is Odd.
```

Run-2

Output

```
Enter a Number: 4
Entered a Number is Even.
```

Fig 3.17 - multiple 'if' statements.

The first line of program is same, as the last example program. Next, we put an 'if condition' where we compare two values, i.e. if remainder of the number when divided by two is zero, then it is an even number. Remember to put a ":" after the condition, in an 'if' statement. Then we repeat the same process to check whether the number is odd. Let's run the program twice and by entering an even and odd number, check whether the conditions are working. This way, we know now that both if conditions are working. Additionally, we also learnt that if the condition does not matches, the corresponding 'if statement' is not executed.

#### IF-ELSE

Instead, a good choice to write the same code is as follows, where we check for a condition in 'if statement' and the alternate condition is handled by an 'else:' statement, which handles any other possible outcomes, as shown in Fig 3.18.

```
1. num=eval (input("Enter a Number: "))
2. if(num%2==0):
3:   print("Entered Number Is Even. ")
4. else:
5:   print("Entered Number Is Odd. ")
```

Fig 3.18 - 'if-else' statement.

#### IF-ELIF-ELSE

Let's take a step ahead. Some people treat Zero neither as even nor odd. It means, now we have 3 possibilities. The last program will treat zero as even integer. So, let's change it by incorporating 'elif' which is short of 'else-if statement'. We changed the above program and tried to incorporate all possible outcomes of the program by using 'if-elif-else', as shown in Fig 3.19.

```
1. num=eval (input("Enter a Number: "))
2. if(num%2==0):
3:   print("Entered Number Is Zero. ")
4. elif(num%2==1):
5:   print("Entered Number Is Odd. ")
6. else:
7:   print("Entered Number Is Even. ")
```

Output

```
Enter a Number: 0
Entered Number is Even.
```

Fig 3.19 - 'if-elif-else' statement, which does not provide the desired result.

However, it did not work, and the following outcome was displayed. What could have gone wrong? Let's discuss. We checked the condition if remainder is 0, then it is even. But since our number is zero, so is its remainder. And hence, this highlights that in 'if-elif-else' statement. If all other fail, only then the last part executes. Let's change our program accordingly where we check first if the input from user is zero, else-if the remainder is 1 for odd, otherwise it will be an even number; as shown in Fig.3.20. Corresponding output after running the program thrice, to check for all possible answers:

<pre>1. num=eval(input("Enter a Number: ")) 2. if(num==0): 3.     print("Entered Number Is Zero. ") 4. elif(num%2==1): 5.     print("Entered Number Is Odd. ") 6. else: 7.     print("Entered Number Is Even. ")</pre>	<p><b>Output</b></p> <pre>Enter a Number: 0 Entered Number is Zero. ----- Enter a Number: 6 Entered Number is Even. ----- Enter a Number: 7 Entered Number is Odd.</pre>
<p>Fig 3.20 - Code reprints 'if-elif-else' statement, after tuning.</p>	

Alternate method is to use logical operators '&&' (and) '||' (or) and '!=' (not), to combine multiple conditions in 'if-else' statement. This we leave as an activity for practice.

### 3.9 Iteration and Loop

Computer programs are the most useful in daily life where same task is to be executed in iterative fashion. Rather than write code for every part of same iteration, incorporate iterations in the programming language, to write small code for iterations and execute effectively. The most commonly used is the 'for loop'.

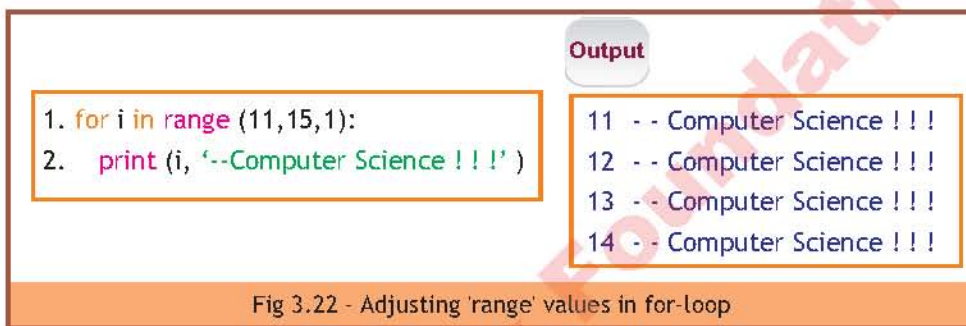
In 'for loop', we need to assign the starting point and the terminating condition, and the loop will execute the instruction(s) for the defined range. For example, you want to display your name five times. So, rather than writing the print() function with a subject name five times, better alternative is to place the print() function once within the for loop and define the range to be five; as shown in Fig 3.21.

<pre>1. for i in range (5): 2.     print (i, '--Computer Science !!!')</pre>	<p><b>Output</b></p> <pre>0 - - Computer Science !!! 1 - - Computer Science !!! 2 - - Computer Science !!! 3 - - Computer Science !!! 4 - - Computer Science !!!</pre>
<p>Fig 3.21 - A sample for-loop</p>	



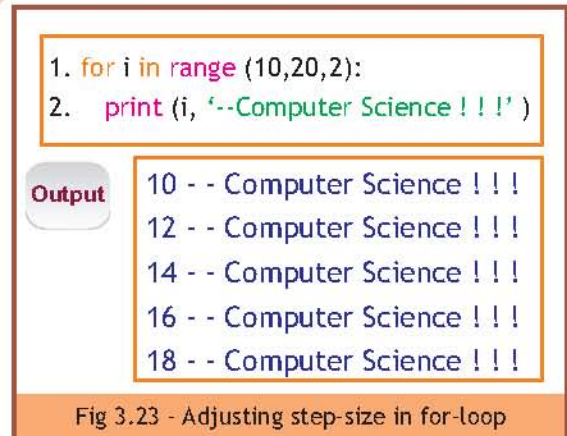
The output reflects that by default, the index (i) of the for loop starts with 0 with step of 1. Alternatively, we can define the starting point and terminating condition along with the step-size for a 'for loop'. Here another important aspect to note is the indentation of the code, which not only provides better readability and visibility of the code as the lines of code increases but also helps in outline the composition of the code and ultimately the overall program. It is quite critical to follow the 'indentation rules' in Python, as the line/ lines of code if not correctly indented can result in logical bugs and unforeseen outcome of the program. The ':' after the for loop statement shows that the lines of code that follow are to be executed in the loop; and hence need to be indented for the understanding of Python interpreter.

The program in Fig 3.22 starts the loop index from 11, terminating value is provided to be 15 with a step-size of 1.

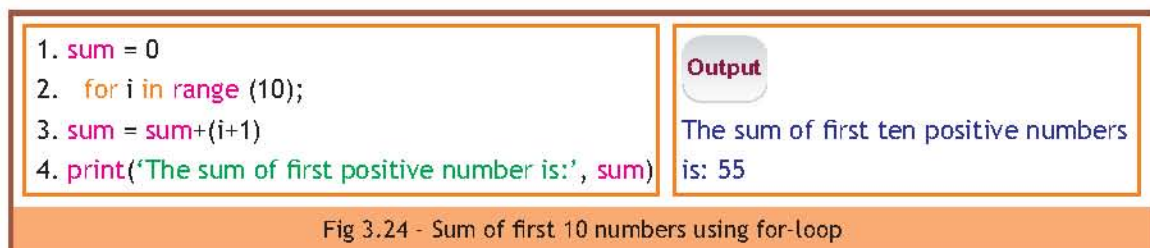


When the terminating condition in a loop met, the loop stop. In other words, the value we provide for terminating condition of the loop is not inclusive. As far as the step-size is concerned, we can provide any non-zero value, according to which the index value will update itself. Fig 3.23 shows a for loop with step-size 2.

For the sake of practice, you may explore for-loops further, by assigning a negative value to step-size, in range function. Thus, for any series and sequence, the value of range can be set accordingly.

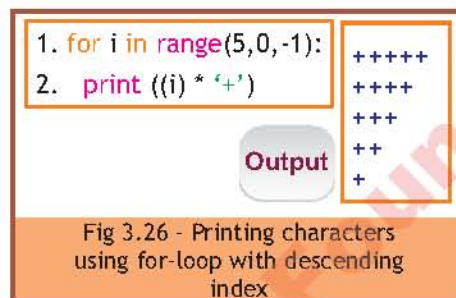
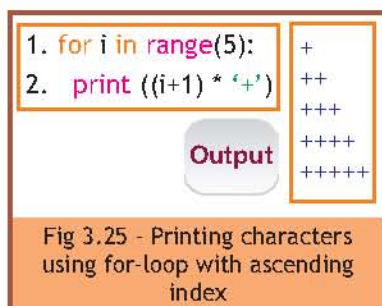


After understanding the syntax and corresponding outputs, let's explore in what other ways, we can utilize 'for loop'.



In the Fig 3.24, first, we initialized a variable namely 'sum' to 0. Next, we set the range of the for-loop and with a single statement; we were able to sum-up the first 10 positive integers. Just, change the range value to 100 and the same program of 4 lines of code, will calculate the sum of first 100 positive integers.

To emphasize, how to use index effectively, programs in Fig 3.25 & 3.26 use a character which is displayed, based on the index number in iterations. The range value defines the height of the triangles in both these figures which will execute 5 times. Since, the index starts with 0, therefore index is incremented before multiplying with character to be displayed as 'i+1'. Otherwise, the first line would have been blank as index value of 0 multiplied by anything will result in nothing. Consequently, the same print statement reflects an additional character in each line, specially in first and last line while using descending index. So, in the latter, 'i+1' is not required.



### Activity 3

Activity: Take input from the user which determines the height of the diamond and prints a hollow diamond (◇).

## 3.10 List

Now that we have a pretty good idea about variables and how to use them for I/O, let's take things to the next level. Assuming that in a class of 25 students, the teacher takes a test and marks are entered in a computer program. So 25 variables are required to store the numbers of each student in the program. In case of increase in number of students, additional variables need to be created and remembered. Though, it is correct but now effective specially how many variable names can we create and keep track of? Python provides a data type of 'List', which is alterable and can contain many values.

A list is defined with a name assigned with '[]', later we can assign the values. Alternatively, it is a good idea to initialize the list at the time of creation; if user input is not desired to run the program. For example,

```
Sngl_Dgt_Odd_Nums = [1,3,5,7,9]
```

Printing the list is simply straight forward, we have to pass the name of the list to print() and the whole list will be displayed. However, if a particular value needs to be accessed or displayed, the corresponding index of the list can be directly accessed and needs to be assigned like:

```
Print(Sngl_Dgt_Odd_Nums[2])
```

and the output will be '5', which is the value on 2nd index where index value starts from 0.

Another aspect of using list is that it provides some basic functions via which handling and manipulating a list is fun to learn and use. The len() function provides with the total number of



elements in a list. The `min()` and `max()` functions can be used to extract the smallest and largest element of the list, respectively. While `sum()` function when applied to the list, sums up all the elements of the list and result is displayed as shown in Fig 3.27.

```
1. Sngl_Dgt_Odd_Nums = [1,3,5,7,9]
2. for i in range (len (Sngl_Dgt_Odd_Nums)):
3. print('List element number',i+1, ':',Sngl_Dgt_Odd_Nums[i])
4. print('Sum of all element of List: ',sum(Sngl_Dgt_Odd_Nums))
5. print('Smallest number in the List ',min(Sngl_Dgt_Odd_Nums))
6. print('Largest number in the List: ',max(Sngl_Dgt_Odd_Nums))
```

```
List element number 1 : 1
List element number 2 : 3
List element number 3 : 5
List element number 4 : 7
List element number 5 : 9
Sum of all elements of List: 25
Smallest number in the List: 1
Largest number in the List: 9
```

Fig 3.27 - List Function Utilization

Another useful function is `insert()`, which inserts an element in the list, as shown in Fig 3.28.

```
1. Sngl_Dgt_Odd_Nums = [1,3,5,7,9]
2. Sngl_Dgt_Odd_Nums.insert (2,5)
3. print('List: ',Sngl_Dgt_Odd_Nums)
4. print('First 5 is found on index: ', Sngl_Dgt_Odd_Nums.index(5))
```

Output

```
List : [1,3,5,7,9]
First 5 is found on index: 2
```

Fig 3.28 - Inserting of a number in a list

A list can have duplicate values, too. If we insert another value of 5, other than the one at index number 2, let's say at the end of the list; the list will now hold more than one values of 5, i.e. on index numbers 2 and 5. In Fig 3.28 the `index()` function on the last line returns the position of a number in the list. In case of multiple occurrences of the said number, the placement of first match is returned.

Next, let's introduce some dynamicity in our code by introducing random numbers, as shown in the first line of the following code. Among the modules that Python provides for convenience, is the `random` module which consists of many inbuilt functions like `randint()` and can be used by importing in the code, with the 'import' keyword. Once defined in the start of the program, the function can be used anywhere in the program to generate a random number, in the given range. Like in line 4 which is inside a for-loop that will be executed 10 times where in every iteration the `randint()` function will generate a number between 1 and 99.



#### Activity 4

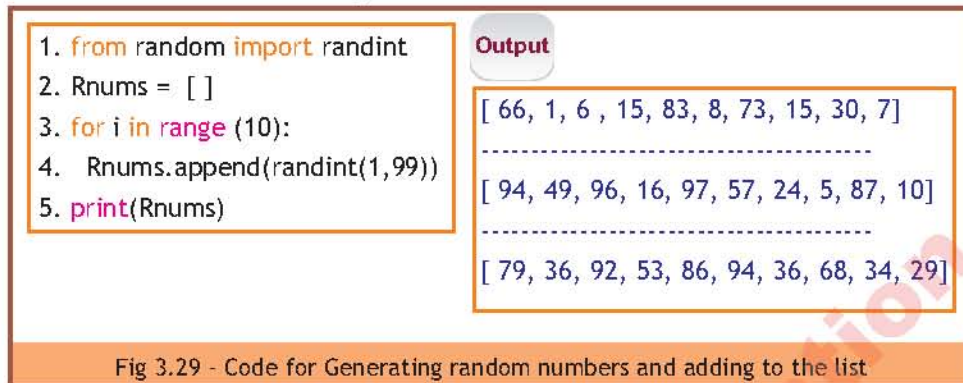
- Take a random 2 digit number.
- Subtract it from 100.
- Calculate the difference.
- Print the difference (magnitude only).



#### Tip

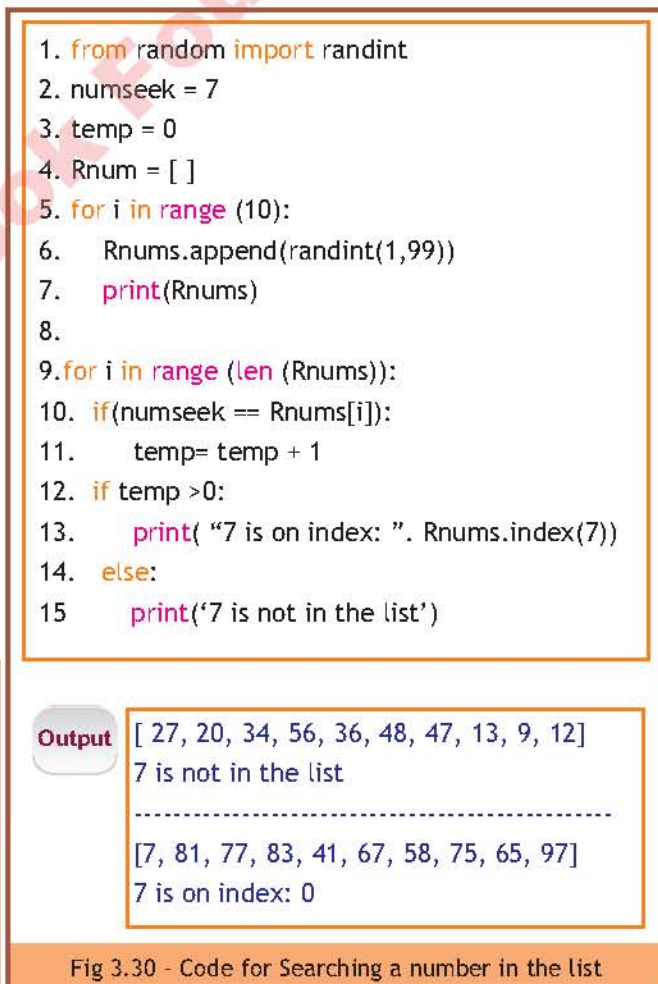
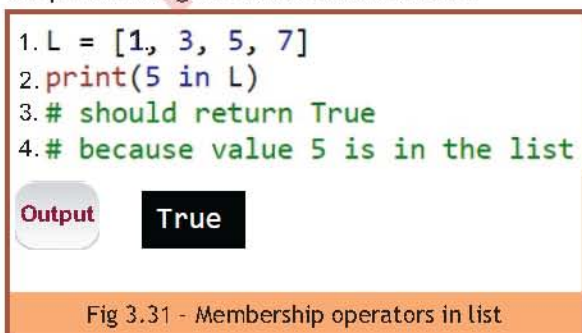
`abs()` function returns the value/magnitude of an integer only and discards "-ve" sign. This means that `abs()` will always returns a positive number

So, initially an empty list is defined and with the help of for loop, the list is populated with 10 random numbers. The outputs of running the same code shows different randomly generated values in each execution, as shown in Fig 3.29.



Let us extend the same code, where we initially populate random numbers and later, try to find a particular number in the list. The two possible outcomes are either the number is in the list or not. In Fig 3.30 shown is the code along with the both possible outcomes.

As discussed earlier, the membership operators can be used for checking whether a particular element exists in an object, like list. For better understanding, in Fig 3.31, a list is defined in the line 1 which consists of some members. In line 2, we check for whether the value 5 is in the list and corresponding result of either true / false gets printed. The last 2 lines are commented using '#' and are highlighted in a different color by the IDE. Since, the value of 5 exists in the list, therefore the output reflects 'True'. The membership operators are quite useful to control the flow of the program, dynamically by incorporating respective logic based on the values.





## 3.11 Functions in Python

So far, as you have experienced throughout this chapter that we started from a single line of code and now can develop tens of it. Additionally, via conditions and iterations we are able to not only write it effectively but also control the flow of the program. However, as the code lengthens it is hard to keep track of the logic you have applied, especially in case of if multiple algorithms co-exist.

So, the solution is to divide the code into segments through which it becomes handy to retain and understand large program. Each segment does a separate task and can be called upon to fulfill that particular task, if required from the main program or another segment. Such segments are called **functions**. In Python **'def'** is used to define a function with a unique name given to each function, followed by **':'**. For example,

1. `Def my_first_func():`
2. `print('This line is printed, when my_first_func() is called')`

```
1. def draw_line ():  
2.     print( '_' * 10)
```

So, let's consider another simple example where we want to draw a line on the screen, which is required to be done frequently and hence we define a **draw\_line()** function.

However, if we want to extend it further and divide the drawing of vertical lines too, we will define another function and change the names of the functions, such that they are meaningful and easy to portray about their respective working. Example code and corresponding output is as , as shown in Fig 3.32.

```
1. def draw_hline ():  
2.     print( '_' * 10)  
3. def draw_vline ():  
4.     print( '|')  
5. draw_hline  
6. draw_vline
```

Output

```
-----  
|
```

Fig 3.32 - A sample function definition to draw a line on console

As a next step, let's adjust the number of vertical lines to be two, such that one is in the start and other in the end of the horizontal line and place them just under the horizontal line. Since, the functionality of the function changed, so we may change the name of the functions as well, as reflected in the Fig 3.33.

Here, the **draw\_v\_parallel\_line()** function for drawing vertical lines is called 4 times. The functionality is defined once in the function and the same result is obtained, every time. The **draw\_h\_line()** function is called in lines 5 and 10.

```
1. def draw_h_line ():  
2.     print( '_' * 10)  
3. def draw_v_parallel_line ():  
4.     print( '|', ' ', *26, '|')  
5. draw_h_line()  
6. draw_v_parallel_lines()  
7. draw_v_parallel_lines()  
8. draw_v_parallel_lines()  
9. draw_v_parallel_lines()  
10. draw_h_lines()
```

Output

```
-----  
|                               |  
|                               |  
|                               |  
|                               |  
-----
```

Fig 3.33 - Multiple Functions in Program



### Activity 5

Take a number as input from the user, using a function, print a mathematical table.

### 3.11.1 Arguments of Function

The beauty of defining function is that, the same functionality can be achieved for different values. This is done by passing arguments to the function. The function needs to be defined in such a way, that it accepts a value and stores it in a variable and uses it inside the function. The point to remember is that, the variable defined inside the function, does not exist outside the function and is called **local variable**; as its scope is limited to the function itself. And that is the very reason; to test a variable inside the function is easy as compared to the variables in the main program, called **global variables**.

So, let us define an argument in each of the functions, as shown in Fig 3.34. Arguments are variables and we can name them anyway we want, as per the rules defined by Python. As you might have observed that whenever the function is called, rather than a variable, a value is being passed. This way, the value is assigned to that argument in the function and is used inside the function. The output is same as that of Fig 3.33. Try it and experience it yourself.

To make our functions more dynamic, why not use the argument portion of a function, to its potential. Let's extend the number of arguments and also allow the functionality to print the character which is also passed in the argument. This way, each function now, accepts a character and a number and the function will display accordingly, as shown in Fig 3.35.

```
1. def draw_hline (n):
2.     print( '_' * 10)
3. def draw_v_parallel_line (m):
4.     print( '|', ' ', *m, '|')
5. draw_hline(10)
6. draw_v_parallel_lines(26)
7. draw_v_parallel_lines(26)
8. draw_v_parallel_lines(26)
9. draw_v_parallel_lines(26)
10. draw_hlines()
```

Fig 3.34 - Function with Single Argument

Note that, in both the functions, the argument sequence is not the same. This is to emphasize that sequence of arguments is the choice of the developer. But, good programming technique is to stick to the same sequence this way; fewer errors are encountered and are easier to call the function without referring to the function, again and again. The important aspect is the order of arguments defined in the function, should be the same while calling the function.

```
1. def draw_hline (n,o):
2.     print(o*n)
3. def draw_v_parallel_lines (l,m):
4.     print( '|', ' ', *m, '|')
5. draw_hline(10, '=')
6. draw_v_parallel_lines( '| | ',24)
7. draw_v_parallel_lines( '| | ',24)
8. draw_v_parallel_lines( '| | ',24)
9. draw_v_parallel_lines( '| | ',24)
10. draw_hlines(10, '=')
```

Output



Fig 3.35 - Sample code and output of functions with multiple arguments

```
1. L_of_num = [7,8,9,10,11]
2. def list_sumup ():
3.     { sum = 0
4.     for i in range(5):
5.         sum += L_of_num [i]
6.     return (sum) }
7. print (list_sumup())
```

Output

45

Fig 3.36 - Sample code and output of functions with multiple arguments



### 3.11.2 Function Returns Result

Function is not always, about displaying or printing. Functions are also used to solve an equation and return the result. This is done by using 'return'. For example, the function in the Fig 3.36 sums up the elements of a list and returns the value, which is printed in the main program. The point to note is that a function can return a single value only. A good practice is to enclose the variable or value in '()' after return.

## 3.12 Debugging

Bug, is an unanticipated error which is generally logical in nature. Some are simple, as we discussed earlier in the example where we swapped the values of 2 variables. And to fix it we dry ran the code. However, for complex ones which exist in a code having large number of lines of code and many variables of various data types, manual dry-run will be tedious and hard. Hence, the solution is to debug the code.

### Debugger in IDE

In the Python IDLE menu, locate the Debug and select '**Debugger**'. Correspondingly, you will notice a '**Debug On**' message on the shell prompt before the cursor. Next, run your program and a window titled '**Debug Control**' will pop-up as shown in Fig 3.37 which allows you to monitor and examine the defined variables and the corresponding values, after each instruction. The pop-up window comprises of some buttons that will help you in debugging the code.

- **Step button** is the one that you will use initially, quite frequently; as it allows you to execute the code line-by-line. Every time you press it, the next line of code will execute.
- **Go button** allows you to jump to the breakpoint directly. And after that you can press step iteratively to seek the bug.
- **Over button** is used when there is a function call and you want to avoid it completely. This way, the function is fully executed and value is returned but the debugger skips it.
- **Out button** is used, when you have reached in a function and are examining it in debugger, but wants to skip the remaining portion of the function and come out of it.

Besides this, there are some checkboxes as well. '**Globals**' and '**Locals**' reflect the global and local data respectively. '**Source**' is the file, currently running in the debugger. Lastly, '**Stack**' highlights the function currently executed in the debugger.

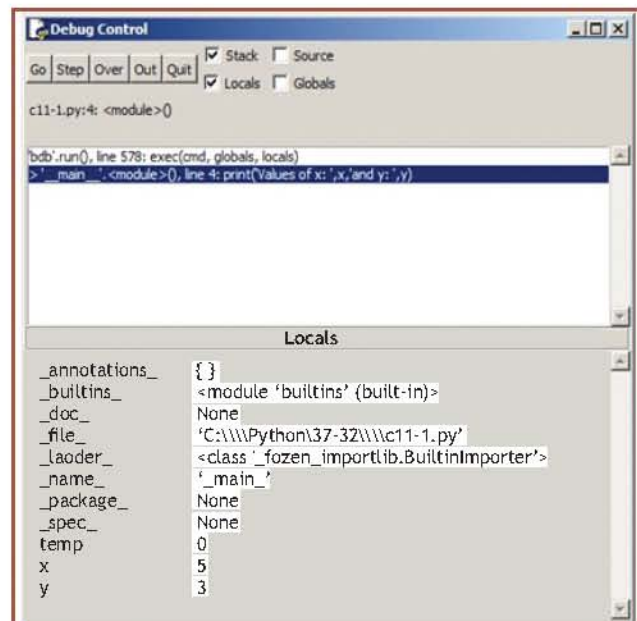


Fig 3.37 - A Debug Control

```

1. x = 5
2. y = 3
3. temp = 0
4. print ('Values of x: ',x,'and y: ',y)
5. print ("Lets's Swap the value of x & y")
6. temp = x
7. y = temp
8. x = y
9. print ('Values of x: ',x,'and y: ',y)

```

Fig 3.38 - Breakpoint in code of Fig.3.11

The most important element, we have kept for the end, the **Breakpoint**. A breakpoint is the line in the code which needs close examination, so the debugger will make a stopover there definitely. Although there may be multiple breakpoints, and debugger will make a stopover on each one of them. To assign a breakpoint is quite easy. After activating the debugger, just right click on the line of code, you want to set as a breakpoint and select '**Set Breakpoint**'. The said line will be highlighted yellow and is now marked as a breakpoint. For every breakpoint, you want to set, repeat the same process on the lines of codes, you want to examine. Next, run and debug your code. To clear the breakpoint, right click on the highlighted line and select '**Clear Breakpoint**'.

For better understanding, let's revisit the earlier code of example Fig 3.11 as an example scenario. Here, we set the breakpoint on the line, where we doubt the values are not as expected and need to examine it for confirmation i.e. line 7, as shown in Fig 3.38.

As soon as we run and start to step-over the code, the values of the variables are initialized and are shown in the Fig 3.37.

However, when we reach the breakpoint, we observe the values of the variables and get a confirmation about the bug, as shown in Fig 3.39. The corresponding output on the shell is also visible.

### Python Debugger (pdb)

Python provides a debugger library namely Python Debugger (pdb). From the command prompt, you may execute the program with 'pdb' parameter, by giving the command:

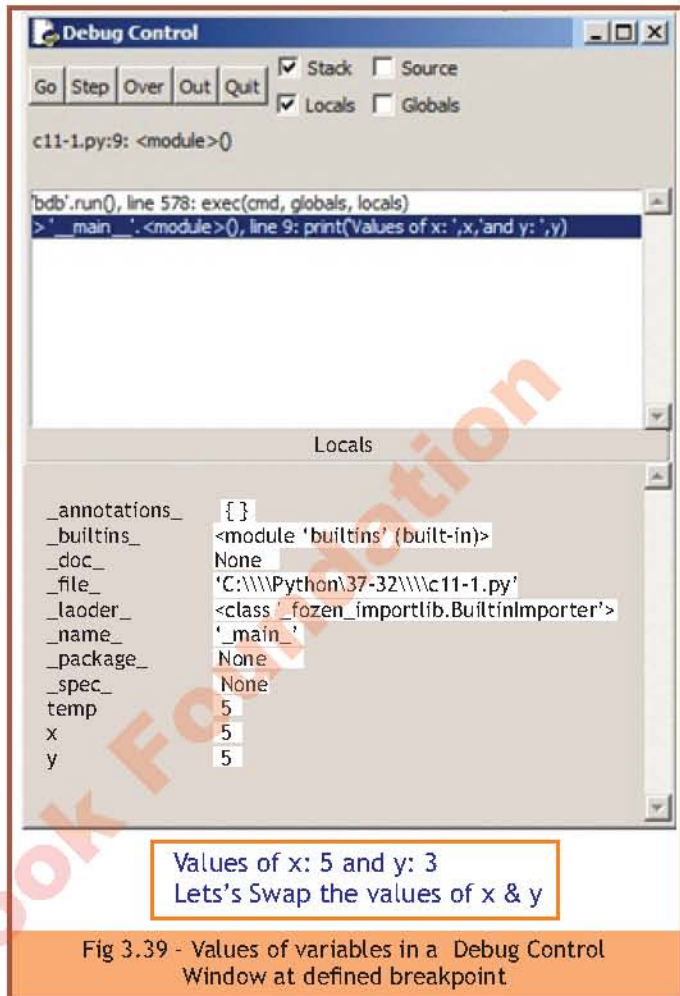


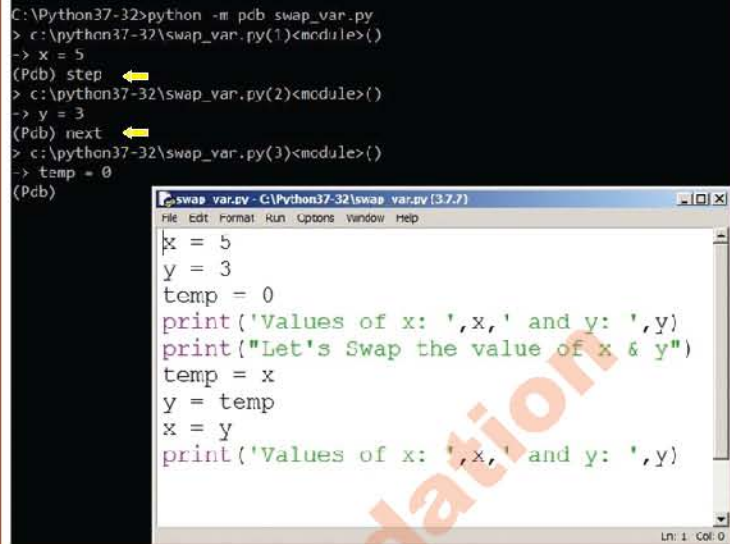
Fig 3.39 - Values of variables in a Debug Control Window at defined breakpoint



> `Python -m pdb swap_var.py`

Adding '-m switch' allows python to run the imported module as script. Thereafter, you can execute through the code line by line using 'step' or 'next' command, as shown in Fig 40.

You may assign breakpoints via 'break "line\_number"', like break 7 can be typed in to inspect line 7, in our example. Subsequently, if 'continue' command is used, the debugger will go directly to the breakpoint. For convenience and clarity, 'list' command is used in Fig 41, such that defined breakpoint and effect of continue command can be shown along with the code, simultaneously.



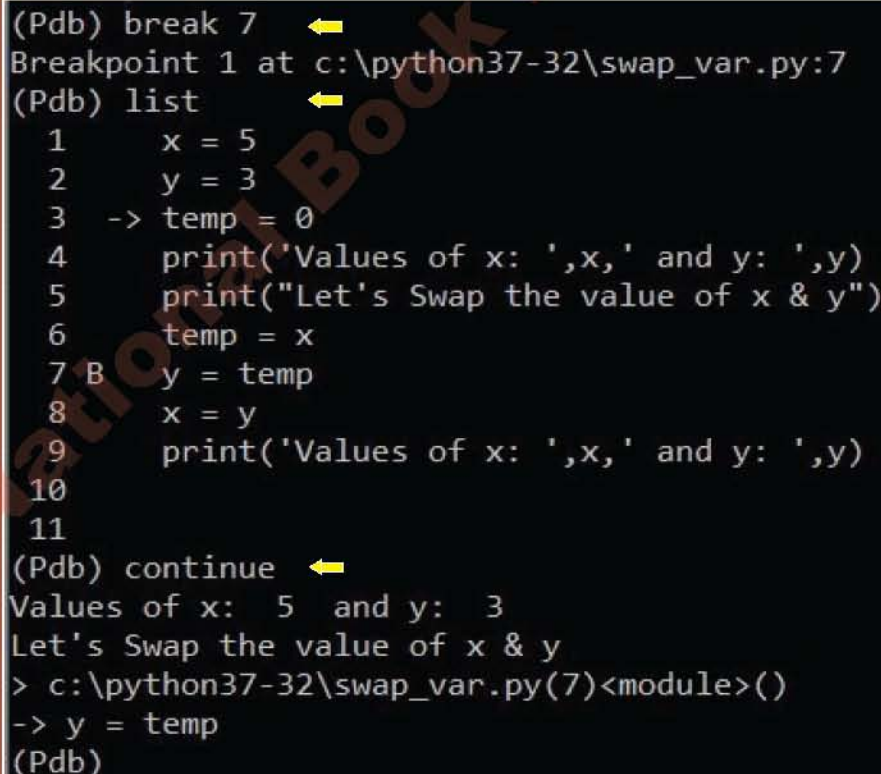
The screenshot shows a Windows Command Prompt window with the following text:

```
C:\Python37-32>python -m pdb swap_var.py
> c:\python37-32\swap_var.py(1)<module>()
-> x = 5
(Pdb) step
> c:\python37-32\swap_var.py(2)<module>()
-> y = 3
(Pdb) next
> c:\python37-32\swap_var.py(3)<module>()
-> temp = 0
(Pdb)
```

Below the command prompt is a window titled 'swap\_var.py - C:\Python37-32\swap\_var.py [3.7.7]'. It displays the following code:

```
x = 5
y = 3
temp = 0
print('Values of x: ',x,' and y: ',y)
print("Let's Swap the value of x & y")
temp = x
y = temp
x = y
print('Values of x: ',x,' and y: ',y)
```

Fig 3.40 - Executing code from Command Prompt



The screenshot shows a Windows Command Prompt window with the following text:

```
(Pdb) break 7
Breakpoint 1 at c:\python37-32\swap_var.py:7
(Pdb) list
1      x = 5
2      y = 3
3  ->  temp = 0
4      print('Values of x: ',x,' and y: ',y)
5      print("Let's Swap the value of x & y")
6      temp = x
7 B    y = temp
8      x = y
9      print('Values of x: ',x,' and y: ',y)
10
11
(Pdb) continue
Values of x: 5 and y: 3
Let's Swap the value of x & y
> c:\python37-32\swap_var.py(7)<module>()
-> y = temp
(Pdb)
```

Fig 3.41 - Break point set from Command Prompt

Alternatively, Python also gives the option to import pdb in the code, as the first line of the program. When the program executes, the debugger stops at the first line. Next, 'step' command can be used.

You may assign breakpoints in the code, using 'pdb.set\_trace()' instruction. As soon as the program executes, it stops on the very first line after 'pdb.set\_trace()' instruction allowing the code to be inspected and executed line-by-line, as shown in Fig 42.

Thereafter, you may type in 'step' on the shell to execute the code line by line. Fig 43 shows the similar sequence of commands that were executed from command prompt, as reflected in Fig 41.

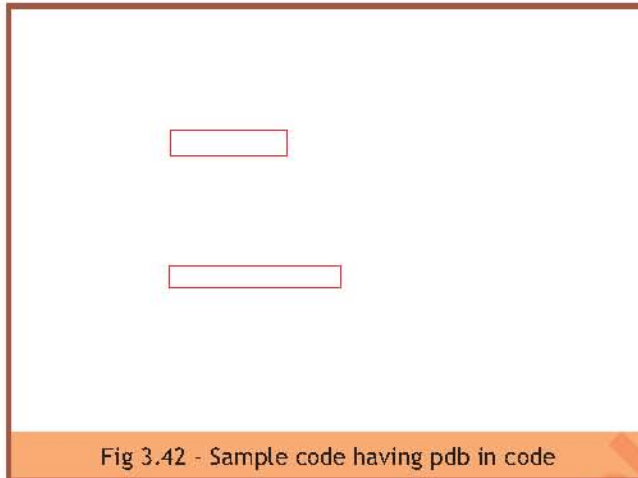


Fig 3.42 - Sample code having pdb in code

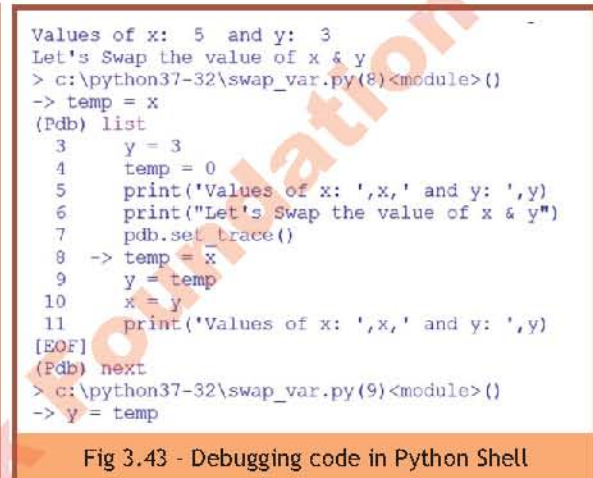


Fig 3.43 - Debugging code in Python Shell

### Debugging via print statements

While writing a program, the first statement you will learn is how to print and if used adequately is the easiest way to debug a program. By placing multiple print statements thoughtfully, a smooth traceability of any bug can be achieved. In Fig.44, we have placed multiple print statements to display the x, y and temp variables, after each statement where we are assigning new values to any of the said variables. The corresponding output allows us to identify the bug.

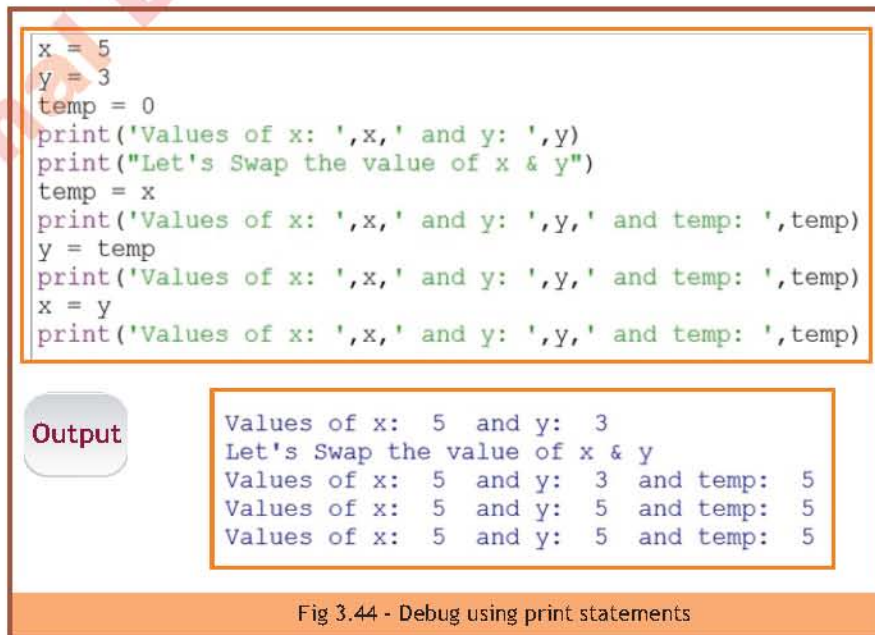


Fig 3.44 - Debug using print statements



## Assert

Python also provides 'assert' keyword which just like a boolean checks for a condition and returns True or False. If the assert statement is True, the statement will execute normally otherwise an 'AssertionError' is returned. Mainly, assert is used to locate logical bugs in code, or any illogical suppositions are taken into consideration. For example, if there is a typing error or miscalculation of multiplying the number with a negative number, in the code as shown in Fig. 45; then using assert before output can help in pinpointing the error.

```
x= 5 # initialize
x = 5 * -1 # may be typing error
assert x> 0
print(x)
```

**Output**

```
assert x> 0
AssertionError
```

Fig 3.45 - An Assert example

### Tip

You can type in 'help' on the shell prompt, at any point in time and a list of available debugging commands will be available. You may type in specifically the 'help "command\_name" ' for further guidance. Like 'help step'.

```
(Pdb) help step
s(step) Execute the current line, stop at the first possible occasion
(either in a function that is called or in the current
function).
```



### Activity 6

Take input from the user which determines the height of the diamond and print a filled diamond (◆).



### Activity 8

Take an odd number as input from the user and draw a 'X' with the help of a character like '+'. The input number defines the size of 'X' to be displayed.



### Activity 7

Write a turtle graphics code to draw a Tic-Tac-Toe board.



### Activity 9

Write a program to calculate how many iterations it takes to subtract a small number from a larger one, where both the numbers are positive integrals.

### Summary

- A set of well-defined instructions which a computer can execute to solve a problem is called a computer program.
- When the program is executed, the program is loaded into main memory from where one instruction at a time is fetched and provided to the processor which executes it.
- Computer Programs are developed in programming languages which provide the development environment via set of rules and keywords to use.
- Python is an open-source programming language. Prologations are provided in code with the help of conditional statements like 'if-elif-else' statements.
- To implement an iterative task, Python provides loops like the for-loop to accomplish the said task effectively and efficiently with lesser lines of code.
- Turtle graphics provides functions to design basic drawings and two dimensional geometrical shapes.
- With the help of a library we can produce result, with just one line of code, rather than writing our own tens of lines of code.
- Functions help in separating the code which is frequently occurring, this way the written function can be called in a single statement, rather than writing the same code again and again.
- The variable inside a function is a local variable and its scope is limited inside the function only, whereas variables in the main program are global variables and its scope covers the whole program.
- Arguments can be passed to functions for more dynamicity of the functionality.
- Function can return the result of the task it performed in a single value.
- Python provides a data type of List, which is alterable and can contain many values.
- Bug, is an unanticipated error which is generally logical in nature.
- Debugger allows examining the defined variables and the corresponding values, after each instruction.
- A breakpoint is the line in the code which needs close examination, so the debugger will make a stopover there definitely.



## Exercise



Select the best answer for the following Multiple-Choice Questions (MCQs).

- Computer Programs are categorized into \_\_\_\_\_.  
a) Interactive Program and Batch Program    b) Interactive Program and Badge Program  
c) Inactive Program and Batch Program    d) Inactive Program and Badge Program
- Python was first introduced in \_\_\_\_\_.  
a) 1990    b) 1994  
c) 1992    d) 2022
- Operator `!=` is used for \_\_\_\_\_.  
a) Equal    b) Logical AND  
c) Not equal    d) Logical OR
- We can design 2D geometrical shapes in Python, using \_\_\_\_\_.  
a) Operators    b) Iterations  
c) Variables    d) Turtle Graphics
- What will the following program do:  

```
for i in range(10,0,-1):  
    print('iteration no-',i)
```

  
a) It will give error    b) It will run 9 times  
c) It will run 10 times    d) It will run 11 times
- The interactive computer program accept some input which is from a user program like \_\_\_\_\_.  
a) Sub program    b) Web browser  
c) Print job    d) Page break
- \_\_\_\_\_ provide basic variable types as str, integer, float, etc.  
a) Print job    b) Page break  
c) Python    d) Procedure
- The `input()` function treats the entered data as a set of \_\_\_\_\_.  
a) Digits    b) Characters  
c) Symbols    d) Arithmetic operators
- Python provide the option of Modulo operator represented as \_\_\_\_\_.  
a) =    b) ==  
c) !    d) %

10. `***` is used to represent \_\_\_\_\_.
  - a) Division
  - b) Floor division
  - c) Multiplication
  - d) Exponentiation
11. If the program is unable to execute the next instruction, the program will reach the state of \_\_\_\_\_.
  - a) Hang
  - b) Reboot
  - c) Shutdown
  - d) Sleep
12. We apply comparison with the help of \_\_\_\_\_ statement.
  - a) For
  - b) While
  - c) If
  - d) But
13. In online IDEs, we can write and run our code without \_\_\_\_\_ in our computer.
  - a) Installing
  - b) Creating
  - c) Performing
  - d) Detecting
14. IDE stands for \_\_\_\_\_.
  - a) Internet Development Environment
  - b) Illegal Development Environment
  - c) Integrated Development Environment
  - d) Installer Development Environment
15. Turtle graphics help us to design basic drawing and \_\_\_\_\_.
  - a) 1D shapes
  - b) 2D shapes
  - c) 3D shapes
  - d) 4D shapes
16. AREPL means \_\_\_\_\_.
  - a) Read-Enter-Print-Loop
  - b) Read-Exit-Print-Loop
  - c) Read-Eval-Print-Loop
  - d) Read-Evolve-Print-Loop
17. To draw a triangle, we need to play with \_\_\_\_\_.
  - a) Angles
  - b) Vertices
  - c) Lines
  - d) Shapes
18. Bug is an unanticipated error which is generally \_\_\_\_\_ in nature.
  - a) Analog
  - b) Digital
  - c) Logical
  - d) Legal





### Give short answers to the following Short Response Questions (SRQs).

1. What are the applications of computer programming in daily life?
2. Write code to take input a number from user and print its mathematical table on screen from 1 to 10.
3. Take an odd number as input from the user, check if it is odd, otherwise ask the user to re-enter an odd number.
4. Write down the main examples of Python based application.
5. Differentiate between global and local variables with the help of suitable example.



### Give long answers to the following Extended Response Questions (ERQs).

1. Explain the applications of Python in different business and technical domains.
2. What are the basic functions that 'List' provides. Elaborate each of them with an example.
3. Write a program for a Dice Rolling Race game for 2 players. Alternatively, generate a random number for each player. If the number rolls out to be 6, then the player gets another roll. Rolls of every iteration is summed up to the previous rolls. The player who reaches 100 first, wins.
4. How to locate and select Debugger in IDLE? Write steps by taking an example into consideration.
5. Write code to print the multiplication of first 10 odd numbers and first 10 even numbers and find the difference of the two. [Hint: Use functions]



#### Mini Project 1: Number Guessing Game

Generate a Random Number and ask the user to guess the number.

For each input, respond the user, whether the guess is right or wrong.

If the guess was right, congratulate the user and inform about how many guesses it took till the right guess.

However for each wrong answer, inform the user to try again. Additionally, also provide a hint whether the guess was on the higher or lower side.

Give the user appropriate number of tries. e.g. if the guess is between 1 to 20, you may provide with 3 retries.



#### Mini Project 2: Motorcycle Rent Calculator

Motorcycles can be rented either based on distance travelled or the time the motorcycle was being used by the client. Both are separately calculated and added together, to generate total bill.

For example hourly rate is Rs. 100 while the distance rate is of Rs. 5/ km. A client travelled only 20 km but the possession of the motorcycle was for 3 hours. This way, the bill of travelled-distance will be Rs. 100/- but the possession-charges will be Rs. 300/-. This way, total bill will be of Rs. 400/-.

However, for existing members, the lesser of the two charges is treated as a "discount".

This way, if the client in above example was already a member, only Rs. 300/- needs to be paid by him.

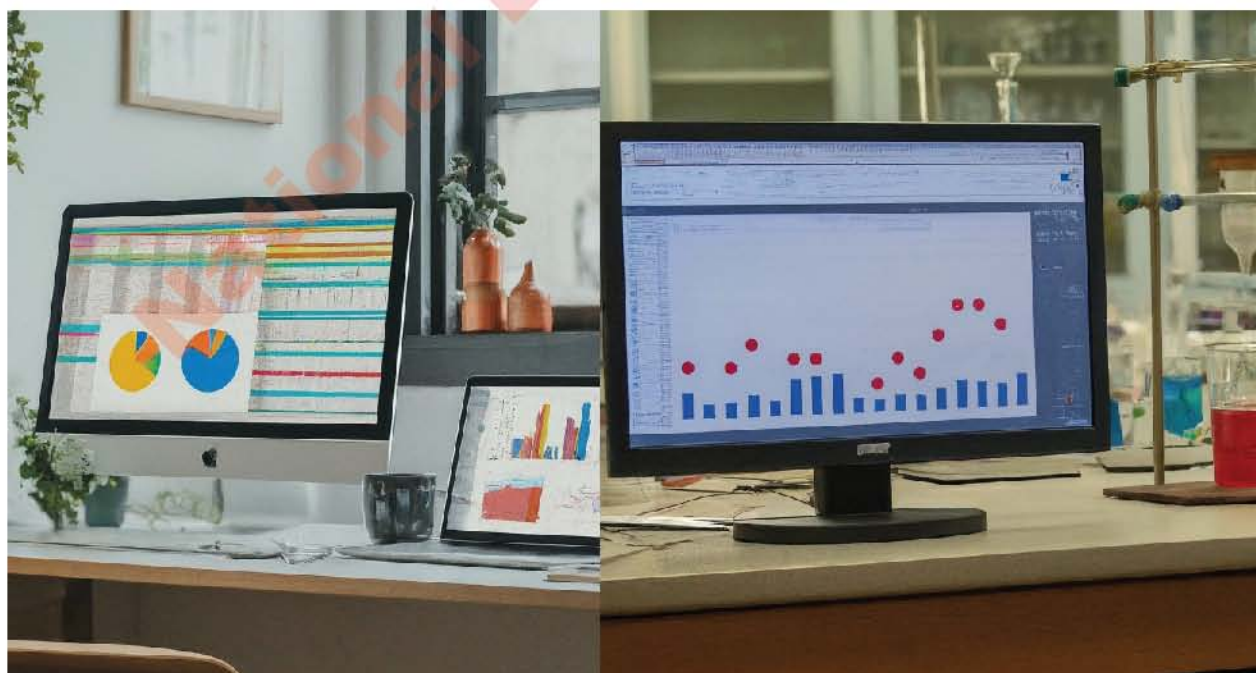
# Unit 04

## Data and Analysis



After completing this lesson, you will be able to:

- relate the role and importance of model building with their real-world applications.
- understand and explain experimental design in data science.
- analyze pre-existing datasets to create summary statistics and data visuals (such as bar charts, pie charts, line graphs, etc.)





## UNIT INTRODUCTION

Data analysis is a process, where data is cleaned, explored, and summarized, providing insights and information for the construction of statistical models to make predictions or draw inferences about real-world phenomena. The procedure helps to reduce the risks inherent in decision-making by providing useful insights and statistics. It is often presented in the form of charts, images, tables, and graphs.

A simple example of data analysis is whenever we decide in our daily lives by evaluating what has happened in the past or what will happen in the future if we make the decision in the same way? Basically, this is the process of analyzing the past or future and making a decision based on that analysis.

### 4.1 Statistical Modeling

Statistical modeling is the use of mathematical models and statistical assumptions to generate sample data and make predictions about the real world. A statistical model is a collection of probability distributions on a set of all possible outcomes of an experiment. Data in a model has a relationship between two or more variables. For example, in a science experiment you collect some data about the height and weight of some specific group of students. You represent the students' weight by variable  $x$  and students' height by variable  $y$ . After plotting the values, you find the relationship  $y = 5x + 2$ . It shows that variable  $y$  is a dependent variable because it is dependent on variable  $x$ . Figure 4.1 shows this relationship graphically.

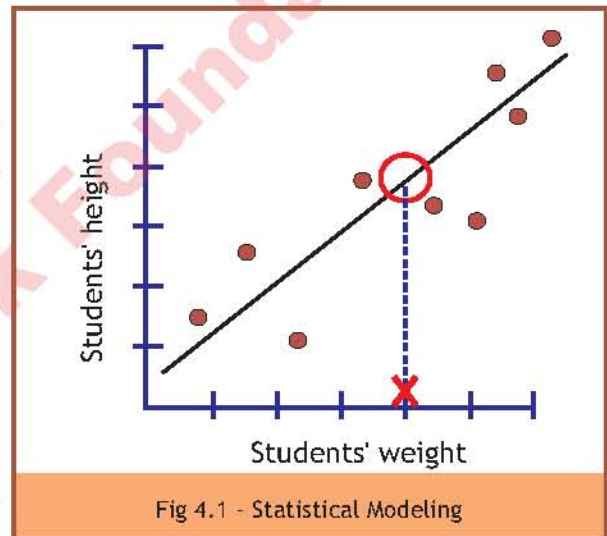


Fig 4.1 - Statistical Modeling

Statistical modeling is described as the mathematical relationship between random and non-random variables.

#### 4.1.1 Use cases for statistical modeling

A use case is a methodology used in system analysis to identify, clarify, and organize a system. A data science use case is a concrete real-world task to be solved using the available data. A use case is a description of how the users who use the system will perform tasks or achieve the goal. In the framework of a particular company, many variables are analyzed using data science techniques in the context of the company's specific industry. Data science use cases can be a problem to be resolved, a hypothesis to be checked, or a question to be answered. Essentially, doing data science means solving real-world use cases.

Although the content of each use case will likely be very different, there are some common things to always keep in mind:

- Data science use case planning is outlining a clear goal and expected outcomes, understanding the scope of work, assessing available resources, providing required data, evaluating risks, and defining KPI (key performance indicators) as a measure of success.
- The most common approaches to solve data science use cases are forecasting, classification, pattern and anomaly detection, recommendations, and image recognition.
- Some data science use cases represent typical tasks across different fields, and you can rely on similar approaches to solve them, such as customer churn rate prediction, customer segmentation, fraud detection, recommendation systems, and price optimization.

#### 4.1.2 How to Solve a Data Science Case Study?

The answer to this question varies from case to case. It depends on the company's business strategy, and on the core of the case study itself. However, the following points can be helpful to outline a general roadmap to follow with any data science use case:

1. **Formulating the right question:** This first step is reviewing available literature.
2. **Data collection:** At this step, we conduct a data inventory and gathering process.
3. **Data wrangling:** The data is cleaned, preprocessed, transformed, manipulated, and mapped from its raw form into a more suitable format.
4. **Data analysis and modeling:** The cleaned data is analyzed from the standpoint of its current status, then fit into a selected predictive statistical model.
5. **Result communication:** It includes sharing the most important findings and ultimate conclusions with management, shareholders, and any other involved parties.

Some examples from the real life as a case study are as follows:

- **Record of Production Goods and Services:** Production analysis may help in improving the quality.
- **Stock Market Data Analysis:** Investors and consumers can get help from predictive analysis of existing data of stock market.
- **Weather Forecasting:** It could be helpful for aviation, agriculture, mountaineering, and businesses.
- **Medical Records:** Data obtained and analyzed at healthcare organizations could be helpful for medical research and timely diagnosis of diseases.
- **Sales Tracking:** A good data analysis helps to plan next strategy model to prevent loss and get profit.
- **Population Record:** The analysis of population data is helpful for government and other organizations for better planning and distribution of resources.



- **Educational Data:** It helps in better planning of educational resources for students and teachers.
- **Natural Disaster Prediction:** It helps the people to take preventive measures against earthquakes, floods, landslides, snowfall, cyclones, and other natural disasters.
- **Pandemic Analysis:** It helps in controlling the pandemic and taking preventive measures timely.

### A sample case study about weather forecasting:

Weather conditions play significant role in our daily life, from dressing to travelling, planning activities and events etc. Unfavourable weather conditions can cause damage to life and properties. The following are steps that how predictive analysis about weather forecast can be performed and would be helpful in our real life.



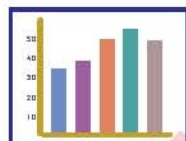
**1. Formulating the right question:** The existing data about the comparable weather situations will be analyzed and reviewed. You can observe from your past experiences that which weather conditions are likely to occur before a rainy day, a sunny day or otherwise.

**2. Data collection:** In this step you can collect data about temperature, amount of rainfall, wind speed and direction, and barometric pressure. You can collect this data by yourself for learning purposes, by using the thermometer, rain gauge, compass, and barometer. For accurate data collection an existing data from meteorological department is acquired.



**3. Data wrangling:** At this stage the collected data will be rearranged, processed, and grouped for the next stage. The data will be cleaned by removing duplicate values, impute missing values etc.

**4. Data analysis and modeling:** The cleaned data will be fitted to some statistical model for analysis.



**5. Result communication:** The accuracy of our results obtained from the statistical model will determine its reliability. Such models are trained on sample data and can be used to predict the weather condition. For example, when we give some new data about temperature, amount of rainfall, wind speed and direction, atmospheric pressure to this model, it will give us the prediction that how likely be the weather considering this new data.

### 4.1.3 Statistical modeling techniques

Data gathering is the foundation of statistical modeling. The data may come from the cloud, spreadsheets, databases, or other sources. There are two categories of statistical modeling methods used in data analysis: Supervised learning and unsupervised learning.

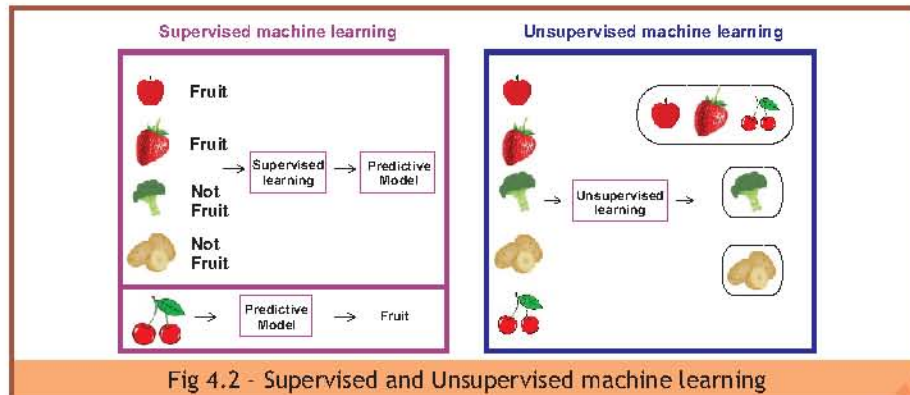


Fig 4.2 - Supervised and Unsupervised machine learning

## Supervised learning

In the supervised learning model, the algorithm uses a labeled dataset for learning, with an answer key the algorithm uses to determine accuracy as it trains on the data. For example, if you give a dataset of fruit and vegetables with labels “Fruit” and “Not Fruit”. The model will learn from the input data and labels that how does a fruit or vegetable should look like. Then if you give it some fruit or vegetable without label, it will recognize it and assign it the correct label. Supervised learning techniques in statistical modeling include regression model and classification model.

**Regression model:** If the result, label, or outcome of a model is continuous value then it will be called regression. The most common regression model is a linear model. A linear regression model is a mathematical equation that allows us to predict a response for a given predictor value. The variable which is used for prediction is known as independent variable( $x$ ) and the variable to be predicted based on the value of variable  $x$  is known as dependent variable( $y$ ). In simple linear regression the equation of the line is  $y = mx + b$ , where  $m$  is the slope and  $b$  is the  $y$  intercept. If the values of independent variable is plotted along the  $x$ -axis and the values of dependent variable are plotted along the  $y$ -axis and a line is drawn then the intercept and slope will be as follows:

**Intercept:** It is the point where the graph line meets the  $y$ -axis. It is also called  $y$ -intercept.

**Slope:** It is calculated by the term rise/run where rise is the distance from the  $x$ -axis to the  $y$ -intercept, and the run is the distance from  $y$ -axis to the  $y$ -intercept.

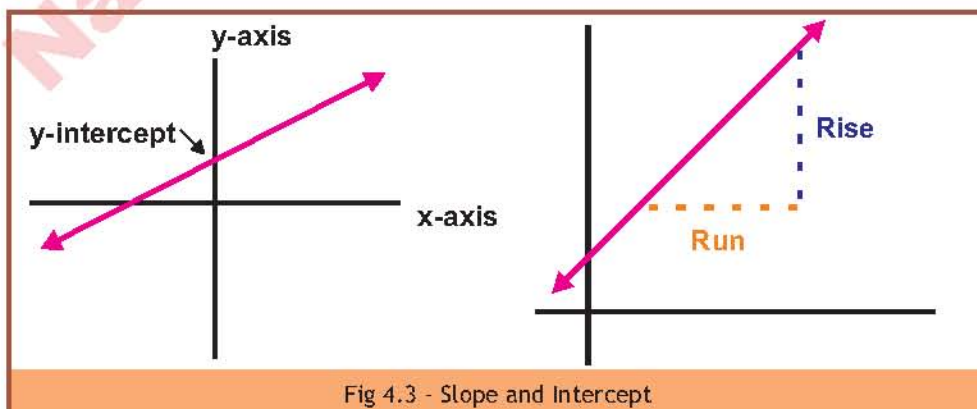
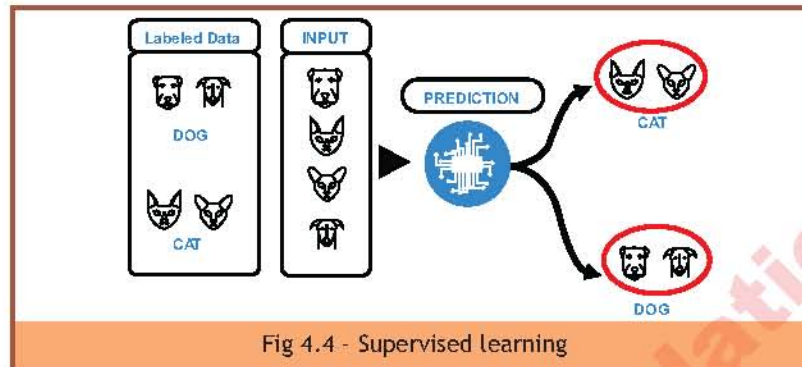


Fig 4.3 - Slope and Intercept



**Classification model:** If the result, label, or outcome of a model is a discrete value then it will be called classification. For example, if we predict that will a certain employ get raise in salary or not then it will be classification example. But if we make a prediction that how much salary raise an employee will get, based on some statistical data, it will be regression model.

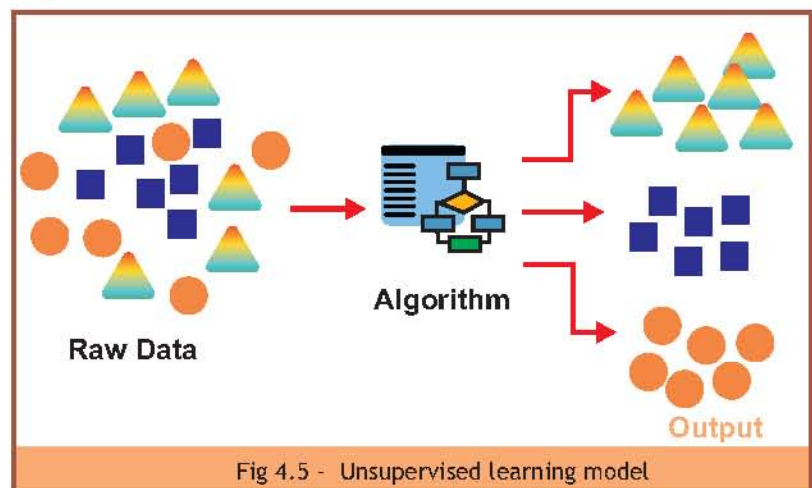


### Unsupervised learning

In the unsupervised learning model, the algorithm is given unlabeled data and attempts to extract features and determine patterns independently. Clustering algorithms and association rules are examples of unsupervised learning. K-mean clustering algorithms and reinforcement learning algorithms are used in unsupervised learning. The term clustering means the group of data items with maximum similarities. For example, to reduce the churn rate of customer a telecom company observes the customers' usage and divide them into three clusters, the customers with long call duration, the customers with heavy internet usage, and the customers with short calls and average internet usage. Now the company will give economical call rates to the cluster of having long calls, and attractive internet packages to the customer with heavy internet usage. The term association means how likely is to do second action if first action is done. For example, in a supermarket customer 1 buys bread, milk, tea and tissues. Customer 2 buys bread, milk, coffee, and eggs. Now if the customer 3 buys bread, then there are more chances that he will also buy milk. In this example an association between bread and milk has been found.

#### K-means clustering:

The algorithm combines a specified number of data points into specific groupings based on similarities. An example based on the image could be a machine learning algorithm that instead of classifying by known labels separates data into groups where members are similar as possible as shown in Figure 4.5.



## 4.1.4 Build a statistical model using Python

You can build statistical models to perform predictive analysis. Several tools are available for this task such as MS-Excel, Weka, R Studio and Python. Many datasets are available on the internet. [www.kaggle.com](http://www.kaggle.com) and <https://github.com/> are the most popular platform to download various datasets. However, to create your experimental statistical model you don't need to download any dataset. You can create your own dataset by

generating some random numbers by using python code. We will plot these numbers on a linear graph by using correlation coefficients  $y = mx + c$  ( $y = 3x + 4$ , where  $x$  will also be generated randomly) using slope and intercept. You could write python code in your favorite IDE if you already had installed any, else you can open browser on the computer and type the URL <https://colab.google/> and press enter key. The online compiler for Python will open, click on “Open Colab” button on the top right corner of the browser window ( See Figure 4.6)



Fig 4.6 - Build a statistical model using Python

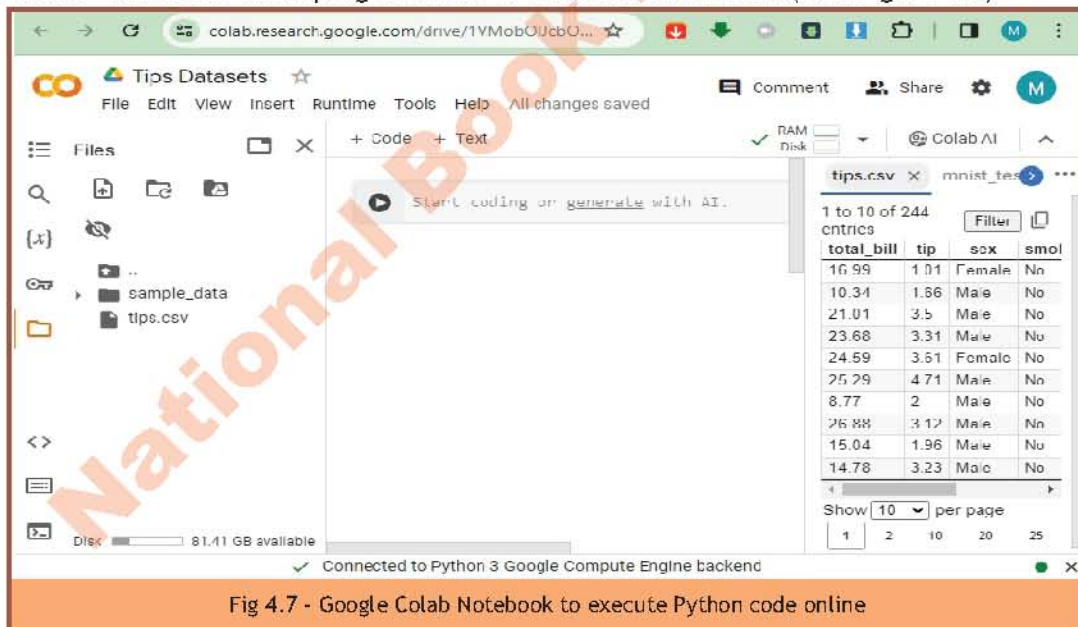


Fig 4.7 - Google Colab Notebook to execute Python code online



### Teacher's Guide

To explore more tools to build statistical model using python following are some useful links to download Weka and R Studio:

<https://www.weka.io>

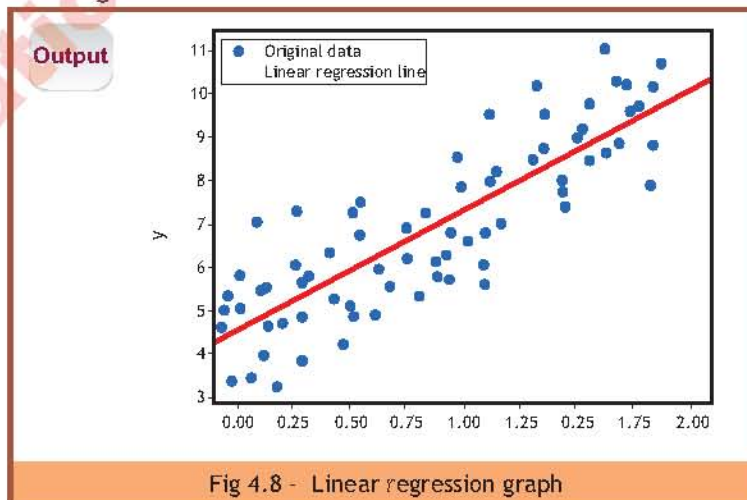
<https://www.rstudio.com/tags/rstudio-ide>



Type the following code in the browser window:

```
1. # Import necessary libraries
2. import numpy as np
3. from sklearn.linear_model import LinearRegression
4. import matplotlib.pyplot as plt
5. # Generate a small dataset
6. np.random.seed(42)
7. X = 2 * np.random.rand(100, 1)
8. y = 4 + 3 * X + np.random.randn(100, 1)
9. # Create and fit the linear regression model
10. model = LinearRegression()
11. model.fit(X, y)
12. # Make predictions on new data points
13. X_new = np.array([[0], [2]])
14. y_pred = model.predict(X_new)
15. # Plot the original data and the linear regression line
16. plt.scatter(X, y, label='Original data')
17. plt.plot(X_new, y_pred, 'r-', label='Linear regression line')
18. plt.xlabel('X')
19. plt.ylabel('Y')
20. plt.legend()
21. plt.show()
```

Click the Run button or press Ctrl+E keys, the linear regression graph will be fitted on newly generated data set. See Figure 4.8.



## 4.2 Experimental Design in Data Science

Experimental design is about precise planning and design to ensure that you have the appropriate data and design for your analysis or studies so that erroneous conclusions can be prevented.

### 4.2.1 Experimentation in data science as a tool

Experimental design is a tool or technique used to organize, conduct, and clarify the results of an experiment efficiently. It leads to the important factors that must be investigated. An experiment will produce precise and trustworthy results if you select the right sample size and plan it carefully. Experimental design is used in many disciplines like engineering, psychology, agriculture, and medicine.

#### Experimental Design Flow

In data science, the experimental design flow often adheres to a standardized procedure to guarantee careful preparation, execution, and evaluation of experiments. Here is a general description of the data science experimental design flow. It will indicate what the research question is. Clearly define the research topic or problem statement that your experiment will attempt to solve. This will serve as the design's compass throughout.

- **Develop Hypotheses:** The first step is developing hypotheses which will clarify the differences and correlations between various variables involved in the experiment.
- **Find the variables:** In this step you must decide which variables will be fixed and which will be changed. The fixed variables will be called independent variables. The values that will change according to fixed variable will be called dependent variables. For example, if the salary of the person will be higher for the employees with more work experience, then years of experience will be independent variable and salary will be dependent variable. You should also consider other potential confounding factors that might need to be taken into consideration in this case the skill type of an employee can also affect the salary. For example, employees having C++ programming skills might be paid higher than employees having Java programming skills.
- **Determine the Experimental Design:** Choose an appropriate experimental design according to the nature of research and available resources. Factorial designs, randomized block designs, totally randomized designs, and more are examples of common designs. Each design has its own benefits and factors to consider. (The details of these experimental designs is beyond the scope of this book.)
- **Calculating the Needed Sample Size:** Calculate the necessary sample size to generate appropriate statistical results and identify their significant effects.
- **Random Assignment and Selection:** To produce more accurate results choose random samples. It ensures that groups are identical in there is no bias. It will ensure that any change in dependent variable is purely based on the change in independent variable.
- **Carry out the experiment:** After taking careful sample size follow the instructions and gather the data. While collecting data about the samples, you should strictly follow the defined



methodology.

Fig 4.9 - Experimentation in data science as a tool

- **Data Analysis:** Use the collected data for appropriate statistical analysis to evaluate the hypotheses and develop conclusions. Depending on the experimental design and research issue, this may require different procedures including hypothesis testing, regression analysis, ANOVA, and other statistical methods. ANOVA (analysis of variance) is a statistical test for detecting differences in the group means, when there is one dependent variable and one or more independent variable.
- **Interpreting and Drawing Conclusions:** After data analysis you will reach to the conclusion. Observe the results carefully and consider any practical factors which might affect the results.
- **Discuss and Report:** Give a clear and concise presentation of the design of the experiment, methodology, findings, and conclusions. To enable reproducibility and transparency, the experiment should be thoroughly documented. In this way if the same experiment is repeated by different group of data scientists, it will produce more or less the same results.

### Principles of Experimental Design

The fundamentals of this design determine whether an experiment is successful. These principles of experimental design consist of:

**Principle of Randomization:** According to this principle you divide the people into various treatment groups randomly. The randomization technique ensures that there is no bias or preplanned grouping for example some groups might have people with one capability and others might have some other strengths or weaknesses, such grouping cannot provide accurate results. Therefore, groups should be decided randomly. For example, if we conduct an experiment about the effectiveness of a new medicine for asthma. There are 100 patients. The principle of randomization means we must randomly make the two groups of patients. 50 patients will be in each group. Some people in the first group might have severe asthmatic problem while the other might have less chronic condition.

**Principle of local Control:** A control group is a group that has not received any treatment; hence a control group is used to compare the treatment group's findings to its outcomes. This approach guarantees that the treatment is the reason that any other factors are to blame for the outcomes. In our example we will add the new medicine to one group and the other group will be treated

with their regular medication. The group which will not be given the new medicine is called control group.

**Principle of Blocking:** Grouping participants according to a particular trait that could have an impact on the results is known as blocking. This principle makes sure that the results aren't affected by group differences. For example, the patients belongs to two different genders, male and female. So we can make two blocks, one for women and other for men. Suppose there are 56 women and 44 men, we will make two groups of women 28 each and two groups of men 22 each. One group of 28 women and 22 men will be treated with new medicine while the other group of 28 women and 22 men will be treated with their regular medicines.

**Principle of Replication:** If you repeat the experiment again and again, it is called replication. The replication ensures that results are reliable. This principle ensures that experimental outcomes are not coincident, erroneous or a result of some randomness. In above example if we get the results that new medicine is more effective than the previously used medicines, then different data scientists should repeat this test for new medicine, before finally declaring it effective. Because there might be some coincidence, or patients taking the new medicine might have some psychological impact of taking new medicine. Therefore, for principle of replication ensures that these experiments should be repeated with different group of patients, may be from some other demographic background. If the medicine gives the same results repeatedly then we can declare that the new medicine is more effective.

#### 4.2.2 Correlation and causation

**Correlation** describes an association between different types of variables. It is a statistical measure of the degree to which changes to value of one variable predicts the change in the value of other variable. It is a statistical indicator of the relationship between variables. These variables change together which means they co-vary. But this covariation is not necessarily due to a direct or indirect causal link.

**Causation** means that changes in one variable brings about changes in the other. There is a cause-and-effect relationship between variables. The two variables are correlated with each other and there is also a causal link between them. A correlation does not imply causation, but causation always implies correlation.

For example , a smart phone user complains, "Whenever I try to send text message, my phone lags.." A quick look at the smartphone confirmed that there were five game apps open at the same time plus Facebook and YouTube. The act of trying to send a text message wasn't causing the lagging, the lack of RAM was causing the phone to lag. But the user immediately connected it with the last action she was doing before the lags. She was implying a causation where there was only a correlation.



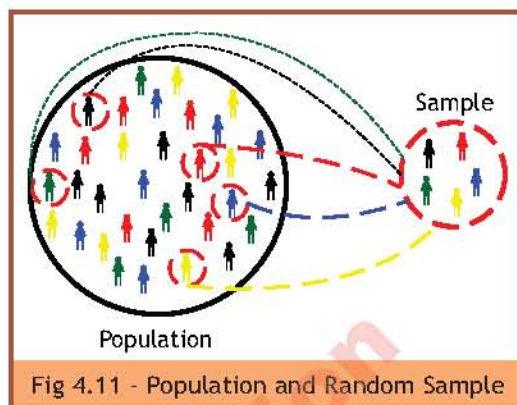
Fig 4.10 - Correlation and causation



### 4.2.3 Population and Random Sample

Population and random sample are the terms used in statistics. Population is a complete collection or set of similar items or events, which are of interest or under consideration regarding some problem or experiment. Therefore, a population is a complete set of people with a specialized set of characteristics.

Whereas, a random sample is a subset of population. each member of the population has an equal chance of being selected as a sample. A population and random sample is shown in Figure 4.11.



### 4.2.4 Parameter and Statistic

Parameter is a number or quantity in statistical population, that describes an aspect of entire population. While a statistic is a number or quantity that describes characteristics or an aspect of a sample. Parameters are the key things we want to learn about, they are usually unknown. Sample's statistics give us estimates for parameters. There will always be some uncertainty about how accurate estimates are. More certainty gives us more useful knowledge. For every parameter we want to learn about we can get a sample and calculate a sample statistic, which gives us an estimate of the parameter. Some Important Examples are:

Table 4.1 - Parameter and Statistic

Parameters	Statistics
Average income for Pakistan.	Average income for government employees in Pakistan.
Mean weight of Persian cats.	Mean weight of 100 Persian cats.
Proportion of all people who prefer soft drinks.	Proportion of random sample of 500 persons who prefer soft drinks.
Standard deviation of all the transaction times in the state bank.	Standard deviation of a random sample of 1000 transaction times in the state bank.



Tip

A parameter is a fixed, unknown numerical value, while the statistic is a known number and a variable which depends on the portion of the population.

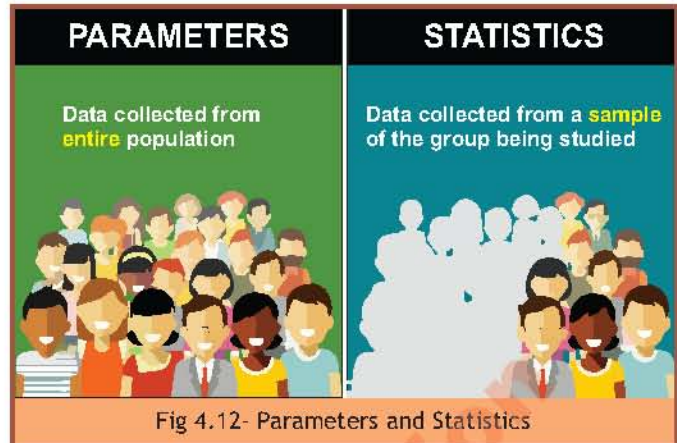
Figure 4.12 shows the difference between parameters and statistics.

Mean, median and mode are different types of averages used to represent typical values in a population. Therefore you can say that the mean value of a population is parameter while the mean value of a sample is statistics. Some examples of parameters and statistics are as follows:

The typical age of people in a country

The typical profits of a company

The typical range of an electric car



## 4.2.5 Data collection methods

Data collection is the process of collecting, measuring and analyzing different types of information using a set of standard validated techniques. The main objective of data collection is to gather information-rich and reliable data, and analyze them to make critical business decisions. Once the data is collected, it goes through a rigorous process of data cleaning and data processing to make this data truly useful for businesses. There are two main methods of data collection in research based on the information that is required, namely:

- Primary Data Collection
- Secondary Data Collection

### Primary Data Collection Methods

Primary data refers to data collected from firsthand experience directly from the main source. It refers to data that has never been used in the past. Here are some of the most common primary data collection methods:

#### Interviews

Interviews are a direct method of data collection. It is simply a process in which the interviewer asks questions, and the interviewee responds to them. It provides a high degree of flexibility because questions can be adjusted and changed anytime according to the situation.



#### Observations



In this method, researchers observe a situation around them and record the findings. It can be used to evaluate the behavior of different people in controlled (everyone knows they are being observed) and uncontrolled (no one knows they are being observed) situations. This method is highly effective because it is straightforward and not directly dependent on other participants. For example,



a person looks at random people that walk their pets on a busy street, and then uses this data to decide whether or not to open a pet food store in that area.

### Surveys and Questionnaires

Surveys and questionnaires provide a broad perspective from large groups of people. They can be conducted face-to-face, mailed, or even posted on the Internet to get respondents from anywhere in the world. The answers can be yes or no, true or false, multiple choice, and even open-ended questions.



### Focus Groups

A focus group is similar to an interview, but it is conducted with a group of people who all have something in common. The data collected is similar to in-person interviews, but they offer a better understanding of why a certain group of people thinks in a particular way. However, some drawbacks of this method are lack of privacy and domination of the interview by one or two participants. Focus groups can also be time-consuming and challenging, but they help reveal some of the best information for complex situations.

### Oral Histories

Oral histories also involve asking questions like interviews and focus groups. However, it is defined more precisely, and the data collected is linked to a single phenomenon. It involves collecting the opinions and personal experiences of people in a particular event that they were involved in. For example, it can help in studying the effect of a new product in a particular community.



### Secondary Data Collection Methods

Secondary data refers to data that has already been collected by someone else. It is much more inexpensive and easier to collect than primary data. While primary data collection provides more authentic and original data, there are numerous instances where secondary data collection provides great value to organizations.

Here are some of the most common secondary data collection methods:

#### Internet

The use of the Internet has become one of the most popular secondary data collection methods in recent times. There is a large pool of free and paid research resources that can be easily accessed on the Internet. While this method is a fast and easy way of data collection, you should only source from authentic sites while collecting information.



### Government Archives

There is lots of data available from government archives that you can make use of. The most important advantage is that the data in government archives are authentic and verifiable. The challenge, however, is that data is not always readily available due to several factors. For example, criminal records can come

under classified information and are difficult for anyone to have access to them.

### Libraries

Most researchers donate several copies of their academic research to libraries. You can collect important and authentic information based on different research contexts. Libraries also serve as a storehouse for business directories, annual reports and other similar documents that help businesses in their research.



## 4.2.6 Real world experimentation examples

**Facebook A/B testing:** A/B testing, also known as split testing, is a method used by companies like Facebook to compare two versions of something to figure out which one performs better. In the context of Facebook or other online platforms, this often involves testing different versions of a webpage, app feature, or advertisement to see which one leads to better results.

**AirBnB uses data science to help renters set their prices:** Airbnb uses statistical experimentation, such as A/B testing, to enhance its platform and user experience. For instance, when testing a new feature like a redesigned booking flow or a different search algorithm, Airbnb would create two versions (A and B) and randomly assign users to each group. By collecting and analyzing data on user interactions, conversion rates, or other relevant metrics, Airbnb can determine which version performs better.

This data-driven approach allows Airbnb to make informed decisions about implementing changes that positively impact user satisfaction, booking rates, or other key performance indicators. Through ongoing experimentation, Airbnb continuously refines its platform, ensuring that updates are based on empirical evidence rather than assumptions, ultimately contributing to a more effective and user-friendly service.

**YouTube and statistical experimentations:** YouTube utilizes statistical experimentation to improve user engagement, content discovery, and overall user experience. A/B testing is one of the key methodologies they use. For example, when introducing a new recommendation algorithm or modifying the layout of the platform, YouTube may create two versions (A and B) and randomly assign users to each group. YouTube collects data on user interactions, such as click-through rates, watch time, and user feedback, to evaluate the performance of each version. By comparing metrics between the control group (A) and the experimental group (B), YouTube can make data-driven decisions about the effectiveness of the changes. This iterative process allows them to fine-tune algorithms, recommend more relevant content to users, and optimize features to enhance overall satisfaction.

## 4.3 Analyze pre-existing datasets to create summary statistics and data visuals. (bar charts, pie charts, line graphs, etc.)

To analyze a dataset a series of steps is involved. By following all those steps, you can extract meaningful insight and information from the data. These steps include:



- Data exploration
- Data cleaning

You can eliminate unwanted data to get more meaningful insight. Summary statistics are calculated to understand the tendency (how closely similar is data) and dispersion (how scatter is the data). After computing, mean, median, mode you will calculate count and frequently etc. After all these steps of data exploration, data cleaning, calculation of summary statistics, data will be ready for visualization. Various diagrams/tools are used for data visualization like charts, graphs, boxplots etc. various graphs are shown in Figure 4.13.

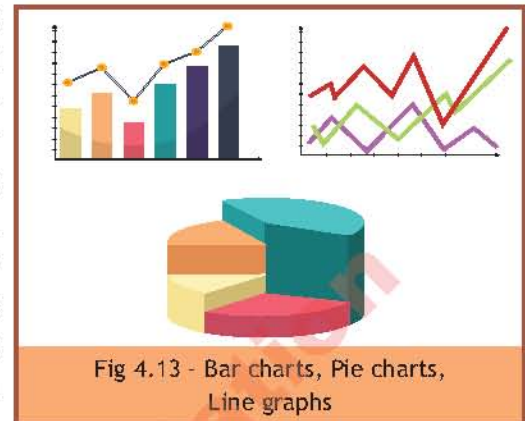


Fig 4.13 - Bar charts, Pie charts, Line graphs

### 4.3.1 Data products (charts, graphs, statistics)

A data product is a tool that uses data to help businesses to improve their decisions and business processes. Data products, in turn, leverage the results of data analysis to automate processes, provide real-time recommendations, or deliver data-driven services to users or organizations.

In simple words, data analysis is the process of deriving insights and knowledge from data, while data products are the applications or tools that use those insights to deliver value to users. Together, they form the core of data science, enabling organizations to harness the power of data for decision-making and innovation.

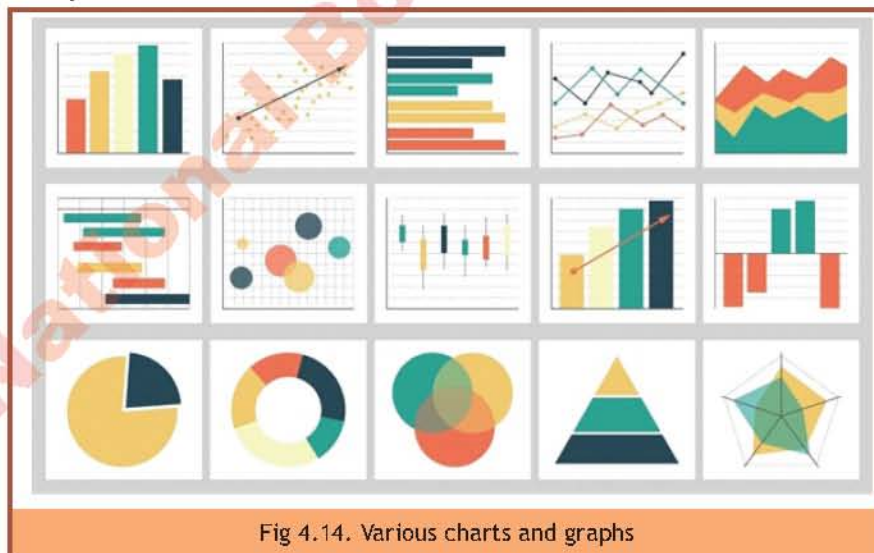


Fig 4.14. Various charts and graphs

### 4.3.2 Data Visualization

Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see

and understand trends, outliers, and patterns in data. Additionally, it provides an excellent way for employees or business owners to present data to non-technical audiences without confusion.

In the world of Big Data, data visualization tools and technologies are essential to analyze massive amounts of information and make data-driven decisions.

### 4.3.3 Data analysis through Python

In today's world, a lot of data is being generated daily. And sometimes to analyze this data for certain trends, patterns may become difficult if the data is in its raw format. To overcome this data visualization comes into play. Data visualization provides a good, organized pictorial representation of the data which makes it easier to understand, observe, and analyze data. Python provides various libraries that come with different features for visualizing data. All these can support various types of graphs.

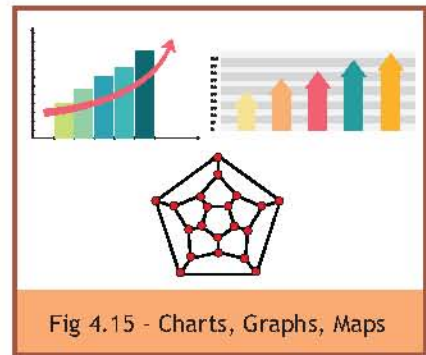


Fig 4.15 - Charts, Graphs, Maps

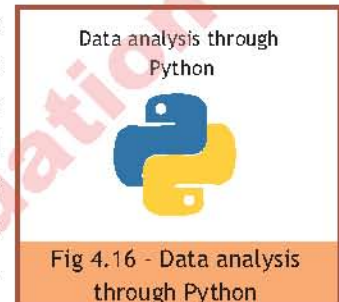


Fig 4.16 - Data analysis through Python

#### Dataset of Tips in a restaurant (Tips.csv)

Tips database is the record of the tip given by the customers in a restaurant for two and a half months in the early 1990s. It contains 6 columns such as total bill, tip, male/female, smoker, day, time, size.



You can download the tips database from the Internet (<https://www.mongodb.com> or <https://www.kaggle.com/datasets/jsphyg/tipping>) To analyze dataset or database in Python you need to install certain libraries. Pandas is the most important library to install for data analysis. To install pandas library the following is the command:

```
pip install pandas
```

To run the Python code you can use google colab as mentioned earlier in this unit. However, this time you need to upload the dataset "tips.csv" to your google drive, so that Python code in colab notebook can access it directly. You can see the dataset in the left pane of the browser in Figure 4.17.

Example Python Code is as following:

```
import pandas as pd
# reading the database
data = pd.read_csv("tips.csv")
# printing the top 10 rows
display(data.head(10))
```



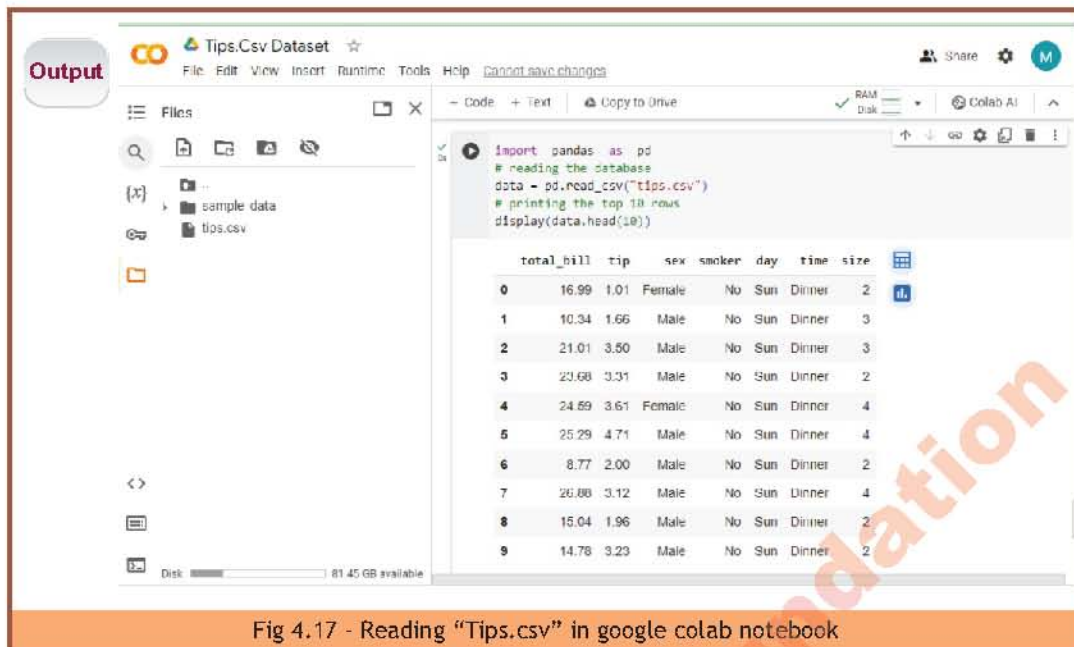


Fig 4.17 - Reading "Tips.csv" in google colab notebook

**Matplotlib:** Matplotlib is an easy-to-use, low-level data visualization library that is built on NumPy arrays. It consists of various plots like scatter plot, line plot, histogram, etc. Matplotlib provides a lot of flexibility. To install this type the below command in the terminal.

**Scatter Plot:** Scatter plots are used to observe relationships between variables and uses dots to represent the relationship between them. The scatter() method in the matplotlib library is used to draw a scatter plot. In the browser window click on + Code button to type new piece of code given below:

1. import pandas as pd
2. import matplotlib.pyplot as plt
3. #reading the database
4. data = pd.read\_csv('tips.csv')
5. #Scatter plot with day against tip
6. plt.scatter(data['day'], data['tip'])
7. #Adding Title to the Plot
8. plt.title("Scatter Plot")
9. #Setting the X and Y labels
10. plt.xlabel('Day')
11. plt.ylabel('Tip')
12. plt.show()

Output

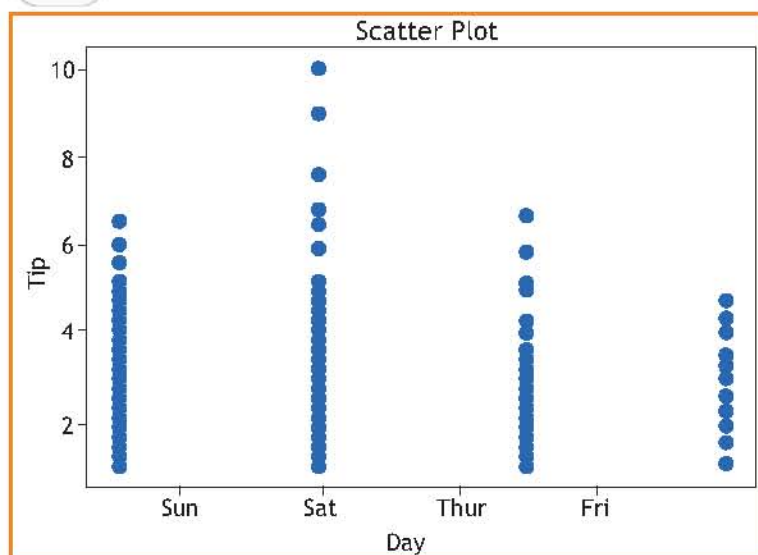


Fig 4.18 - Python Code

**Line Chart:** Line Chart is used to represent a relationship between two data X and Y on a different axis. It is plotted using the plot() function. You can use the following code to make line chart.

```
1. import pandas as pd
2. import matplotlib.pyplot as plt
3. # reading the database
4. data = pd.read_csv("tips.csv")
5. # Scatter plot with day against tip
6. plt.plot(data['tip'])
7. plt.plot(data['size'])
8. # Adding Title to the Plot
9. plt.title("Scatter Plot")
10. # Setting the X and Y labels
11. plt.xlabel('Day')
12. plt.ylabel('Tip')
13. plt.show()
```

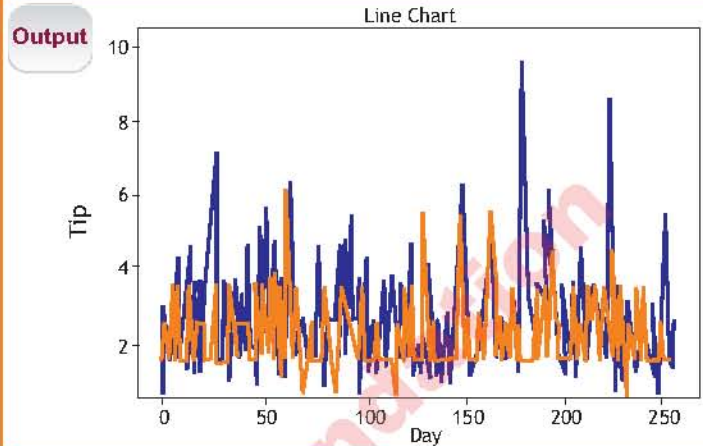


Fig 4.19 - Python Code

**Bar Chart:** A bar plot or bar chart is a graph that represents the category of data with rectangular bars with lengths and heights that is proportional to the values which they represent. It can be created using the bar() method.

```
1. import pandas as pd
2. import matplotlib.pyplot as plt
3. # reading the database
4. data = pd.read_csv("tips.csv")
5. # Bar chart with day against tip
6. plt.bar(data['day'], data['tip'])
7. plt.title("Bar Chart")
8. # Setting the X and Y labels
9. plt.xlabel('Day')
10. plt.ylabel('Tip')
11. # Adding the legends
12. plt.show()
```

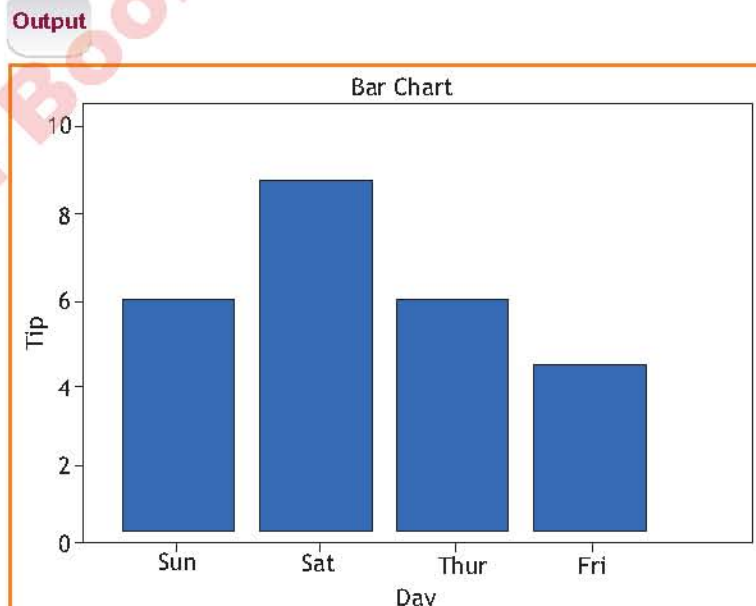


Fig 4.20 - Bar Chart

**Histogram:** A histogram is basically used to represent data in the form of some groups. It is a type of bar plot where the X-axis represents the bin ranges while the Y-axis gives information about frequency. The hist() function is used to compute and create a histogram. In histogram, if we pass



categorical data then it will automatically compute the frequency of that data i.e. how often each value occurred.

```
1. import pandas as pd
2. import matplotlib.pyplot as plt
3. # reading the database
4. data = pd.read_csv("tips.csv")
5. # histogram of total_bills
6. plt.hist(data['total_bill'])
7. plt.title("Histogram")
8. # Adding the legends
9. plt.show()
```

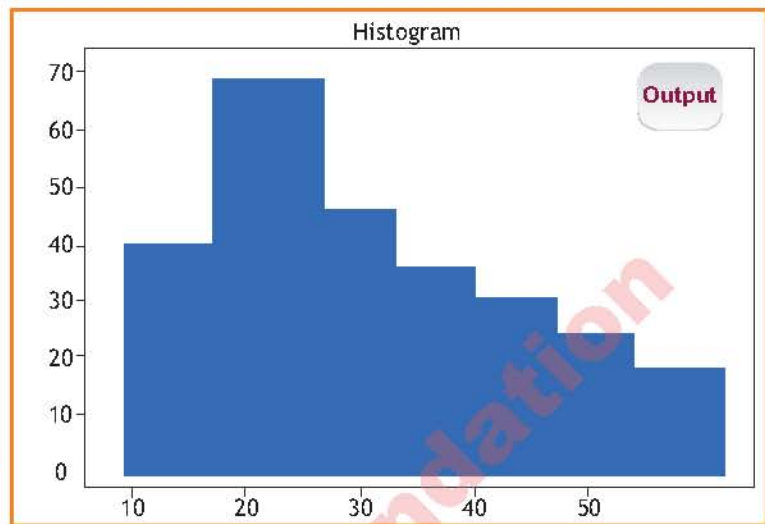


Fig 4.21 - Histogram

**Boxplot:** A boxplot, also known as a box-and-whisker plot, is used to represent distribution of a dataset. It provides a summary of key statistical measures, allowing you to visualize the central tendency, spread, and identify potential outliers in the data. The `boxplot()` function is used to compute a boxplot

```
1. import pandas as pd
2. import matplotlib.pyplot as plt
3. # reading the database
4. data = pd.read_csv("tips.csv")
5. # boxplot of total_bills
6. plt.boxplot(data['total_bill'])
7. plt.title("Boxplot")
8. # Adding the legends
9. plt.show()
```

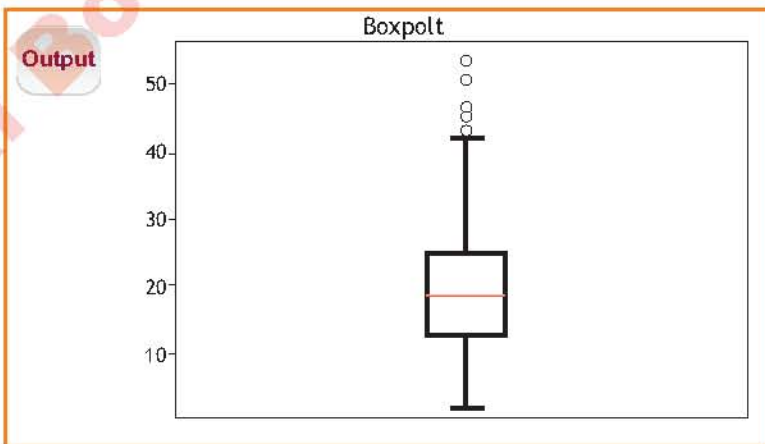


Fig 4.22- Boxplot

**Pie Chart:** A pie chart is a circular statistical chart, which is divided into sectors to illustrate numerical proportion. Given a set of categories or groups with their corresponding values you can make use of the `pie` function from `matplotlib` to create a pie chart in Python. By default, the area of the slices will be calculated as each value divided by the sum of values. With `pyplot`, you can use the `pie()` function to draw pie charts.

```

1. import pandas as pd
2. import matplotlib.pyplot as plt
3. # reading the database
4. data = pd.read_csv("tips.csv")
5. # Pie chart of tips on various days
6. plt.pie(data['day'].value_counts(),
    autopct="%.1f%%", radius=1,
    labels=['Sat', 'Sun', 'Thur', 'Fri'])
7. #Adding the legends
8. plt.title("Pie Chart")
9. plt.show()

```

Output

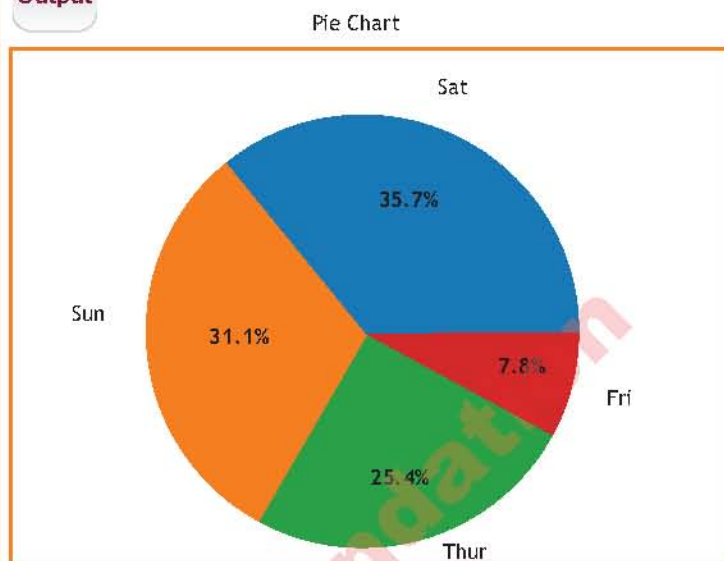


Fig 4.23- Pie Chart

Note: you can display more than one graph in a single instance of code, according to the need of data representation. Here, only one graph in each piece of code is shown for simplicity and better understanding of the students.



### Summary

- **Data analysis:** is the primary step, where data is cleaned, explored, and summarized, providing insights and information for the construction of statistical models to make predictions or draw inferences about real-world phenomena.
- **Statistical modeling:** is the use of mathematical models and statistical assumptions to generate sample data and make predictions about the real world.
- **Statistical model:** is a collection of probability distributions on a set of all possible outcomes of an experiment.
- **Use case:** is a methodology used in system analysis to identify, clarify, and organize a system. A use case is a description of how the users will perform tasks or achieve the goal.
- **Supervised learning model:** uses a labeled dataset for learning, with an answer key to determine accuracy as it trains on the data.
- **Regression model:** is predictive model designed to analyze the relationship between independent and dependent variables.
- **Linear regression model:** is a mathematical equation that allows us to predict a response for a given predictor value.
- **Classification model:** analyzes and classifies a large and complex set of data points. Common models include decision trees, Naive Bayes, the nearest neighbor, random forests, and neural networking models.
- **Unsupervised learning:** is given unlabeled data and attempts to extract features and determine patterns independently.
- **K-means clustering:** combines a specified number of data points into specific groupings based on similarities.
- **Reinforcement learning:** involves training the algorithm to iterate over many attempts using deep learning.
- **Experimental design:** is about precise planning and design to ensure that you have the appropriate data and design for your analysis or studies so that erroneous conclusions can be prevented.
- **Randomization:** is the technique of dividing people into various treatment groups at random. This approach guarantees that prejudice or innate variations between the groups did not affect the outcomes.
- **Control group:** is a group that has not received any treatment, hence a control group is used to compare the treatment group's findings to its outcomes. This approach guarantees that the treatment is the reason that any other factors are to blame for the outcomes.

- Blocking: is grouping the participants according to a particular trait that could have an impact on the results. This principle makes sure that the results aren't affected by group differences.
- Correlation: describes an association between types of variables.
- Causation means that changes in one variable bring about changes in the other variable.
- Population is a complete set of people with a specialized set of characteristics.
- Sample: is a subset of the population.
- Parameter: is number that describes something (usually mean) about the whole population
- Statistics: are number that describes something (usually mean) about the sample or describe properties of sample.
- Mean, MEDIAN AND Mode are different types of averages used to represent typical values in a population.
- Data collection: is the process of collecting, measuring, and analyzing different types of information using a set of standard validated techniques.
- Primary data: refers to data collected from firsthand experience directly from the main source. It refers to data that has never been used in the past. Here are some of the most common primary data collection methods:
  - Interviews: are a direct method of data collection.
  - Observations: is the method, in which researchers observe a situation around them and record the findings.
  - Surveys and Questionnaires: provide a broad perspective from large groups of people.
  - Focus Groups: is similar to an interview, but it is conducted with a group of people who all have something in common.
  - Oral Histories: involve asking questions like interviews and focus groups. However, it is defined more precisely, and the data collected is linked to a single phenomenon.
- Secondary data: refers to data that has already been collected by someone else. It is much more inexpensive and easier to collect than primary data.:
- Internet: is one of the most popular secondary data collection methods in recent times.
- Government Archives: contains a lot of authentic and verifiable data.
- Libraries: are important and authentic sources of information based on different research contexts.
- Data visualization: is the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide an



accessible way to see and understand trends, outliers, and patterns in data.

- **Tips database:** is the record of the tip given by the customers in a restaurant for two and a half months in the early 1990s. It contains 6 columns such as total bill, tip, sex, smoker, day, time, size.
- **Matplotlib:** is an easy-to-use, low-level data visualization library that is built on NumPy arrays. It consists of various plots like scatter plot, line plot, histogram, etc. Matplotlib provides a lot of flexibility.
- **Scatter Plot:** Scatter plots are used to observe relationships between variables and uses dots to represent the relationship between them.
- **Line Chart:** is used to represent a relationship between two data X and Y on a different axis.
- **Bar Chart** is a graph that represents the category of data with rectangular bars with lengths and heights that is proportional to the values which they represent.
- **Histogram:** is used to represent data in the form of groups. It is a type of bar plot where the X-axis represents the binary ranges while the Y-axis gives information about frequency.
- **Boxplot:** is used to represent distribution of a dataset. It provides a summary of key statistical measures, allowing you to visualize the central tendency, spread, and identify potential outliers in the data.

Select the best answer for the following Multiple-Choice Questions (MCQs).

- Select the best answer for the following Multiple-Choice Questions (MCQs).





### Give short answers to the following Short Response Questions (SRQs).

1. List out the parameters and statistics from given statements:
  - a) Average length of height of a giraffe.
  - b) Average weight of watermelon.
  - c) There are 430 doctors in a hospital.
  - d) Average age of students of 6th class in a school is 12 years.
  - e) The number of a basketball team players having height above 6 feet.
2. If you want to make a report regarding the products exported from Pakistan in last five years, how libraries can help you to collect data? Write steps.
3. Make a pie chart of vegetable prices in the market. Consider five to ten vegetables.
4. Enlist steps to represent the monthly temperatures of a Pakistani city in 2023 from January till December using a line graph.



### Give long answers to the following Extended Response Questions (ERQs).

1. Simulate on paper, an experimental design for awareness of food security (Narrative Visualization).
2. Sketch primary data collection methods in context of disease outbreak, like seasonal flu.
3. Argue about the use of statistical modeling techniques. Highlight all techniques discussed in this unit.
4. Compare linear regression and classification. Emphasize on their respective roles in statistical modeling.
5. Defend either of supervised learning and unsupervised learning. Give reasons for your preference to the other.
6. Write a Python code to generate a dataset with two variables where  $y = x^2 + 2x$ . Fit scatter plot and box plot on this data.
7. Relate some real world examples (other than Airbnb, Facebook and YouTube) where data science was used to improve marketing strategies and enhance the business.



#### Activity 1

Practice all the Python code given in this Unit.

**Objective:** To help students develop the skill of data analysis by using Python.

**Outcome:** This activity will help students understand the basic syntax of Python for data analysis.



## Activity 2

### Data collection by interview method:

**Objective:** To help students understand the process of data collection through survey and questionnaires.

- Divide the students into small groups(not more than 3 students in each group).
- Ask the students to collect data by interviewing their fellow students, about their academic performance, study schedule, outdoor games etc.
- Ask them to organize data and plot them by using their favorite tool.
- Supervise them to draw conclusions about correlation among various factors and academic performance.
- Various groups will compare their results/conclusions and find the similarities and differences.

**Outcome:** This activity will help students understand positive or negative correlation among academic and not academic activities. They will also develop the skill of conduction interview as an interviewee as well as interviewer.



## Activity 3

### Build more statistical models on new datasets.

**Objective:** To help students develop the skill of data analysis by using Python and other tools.

- Ask the students to search and download some dataset of their choice from the internet.
- Perform the steps of data cleaning and feature selection.
- Visualize the data and create various graphs according to the nature of dataset. Students can use other tools like MS-Excel, Weka, Pytorch, R studio etc.
- Supervise and help them finding easy and simple tutorials on internet, students can later share small video tutorial through Whatsapp.

**Outcome:** This activity will help students learn new tools at their own.



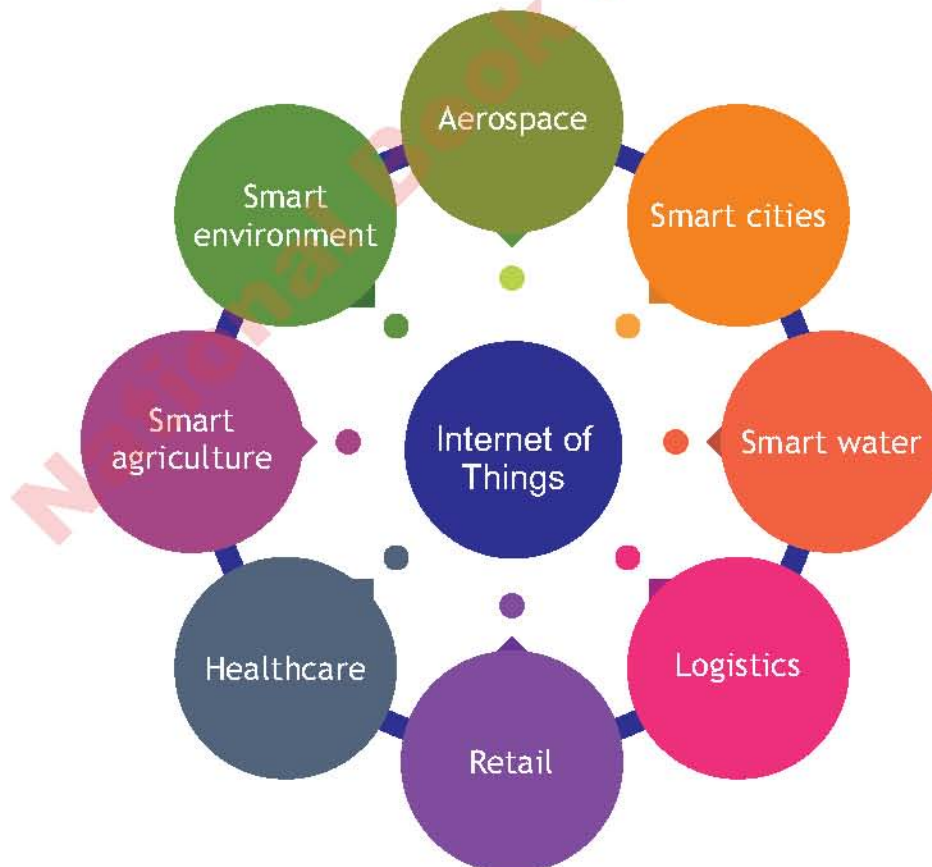
## Unit 05

# APPLICATION OF COMPUTER SCIENCE



After completing this lesson, you will be able to:

- describe technologies that are the foundation of IoT systems, Cloud Computing, and Blockchain.
- evaluate how different stakeholders' culture, values, and (sometimes conflicting) interest affect AI system design.



## UNIT INTRODUCTION

This unit is dedicated to different technologies that have enabled Internet of Things (IoT) and Blockchain applications. It also discusses how different stakeholders' culture, values and (sometimes conflicting) interests affect AI System designs.

### 5.1 Introduction to Internet of Things

Internet of Things (IoT) is made up of two words, that is, Internet and Things. Things here means physical devices such as, smart mobiles, smart watches, smart security systems, medical sensors, home appliances, smart light bulb, smart door bell, smart door lock, smart TV, etc. Internet is through which these devices are connected for communication. These devices are smaller in size, have higher processing power and use longer battery power, due to advancement in technologies. The wireless technologies used for creating IoT networks are Wi-Fi, Bluetooth, Zigbee and NFC (Near Field Communication). Smarthome with IoT devices is shown in Fig.5.1.



Fig 5.1 - Smart home with IoT devices

### 5.2 Technologies that Enabled Internet of Things

The following are the technologies that enabled Internet of Things.

- Wireless Sensor Networks (WSNs)
- Cloud Computing
- Big Data Analytics
- Communication Protocols
- Embedded Systems

#### 5.2.1 Wireless Sensor Network

It is a network of physical devices, appliances, machines and physical objects that can communicate through wireless channels. The devices of a WSN are embedded with sensors and software. Three types of sensors are used to monitor environmental and physical conditions. These are environmental sensors, industrial sensors and motion detection sensors.

**Environmental sensors** are used for measuring and monitoring environmental conditions. These include sensors such as temperature, humidity, air quality, air pressure, wind speed and



#### Teacher's Guide

"How the Internet of Things Works"

<https://www.techtarget.com/iotagenda/definition/Internet-of-Things-IoT>

This Computerworld article enlightens about the technical aspects of IoT.



light sensors as shown in Fig.5.2.



**Industrial sensors** are used for measuring and monitoring of physical quantities in manufacturing process. These include heat, pressure, gas, vibration, viscosity, fluid flow and magnetic field sensors.



**Motion detection sensors** are used to detect movement of humans or objects in its surrounding. These are commonly used to prevent unauthorized movement in restricted areas. These include ultrasonic, proximity, vibration, Passive Infrared (PIR) sensors.



Proximity Sensor



PIR Sensor used for detecting motion



Ultrasonic Sensor for measuring distance

Fig 5.4 - Motion Detection Sensors

Some examples of WSNs used in IoT systems are:

- Surveillance systems
- Weather monitoring systems
- Structural health monitoring systems in civil engineering
- Smart grid electricity network to monitor and manage the transport of electricity

### 5.2.2 Cloud Computing

It is an IT infrastructure that provides data storage, databases, servers, networking and application software services over the Internet.

There are four types of cloud computing models, public cloud, private cloud, community cloud and hybrid cloud as shown in Fig. 5.5.



#### Teacher's Guide

"Cloud Computing for Kids"

<https://cloudacademy.com/>

This resource provides engaging interactive activities and games especially tailored to introduce children the concept of cloud computing in an accessible manner.

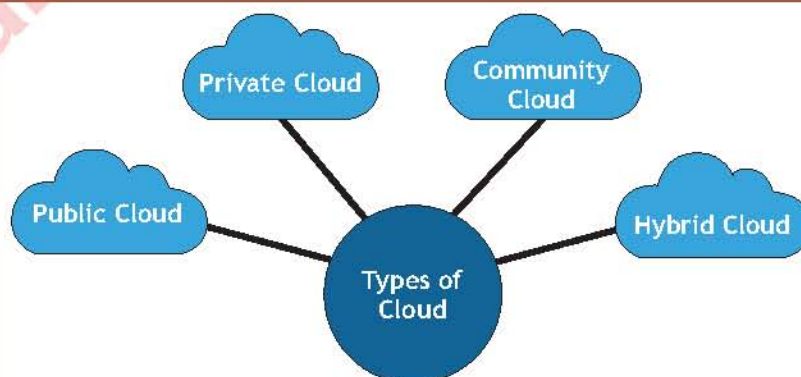


Fig 5.5 - Types of cloud models

#### Public Cloud

In this cloud computing model, resources are owned and operated by a cloud service provider instead of being installed in-house on local devices. The resources that are delivered to



businesses and organizations over the Internet include servers, software and storage. These resources are managed and maintained by the cloud service provider. Public cloud service providers offer security to customers because of sensitive data stored in the cloud. Public cloud provides high speed connectivity to ensure rapid access to applications and data.

It provides services to remote users all over the world. The services can be free or offered on pay-per-usage or on subscription charges. Some examples of cloud computing are Amazon Web Services (AWS), Microsoft Azure and Google Cloud.

### **Private Cloud**

This cloud computing model runs on a privately owned IT infrastructure that is used inside a single organization. In other words, cloud computing is dedicated to a single organization. This computing model requires huge investment as opposed to public cloud computing model. The user organization is responsible for the operation and maintenance of infrastructure (hardware, software, network, etc.). Private cloud solution provides more control over servers and computer network. It offers better performance, reliability and greater flexibility compared to public cloud.

### **Community Cloud**

Community cloud model is a cloud infrastructure that provides access and services to multiple organizations to share information. It is suitable for organizations who have common concerns or needs such as application, security, policy, mission and compliance. It is owned, managed and operated by one or more organizations in the community or a third party. It can be on-site or off-site. It is cost effective as it is shared by many organizations. It is more secure than public cloud but less secure than private cloud.

### **Hybrid Cloud**

Hybrid cloud is a combination of public and private cloud. This cloud computing model uses technology that allows data and applications to be shared between public and private clouds. The organizations use public cloud for performing certain tasks and store their sensitive and critical data and applications on the private cloud for fast access and security reasons.

## **5.2.3 Big Data Analytics**

Big data analytics is concerned with collecting extremely big data from various resources and using analytics software to find out market trends, patterns and insights. It is used to make better business decisions and design smart solutions to provide better customer services and generate more revenue.

Huge volume of data is created in our daily life from various digital resources. These include banking transactions, customer databases, medical records, emails, mobile apps, etc. which cannot be handled by traditional databases. The data gathered by IoT devices is transformed into actionable information by big data analytics to improve decision-making.

The following are some areas of big data analytics.

- Healthcare industry
- Transport industry
- Banking and financial services
- Weather patterns and trends
- Insurance industry
- Digital marketing
- Media and entertainment
- Business insights
- Government sector

### 5.2.4 Communication Protocols

IoT communication protocol is a set of rules that govern how data is transferred and understood between physical devices, sensors and the end users. Protocols provide exchange of information and methods of connection between IoT devices and end users. It is needed for proper functioning of IoT-enabled devices.

### 5.2.5 Embedded Systems

Embedded systems are small electronic devices which are combination of hardware and firmware. Firmware is software that is permanently installed in a device or electronic machine. Embedded systems have a microcontroller or a microprocessor that controls the operation of embedded systems through firmware. Embedded system is designed to perform a specific task. Some embedded system are shown in Fig.5.6.

Embedded systems have various applications in IoT such as they can monitor a room's temperature and automatically adjust the thermostat. Embedded systems are used in a large variety of devices. These include medical equipment, home appliances, point-of-sale terminals, vending machines, ticket scanners, automotive devices and industrial equipment. Embedded systems help in implementing various IoT solutions.

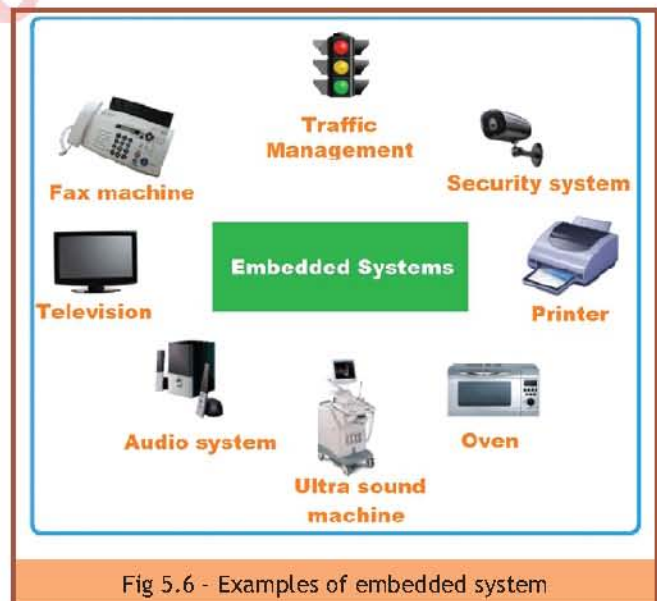


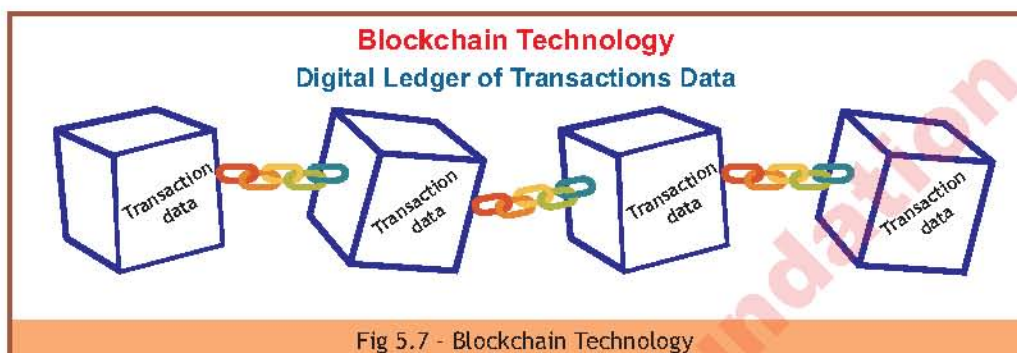
Fig 5.6 - Examples of embedded system



## 5.3 Blockchain

### 5.3.1 Introduction to Blockchain

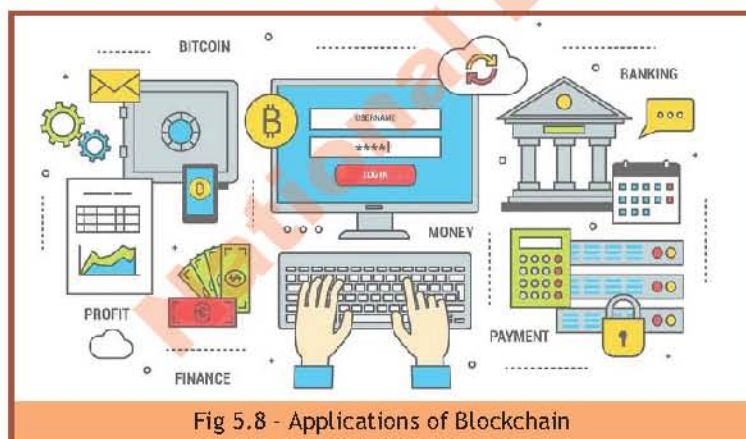
A Blockchain is a digital ledger (database) of transactions that is shared and maintained by network users. The ledger is a list of continuously growing records, known as blocks. The blocks are chained together as shown in Fig.5.7, using cryptography. Cryptography ensures that the transaction data is secure and immutable. It cannot be hacked or altered.



### 5.4 Technologies that Enabled Blockchain

There are three technologies that enabled Blockchain.

- Cryptography
- Blockchain networks
- Transaction process on the network



#### Teacher's Guide

"Blockchain for Kids: A Gentle Introduction"

<https://coinmarketcap.com/currencies/save-the-kids/>

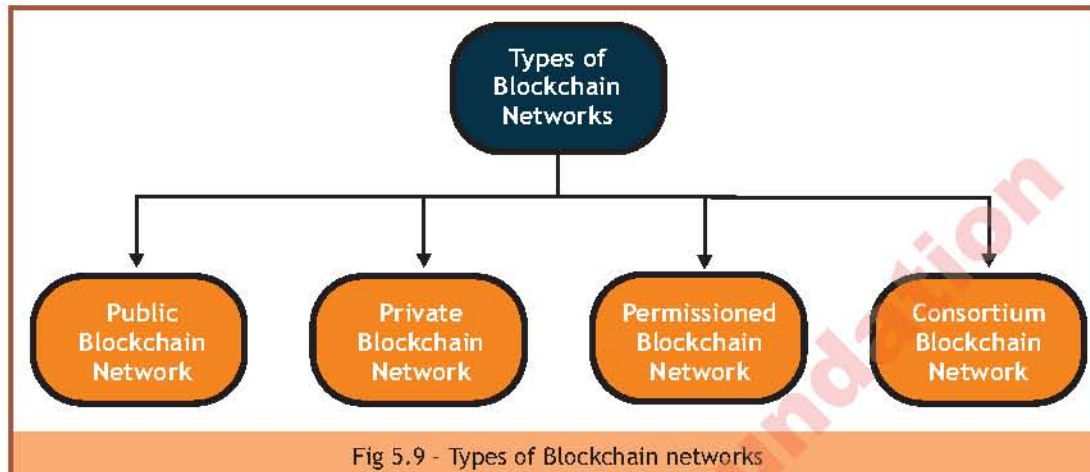
CoinMarketCap utilizes a narrative-driven approach in this resource to elucidate blockchain to younger audiences, introducing essential concepts in an enjoyable and captivating manner.

#### 5.4.1 Cryptography

Cryptography is technology that protects information and communications by coding it in such a way that only those for whom it is intended can read and process it. Cryptography provides confidentiality, integrity and authentication. The topic of cryptography is explained in Unit 1.

## 5.4.2 Blockchain Networks

There are different ways of building Blockchain networks. These are public, private, permissioned and consortium networks as shown in Fig.5.9. The purpose of all these networks is to share the ledger among all the participants using Blockchain.



### Public Blockchain Network

Public Blockchain Network is one that anyone can join in the world. It is an open platform for the public in which data is distributed across all the participants. It is adapted by many organizations as it is open to public. It requires substantial computing power and security is weak.

Public Blockchain network is used for exchanging cryptocurrency such as Bitcoin. Cryptocurrency is a type of digital currency used by people to make direct payments to each other through an online system without the need for a bank.

### Private Blockchain Network

Private Blockchain Network is managed by a single organization or private business. Public access to private Blockchain network is restricted. Organizations or private businesses can customize user access and authorization as it is administered by a single entity. It is more secure compared to public Blockchain network.

Private Blockchain networks are often used by businesses, organizations, and government agencies to implement secure and transparent record-keeping systems.

### Permissioned Blockchain Network

Permissioned Blockchain Network is generally set up by organizations who have built a private Blockchain network. In this network, only authorized individuals are allowed access and can perform or view transactions. In this case, anyone who wants to join needs permission. It is a safe and secure network and has advantage of better transparency.

Permissioned Blockchain networks are used for a variety of applications like supply chain management, trade finance and clearing and settlement of financial assets.



## Consortium Blockchain Network

Consortium Blockchain Networks are permissioned Blockchain networks in which multiple organizations are responsible for managing a single consortium Blockchain network. It is more complex to set up because it requires collaboration between several organizations or businesses. It is safe and secure network.

Consortium Blockchain networks are suitable for finance and banking, insurance and asset trading. Banks also use it to form a group that collects and stores needed information on all debtors in a database.

### 5.4.3 Transaction Process on the Network

The Blockchain technology allows individuals to directly make deals with each other without involvement of intermediaries such as bank or government. Each transaction is verified by the participants in a peer-to-peer network. When a deal is made between two participants, the transaction is permanent. It is added to the ledger and is irreversible. Blockchain facilitates faster transactions than traditional databases as there are no intermediaries. Blockchain uses decentralization of data. The data is stored in millions of computers all over the world that are connected to the Blockchain.

## 5.5 Integration of Blockchain and IoT

Unimaginable number of IoT devices are interconnected all over the world. These devices collect and exchange vast amount of data through sensors which is vulnerable to Cyber-attacks. Therefore, it is crucial to take security measures against Cyber-attacks such as hacking and breaches.

The decentralized ledger systems of Blockchain technology provides transparency and immutable recording of transactions. Blocks containing information about transactions linked together through cryptography store and transmit transactions in a more secure format. Blockchain technology does not allow transaction data to be altered.

The integration of IoT and Blockchain technologies allow to enhance the overall performance and reliability in managing the IoT networks. Moreover, the merger of IoT with Blockchain provides faster communication between the devices as it does not require intermediaries.

Integration of IoT and Blockchain technologies are providing security and enhancing connectivity in many areas including supply chain, energy grid, healthcare, automotive sector, agriculture, and water management as show in Fig. 5.10.

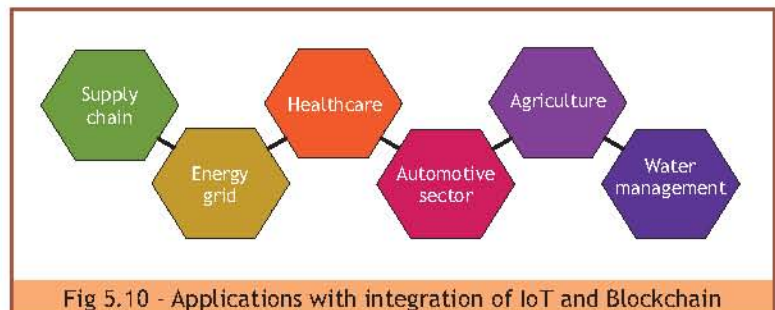


Fig 5.10 - Applications with integration of IoT and Blockchain

## 5.6 Stakeholders Interests in AI Systems

Artificial Intelligence (AI) is a branch of computer science to make computer-controlled machines that can intelligently perform human-like tasks. Using AI-based machines is like fixing an intelligent brain in the machines that has decision-making capabilities to solve complex problems.

### 5.6.1 Stakeholders in AI Systems

Stakeholder is an individual or a group who are affected by AI system or have interest in its social impact. AI is transforming the world in various areas but it also poses ethical, social and legal challenges that affect the stakeholders. Stakeholders may be end users, developers, customers, suppliers, employees, regulators, funders, owners, policy makers and the community.

### 5.6.2 Positive and Negative Impacts of AI Systems

The decision-making capabilities of AI are replacing human-decision making across all the industries. AI is like we have installed an intelligent brain in the computer that can make intelligent decisions to solve complex problems. However, there is no guarantee that all the predictions made by the AI algorithms will be correct as these are based on data pattern and machine learning. Therefore, we can say that AI systems have both positive and negative impact.

#### Positive Impact of AI Systems

##### ● Unbiased Decisions

Decisions made by AI are based on reason and logic rather than driven by emotions. Therefore, it does not have biased views. For example, AI-based recruitment system will hire job applicants based on their skills and qualification, not ethnicity or demographics.

##### ● Around the Clock Availability

AI systems can work 24x7 without any breaks with high efficiency and accuracy which is not possible for human being. For example, online customer support Chatbot can provide assistance to customers anytime around the clock by answering their common questions.

##### ● Zero Risk

AI robots can perform many tasks to overcome risks in hazardous environments such as defusing a bomb or in manufacturing facility.

##### ● Reduction in Human Error

Properly programmed AI systems reduce human errors and provide high accuracy in performing various tasks. For example, robotic surgery systems can perform complex operations with precision ensuring patient's safety and healthcare.

#### Negative Impact of AI Systems

##### ● Less Creativity and Emotions

AI systems cannot be creative in decision-making. AI does not have the ability to experience



emotions or personal connections. AI relies on algorithms and data inputs to make decision, while human creativity is influenced by experience, emotions and imagination. Humans have the ability to generate ideas for innovation and progress which is not possible with AI.

#### ● **Lack of Employment**

Robots are being utilized in various industries to replace humans which may increase unemployment. For example, robots are replacing humans in manufacturing industry. Robots are also replacing humans in hotels by carrying out tasks such as cleaning, restocking bathrooms supplies, checking rooms and luggage transportation.

#### ● **Ethical and Moral Concerns**

It is difficult to incorporate ethics and morality into AI systems. People are concerned that one day, AI may grow uncontrollably and eventually destroy humanity.

#### ● **High Cost**

Creating machines that simulate human intelligence is very costly as it requires plenty of time, resources and also latest hardware and software.

### **5.6.3 Conflicting Requirements of Stakeholders in the Development of AI Systems**

AI application developers have to consider and manage many conflicting requirements of stakeholders to avoid failure of the resulting software. Stakeholders of an AI application of a specific area, such as, healthcare, education, supply chain, marketing, etc., may have many diverse expectation, desires and goals in conflicting situations. Conflicts among the stakeholders should be resolved, not suppressed. Conflicts in requirements between different stakeholders are unavoidable. They express their views or disagreements on some decisions or values proposed in an AI application. Different stakeholders may have similar needs but may differ on how to implement them. Conflicts can be due to different interpretation or understanding of the given problem. Conflicting requirements of stakeholders make the developers' task of AI application design more difficult as they have to satisfy all the stakeholders.

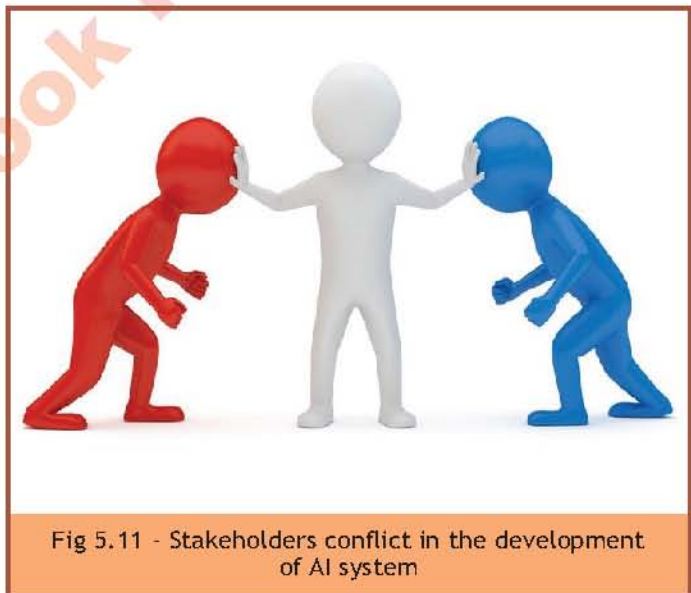


Fig 5.11 - Stakeholders conflict in the development of AI system

### Summary

- Internet of Things (IoT) is made up of two words, that is, Internet and Things. Things here means physical devices such as, smart mobiles, smart watches, smart security systems, medical sensors, home appliances, smart light bulb, smart door bell, smart door lock, smart TV, etc. Internet is through which these devices are connected for communication.
- Wireless Sensor Network (WSN) is a network of physical devices, appliances, machines and physical objects that can communicate through wireless channels. The devices of a WSN are embedded with sensors and software.
- Environmental Sensors are used for measuring and monitoring environmental conditions. These include sensors such as temperature, humidity, air quality, air pressure, wind speed and light sensors.
- Industrial Sensors are used for measuring and monitoring of physical quantities in manufacturing process. These include heat, pressure, gas, vibration, viscosity, fluid flow and magnetic field sensors.
- Motion Detection Sensors are used to detect movement of humans or objects in its surrounding. These are commonly used to prevent unauthorized movement in restricted areas. These include ultrasonic, proximity, vibration, Passive Infrared (PIR) sensors.
- Cloud Computing is an IT infrastructure that provides data storage, databases, servers, networking and application software services over the Internet.
- In Public Cloud computing model, resources are owned and operated by a cloud service provider instead of being installed in-house on local devices. The resources that are delivered to businesses and organizations over the Internet include servers, software and storage. These resources are managed and maintained by the cloud service provider.
- Private Cloud computing model runs on a privately owned IT infrastructure that is used inside a single organization. In other words, cloud computing is dedicated to a single organization. This computing model requires huge investment as opposed to public cloud computing model. The user organization is responsible for the operation and maintenance of infrastructure (hardware, software, network, etc.).
- Community Cloud model is a cloud infrastructure that provides access and services to multiple organizations to share information. It is suitable for organizations who have common concerns or needs such as application, security, policy, mission and compliance considerations. It is owned, managed and operated by one or more organizations in the community or a third party.
- Hybrid Cloud is a combination of public and private cloud. This cloud computing model



uses technology that allows data and applications to be shared between public and private clouds. Hybrid cloud is a combination of public and private cloud. This cloud computing model uses technology that allows data and applications to be shared between public and private clouds.

- Big Data Analytics is concerned with collecting extremely big data from various resources and using analytics software to find out market trends, patterns and insights.
- IoT Communication Protocol is a set of rules that govern how data is transferred and understood between physical devices, sensors and the end users. Protocols provide exchange of information and methods of connection between IoT devices and end users.
- Embedded Systems are small electronic devices which are combination of hardware and firmware. Firmware is software that is permanently installed in a device or electronic machine. Embedded systems have a microcontroller or a microprocessor that controls the operation of embedded system through firmware.
- A Blockchain is a digital ledger (database) of transactions that is shared and maintained by network users. The ledger is a list of continuously growing records, known as blocks. The blocks are chained together using cryptography. Cryptography ensures that the transaction data is secure and immutable.
- There are different ways of building Blockchain networks. These are public, private, permissioned and consortium networks. The purpose of all these networks is to share the ledger among all the participants using Blockchain.
- Public Blockchain Network is one that anyone can join in the world. It is an open platform for the public in which data is distributed across all the participants. It is adapted by many organizations as it is open to public.
- Private Blockchain Network is managed by a single organization or private business. Public access to Blockchain network is restricted. Organizations or private businesses can customize user access and authorization as it is administered by a single entity.
- Permissioned Blockchain Network is generally set up by organizations who have built a private Blockchain network. In this network, only authorized individuals are allowed access and can perform or view transactions.
- Consortium Blockchain Networks are permissioned Blockchain networks in which multiple organizations are responsible for managing a single consortium Blockchain network. It is more complex to set up because it requires collaboration between several organizations or businesses.
- Cryptography is technology that protects information and communications by coding it in such a way that only those for whom it is intended can read and process it. Cryptography provides confidentiality, integrity and authentication.



### Exercise

Select the best answer for the following Multiple-Choice Questions (MCQs).

1. Fluid flow sensor is a type of:
  - a) Environmental sensor
  - b) Industrial sensor
  - c) Motion detection sensor
  - d) Climate control sensor
2. The cloud model that provides services to multiple organizations is known as:
  - a) Public cloud
  - b) Private cloud
  - c) Community cloud
  - d) Hybrid cloud
3. The cloud model that provides best security of data is:
  - a) Public cloud
  - b) Private cloud
  - c) Community cloud
  - d) Hybrid cloud
4. Exchange of information between IoT devices and end users is provided by:
  - a) Cloud computing
  - b) Big data analytics
  - c) Embedded systems
  - d) Communication protocols
5. The Blockchain network that is most difficult to set up is:
  - a) Public Blockchain network
  - b) Private Blockchain network
  - c) Permissioned Blockchain network
  - d) Consortium Blockchain network
6. The Blockchain network that is open for everyone to join is called:
  - a) Public Blockchain network
  - b) Private Blockchain network
  - c) Permissioned Blockchain network
  - d) Consortium Blockchain network
7. The sensors used to prevent unauthorized movement in restricted areas are called:
  - a) Environment sensors
  - b) Industrial sensors
  - c) Motion detection sensors
  - d) Restricted area sensors
8. Sensors used for measuring humidity, air quality, air pressure, temperature and wind speed are called:
  - a) Environment sensors
  - b) Industrial sensors
  - c) Motion detection sensors
  - d) Measurement sensors
9. Software that is permanently installed in an electronic machine is known as:
  - a) System software
  - b) Application software
  - c) Machine software
  - d) Firmware
10. A digital ledger of transactions shared and maintained by network users is called:
  - a) Database
  - b) Accounting software
  - c) Blockchain
  - d) Network ledger





### Give short answers to the following Short Response Questions (SRQs).

1. How IoT can enhance our daily life?
2. Provide 3 examples of WSNs used in IoT system.
3. Differentiate between public cloud model and private cloud models.
4. What is Blockchain? Why data stored in a Blockchain is secure?
5. Why integration of Blockchain and IoT is beneficial?
6. Define permissioned Blockchain network.



### Give long answers to the following Extended Response Questions (ERQs).

1. What is Blockchain technology? Describe in detail how transactions are processed using Blockchain technology.
2. Briefly explain the role of following technologies that enabled IoT.  
a. Cloud computing   b. Communication protocols   c. Embedded systems
3. Criticize the negative impacts of AI systems in the domain of education and learning of students.
4. Examine the reasons behind the conflicting requirements among stakeholders during the development of AI systems.
5. Consider creating a cutting-edge system for language learning. The priorities of teachers, learners, and programmers will all differ. How would it help to make the new language learning system better by incorporating the varying priorities? How can AI be added to it?



#### Activity 1

Take an example of an IoT application and identify how each component of connectivity, processing, AI, cloud computing, and battery power enables that application.



#### Activity 2

Have the students work through a Blockchain application where the students act like Blockchain users and maintain a ledger through their peer-to-peer network in class.



#### Activity 3

Have a discussion in class where you divide the class into groups. Each group is further divided into three roles where one role represents advertisers that are trying to sell their products on a social media platform, the second role is using social media to connect with friends, and the third role is the developers who is trying to sign up more users and trying to get the existing users to spend more time on their application. Help the discussion on how each role has different expectations from the AI algorithms being used in the application.

**National Book Foundation**



Approved by National Curriculum Council, Secretariat  
Ministry of Federal Education & Professional Training  
vide letter No. F.1-1 (2024)-NCC/DEA/Dir/English, Dated: 04th March 2024

National Book Foundation



National Book Foundation  
as  
Federal Textbook Board  
Islamabad

