

Test Edition



**Based on New Sindh Curriculum of Computer Science 2024
Aligned with National Curriculum of Pakistan 2022-23**

The Textbook of
**Computer
Science**

For Grade XI

Sindh Textbook Board

Test Edition



The Textbook of
**Computer
Science**

For Grade **XI**



Sindh Textbook Board

All rights are reserved with the **SINDH TEXTBOOK, BOARD, JAMSHORO.**

Prepared by **SINDH TEXTBOOK, BOARD, JAMSHORO.**

Reviewed by **Directorate of Curriculum Assessment and Research Sindh, Jamshoro**

Prescribed by the Board of Intermediate and Secondary Education Hyderabad, Sukkur, Shaheed Benazirabad, Larkana, Mirpurkhas and Karachi for Secondary School Examination.

Approved by the **School Education and Literacy Department, Government of Sindh.**

Letter No. SELD/HCW/18/2018: Dated 22nd July, 2026 for the province of Sindh.

Patron in Chief

Parvaiz Ahmed Baloch
Chairman, Sindh Textbook Board.

Supervisor

Moona Aziz Pathan
Sindh Textbook Board.

Authors

- ◆ **Dr. Muhammad Ahsan Ansari**
- ◆ **Mr. Sajid Ahmed Qazi**
- ◆ **Mr. Muhammad Arif Rao**
- ◆ **Ms. Zufishan Shaikh**
- ◆ **Mr. Syed Hassan Raza Zaidi**
- ◆ **Mr. Touqeer Jalal**
- ◆ **Engr. Mahaveer Rathi**

Reviewers

- ◆ **Dr. Naeem Ahmed Mahoto**
- ◆ **Dr. Riaz Ahmed Shaikh**
- ◆ **Mr. Muhammad Zahid Shaikh**
- ◆ **Mr. Ajmal Saeed**
- ◆ **Ms. Shahnaila Daudpota**
- ◆ **Mr. Abdul Ghafoor Brohi**
- ◆ **Mr. Imran Pathan**
- ◆ **Mr. Abdul Ghaffar Keerio**
- ◆ **Mr. Asad Ali Zubedi**
- ◆ **Mr. Mushtaque Ahmed Ansari**
- ◆ **Ms. Moona Aziz Pathan**

Designing & Layout

Rumisha Graphics

Printed at:

PREFACE

We are living in an age where Science and Technology shape nearly every dimension of human life. From the way we communicate to the way we learn, work, and solve problems, technology has become deeply woven into our daily existence and Computer Science stands at the very heart of this transformation.

This is the first time that Computer Science textbooks for Grade XI and Grade XII are being published separately, allowing each book to be developed with greater focus and depth suited to its respective grade level. These textbooks have also been aligned with the National Curriculum of Computer Science 2023 developed by National Curriculum Council (NCC), ensuring that students across the country learn from a consistent, nationally recognized standard. As updated textbooks, they provide students with new and relevant content, keeping pace with the rapid and multidirectional developments taking place across all branches of science.

This Computer Science textbook for Grade XI has been developed in line with the revised curriculum and reviewed by an independent team from the Directorate of Curriculum Assessment and Research, Jamshoro. Recognizing the growing significance of Computer Science in today's world, its content has been thoughtfully restructured to meet the evolving needs of learners.

Among the new additions in this book are Introductory Paragraphs, Information Boxes, Summaries, a Glossary of Key Terms, and a dedicated Career Education section, alongside a wide range of practical exercises. A Teaching Guide section has also been introduced for the first time, offering teachers structured support to deliver lessons more effectively. These features have been incorporated to spark curiosity, deepen understanding, and guide students toward future academic and professional pathways, while also enhancing the overall usefulness of the book for both students and teachers.

The Sindh Textbook Board has invested considerable effort and resources into the publication of this textbook. The authors have worked diligently to present concepts with clarity and effectiveness. Nonetheless, some gaps or oversights may remain, teachers and readers are warmly encouraged to share their feedback, corrections, and suggestions to help improve future editions.

In closing, I extend my sincere gratitude to our Authors, Editors, Reviewers, and Subject Specialists for their unwavering dedication to this project, as well as to the entire team and everyone who contributed to bringing these textbooks to life

Chairman
Sindh Textbook Board

CONTENTS

Unit No.	Unit Name	Page No.
1	Computer Systems	01 - 70
2	Computational Thinking & Algorithm	71 - 90
3	Programming Fundamentals	91 - 138
4	Data and Analysis	139 - 173
5	Application and Impacts of Computing	174 - 198
6	Digital Literacy	199 - 218
	Glossary	219 - 225
	Teaching Guide	226 - 234

Unit

1

Computer Systems

Student Learning Outcomes:

After the completion of this unit students will be able to:

- ◆ Differentiate between analog and digital data representation.
- ◆ Understand and apply logic gates in digital systems.
- ◆ Construct truth tables using Boolean operators.
- ◆ Build logic circuits from Boolean functions and logic diagrams.
- ◆ Use the given simulation tool (LogiSim) for logic gate representation.
- ◆ Identify and understand stages of the software development life cycle (analysis, design, coding and testing etc.).
- ◆ Software development methodologies.
- ◆ Understand the importance of SDLC.
- ◆ Understand water fall model with case study.
- ◆ Understand agile model with case study.
- ◆ Understand and explain the communication models.
- ◆ Explain 7 layered OSI model and role of each layer.
- ◆ Identify the purpose of protocols at each layer.
- ◆ Understand TCP/IP model.

Introduction to Unit:

This unit introduces three important areas of computing. First, how computers store and process information using simple ON and OFF signals, called digital data, and how basic building blocks called logic gates are used to perform operations. Students will use a tool “LogiSim” for building logic circuits. Second, we will discover how software is created step by step through a process called the Software Development Life Cycle (SDLC), and we will compare two popular approaches - the Waterfall model and the Agile model using real examples. Finally, we will learn how computers and devices communicate with each other over a network, including a well-known seven-layer model called the OSI model and the TCP/IP model that powers the internet. By the end of this unit, we will have a clear understanding of how computers think, how software is built, and how devices talk to each other.

1.1.1 Discrete and Continuous Quantities:

In our daily life, we deal with two types of quantities: **discrete** and **continuous**.

Discrete Quantities are separate, countable items with specific values. We can count them one by one. Examples include:

- Number of students in a classroom
- Number of books on a shelf
- Number of cars in a parking lot
- Number of fingers on our hand

Continuous quantities can take any value within a range. They flow smoothly without breaks. Examples include:

- Temperature of water
- Height of a person
- Time passing on a clock
- Speed of a moving car

Simple Analogy: Stairs vs Ramp

Think about climbing stairs versus walking up a ramp:

Table 1.1: Analogy of Discrete vs Continuous behavior

Stairs (Discrete)	Ramp (Continuous)
We step on specific heights only	We can stop at any height
We cannot stand between two steps	We can stand anywhere
Values are fixed (step 1, step 2, step 3)	Values can be anything (1.5m, 1.73m, 2.1m)
Similar to digital signals	Similar to analog signals

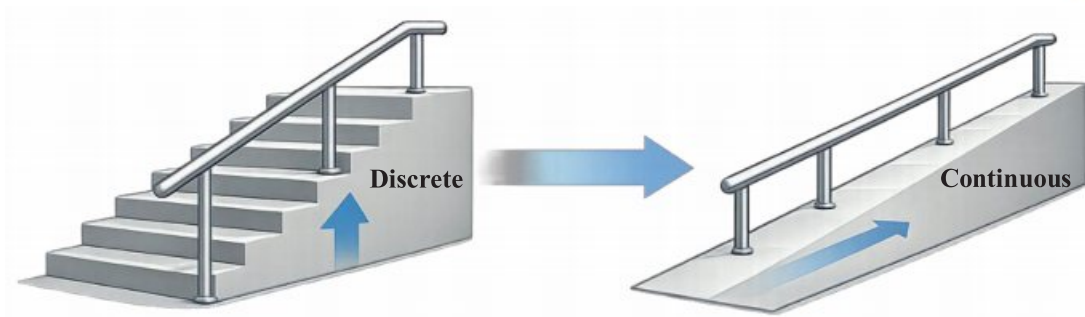


Figure 1.1: Stairs and Ramp Comparison

1.1.2 Introduction to Digital Systems:

A **digital system** is an electronic system that uses only two values: **0** and **1**. These two values are called **binary digits** or **bits**.

- **0** represents OFF, LOW, or FALSE
- **1** represents ON, HIGH, or TRUE

Why Use Digital Systems?

Digital systems are used in almost all modern electronic devices because they are:

- **Reliable:** Less affected by noise and interference
- **Accurate:** No loss of quality when copying data
- **Easy to design:** Only two states to handle
- **Programmable:** Can be easily controlled by software

Table 1.2: Examples of Digital Systems

Device	How it uses digital signals
Computer	Processes data as 0s and 1s
Smartphone	Stores photos, music as binary data
Digital watch	Displays time using electronic circuits
Calculator	Performs math using logic gates
Traffic light	Switches between states (red, yellow, green)
Digital camera	Captures images as binary data

1.1.3 Analog and Digital Signals:

Signal:

A signal is a function of one or more variables that indicate some (usually physical) phenomenon. Signals are carriers of information. They can convey different types of information depending on the application. Signals can be of different forms.

Analog Signals:

Analog signals are continuous signals that can take any value within a range. They change smoothly over time, like a wave.

Examples of analog signals:

- Human voice
- Music from a guitar
- Temperature changes
- Radio waves
- Voltage in old telephones

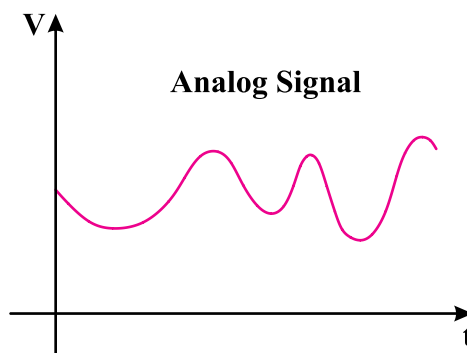


Figure 1.2: Analog Signal

Digital Signals:

Digital signals have only two levels: HIGH (1) and LOW (0). They jump between these two values instantly, creating a square wave pattern.

Examples of digital signals:

- Data in computers
- USB communication
- HDMI video signals
- Data on the internet

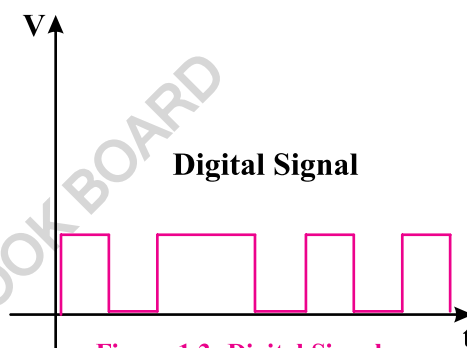


Figure 1.3: Digital Signal
(Include a square wave graph showing only HIGH and LOW levels)

Table 1.3: Comparison of Analog and Digital Signal

Feature	Analog Data Representation	Digital Data Representation
Nature of Data	Continuous: Data flows smoothly and can take any value.	Discrete: Data consists of separate, countable values.
Basic Unit	Represented by physical quantities (like voltage or frequency).	Represented by Binary Digits (Bits) : 0 and 1.
Representation Form	Uses a Smooth Wave pattern (Sine wave).	Uses a Square Wave pattern (Steps).
Range of Values	Infinite: There are no fixed steps between values.	Fixed: Only two levels exist (HIGH/LOW).
Reliability	Low: Easily affected by “noise” or interference.	High: Very resistant to noise and interference.
Common Examples	Human voice, mercury thermometers, vinyl records.	Computers, smartphones, digital watches.

1.1.4 Boolean Algebra:

Boolean algebra is a branch of mathematics that deals with only two values: **TRUE (1)** and **FALSE (0)**. It was developed by **George Boole**, an English mathematician, in the 1850s. Boolean algebra is the foundation of all digital electronics and computer programming. It is a form of symbolic logic that shows how logic operations are performed. A Boolean expression represents the outcome when logic operations are applied to variables.

Table 1.4: Key Concepts in Boolean Algebra

Concept	Description	Example
Variables	Letters that hold values 0 or 1	A, B, C, X, Y
Constants	Fixed values	0 or 1
Operations	Actions performed on variables	AND, OR, NOT
Expressions	Combinations of variables and operations	A AND B, A OR B

1.1.5 Boolean Operations:

There are three basic Boolean operations:

1. AND Operation ($\cdot$):

The AND operation gives output 1 **only when ALL inputs are 1**.

Symbol: $\cdot$ (dot) or $\wedge$ (wedge)

Expression: $Y = A \cdot B$ or $Y = A \text{ AND } B$

Real-life example: A door with two locks. The door opens only when BOTH locks are unlocked.

Truth Table:

Input		Output
A	B	$Y = A \cdot B$
0	0	0
0	1	0
1	0	0
1	1	1

2. OR Operation (+):

The OR operation gives output 1 **when ANY input is 1**.

Symbol: + (plus) or $\vee$ (vee)

Expression: $Y = A + B$ or $Y = A \text{ OR } B$

Real-life example: A room with two doors. We can enter if EITHER door is open.

Truth Table:

Input		Output
A	B	$Y = A + B$
0	0	0
0	1	1
1	0	1
1	1	1

3. NOT Operation ($\neg$ or '):

The NOT operation **inverts** the input. If input is 1, output is 0. If input is 0, output is 1. NOT operation is a unary operation, meaning that it operates on only one operand.

Symbol: $\neg$ (negation sign) or ' (prime/apostrophe) or $\bar{}$ (bar)

Expression: $Y = A'$ or $Y = \text{NOT } A$ or $Y = \bar{A}$ or $Y = \neg A$

Real-life example: A light switch. When the switch is ON, the room is lit. When OFF, the room is dark.

Truth Table:

Input	Output
A	$Y = A'$
0	1
1	0

1.1.6 Truth Tables:

A **truth table** shows all possible input combinations and their corresponding outputs for a logic gate or circuit.

How to create a truth table:

- I. Count the number of inputs (n)
- II. Calculate total rows = 2^n
- III. List all input combinations
- IV. Calculate output for each combination

Example: For 2 inputs, rows = $2^2 = 4$
 : For 3 inputs, rows = $2^3 = 8$

Example of a 2-Input Template:

When building a table, we always list inputs in binary order to ensure we do not miss any combination.

Combination	Input		Output
#	A	B	Logical result
1	0	0	Result would be 0 (zero) or 1 (one) depending on the values of the inputs and the operation performed
2	0	1	
3	1	0	
4	1	1	

1.1.7 Logic Gates:

Logic gates are electronic circuits that perform Boolean operations. They are the building blocks of all digital systems.

Each gate has:

- One or more **inputs**
- Only **one output**
- A specific **function**

Logic Gate Symbols:

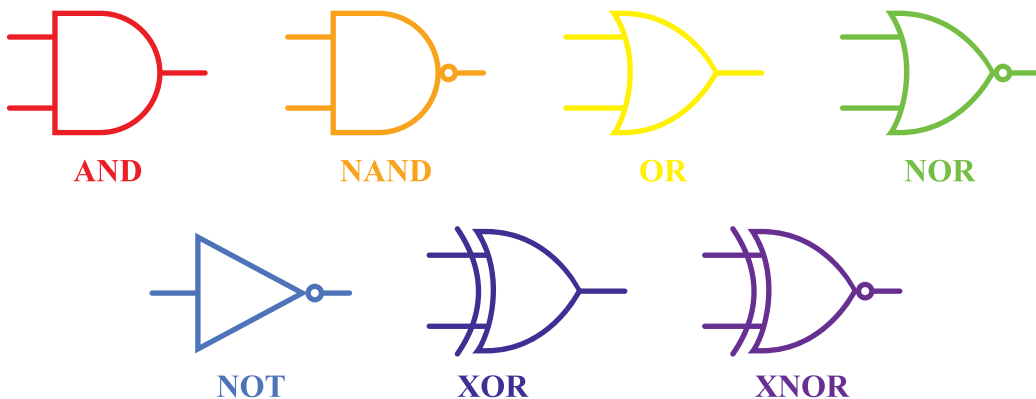


Figure 1.4: Logic Gate Symbols

1. AND Gate:

The **AND gate** produces output 1 only when **all inputs are 1**.

Symbol:



Figure 1.5: AND Gate Symbol

Boolean Expression: $Y = A \cdot B$

Note: Truth Table of 2-input AND gate is same as 2-input AND operation given in Section 1.1.5

Truth Table (3-input AND gate):

Input			Output
A	B	C	$Y = A \cdot B \cdot C$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Real-life example: A car starts only when the key is inserted AND the driver presses the start button.

2. OR Gate:

The **OR gate** produces output 1 when **any input is 1**, will produce output 0 only when **all the inputs are 0**.

Symbol:

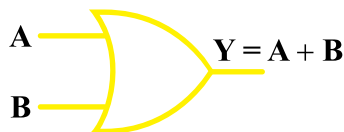


Figure 1.6: OR Gate Symbol

Boolean Expression: $Y = A + B$

Note: Truth Table of 2-input OR gate is same as 2-input OR operation given in Section 1.1.5.

Truth Table (3-input OR gate):

Input			Output
A	B	C	$Y = A + B + C$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Real-life example: An alarm sounds if the front door OR the back door is opened.

3. NOT Gate (Inverter):

The **NOT gate** inverts/ complements the input. It is also called an **inverter**.

Symbol:

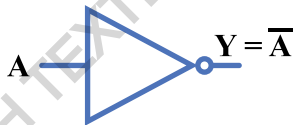


Figure 1.7: NOT Gate Symbol

Boolean Expression: $Y = A'$ or $Y = \bar{A}$ or $Y = \neg A$

Truth Table (NOT gate):

Input	Output
A	$Y = A'$
0	1
1	0

Real-life example:

A refrigerator light turns ON when the door is open and turns OFF when the door is closed.

4. NAND Gate:

The **NAND gate** is the opposite of AND gate. It produces output 0 only when **all inputs are 1**. NAND means "NOT AND".

Symbol:

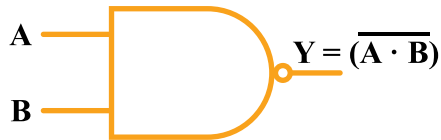


Figure 1.8: NAND Gate Symbol and equivalent circuit

Boolean Expression: $Y = (A \cdot B)'$ or $Y = A \text{ NAND } B$

Truth Table (2-input NAND gate):

Input		Output
A	B	$Y = (A \cdot B)'$
0	0	1
0	1	1
1	0	1
1	1	0



NAND gate is called a **universal gate** because any other gate can be built using only NAND gates.

5. NOR Gate:

The **NOR gate** is the opposite of OR gate. It produces output 1 only when **all inputs are 0**. NOR means "NOT OR".

Symbol:

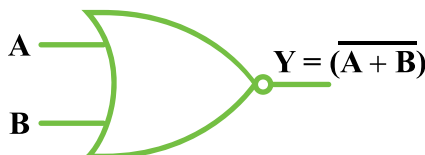


Figure 1.9: NOR Gate Symbol and equivalent circuit

Boolean Expression: $Y = (A + B)'$ or $Y = A \text{ NOR } B$

Truth Table (2-input NOR gate):

Input		Output
A	B	$Y = (A + B)'$
0	0	1
0	1	0
1	0	0
1	1	0



NOR gate is also a **universal gate**.

6. XOR Gate (Exclusive OR):

The **XOR gate** produces output 1 when the inputs are **different**.

Symbol:

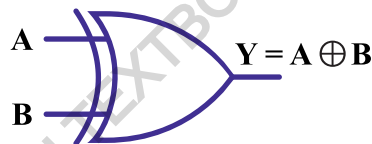


Figure 1.10: XOR Gate Symbol

Boolean Expression: $Y = A \oplus B$ or $Y = A'B + AB'$

Truth Table:

Input		Output
A	B	$Y = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Memory trick: XOR gives 1 when inputs are "eXclusively" different.

Real-life example: A two-way light switch. The light changes state when **EITHER** switch changes, but not when both change together.

7. XNOR Gate (Exclusive NOR):

The **XNOR** gate produces output 1 when the inputs are **same**.

Symbol:

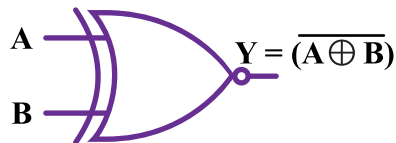


Figure 1.11: XNOR Gate Symbol

Boolean Expression: $Y = (A \oplus B)'$ or $Y = AB + A'B'$

Truth Table:

Input		Output
A	B	$Y = (A \oplus B)'$
0	0	1
0	1	0
1	0	0
1	1	1

Memory trick: XNOR gives 1 when inputs are the SAME.

Table 1.5: Summary of All Logic Gates

Gate	Symbol	Expression	Output = 1 when
AND	D-shape	$Y = A \cdot B$	All inputs are 1
OR	Arrow-shape	$Y = A + B$	Any input is 1
NOT	Triangle	$Y = A'$	Input is 0
NAND	AND + bubble	$Y = (A \cdot B)'$	Any input is 0
NOR	OR + bubble	$Y = (A + B)'$	All inputs are 0
XOR	OR + line	$Y = A \oplus B$	Inputs are different
XNOR	XOR + bubble	$Y = (A \oplus B)'$	Inputs are same

1.1.8 Logic Expressions:

A **logic expression** (Boolean expression) is a mathematical statement that uses binary (Boolean) variables and logical (Boolean) operators.

i) Components of Logic Expressions:

1. **Variables:** A, B, C, X, Y (can be 0 or 1 only)
2. **Operators:** AND ($\cdot$), OR ($+$), NOT ($'$)
3. **Parentheses:** Control order of operations

ii) Order of Operations (Precedence):

1. **Parentheses:** First (highest)
2. **NOT:** Second
3. **AND:** Third
4. **OR:** Last (lowest)

Example 1: Evaluate $Y = A' + B$ for $A = 1, B = 0$

Step 1: Calculate $A' = 1' = 0$

Step 2: Calculate $Y = 0 + 0 = 0$

Answer: $Y = 0$

Example 2: Evaluate $Y = (A + B) \cdot C$ for $A = 1, B = 0, C = 1$

Step 1: Calculate $(A + B) = 1 + 0 = 1$

Step 2: Calculate $Y = 1 \cdot 1 = 1$

Answer: $Y = 1$

Example 3: Evaluate $Y = A \cdot B + C'$ for $A = 1, B = 1, C = 1$

Step 1: Calculate $C' = 1' = 0$

Step 2: Calculate $A \cdot B = 1 \cdot 1 = 1$

Step 3: Calculate $Y = 1 + 0 = 1$

Answer: $Y = 1$

1.1.9 Creating Truth Tables from Logic Expressions:

Steps:

1. Identify all variables
2. Calculate number of rows (2^n , where n is the number of variables)
3. List all input combinations
4. Calculate intermediate values for the operation with highest precedence using truth table of that operation, do the same for the operation with lower precedence.
5. Calculate final output

Example: Create truth table for $Y = A \cdot B + C$

Variables: A, B, C (3 variables)

Rows: $2^3 = 8$ rows

Input			intermediate	Output
A	B	C	$A \cdot B$	$Y = A \cdot B + C$
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	0	1
1	0	0	0	0
1	0	1	0	1
1	1	0	1	1
1	1	1	1	1

1.1.10 Creating logical expression from Truth Table:

Minterm and Maxterms are canonical forms used in Boolean algebra to represent logical functions.

Minterm is a product (**AND**) of all variables (true or complemented) that output a **1**, used in sum of products (**SOP**). A variable is complemented (**A'**) if its value is **0** and uncomplemented (**A**) if its value is **1**.

Maxterm is a sum (**OR**) of all variables (true or complemented) that output a **0**, used in product of sums (**POS**). A variable is complemented (**A'**) if its value is **1** and uncomplemented (**A**) if its value is **0**.

Example: for two variables, A and B

A	B	Minterm	Maxterm
0	0	$A'B'$	$A+B$
0	1	$A'B$	$A+B'$
1	0	AB'	$A'+B$
1	1	AB	$A'+B'$

From the given truth table identify all the minterms and sum them up to form a Boolean expression.

Example:

Input		Output	
A	B	Y	
0	0	0	Maxterm
0	1	1	Minterm
1	0	0	Maxterm
1	1	1	Minterm

the truth table above.

there are two minterms $A'B$ and AB .

Therefore output in **SOP** is $Y = A'B + AB$ or

there are two maxterms $A + B$ and $A' + B$

Therefore output in **POS** is $Y = (A + B) \cdot (A' + B)$



Note:

Onward we will use SOP for creating and representing the logical expressions.

1.1.11 Logic Diagrams:

A **logic diagram** shows how logic gates are connected to create a circuit.

Steps to Draw Logic Diagrams:

1. Identify the expression
2. Break it into parts based on the precedence of the operation.
3. Draw gate for each part, operation with the highest precedence will be drawn first and with lower will be drawn after.
4. Connect gates with lines

Example: Draw logic diagram for $Y = A \cdot B + C$

AND operation has higher precedence, so we will draw AND gate with inputs A and B first, than an OR gate with one input from AND gate, drawn previously, and other input from C.

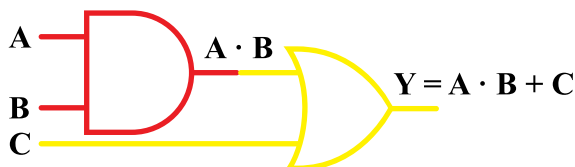


Figure 1.12: Logic Diagram for $Y = A \cdot B + C$

Example: Draw logic diagram for $Y = (A + B) \cdot C'$

Parenthesis have higher precedence in this case, so we will draw OR gate with inputs A and B first, then AND gate with one input from OR gate, drawn previously, and other input from C.

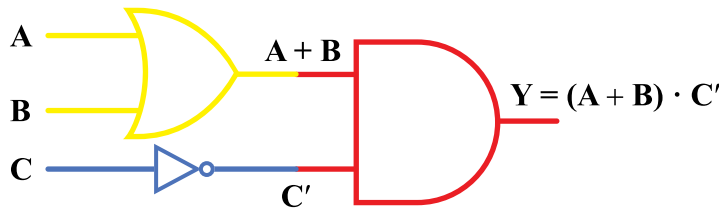


Figure 1.13: Logic Diagram for $Y = (A + B) \cdot C'$

1.1.12 Introduction to Karnaugh Maps (K-Maps):

A Karnaugh map (K-map) is a graphical tool used to simplify Boolean expressions. It was invented by Maurice Karnaugh in 1953. K-maps help us reduce the number of logic gates needed in a circuit, making circuits simpler, cheaper, and faster.

Why Use K-Maps?

When we have a Boolean expression with many terms, simplifying it using Boolean algebra can be difficult and time-consuming. K-maps provide a visual and systematic method to find the simplest form.

Benefits of K-Maps:

- Easier than algebraic simplification
- Visual method - you can see patterns
- Always gives the simplest result
- Reduces errors
- Works well for 2, 3, and 4 variables

Structure of K-Maps:

A K-map is a grid where:

- Each cell represents one row of the truth table
- Adjacent cells differ by only one variable (Gray code order)
- Adjacent cells, including wrapping

around with a 1 can be grouped to simplify expressions. In general, we can make groups of 2^n cells (1, 2, 4, 8...), where $n \geq 0$. Make as large as possible groups of cells, even if the groups are overlapping.

Important Rule: In K-maps, cells that are next to each other (including wrapping around edges) differ by only ONE bit. This is called Gray code arrangement.

	B = 0	B = 1
A = 0	0	1
A = 1	1	1

Two-Variable K-Map (2 Variables):

For a Boolean expression with 2 variables (A and B), we need a K-map with 4 cells ($2^2 = 4$), a 2×2 grid (two rows and two columns).

Structure:

	B = 0	B = 1
A = 0	Cell 0 (A'B')	Cell 1 (A'B)
A = 1	Cell 2 (AB')	Cell 3 (AB)

Cell Values:

- Cell 0: A = 0, B = 0 → represents term A'B' (minterm m_0)
- Cell 1: A = 0, B = 1 → represents term A'B (minterm m_1)
- Cell 2: A = 1, B = 0 → represents term AB' (minterm m_2)
- Cell 3: A = 1, B = 1 → represents term AB (minterm m_3)

Example 1: Simplify using 2-Variable K-Map:

Given Truth Table:

Input		Output
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

Boolean Expression: $Y = A'B + AB' + AB$

Step 1: Draw the K-map and fill in values from truth table

	B = 0	B = 1
A = 0	0	1
A = 1	1	1

Step 2: Group the 1s

We can make two groups:

- Group 1: Bottom row ($A = 1, B = 0$) and ($A = 1, B = 1$), from this group, we have

$$\begin{array}{r} A \quad B' \\ A \quad B \\ \hline A \quad x \end{array}$$

	B = 0	B = 1
A = 0	0	1
A = 1	1	1

B and B' will be canceled out $\rightarrow$ This gives us A

- Group 2: Right column ($A = 0, B = 1$) and ($A = 1, B = 1$) from this group, we have

$$\begin{array}{r} A' \quad B \\ A \quad B \\ \hline x \quad B \end{array}$$

- A and A' will be canceled out $\rightarrow$ This gives us B

Step 3: Write the simplified expression

$$Y = A + B$$

Verification: This matches the OR gate truth table!

Three-Variable K-Map (3 Variables):

For 3 variables (A, B, C), we need a K-map with 8 cells ($2^3 = 8$), this can be a 2×4 grid (two rows and four columns) or a 4×2 grid (four rows and two columns). Structure:

	BC = 00	BC = 01	BC = 11	BC = 10
A = 0	Cell 0 (A'B'C')	Cell 1 (A'B'C)	Cell 3 (A'BC)	Cell 2 (A'BC')
A = 1	Cell 4 (AB'C')	Cell 5 (AB'C)	Cell 7 (ABC)	Cell 6 (ABC')

Or

	C=0	C=1
AB = 00	Cell 0 (A'B'C')	Cell 1 (A'B'C)
AB = 01	Cell 2 (A'BC')	Cell 3 (A'BC)
AB = 11	Cell 6 (ABC')	Cell 7 (ABC)
AB = 10	Cell 4 (AB'C')	Cell 5 (AB'C)



Notice the column order is 00, 01, 11, 10 (NOT 00, 01, 10, 11). This is Gray code - only one bit changes between adjacent columns.

Gray Code Sequence:

- $00 \rightarrow 01$ (C changes)
- $01 \rightarrow 11$ (B changes)
- $11 \rightarrow 10$ (C changes)

This ensures adjacent cells differ by only one variable.

Example 2: Simplify using 3-Variable K-Map

Simplify: $Y = A'B'C + A'BC + AB'C + ABC$

Step 1: Identify the minterms

- $A'B'C = 001 = m_1$
- $A'BC = 011 = m_3$
- $AB'C = 101 = m_5$
- $ABC = 111 = m_7$

Step 2: Fill the K-map with 1s in those positions

	BC = 00	BC = 01	BC = 11	BC = 10
A = 0	0	1	1	0
A = 1	0	1	1	0

Step 3: Group the 1s

	BC = 00	BC = 01	BC = 11	BC = 10
A = 0	0	1	1	0
A = 1	0	1	1	0

Looking at the K-map, we see that cells m_1, m_3, m_5, m_7 form a square group of 4 in columns $BC = 01$ and $BC = 11$.

What do these cells have in common? $C = 1$ in all of them!

Step 4: Write simplified expression

$$Y = C$$

Original expression: $A'B'C + A'BC + AB'C + ABC$ (4 terms, 12 literals)

Simplified expression: C (1 term, 1 literal)

This is a huge simplification!

Example 3: Simplify using 3-Variable K-Map

Simplify: $Y = A'B'C' + A'BC' + A'B'C + A'BC + AB'C + ABC$

Step 1: Identify the minterms

- $A'B'C' = 000 = m_0$
- $A'B'C = 001 = m_1$
- $A'BC' = 010 = m_2$

Unit 1: Computer Systems

- $A'BC = 011 = m_3$
- $AB'C = 101 = m_5$
- $ABC = 111 = m_7$

Step 2: Fill the K-map with 1s in those positions

	BC = 00	BC = 01	BC = 11	BC = 10
A = 0	1	1	1	1
A = 1	0	1	1	0

Step 3: Group the 1s

	BC = 00	BC = 01	BC = 11	BC = 10
A = 0	1	1	1	1
A = 1	0	1	1	0

Looking at the K-map, we see that cells m_1, m_3, m_5, m_7 form a square group of 4 in columns $BC = 01$ and $BC = 11$ also cells m_0, m_1, m_2, m_3 form a Horizontal group of 4 in row $A = 0$.

What do the former group have in common? $C = 1$ in all of them!

What do the latter group have in common? $A = 0$ in all of them!

Step 4: Write simplified expression

$$Y = A + C$$

Original expression: $A'B'C' + A'BC' + A'B'C + A'BC + AB'C + ABC$ (6 terms, 18 literals)

Simplified expression: $A' + C$ (2 terms, 2 literals)

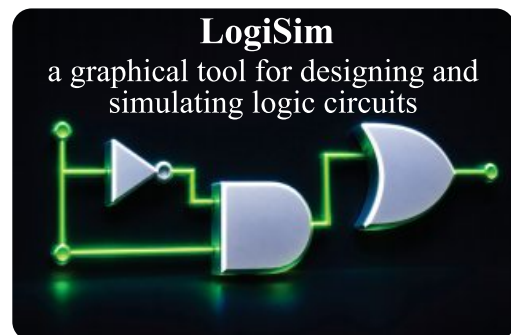
Another huge simplification!

Simulation tool for Logic Gates (LogiSim):

1.1.13 Downloading and Installing LogiSim Evolution:

1. Downloading the Software:

LogiSim Evolution is available free of charge from its official GitHub repository. Figure 1.14 shows the GitHub releases page where the software can be downloaded.



<https://github.com/logisim-evolution/logisim-evolution/releases>

logisim-evolution / logisim-evolution

Digital logic design tool and simulator

Releases

Latest release: v3.9.0 (Logisim-Evolution)

Released: February 2025 | Requires Java 21 or newer

Assets (Download Files):

- Windows Installer (64-bit) — RECOMMENDED
- Platform-independent JAR file (requires Java)

← Click here to download for Windows

28.4 MB
12.1 MB

Installation Steps (Windows):

- Click the Windows Installer link above to download the .msi file.
- Also download and install Java 21 from <https://adoptium.net> if not already installed.
- Double-click the downloaded .msi file and follow the installation wizard.
- Launch Logisim Evolution from the Start menu or desktop shortcut.

System Requirements

- Operating System: Windows 10 / 11 (64-bit)
- Java Runtime: Java 21 or newer (download free from adoptium.net)
- RAM: 512 MB minimum | Disk Space: 100 MB | Screen: 1024x768 or higher

Figure 1.14: LogiSim Evolution Download Page (GitHub Releases)

Follow these steps to download LogiSim Evolution for Windows:

- Open a web browser and go to: <https://github.com/logisim-evolution/logisim-evolution/releases>
- Click on the latest release (for example, v3.9.0 or newer).
- Under 'Assets', click the file named logisim-evolution-X.X.X-win-x86_64-installer.msi to download the Windows installer.
- If Java 21 is not installed, also download and install it from <https://adoptium.net> before proceeding.



Note:

The .msi installer for Windows includes everything needed to run LogiSim Evolution on a 64-bit Windows system. If you are unsure whether Java is installed, you can check by opening the Windows Command Prompt and typing: `java -version`.

2. Installing LogiSim Evolution on Windows:

Once the installer file has been downloaded, follow these steps to install the software:

- Locate the downloaded .msi file in your Downloads folder.
- Double-click the file to launch the installation wizard.
- When Windows asks 'Do you want to allow this app to make changes to your device?', click Yes.

- Follow the on-screen instructions: accept the license agreement, choose an installation folder (the default is recommended), and click Install.
- When the installation is complete, click Finish. A shortcut is added to the Windows Start menu and optionally to the desktop.
- To launch LogiSim Evolution, click the shortcut in the Start menu or double-click the desktop icon.

3. The LogiSim Evolution Interface:

When LogiSim Evolution starts, it opens with a blank project ready for circuit design. The interface is divided into four main areas. Figure 1.15 shows the complete interface and Figure 1.16 shows the interface with each area labelled.

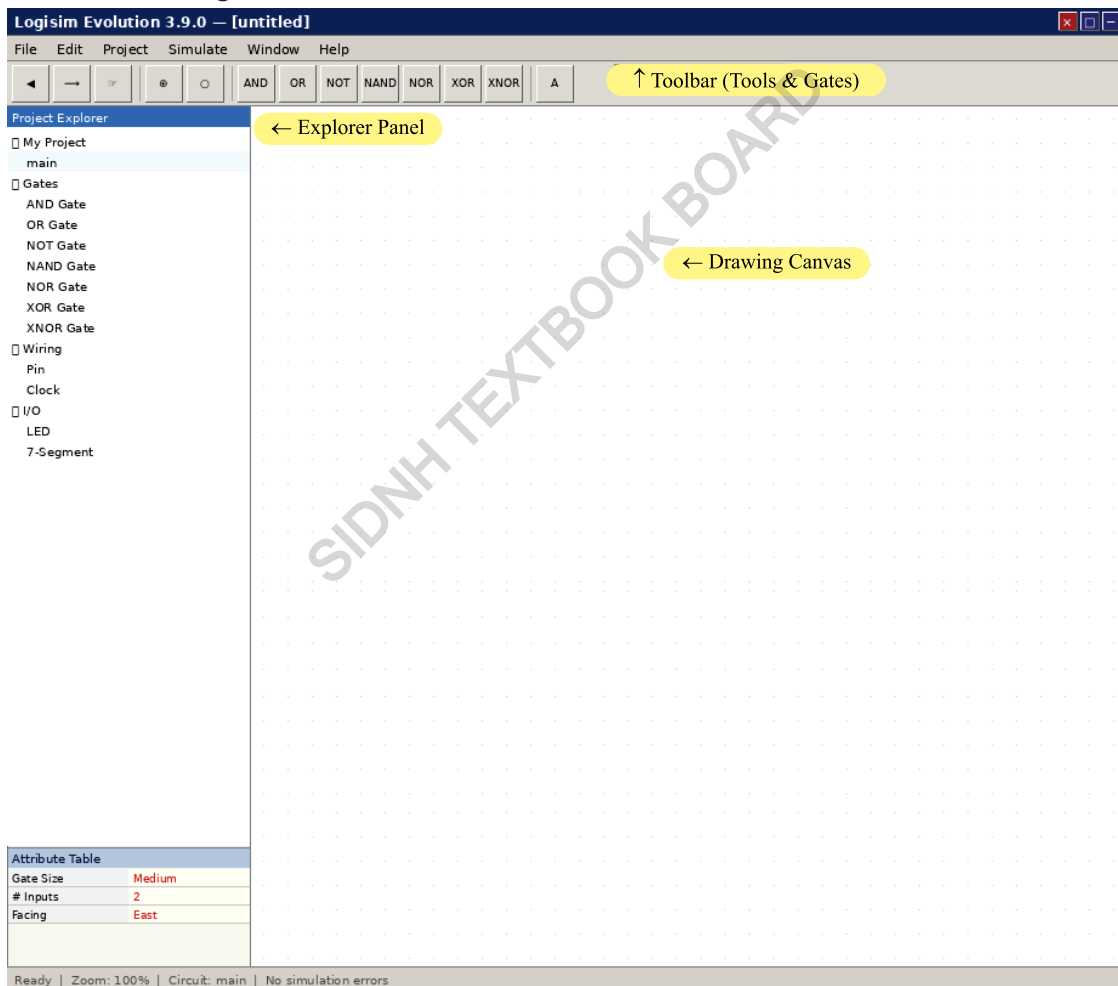


Figure 1.15: LogiSim Evolution Main Interface

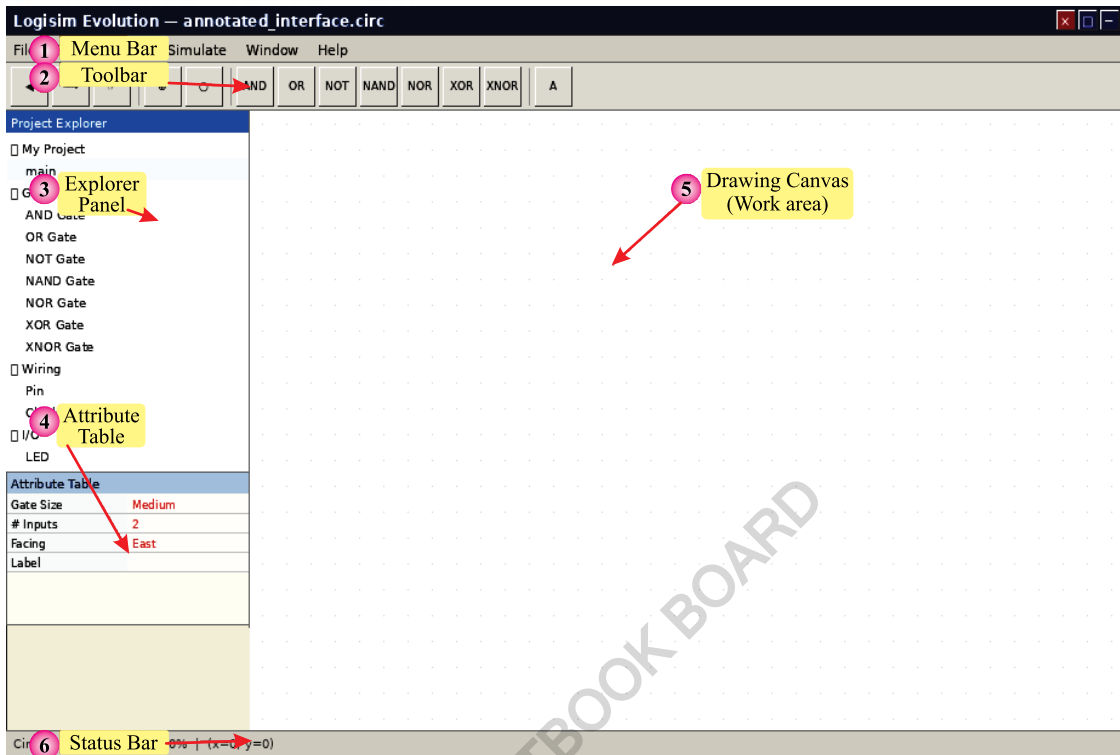


Figure 1.16: LogiSim Evolution Interface Annotated Areas

(a) The Menu Bar:

The Menu Bar at the top of the window provides access to all program functions organized into six menus:

Table 1.6: Menu Bar Menus and Their Main Functions

Menu	Main Functions
File	New project, Open, Save, Save As, Print, Export image, Exit
Edit	Undo, Redo, Cut, Copy, Paste, Select All, Duplicate
Project	Add new circuit, Load library, Edit circuit appearance, Project options
Simulate	Start/stop simulation, Reset, Step clock, Simulation speed settings
Window	Zoom in/out, toggle panels, switch between open projects
Help	Online documentation, About LogiSim Evolution

(b) The Toolbar:

The Toolbar provides one-click access to the most frequently used tools and gates. Table 1.7 describes each tool in the default toolbar.

Table 1.7: Toolbar Tools and Their Functions

Tool	Icon	Icon Description	Function
Edit / Select Tool		Arrow cursor	Select, move, and edit components already on the canvas
Wiring Tool		Line with two dots	Draw wires between component pins (horizontal or vertical only)
Poke Tool		Pointing finger (☞)	Toggle input values during simulation; inspect wire states
Text Tool	A	Letter A	Add text labels and annotations to the circuit
Input Pin		Square with a 0 inside	Add a logic input (switch) to the circuit
Output Pin		Circle with a 0 inside	Add a logic output (LED) to the circuit
AND Gate		D-shaped symbol	Place a 2-input AND gate on the canvas
OR Gate		Arrow-shaped symbol	Place a 2-input OR gate on the canvas
NOT Gate		Triangle with bubble	Place a NOT (inverter) gate on the canvas
NAND Gate		AND + bubble	Place a NAND gate (NOT AND) on the canvas
NOR Gate		OR + bubble	Place a NOR gate (NOT OR) on the canvas
XOR / XNOR Gate		OR+ extra line / XOR + bubble	Place an Exclusive OR gate (XOR), or exclusive NOR gate (XNOR), on the canvas

(c) The Explorer Panel:

The Explorer Panel on the left side of the window contains two sections. The upper section shows your project structure, including all circuits in the project. The lower section shows component libraries (Gates, Wiring, I/O, Memory, etc.) from which you can select and place components onto the canvas. Clicking any component in the Explorer Panel selects it for placement on the canvas.

(d) The Attribute Table:

The Attribute Table appears below the Explorer Panel. When you click on a component (either in the Explorer Panel or on the canvas), its properties appear in the Attribute Table. For example, selecting an AND gate shows attributes such as Gate Size, Number of Inputs, and Facing (direction). You can click on any attribute value to change it.

(e) The Drawing Canvas:

The Drawing Canvas is the large white area on the right side of the window. This is where you design your circuit by placing components and connecting them with wires. The canvas has a grey dot grid to help you align components neatly. You can zoom in and out using Ctrl+= and Ctrl+- or using the mouse scroll wheel. The canvas background is white by default; the grid dots are light grey and do not appear in printed or exported versions of the circuit.

4. Understanding Wire Colors in Simulation:

During simulation, LogiSim Evolution uses wire colors to show the logical state of each wire. Table 1.8 explains what each color means.

Table 1.8: Wire Color Meanings in LogiSim Evolution

Wire Color	Meaning	What to Do
Bright green	Logic 1 (HIGH) the wire carries a 1 signal	This is correct. The signal is active.
Dark green	Logic 0 (LOW) the wire carries a 0 signal	This is correct. The signal is inactive.
Blue	Unknown value , the wire has no defined signal	Check that all inputs are connected to a pin.
Grey	Unconnected , the wire is not connected to anything	Connect both ends of the wire to component pins.
Orange/Red	Error, short circuit or multiple outputs connected	Check for wires connecting two outputs together.

IMPORTANT Before testing a circuit, make sure no wires are blue or grey. Blue means an input has not been connected to a Pin, and grey means a wire is floating and not connected to any component at one or both ends.

5. Building Your First Circuit, Step-by-Step Guide:

This section guides you through building a simple 2-input AND gate circuit from scratch. The circuit will have two inputs (A and B) and one output (Y), where $Y = A \cdot B$.

Step 1: Start a New Project

- Launch LogiSim Evolution from the Start menu or desktop shortcut.
- A new blank project opens automatically with a circuit named 'main'. If you want to start fresh, select File → New.
- Before building, save the file: select File → Save As, type a name such as AND_gate_circuit, and click Save. Files are saved with the .circ extension.

Step 2: Place an AND Gate

Figure 1.17 shows how to select a gate from the Explorer Panel and place it on the canvas.

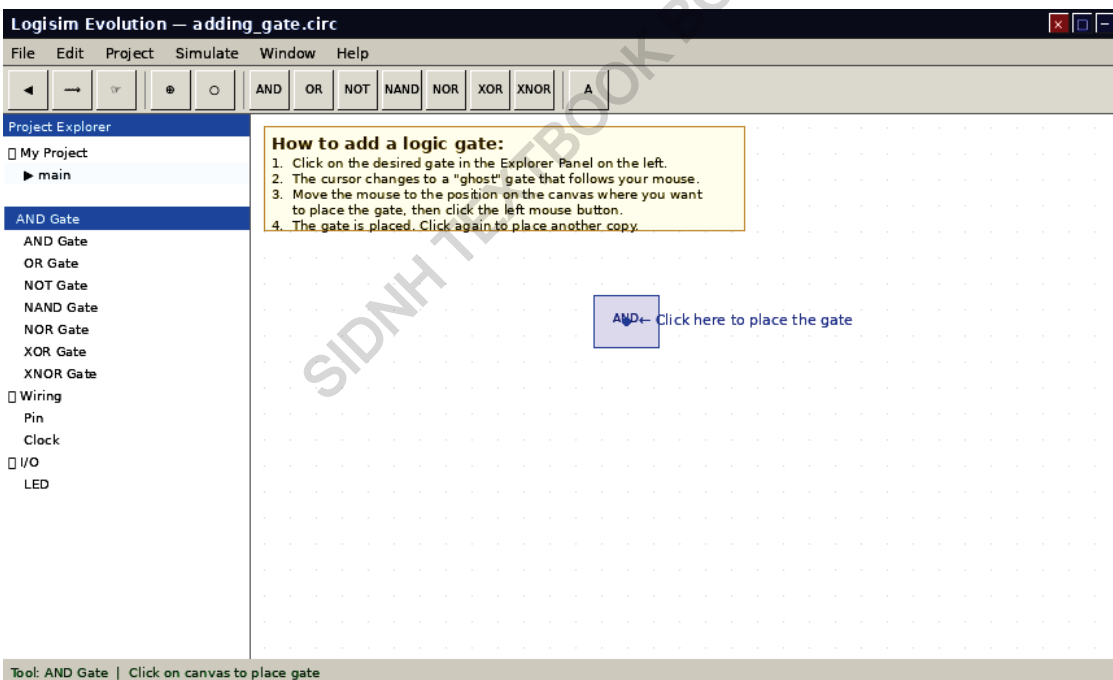


Figure 1.17: Selecting and Placing an AND Gate from the Explorer Panel

- In the Explorer Panel, click on 'Gates' to expand the library.
- Click on 'AND Gate'. The cursor changes to show a ghost image of the gate following your mouse.

- Move the mouse to the center of the canvas and click the left mouse button to place the gate.
- Press Escape or click the Select Tool (arrow) in the toolbar to stop placing gates.



You can also find gates directly in the toolbar and click their icons to select them for placement. For example, the AND icon in the toolbar is the D-shaped symbol.

Step 3: Place Input and Output Pins

Inputs and outputs are represented by Pins. An Input Pin is a square; it acts as a switch you can toggle. An Output Pin is a circle; it displays the result.

- Click the Input Pin tool in the toolbar (square icon).
 - Click on the canvas to the left of the AND gate, place one pin for input A.
 - Place a second Input Pin below the first one, this will be input B.
 - Click the Output Pin tool (circle icon).
 - Place an Output Pin to the right of the AND gate, this will be the output Y.
- To label the pins, click the Select Tool (arrow), then double-click each pin and type its label (A, B, or Y) in the 'Label' field in the Attribute Table.

Step 4: Draw Wires

Figure 1.18 shows the wiring process. Wires in LogiSim Evolution must always be horizontal or vertical, diagonal wires are not allowed.

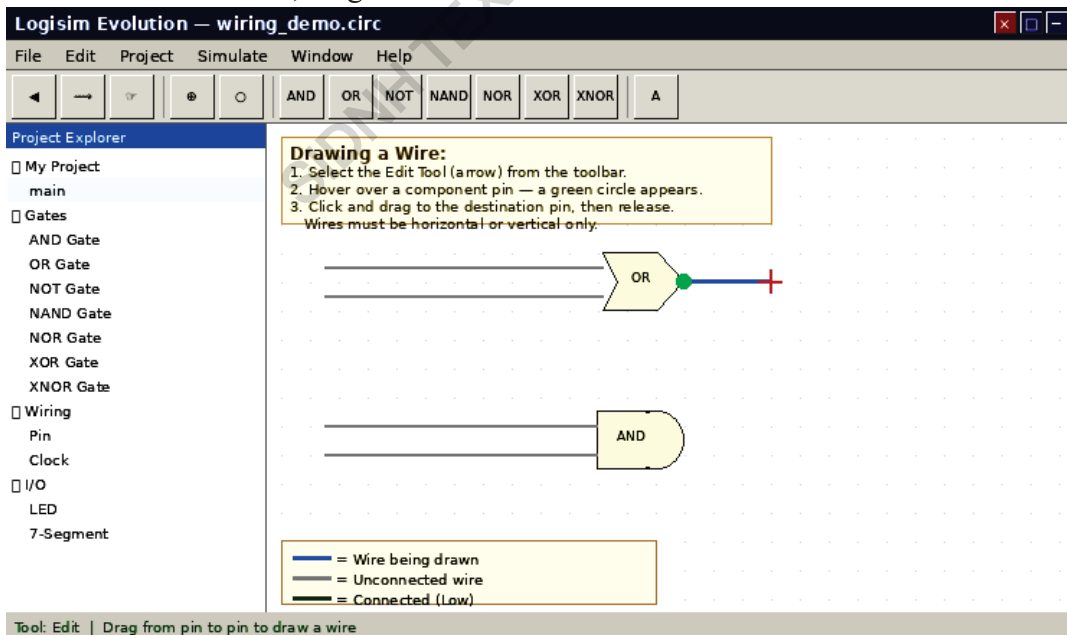


Figure 1.18: Drawing Wires Between Component Pins

Unit 1: Computer Systems

- Select the Edit Tool (arrow cursor) from the toolbar.
- Move the cursor over a pin end of the Input Pin A. A small green circle appears, indicating a valid connection point.
- Click and hold the mouse button, then drag the wire to the top input pin of the AND gate.
- Release the mouse button. The wire connects and changes color (green for connected, grey if not properly connected).
- Repeat to connect Input Pin B to the bottom input of the AND gate.
- Connect the output of the AND gate to the Output Pin Y.



If a wire shows in blue after connecting, it means one end is not properly connected. Zoom in (Ctrl+=) and carefully re-draw the wire, making sure it starts and ends exactly on a component pin (green dot).

Step 5: Test the Circuit with the Poke Tool

Figure 1.19 shows the completed AND gate circuit with the simulation running. The Poke Tool is used to toggle input values and observe the output.

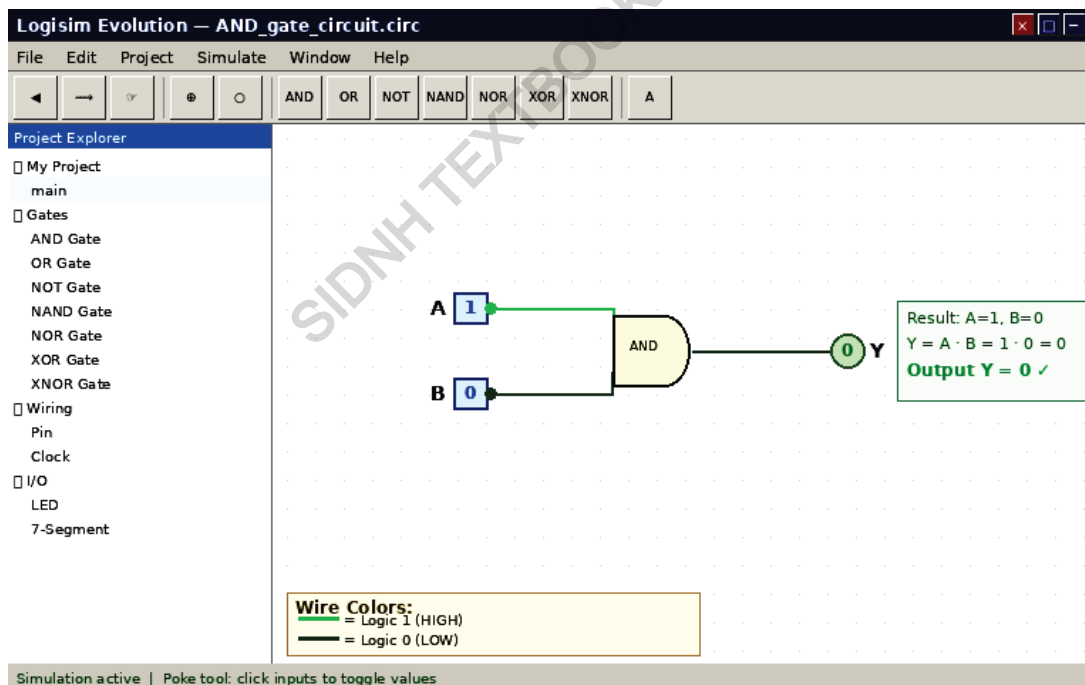


Figure 1.19: Completed AND Gate Circuit with Simulation Running

- Select the Poke Tool from the toolbar (the pointing finger icon).
- Click on Input Pin A to toggle its value between 0 and 1.

- Click on Input Pin B to toggle its value.
- Observe the output pin Y and the wire colors bright green means 1 (HIGH), dark green means 0 (LOW).
- Test all four input combinations (0,0), (0,1), (1,0), and (1,1) and record the output Y each time.

The output Y should match the AND gate truth table: $Y = 1$ only when both $A = 1$ AND $B = 1$.

Table 1.9: Expected Simulation Results AND Gate

Input A	Input B	Expected Output $Y = A \cdot B$	Wire Color of Y
0	0	0	Dark green
0	1	0	Dark green
1	0	0	Dark green
1	1	1	Bright green

Step 6: Save Your Circuit

It is important to save your work regularly. Figure 1.20 shows the Save As dialog.

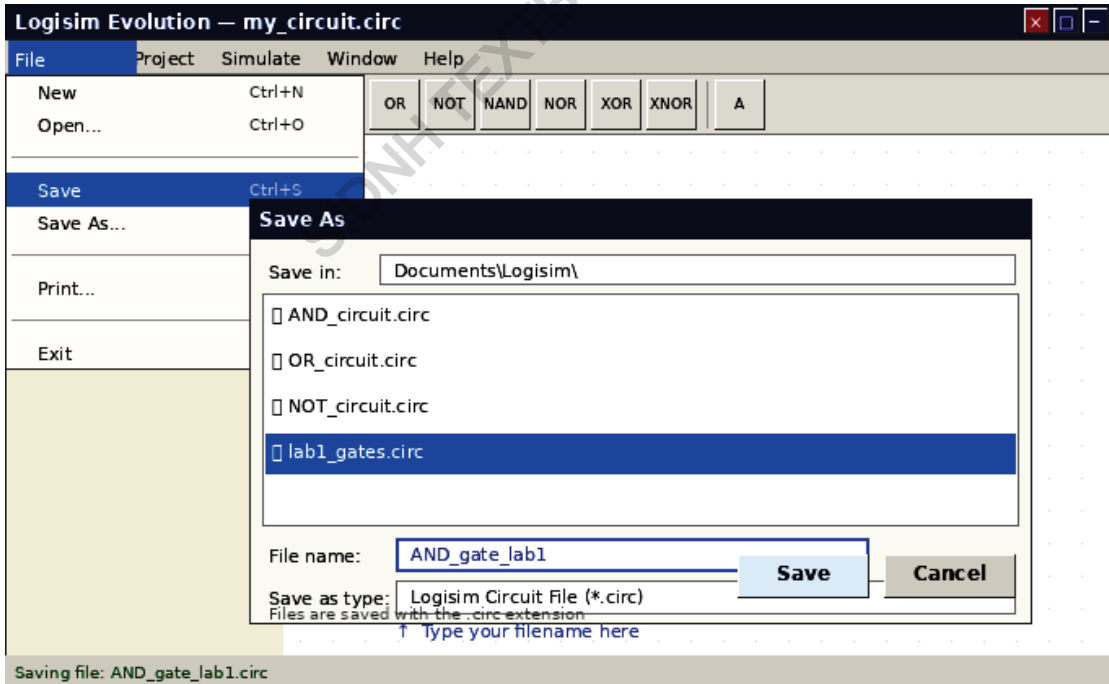


Figure 1.20: Saving a Circuit Using the Save As Dialog

- Select File → Save (Ctrl+S) to save using the current filename.
- Select File → Save As (Ctrl+Shift+S) to save with a new name or in a different location.
- Choose a meaningful filename such as AND_gate_lab1.circ and click Save.
- LogiSim Evolution saves files with the .circ extension. These files can be opened on any computer with LogiSim installed.

6. Placing and Testing Other Logic Gates:

LogiSim Evolution includes all seven standard logic gates. The procedure for placing, wiring, and testing each gate is the same as described in Section 5. Table 1.10 summarizes all available gates with their Boolean expressions and truth table outputs.

Table 1.10: Logic Gates Available in LogiSim Evolution

Gate	Symbol	Boolean Expression	Output = 1 When
AND		$Y = A \cdot B$	All inputs are 1
OR		$Y = A + B$	Any input is 1
NOT		$Y = A'$	Input is 0 (inverts)
NAND		$Y = (A \cdot B)'$	Any input is 0
NOR		$Y = (A + B)'$	All inputs are 0
XOR		$Y = A \oplus B$	Inputs are DIFFERENT
XNOR		$Y = (A \oplus B)'$	Inputs are the SAME

(a) Changing the Number of Inputs:

By default, most gates in LogiSim Evolution have 2 inputs. To change this:

- Select the gate on the canvas using the Select Tool (arrow).
- In the Attribute Table (lower-left panel), find the attribute 'Number of Inputs'.
- Click the value and type the new number (between 2 and 32).
- The gate symbol on the canvas updates immediately to show the new number of input pins.

(b) Changing Gate Facing Direction:

By default, gates face East (inputs on the left, output on the right). To rotate a gate:

- Select the gate using the Select Tool.
- In the Attribute Table, find the attribute 'Facing'.
- Click the value and choose North, South, East, or West.
- Alternatively, select the gate and press Ctrl+R to rotate it 90 degrees clockwise.

7. Building a Multi-Gate Circuit:

The real power of LogiSim Evolution is the ability to combine multiple gates to build more complex circuits. Figure 1.21 shows the simulation of a circuit that implements the Boolean expression $Y = A \cdot B + C$, which requires an AND gate and an OR gate.

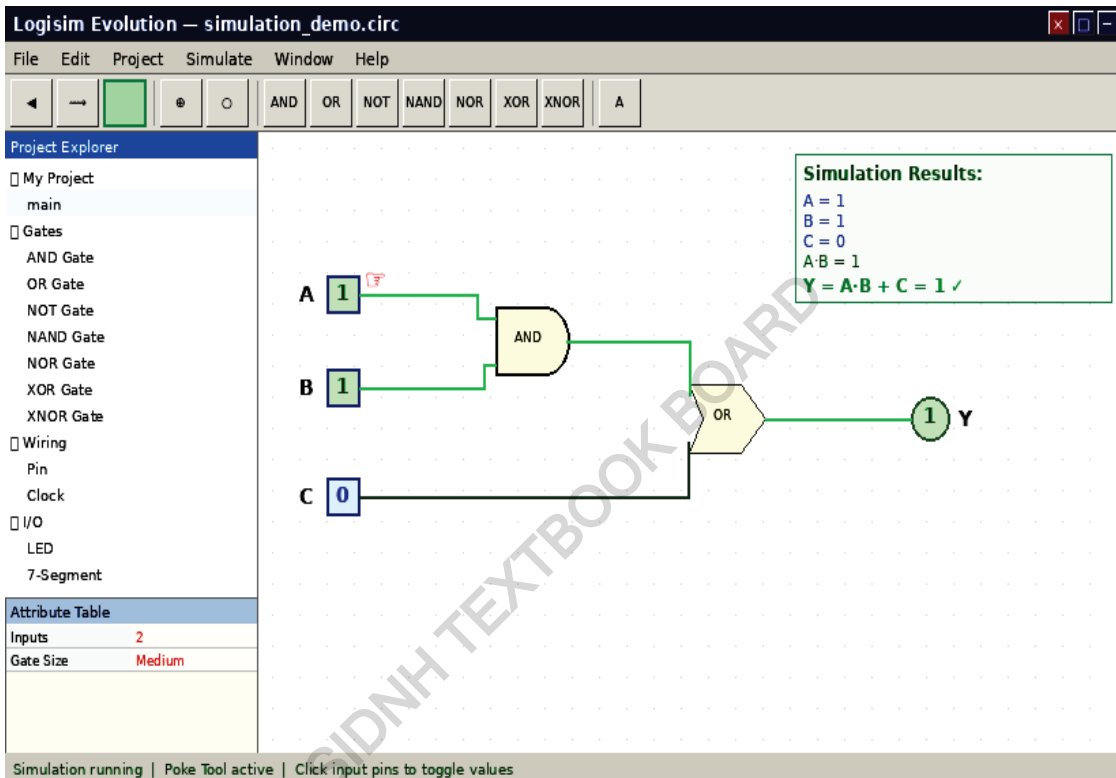


Figure 1.21: Multi-Gate Circuit Simulating $Y = A \cdot B + C$

Steps to build the circuit $Y = A \cdot B + C$:

- Place three Input Pins and label them A, B, and C.
- Place one AND gate and one OR gate on the canvas.
- Connect Input A and Input B to the two inputs of the AND gate.
- Connect the output of the AND gate to one input of the OR gate.
- Connect Input C to the other input of the OR gate.
- Place an Output Pin and connect it to the output of the OR gate. Label it Y.
- Use the Poke Tool to test all 8 input combinations (A,B,C) and verify that the output Y matches the truth table.

Table 1.11: Expected Results $Y = A \cdot B + C$

A	B	C	$A \cdot B$	$Y = A \cdot B + C$
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	0	1
1	0	0	0	0
1	0	1	0	1
1	1	0	1	1
1	1	1	1	1

1.2 Software Development Life Cycle (SDLC):

1.2.1 What is Software Development?

Software development is the process of creating software applications. It includes:

- Planning what the software should do (function)
- Designing how it will work (algorithm)
- Writing the code (programming)
- Testing (debugging)
- Releasing to users (deployment)
- Fixing problems after release (support)

Software development is like building a house. We need a plan, materials, workers, and inspections before people can live in it.



1.2.2 What is SDLC?

SDLC (Software Development Life Cycle) is a structured process that guides software development from beginning to end. It provides a clear framework with specific phases and deliverables.

Table:1.12 Benefits of using SDLC

Benefit	Explanation
Clear goals	Everyone knows what to build
Better planning	Resources and time are managed well
Quality assurance	Testing ensures fewer bugs
Risk reduction	Problems are found early
Documentation	Records help future maintenance
Team communication	Everyone follows the same process

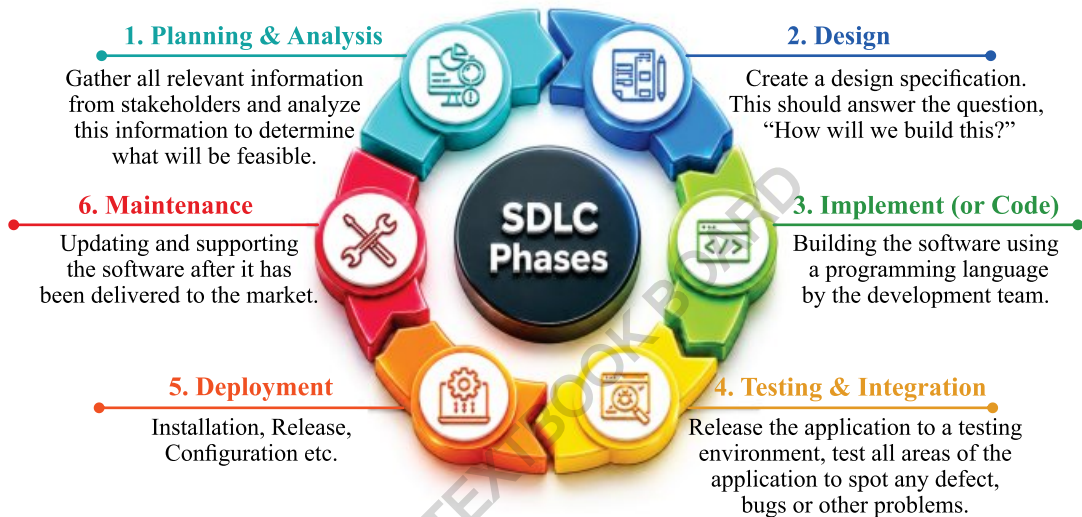


Figure 1.22: SDLC Cycle Diagram

1.2.3 Phases of SDLC:

SDLC has six main phases:

Phase 1: Requirement Analysis (Planning)

This is the first and most important phase. The team gathers information about what the software should do.

Activities:

- Meet with customers/users
- Understand their problems and needs
- List all features required
- Create a document called **SRS (Software Requirements Specification)**

Deliverable: SRS Document

Example: For a school management system:

- Students should be able to see their grades
- Teachers should enter attendance
- Parents should receive notifications

Phase 2: System Design

Based on requirements, the team designs how the software will work and look.

Activities:

- Create **High-Level Design (HLD)**: Overall structure
- Create **Low-Level Design (LLD)**: Detailed component design
- Design database structure
- Design user interface (screens)

Deliverable: Design Document

Tools used:

- Flowcharts
- UML Diagrams
- ER Diagrams
- Wireframes

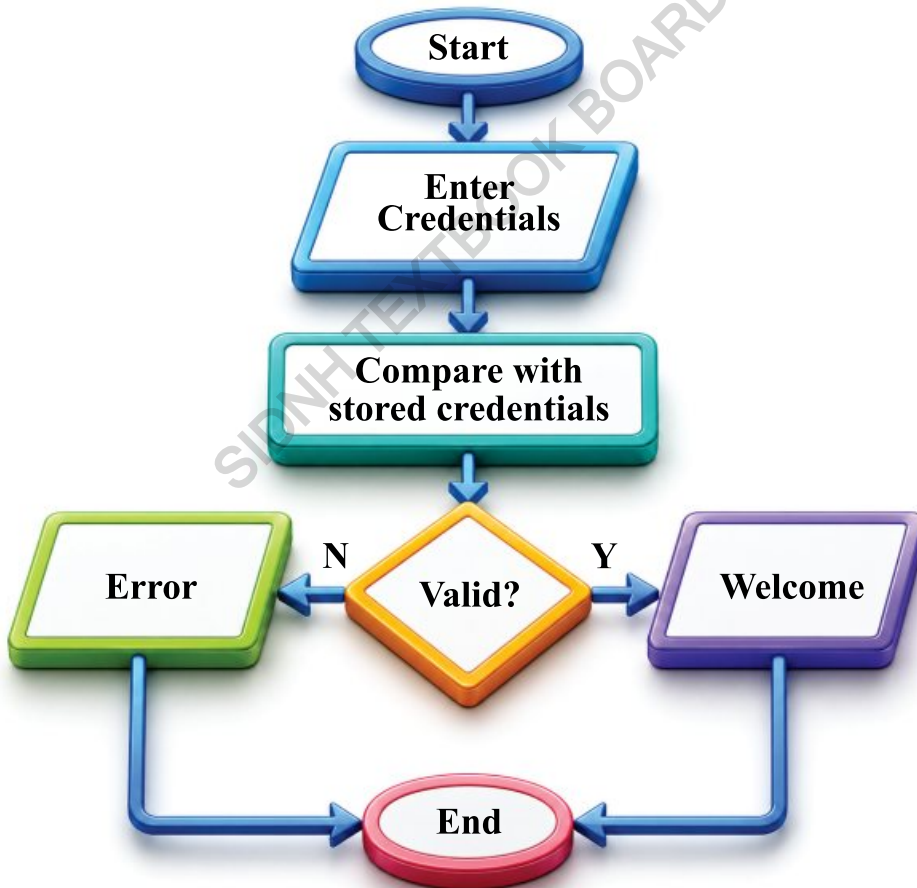


Figure 1.23: Example of a Simple Flowchart

Phase 3: Implementation (Coding)

Developers write the actual code based on design documents.

Activities:

- Write code in programming languages (Python, Java, C++, etc.)
- Follow coding standards
- Use version control (Git)
- Document the code

Deliverable: Working code/software

Best Practices:

- Write clean, readable code
- Add comments to explain complex parts
- Test code as we write
- Review code with team members

Phase 4: Testing

The software is tested to find and fix errors (bugs).

Table 1.13 : Types of Testing

Testing Type	Description	Who Does It
Unit Testing	Test individual components	Developers
Integration Testing	Test combined modules	Testing team
System Testing	Test complete system	Testing team
User Acceptance Testing (UAT)	Test by actual users	End users

Testing Methods:

Black Box Testing:

- Tester does NOT see the code
- Tests only inputs and outputs
- Checks if software works correctly for users
- Example: Enter username and password, check if login works

White Box Testing:

- Tester CAN see the code
- Tests internal logic and structure
- Checks if code is written correctly
- Example: Check if all conditions in the code are tested

Table 1.14 : Comparison of Black box and White box testing

Black Box Testing	White Box Testing
Does not need code knowledge	Needs programming knowledge
Tests functionality	Tests code structure
Done by testers	Done by developers
Tests what software does	Tests how software does it

Deliverable: Test reports, bug-free software

Phase 5: Deployment

The software is released to users.

Table 1.15: Deployment strategies

Strategy	Description
Direct Deployment	Replace old system immediately
Phased Rollout	Release to groups of users gradually
Pilot Release	Release to small group first, then everyone
Parallel Running	Run old and new systems together temporarily

Deliverable: Live software available to users

Phase 6: Maintenance

After release, the software needs ongoing support and updates.

Table 1.16 : Types of Maintenance

Type	Description	Example
Corrective	Fix bugs found after release	Fixing login error
Adaptive	Change for new environments	Update for new Windows version
Perfective	Add new features or improve	Add dark mode
Preventive	Prevent future problems	Improve code for better performance

Deliverable: Updated versions, patches, support

1.2.4 SDLC Models:

Different projects need different approaches. SDLC models define how phases are organized.

(a) Waterfall Model:

The **Waterfall model** is a linear, sequential approach. Each phase must be completed before the next begins. There is no going back.

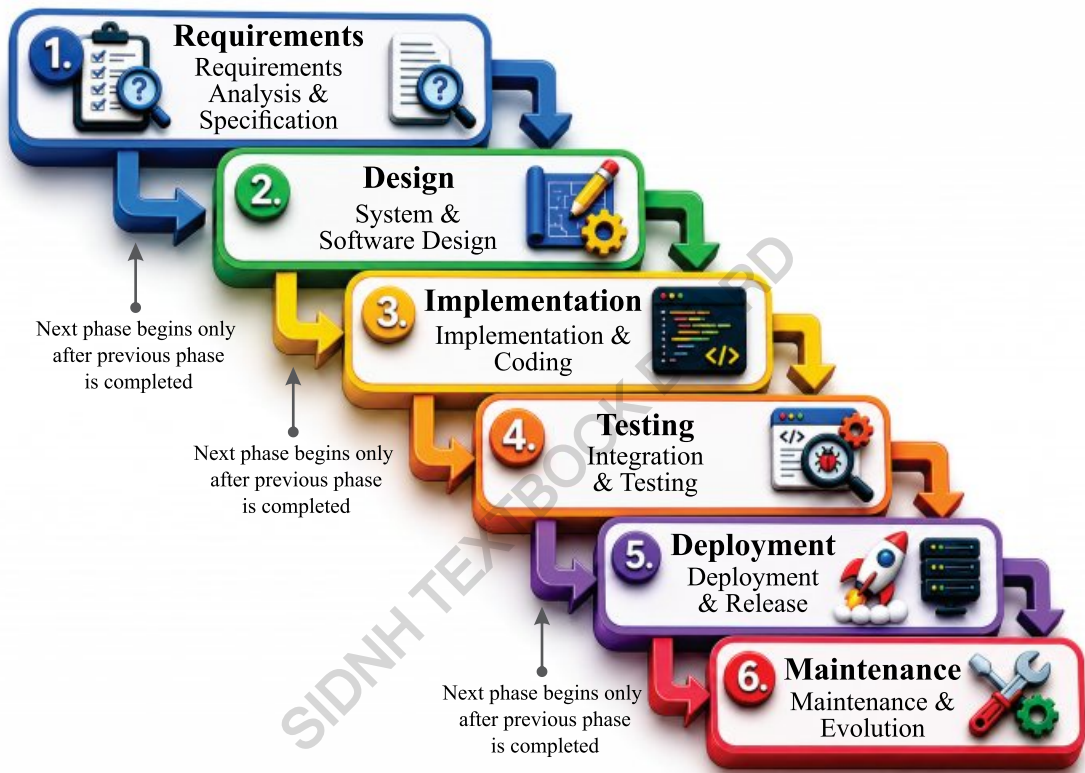


Figure 1.24: Waterfall Model Diagram

Characteristics:

- Sequential (one phase after another)
- No overlapping phases
- Heavy documentation
- Changes are difficult and expensive

When to Use Waterfall:

- Requirements are clear and fixed
- Project is small to medium size
- Technology is well understood
- No major changes expected

Table 1.17 : Advantages of using Waterfall model

Advantage	Explanation
Simple to understand	Clear, linear process
Easy to manage	Defined phases and milestones
Well documented	Complete records at each phase
Works for small projects	When requirements don't change

Table 1.18: Disadvantages of using Waterfall model

Disadvantage	Explanation
Not flexible	Hard to make changes
Late testing	Bugs found late in process
High risk	Problems discovered late are costly
Not suitable for complex projects	Cannot handle changing requirements

(b) Agile Model:

The **Agile model** is an iterative approach. Software is developed in small cycles called **sprints** (usually 2-4 weeks).

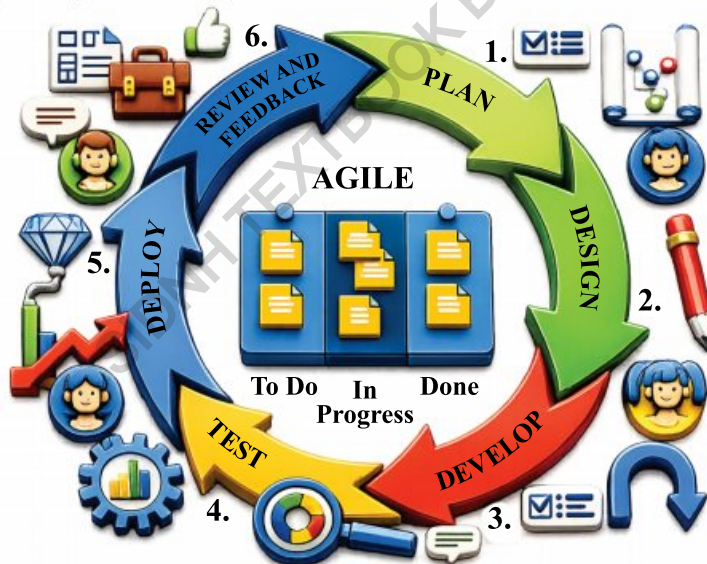


Figure 1.25: Agile Model Diagram

Characteristics:

- Iterative and incremental
- Flexible to changes
- Customer involvement throughout
- Working software delivered frequently
- Team collaboration emphasized

Table 1.19: Key Agile Concepts

Term	Meaning
Sprint	Short development cycle (2-4 weeks)
User Story	Feature described from user's perspective
Scrum	Popular Agile framework
Daily Standup	Short daily team meeting
Product Backlog	List of all features to be developed

When to Use Agile:

- Requirements may change
- Customer wants to see progress frequently
- Project is complex or large
- Innovation and speed are important

Table 1.20: Advantages of using Agile model

Advantages	Explanation
Flexible	Easy to add or change features
Early delivery	Users see working software quickly
Customer satisfaction	Regular feedback and involvement
Lower risk	Problems found and fixed early
Better quality	Continuous testing

Table 1.21 : Disadvantages of using Agile model

Disadvantages	Explanation
Less documentation	May cause problems later
Requires commitment	Customer must be available
Unpredictable	Hard to estimate time and cost
Scope creep	Features may keep expanding

1.2.5: Case Study 1: Waterfall Model

The Al-Noor Library Management System:

1. Background:

Al-Noor High School has 1,200 students and 80 teachers. The school library contains approximately 2,400 books. Before this project, the library used a paper-based system where students recorded their name, class, and book title in a handwritten

register when borrowing a book. This system had several problems: books were frequently lost, records were difficult to search, and the librarian spent considerable time each week compiling overdue lists by hand.

The school administration decided to commission a computerized Library Management System (LMS). The requirements were discussed in detail with the librarian and principal before development began. Because the requirements were clear, fixed, and unlikely to change during development, the project team selected the Waterfall model as the most appropriate SDLC approach.

Table 1.22: Project Overview

Parameter	Details
Project Name	Al-Noor Library Management System (LMS)
Client	Al-Noor High School Administration
Development Team	2 Developers, 1 Tester, 1 Project Manager
Total Duration	6 months (January to June)
SDLC Model	Waterfall Model
Technology Stack	Python 3.10, Tkinter GUI, SQLite database
Platform	Windows 10 (no internet connection required)

2. Why the Waterfall Model Was Selected:

The project team evaluated two options, the Waterfall model and the Agile model, before beginning development. The Waterfall model was chosen for the following reasons:

- **Fixed requirements:** The school knew exactly what the system had to do. No new features were anticipated once development began.
- **Clear deadline:** The system had to be operational before the start of the new academic year, giving the team a firm endpoint.
- **Small team:** With only four team members, the linear structure of Waterfall was easier to manage than iterative sprints.
- **Low complexity:** The system performed well-understood operations (add, search, borrow, return, report) with no experimental features.
- **Adequate documentation:** The school required a written record of all requirements and design decisions for future reference.

3. The Waterfall Model:

The Waterfall model organizes software development into six distinct, sequential phases. Each phase must be fully completed and formally approved before the next phase begins. Unlike the Agile model, there is no mechanism to return to an

earlier phase once it is complete. Figure 1.26 illustrates the sequential flow of the Waterfall model.

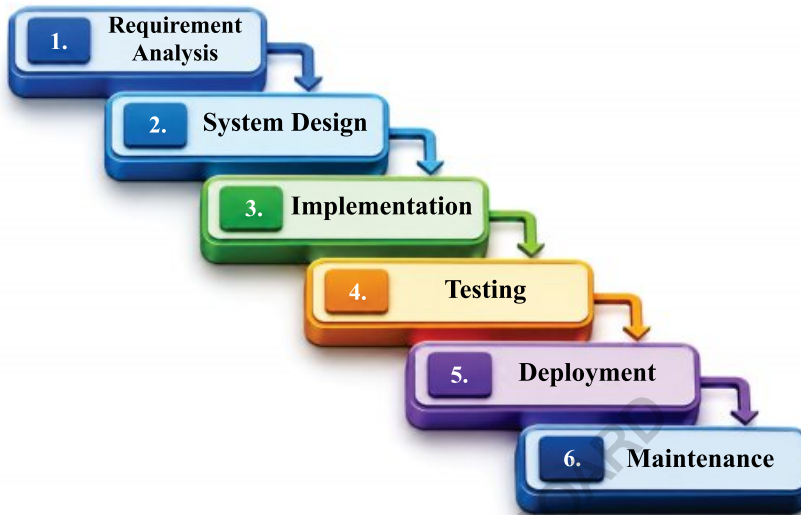


Figure 1.26: Waterfall Model , Sequential Phase Diagram

4. Project Timeline:

The six phases were planned as follows. Figure 1.27 shows the project timeline as a Gantt chart, illustrating the duration and sequence of each phase.

Phase	Jan Wk1-2	Jan Wk3-4	Feb	Mar	Apr	May	Jun+
Requirement Analysis	2 weeks						
System Design		2 weeks					
Implementation			2 months				
Testing					1 month		
Deployment						1 month	
Maintenance							1 month

Figure 1.27: Al-Noor Library System , Project Timeline (Gantt chart)

5. The Six Phases of the Project:

Phase 1: Requirement Analysis (January, Weeks 1–2)

The project manager held two formal meetings with the school librarian, two teachers, and the principal. The team used structured interviews and observation to document all requirements. The following six core requirements were identified:

- The librarian must be able to add, edit, and remove book records.

- Students must be able to search for books by title or author name.
- The system must record the student name, class, book borrowed, date borrowed, and expected return date (14 days).
- The system must display an alert for any book not returned within 14 days.
- The librarian must be able to generate and print a report of all overdue books.
- The system must run on the existing school computers (Windows 10) without requiring internet access.

Deliverable: *Software Requirements Specification (SRS) document, reviewed and signed by the school principal.*



Once the principal signed the SRS document, no new requirements could be added. This is a fundamental rule of the Waterfall model, requirements are frozen after Phase 1.

Phase 2: System Design (January, Weeks 3–4)

Using the signed SRS document, the team produced a full design specification covering the database structure, screen layouts, and the technology stack.

The design decisions made during this phase were:

- Programming language: Python 3.10 with Tkinter for the graphical user interface.
- Database: SQLite, stores all data in a single file on the local computer with no server required.
- Five screens designed: Login, Home Menu, Book Management, Borrow/Return, and Reports.

Figure 1.28 shows the database schema, the structure of the two main database tables and their relationship.

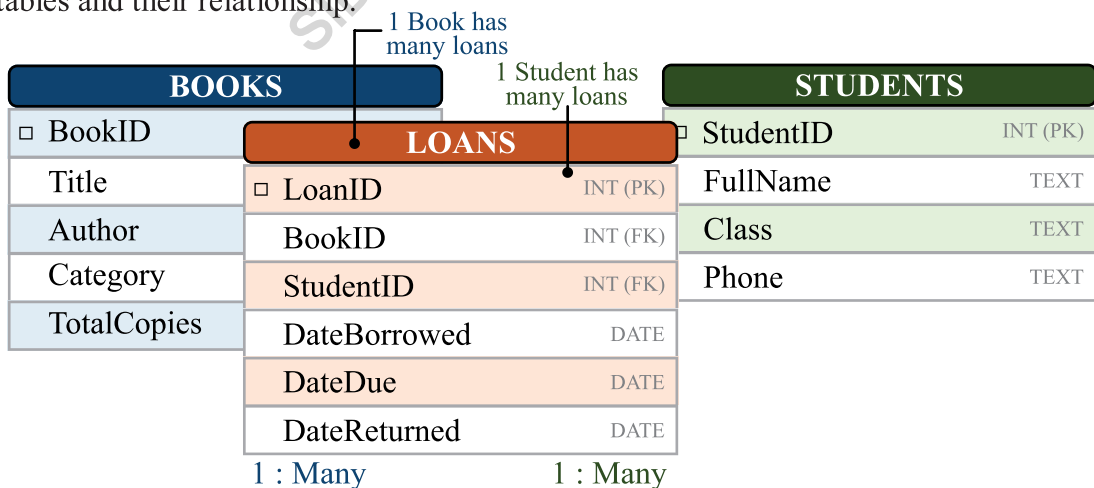


Figure 1.28: Library Management System , Database Schema

Unit 1: Computer Systems

Table 1.23 Screen Design Summary

Screen Name	Purpose	Primary User
Login Screen	Password authentication before access	Librarian
Home Menu	Navigation to all features	Librarian / Student
Book Management	Add, edit, delete, and search book records	Librarian
Borrow / Return	Record borrowing and returning of books	Librarian
Reports	View and print list of overdue books	Librarian

Deliverable: Design Document, all screen layouts, database tables, and system flowcharts.

AI-Noor Library Management System — Book Management [Librarian: Ms. Fatima]

Book Management

Add, Edit, Delete and Search all books in the library database

File Books Loans Students Reports Settings Help

Home Dashboard

Book Management

Borrow / Return

Search Books

Students

Overdue Books

Reports

Settings

Add Book Edit Book Delete Print List Import Search: Digital Logic Find

Filter by: Category: All Availability: All Class Level: All Showing 1-10 of 2,418 books

Book ID	Title	Author	Category	Total	Available	Status
BK-0001	Digital Logic and Computer Design	M. Morris Mano	Computer Sci.	4	3	Available
BK-0002	Introduction to Python Programming	Eric Matthes	Programming	3	2	Available
BK-0003	Network Fundamentals — CCNA	Todd Lammle	Networking	2	0	All Out
BK-0004	Computer Architecture & Organisation	William Stallings	Computer Sci.	3	3	Available
BK-0005	Calculus and Analytical Geometry	Thomas Finney	Mathematics	6	4	Available
BK-0006	English Grammar in Use	Raymond Murphy	English	5	5	Available
BK-0007	C++ How to Program	Deitel & Deitel	Programming	4	1	Available
BK-0008	Physics for Scientists	Serway & Jewett	Physics	4	2	Available
BK-0009	Digital Electronics — Principles	A.P. Malvino	Electronics	3	3	Available
BK-0010	Database Management Systems	Ramakrishnan	Computer Sci.	2	0	All Out

Selected Book Details: BK-0003 — Network Fundamentals — CCNA (Todd Lammle)

Book ID: BK-0003 Category: Networking

Title: Network Fundamentals — CCNA ISBN: 978-1-119-58867-4

Author: Todd Lammle Total Copies: 2

Publisher: Sybex / Wiley Borrowed: 2 (All copies out)

Year: 2019 Available: 0 All Out

Book Management | 2,418 total books | Selected: BK-0003 Ms. Fatima Ahmed | AI-Noor LMS v1.0

Figure 1.29: Book Management Screen LMS.

Phase 3: Implementation, Coding (February and March)

The two developers divided the coding work according to their specializations. Developer 1 built the database layer and all data operations (add, search, borrow, return, and generate reports). Developer 2 built all five graphical screens and connected them to Developer 1's code.

Coding standards followed during this phase:

- All source code was stored in a Git version control repository.
- Each function was documented with a comment explaining its purpose.
- The developers held a 30-minute review meeting every Friday to check progress and resolve any problems.
- Code was tested by the developer who wrote it before being passed to the other developer.

Deliverable: *Working software, Version 1.0 (internal build, not yet tested by the formal tester).*

Phase 4: Testing (April)

A dedicated tester, who had not been involved in coding, tested every feature using a formally prepared test plan. This approach is known as Black Box Testing: the tester checks inputs and outputs without examining the code itself.

Table 1.24: Test Results

Test #	Feature Tested	Expected Result	Actual Result	Status
T-01	Add a new book	Book appears in the database	Saved correctly	✓ Pass
T-02	Search by title (exact)	Correct book displayed	Book found immediately	✓ Pass
T-03	Search with misspelling	Suggestion or no result shown	System crashed	✗ Fail
T-04	Borrow a book	Loan recorded with return date	Loan saved correctly	✓ Pass
T-05	Return a book	Loan marked as returned	Return date recorded	✓ Pass
T-06	Overdue alert after 14 days	Alert displayed automatically	Alert not displayed	✗ Fail
T-07	Print overdue report	Report sent to printer	Printed correctly	✓ Pass
T-08	Login with wrong password	Access denied message shown	Access denied shown	✓ Pass

Two test failures were identified and corrected:

- T-03, Search crash: When a student entered an incomplete or misspelled title, the search function raised an unhandled exception. The fix added input validation and a 'no results found' message for unmatched searches.
- T-06, Overdue alert: The alert was coded to run only when the librarian clicked a dedicated button, not automatically. The fix added an automatic check that runs every time the application starts.
- After applying both fixes, all eight tests were repeated. All tests passed on the second run.

Deliverable: *Formal Test Report, listing every test case, its result, and the corrective action taken.*

Phase 5: Deployment (May)

The approved software was installed on two computers in the school library. A Pilot Release strategy was used: the librarian used the system alone for two weeks before it was opened to students. This allowed any remaining issues to be identified in a controlled environment.

Deployment steps completed:

- Software installed on Library Computer 1 (librarian's desk) and Library Computer 2 (student search terminal).
- The librarian entered all 2,400 existing book records into the system over three working days.
- A one-day training session was held for the librarian and two student helpers.
- A printed Quick-Reference Guide was placed next to each computer.
- After the two-week pilot with no problems, the system was opened to all students on the first day of the new term.

Deliverable: *Live, operational Library Management System installed and in use.*

Phase 6: Maintenance (June onwards)

During the first month of operation, the librarian reported a small number of issues and requested one additional feature. These were handled as maintenance tasks categorized by type.

Table 1.25: Maintenance Tasks

Issue / Request	Maintenance Type	Action Taken
Printer does not print the report on some days	Corrective	Fixed a printer driver compatibility issue in the report module
Library wants to add e-books to the same system	Adaptive	Added a new 'E-Book' category with a download-link field
Search is slow when more than 2,000 books are added	Perfective	Improved the database query speed , search now completes in under 1 second
Automatic daily data backup needed	Preventive	Added automatic backup to a USB drive every day at 5:00 PM

6. Project Outcome:

The Al-Noor Library Management System was delivered on schedule and within the agreed budget. The following results were recorded after the first term of operation:

- The system was completed in exactly 6 months as planned, with no schedule overrun.
- All 2,400 library books were entered into the database within the first week of deployment.
- Students can now locate any available book in under five seconds using the search feature.
- Overdue book returns decreased by 70% in the first term compared to the paper-based system.
- The librarian estimated a saving of approximately three hours of administrative work per week.

Key Observation: *The Waterfall model worked well in this project because the requirements were completely defined before development began and did not change during the six-month period. The model's rigid structure kept the team focused and the project on schedule.*

1.2.6. Case Study 2: Agile Model The Al-Noor Student Mobile Application

1. Background:

Following the successful deployment of the Library Management System (Case Study 1), the Al-Noor High School administration decided to develop a mobile

application for students, teachers, and parents. Unlike the library system, the requirements for the mobile application were not well defined at the outset. Different stakeholders requested different features, and the school expected requirements to evolve as the project progressed.

Because of these characteristics, unclear initial requirements, multiple stakeholders with changing priorities, and a desire to see working software at frequent intervals, the project team selected the Agile model, specifically using the Scrum framework, as the most appropriate development approach.

Table 1.26: Project Overview

Parameter	Details
Project Name	Al-Noor Student Mobile Application
Client	Al-Noor High School (Principal, Teachers, Students, Parents)
Development Team	3 Developers, 1 UI Designer, 1 Tester (Scrum team)
Sprint Duration	2 weeks per sprint
Total Sprints	5 sprints (10 weeks total)
SDLC Model	Agile Model (Scrum Framework)
Platform	Android and iOS (cross-platform using Flutter)

2. Why the Agile Model Was Selected:

The Agile model was selected for the mobile application project for the following reasons:

- **Unclear requirements:** No single stakeholder could define all the required features at the start. Requirements emerged gradually through discussions with students, teachers, and parents.
- **Rapidly changing technology:** Mobile application development involves platforms (Android, iOS) that update frequently. Agile allows the team to adapt to these changes between sprints.
- **Need for early delivery:** The school wanted to see a working application quickly, not wait six months for a complete product.
- **Continuous feedback required:** Students and teachers needed to test features and provide feedback before further development could proceed.
- **Complex project:** Integrating notifications, library booking, attendance marking, and homework submission in a single app is significantly more complex than the library system.

3. The Agile Scrum Cycle:

In the Agile Scrum framework, software is developed in short cycles called sprints. Each sprint follows the same process: planning, daily standups, development, review, and retrospective. At the end of every sprint, working software is delivered to the client for feedback. Figure 1.30 illustrates the Scrum cycle.



Figure 1.30: Agile Scrum Cycle , Sprint-Based Development Process

4. Product Backlog:

Before the first sprint, the project team conducted a requirements-gathering workshop with all stakeholders. Feature requests were written as User Stories, short statements written from the perspective of the user, and compiled into a Product Backlog. Stories were assigned a priority level (High, Medium, or Low) based on stakeholder input.

Table 1.27: Initial Product Backlog

Story ID	User Story	Priority
US-01	As a student, I want to see my class timetable on my phone.	High
US-02	As a student, I want to see my exam results when they are published.	High
US-03	As a parent, I want to receive a notification when my child is absent.	High
US-04	As a student, I want to search and reserve library books from my phone.	High
US-05	As a teacher, I want to mark student attendance on my phone.	Medium
US-06	As a student, I want to submit my homework through the app.	Medium
US-07	As a student, I want to view school announcements and upcoming events.	Medium
US-08	As a parent, I want to view my child's fees and payment history.	Low
US-09	As a student, I want a school bus tracking feature.	Low
US-10	As a teacher, I want to share study notes with students through the app.	Low

Agile Principle: *Unlike the Waterfall model, the Product Backlog is not fixed. After each sprint, new stories can be added, existing stories can be reprioritized, and completed stories are removed. The backlog evolves as the project progresses.*

5. The Five Sprints:

Sprint 1, Class Timetable and Application Setup (Weeks 1–2):

Sprint Goal: Build the basic application structure and deliver the Class Timetable feature (US-01).

Work completed during Sprint 1:

- Days 1–2: The team set up the development environment, created the application project structure, and built the login screen.
- Days 3–8: Developer 1 built the timetable display screen. Developer 2 set up the SQLite database structure. Developer 3 worked on the login authentication logic.
- Days 9–10: The tester checked the timetable feature. The designer finalized the application color scheme and typography.

Sprint Review feedback (received from the principal and five students):

"The timetable is clear and easy to read. However, could you also show the subject teacher's name next to each lesson?"

Action taken: A new sub-task was added to US-01 for inclusion in Sprint 2. No other changes were made to the sprint plan.

Table 1.28: Sprint 1 Summary

Item	Details
User Stories Completed	US-01 (Class Timetable)
Features Delivered	Login screen, Class timetable for all year groups
Bugs Found	None
Key Feedback	Add teacher names to timetable , added to Sprint 2 plan
App Installed On	10 student phones for real-world testing during Sprint 2

Sprint 2, Exam Results and Timetable Improvement (Weeks 3–4):

Sprint Goal: Deliver the Exam Results feature (US-02) and add teacher names to the timetable from Sprint 1 feedback.

Notable events during Sprint 2:

- Daily Standup meetings were held every morning. Each team member answered three questions: What did I complete yesterday? What will I work on today? Is anything blocking my progress?
- Developer 1 built the Exam Results screen. Developer 2 updated the timetable to include teacher names and room numbers.
- The tester found one bug: the results screen crashed when a student had no recorded result for a subject (caused by an unhandled null value). This was fixed on the same day.

Sprint Review feedback:

"The results screen is exactly what we needed. However, parents are now asking , can they see the full attendance record, not just notifications when their child is absent?"

Action taken: A new User Story was added to the Product Backlog, 'As a parent, I want to see my child's full weekly attendance record.' This story was scheduled for Sprint 3.

Sprint 3, Absence Notifications and Teacher Attendance Marking (Weeks 5–6):

Sprint Goal: Deliver parent absence notifications (US-03) and teacher attendance marking (US-05).

This sprint introduced the most technically challenging feature, integrating the application with the school's existing SMS gateway service.

Key events:

- The sprint planning meeting estimated that US-03 would require six development days and US-05 would require three development days.
- During development, the school's SMS provider experienced a two-day outage. The team adapted by temporarily switching to email notifications, allowing development and testing to continue without delay.
- Developer 3 built the connection between the app and the school's existing student records database, enabling real attendance data to trigger notifications automatically.
- Sprint Review results:
- 95% of test notifications were received by parents within two minutes of the teacher marking a student absent.
- Teachers reported that marking attendance on the mobile app took less than one minute per class, compared to approximately ten minutes using the previous paper register.

Sprint 4, Library Booking and School Announcements (Weeks 7- 8):

Sprint Goal: Deliver library book reservation (US-04) and school announcements (US-07).

This sprint connected the mobile application to the Library Management System built in Case Study 1.

Key events:

- The team spent the first two days reviewing the database structure of the existing library system to understand how to read and write data without disrupting the existing application.
- An API (Application Programming Interface) layer was built to allow the mobile app to query the library database without direct access.
- Sprint Review feedback: Students appreciated the book reservation feature but requested that the number of available copies be displayed. This small addition took 30 minutes and was added immediately.

Action taken: The available copies counter was added to the app before the sprint ended, a good example of Agile's ability to incorporate small changes rapidly.

Sprint 5, Homework Submission and Final Testing (Weeks 9–10):

Sprint Goal: Deliver homework submission (US-06) and complete full system testing before the official launch.

Key events:

- Students can photograph their completed homework and submit it directly through the app. Teachers receive a notification and can mark the submission as received.
- The full system test, testing all features together for the first time, identified three bugs: (1) slow loading on older Android phones, (2) images larger than 5 MB caused an upload error, and (3) the login screen did not render correctly on iOS 15.
- All three bugs were fixed within two days.
- The final Sprint Review was attended by the school principal, 20 students, 10 teachers, and 5 parents. All stakeholders approved the application for official launch.

6. Sprint Progress:

Figure 1.31 shows the number of User Stories delivered in each sprint and the cumulative total across the project. Of the original 10 User Stories in the Product Backlog, 5 high- and medium-priority stories were completed in the 10-week development period. The remaining 5 lower-priority stories were deferred to a planned second release.

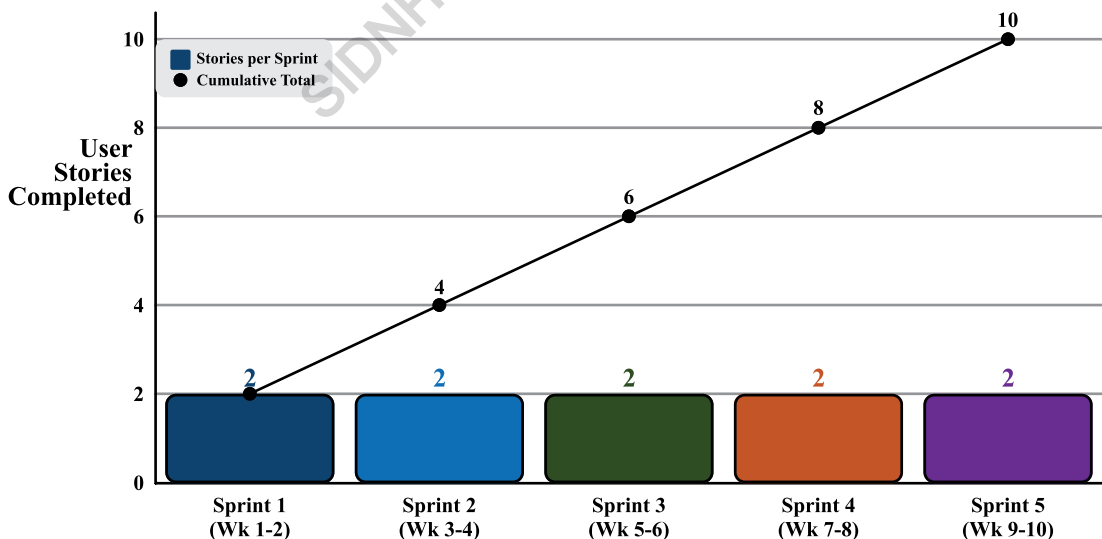


Figure 1.31: Al-Noor Student App, Sprint Progress Chart

Table 1.29: All Sprints Summary

Sprint	Weeks	User Stories Delivered	Key Change Introduced
Sprint 1	1–2	US-01 (Timetable)	Teacher names added to timetable (from review feedback)
Sprint 2	3–4	US-02 (Exam Results)	New story added: full attendance record for parents
Sprint 3	5–6	US-03, US-05 (Notifications, Attendance)	Switched to email during SMS outage; SMS restored
Sprint 4	7–8	US-04, US-07 (Library, Announcements)	Available copies counter added (30-minute change)
Sprint 5	9–10	US-06 (Homework Submission)	Full system test; 3 bugs found and fixed; app launched

7. Project Outcome:

The Al-Noor Student Mobile Application was officially launched at the school assembly at the end of Week 10.

- The application was downloaded by 1,100 students, 80 teachers, and 900 parents within the first week of launch.
- Teacher attendance marking reduced administrative time from approximately 10 minutes to under 1 minute per class.
- The parent notification system resulted in a 45% reduction in unexplained absences within the first month.
- Five of the ten original User Stories were delivered. The remaining five stories were planned for a second release six months later.
- The project received a regional award for 'Best School Technology Initiative'.

Key Observation: *The Agile model's greatest advantage in this project was its ability to incorporate feedback and changing requirements between sprints , for example, adding teacher names to the timetable, creating the full attendance record feature, and adding the book availability counter , without disrupting the overall project plan.*

1.2.7. Comparative Analysis: Waterfall vs Agile

1. Side-by-Side Comparison:

The two case studies illustrate how the same school chose different SDLC models for two different projects, based on the specific characteristics of each project. Table 1.30 compares the two models across the key dimensions observed in this study.

Table 1.30: Waterfall vs Agile , Comparison Based on Case Studies

Dimension	Waterfall (Case Study 1)	Agile (Case Study 2)
Requirements	Fully defined and fixed before coding began	Partially defined; evolved throughout the project
Client involvement	At the start (SRS sign-off) and at the end (delivery)	After every sprint , continuous throughout the project
First working product	After 6 months (full system at deployment)	After 2 weeks (basic login + timetable after Sprint 1)
Response to change	Difficult , changes require going back to Phase 1	Easy , changes are incorporated in the next sprint
Documentation	Heavy SRS, Design Document, Test Report required	Light , emphasis on working software over documentation
Testing	One dedicated phase (April) after all coding complete	Continuous , testing performed within every sprint
Team size	Small (4 members) , easier to manage linearly	Larger (5 members) , Scrum roles and ceremonies required
Risk	Higher , bugs found late are expensive to fix	Lower , problems identified and fixed every sprint
Final result	Complete system delivered at end of 6 months	Partial system delivered after 10 weeks; more to follow
Suitable when	Requirements are fixed and fully understood at the start	Requirements are unclear or expected to change

2. Comparative Analysis Chart:

Figure 1.31 compares the two models across six criteria on a scale of 1 to 5. A higher score indicates better performance on that criterion for the given model. Note that 'Documentation' is scored inversely, a lower score means less documentation required.

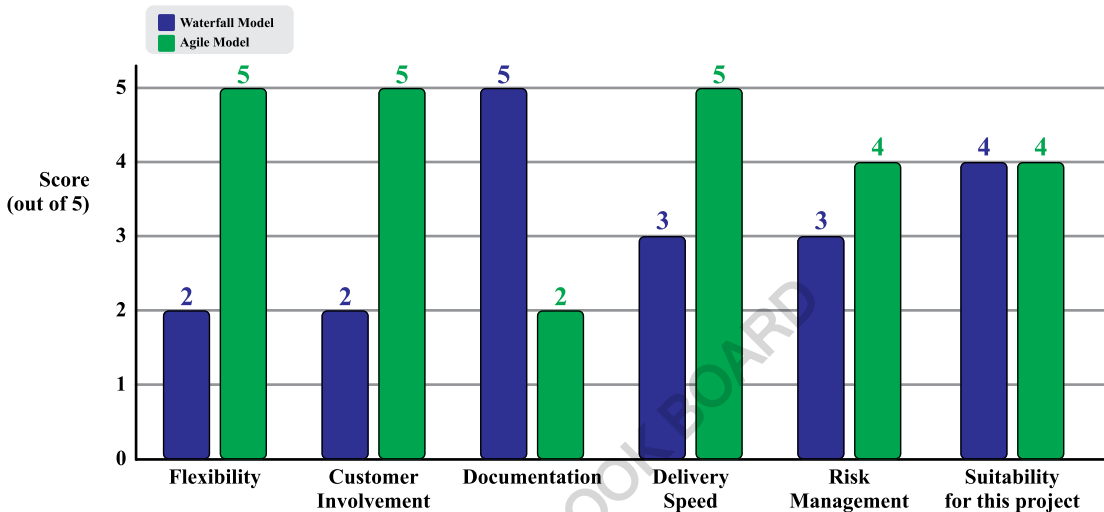


Figure 1.31: Waterfall vs Agile , Comparison by Criteria (Score out of 5)

1.3 Communication Models:



1.3.1 What is a Communication Model?

A communication model is a set of rules and layers that explains how data (information) travels from one device to another through a network. Think of it like the rules of a postal system; both the sender and the receiver must follow the same steps for the letter to arrive correctly.

Why do we need communication models?

- They allow computers from different companies and countries to talk to each other.
- They make it easier to find and fix problems in a network.
- They divide a complex task into smaller, manageable steps called layers.
- They allow engineers to design new technologies without breaking existing ones.

Analogy: Sending data over a network is like sending a letter by post. You write the message, put it in an envelope, write the address, hand it to the post office, which sorts it and delivers it. Each step is like a layer in a communication model.

The two most important communication models are:

- OSI Model (Open Systems Interconnection), a 7-layer reference model used mainly for learning and design.
- TCP/IP Model (Transmission Control Protocol / Internet Protocol), a 4-layer model actually used on the Internet today.

1.3.2 The OSI Model:

The Open Systems Interconnection (OSI) Model



Figure 1.32: 7 layers of OSI, protocols and devices at each layer

The OSI model was developed in 1984 by the International Organization for Standardization (ISO). OSI model provides a universal framework that describes how different computer systems can communicate with each other. The OSI model divides network communication into seven separate layers. Each layer has a specific job, and each layer only communicates with the layers directly above and below it.

Memory Tip, Layers 1 to 7 (bottom to top):

Mnemonic: "Please Do Not Throw Sausage Pizza Away" , P=Physical, D=Data Link, N=Network, T=Transport, S=Session, P=Presentation, A=Application

Table 1.31: OSI Model, Seven Layers Overview

Layer #	Layer Name	Key Protocols / Technologies	Main Job (Simple Description)
7	Application	HTTP, HTTPS, FTP, SMTP, DNS	Provides network services to user applications
6	Presentation	SSL/TLS, JPEG, MP3, ASCII	Translates, encrypts, and compresses data
5	Session	NetBIOS, RPC	Opens, maintains, and closes communication sessions
4	Transport	TCP, UDP	Reliable delivery, error checking, port numbers
3	Network	IP, ICMP, RIP, OSPF	Logical addressing (IP) and routing between networks
2	Data Link	Ethernet, Wi-Fi (802.11), MAC	Physical addressing (MAC) and error detection in frames
1	Physical	Cables, Hubs, Repeaters, Radio	Transmits raw bits over the physical medium

1.3.3 The Seven Layers of the OSI Model in Detail:

In the following sections, each layer is explained in detail with its role, a real-life analogy, the protocols it uses, and its key functions.

1. Layer 7, Application Layer:

The Application Layer is the topmost layer and the one closest to the end user. It provides an interface between the user's application programs (such as a web browser or email client) and the network. This layer does not refer to the application program itself, but to the networking services that the application uses to communicate.

Real-life analogy: *Ordering food at a restaurant, the menu (application layer) is what you interact with directly, even though the kitchen (lower layers) does the actual work.*

Key functions:

- Provides network services directly to user applications (web browsing, email, file transfer).
- Identifies communication partners and ensures sufficient resources are available.
- Supports protocols such as HTTP (web), FTP (file transfer), SMTP (email), and DNS (name resolution).

Table 1.32: Protocols at the Application Layer (Layer 7)

Protocol	Full Name	Purpose
HTTP	HyperText Transfer Protocol	Transfers web pages from server to browser (unencrypted)
HTTPS	HTTP Secure	Same as HTTP but encrypted using TLS
FTP	File Transfer Protocol	Transfers files between computers and servers
SMTP	Simple Mail Transfer Protocol	Sends outgoing email to a mail server
POP3	Post Office Protocol version 3	Retrieves incoming email from a mail server
IMAP	Internet Message Access Protocol	Manages and retrieves email on the server
DNS	Domain Name System	Translates domain names (e.g. google.com) into IP addresses
Telnet	Telnet	Provides remote login access to another computer
SNMP	Simple Network Management Protocol	Monitors and manages network devices

2. Layer 6, Presentation Layer:

The Presentation Layer acts as a translator for the network. It ensures that data sent by one computer can be understood by another, even if they use different data formats. This layer is also responsible for encryption (making data secret) and compression (making data smaller for faster transmission).

Real-life analogy: *A translator at an international meeting who converts one language into another so everyone understands.*

Key functions:

- Translation: Converts data between different formats (e.g. EBCDIC to ASCII, or Unicode to UTF-8).
- Encryption: Scrambles data before transmission so it cannot be read by unauthorised parties (e.g. TLS/SSL).
- Decryption: Unscrambles encrypted data received from the network.
- Compression: Reduces the size of data to speed up transmission (e.g. ZIP, JPEG, MP3).
- Decompression: Restores compressed data to its original size at the receiver.

Protocols and standards at Layer 6:

- SSL/TLS (used for HTTPS secure websites)
- JPEG, PNG (image formats)
- MP3, MPEG (audio and video compression)
- ASCII, Unicode (character encoding standards)

3. Layer 5, Session Layer:

The Session Layer manages communication sessions between two computers. A session is a connection that is established, maintained, and eventually terminated between two devices. This layer ensures that a conversation, once started, can be continued, paused, and resumed in an orderly way.

Real-life analogy: *A telephone call, you say "Hello" to start, speak until you are done, and say "Goodbye" to end. The session layer handles this opening, maintenance, and closing.*

Key functions:

- Session Establishment: Sets up a session between two communicating devices.
- Session Maintenance: Keeps the session active and coordinates data exchange.
- Session Termination: Closes the session properly when communication ends.
- Checkpointing: Saves the progress of a data transfer so it can resume if interrupted, avoiding the need to restart from the beginning.
- Dialog Control: Manages whether communication is half-duplex (one direction at a time) or full-duplex (both directions simultaneously).

Protocols at Layer 5:

- NetBIOS (Network Basic Input/Output System)
- RPC (Remote Procedure Call)
- PPTP (Point-to-Point Tunneling Protocol)

4. Layer 4, Transport Layer:

The Transport Layer is responsible for reliable end-to-end communication between two devices on a network. It breaks large messages into smaller pieces called segments, ensures they all arrive correctly, and reassembles them at the destination. It also uses port numbers to identify which application should receive the data.

Real-life analogy: *A courier company that guarantees your parcel arrives safely and completely, and tracks each package so nothing is lost.*

Key functions:

- Segmentation: Divides large data into smaller segments for transmission.
- Reassembly: Puts segments back together in the correct order at the destination.
- Error Control: Detects missing or damaged segments and requests retransmission.
- Flow Control: Prevents a fast sender from overwhelming a slow receiver.
- Port Numbering: Uses port numbers to route data to the correct application (e.g. port 80 for HTTP, port 443 for HTTPS, port 21 for FTP, port 25 for SMTP).

Table 1.33: TCP vs UDP at the Transport Layer (Layer 4)

Protocol	Full Name	Type	When to Use
TCP	Transmission Control Protocol	Connection-oriented (reliable)	Web browsing, email, file transfer , where accuracy matters
UDP	User Datagram Protocol	Connectionless (fast)	Video streaming, online gaming, live calls , where speed matters more than perfection

5. Layer 3, Network Layer:

The Network Layer is responsible for logical addressing and routing. Every device connected to the Internet has an IP address, which works like a home address. The Network Layer decides the best route for data to travel from the source device to the destination device, even across many different networks. Routers operate at this layer.

Real-life analogy: *A GPS navigation system that finds the best route from your home to a destination, even if the route passes through many towns.*

Key functions:

- Logical Addressing: Assigns IP addresses to every device (IPv4: 32-bit, e.g. 192.168.1.1 | IPv6: 128-bit, e.g. 2001:db8::1).
- Routing: Determines the best path for packets to travel from source to destination across multiple networks.

- **Packet Forwarding:** Routers read the destination IP address and forward each packet toward its destination.
- **Fragmentation:** Breaks packets into smaller pieces when required by the network, and reassembles them at the destination.
- **Error Reporting:** ICMP sends error messages (e.g. the "ping" command uses ICMP to test connectivity).

Table 1.34: Protocols at the Network Layer (Layer 3)

Protocol	Full Name	Purpose
IPv4	Internet Protocol version 4	32-bit addressing; most widely used IP version
IPv6	Internet Protocol version 6	128-bit addressing; designed to replace IPv4
ICMP	Internet Control Message Protocol	Error reporting and network diagnostics (ping)
ARP	Address Resolution Protocol	Resolves IP addresses to MAC addresses on a local network
RIP	Routing Information Protocol	Distance-vector routing protocol for small networks
OSPF	Open Shortest Path First	Link-state routing protocol for larger networks
BGP	Border Gateway Protocol	Routes data between large networks and ISPs on the Internet

6. Layer 2, Data Link Layer:

The Data Link Layer is responsible for communication between two directly connected devices on the same local network (LAN). It organises data into frames, adds physical (MAC) addresses, and detects errors in transmission. Network switches and wireless access points operate at this layer. The Data Link Layer has two sub-layers: LLC (Logical Link Control) and MAC (Media Access Control).

Real-life analogy: *A local postal worker who delivers letters within the same town using house numbers (MAC addresses) as identifiers.*

Key functions:

- **Framing:** Organises bits from the Physical Layer into structured blocks called frames.
- **Physical (MAC) Addressing:** Adds the sender's and receiver's MAC addresses to each frame. A MAC address is a unique hardware address permanently assigned to a network interface card (e.g. AA:BB:CC:DD:EE:FF).
- **Error Detection:** Uses CRC (Cyclic Redundancy Check) to detect frames damaged during transmission.

- Flow Control: Prevents a fast sender from overwhelming a slow receiver on the local network.
- Media Access Control: Manages which device may use the shared network medium at a given time (e.g. CSMA/CD for Ethernet, CSMA/CA for Wi-Fi).

Protocols and technologies at Layer 2:

- Ethernet (IEEE 802.3) , wired local area network standard
- Wi-Fi (IEEE 802.11) , wireless local area network standard
- PPP (Point-to-Point Protocol) , used for direct connections
- MAC addresses , unique hardware identifiers for network interfaces
- Switches , the main hardware device at this layer

7. Layer 1, Physical Layer:

The Physical Layer is the lowest layer of the OSI model. It deals with the actual physical transmission of raw bits (0s and 1s) over a physical medium. This includes electrical signals over copper cables, light pulses over fibre-optic cables, and radio waves for wireless communication. The Physical Layer defines the hardware: cables, connectors, voltage levels, frequencies, and data rates.

Real-life analogy: *The roads and highways that vehicles travel on , without roads, no delivery is possible, regardless of how well the postal system is organised.*

Key functions:

- Bit Transmission: Sends and receives raw bits (0s and 1s) over the physical medium.
- Physical Medium: Defines the type of cable or wireless signal used (copper wire, fibre optic, radio waves, infrared).
- Signal Encoding: Converts bits into electrical signals, light pulses, or radio waves for transmission.
- Data Rate (Bandwidth): Defines the transmission speed (e.g. 100 Mbps, 1 Gbps).
- Physical Topology: Defines the physical arrangement of devices (star, bus, ring, mesh).
- Connectors and Cables: Specifies plug types, cable categories (Cat5, Cat6), and connector standards.

Hardware devices at Layer 1:

- Hubs , broadcast data to all connected devices
- Repeaters , amplify and retransmit signals over long distances
- Network cables (Cat5e, Cat6, Cat7 Ethernet cables)
- Fibre-optic cables , transmit data as pulses of light
- Network Interface Cards (NICs) , the hardware that connects a device to the network

1.3.4 How Data Travels Through the OSI Layers:

When a user sends data (for example, loading a webpage), the data travels downward through all seven layers on the sender's device. At each layer, additional information called a header is added. This process is called encapsulation. When the data arrives at the destination, it travels upward through all seven layers. At each layer, the corresponding header is read and removed. This reverse process is called decapsulation.

Table 1.35: Encapsulation , Data Travelling Down the OSI Layers (Sending, top to bottom)

Layer	Name	What Happens	Data Unit Name
7	Application	User creates data (e.g. types URL). Application header added.	Data
6	Presentation	Data is encrypted and/or compressed. Presentation header added.	Data
5	Session	Session is opened. Session header added.	Data
4	Transport	Data split into segments. TCP/UDP header added (includes port numbers).	Segment
3	Network	IP header added (source and destination IP addresses). Segment becomes a packet.	Packet
2	Data Link	MAC header and trailer added (MAC addresses, error check). Packet becomes a frame.	Frame
1	Physical	Frame converted to electrical signals/light/radio waves and sent over the medium.	Bits

Decapsulation, Data Travelling Up the OSI Layers (Receiving, bottom to top):

At the receiving device, the process is reversed. The Physical Layer receives the raw signals and passes them up as bits. Each layer reads and removes its own header, until the original data reaches the Application Layer and is presented to the user.

Key point: The data unit at each layer has a specific name: Data (Layers 7-5), Segment (Layer 4), Packet (Layer 3), Frame (Layer 2), Bits (Layer 1).

1.3.5 The TCP/IP Model:

The TCP/IP model (Transmission Control Protocol / Internet Protocol) is the practical model that the Internet actually uses today. It was developed in the 1970s by the United States Department of Defense. While the OSI model is used mainly as a reference for learning and design, the TCP/IP model is the one running on every device connected to the Internet, including your smartphone, computer, and smart television.

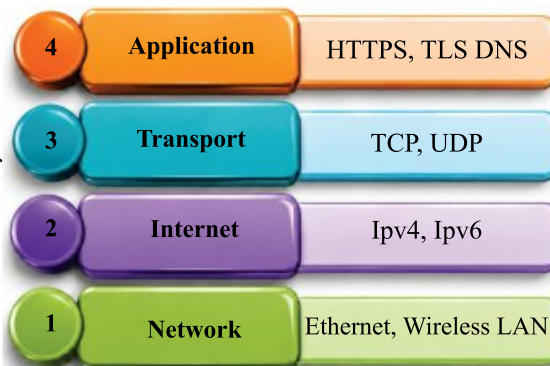


Figure 1.33: 4 layers of TCP/IP model and protocols

Key fact: Every device that connects to the Internet uses the TCP/IP model. When you visit a website, send an email, or make a video call, TCP/IP is working in the background at every step.

The TCP/IP model has four layers. Each layer corresponds to one or more layers of the OSI model:

Table 1.36: TCP/IP Model Layers and Corresponding OSI Layers

TCP/IP Layer	Corresponding OSI Layer(s)	Key Protocols
Application	Application (7), Presentation (6), Session (5)	HTTP, HTTPS, FTP, SMTP, DNS, IMAP, POP3
Transport	Transport (4)	TCP, UDP
Internet	Network (3)	IPv4, IPv6, ICMP, ARP, OSPF, BGP
Network Access	Data Link (2), Physical (1)	Ethernet, Wi-Fi (802.11), Cables, MAC

1. Application Layer (TCP/IP):

The TCP/IP Application Layer combines the functions of the OSI Application, Presentation, and Session layers into a single layer. It provides all the services that user-facing applications need to communicate over the network, including data formatting, encryption, session management, and the actual application protocols (HTTP, FTP, SMTP, DNS, etc.).

2. Transport Layer (TCP/IP):

Identical in function to OSI Layer 4. The Transport Layer provides end-to-end communication services. It uses TCP for reliable, connection-oriented delivery and UDP for fast, connectionless delivery. Port numbers are used at this layer to identify applications.

3. Internet Layer (TCP/IP):

The Internet Layer corresponds to the OSI Network Layer. It is responsible for logical (IP) addressing and routing packets across multiple networks. The Internet Protocol (IP) is the core protocol at this layer. Routers operate at the Internet Layer to forward packets toward their destinations.

4. Network Access Layer (TCP/IP):

The Network Access Layer (also called the Link Layer) combines the OSI Data Link and Physical layers. It handles everything related to the local physical network: MAC addressing, framing, error detection, and the actual physical transmission of bits over cables or wireless signals. Ethernet, Wi-Fi, and the underlying hardware all operate at this layer.

1.3.6 Comparing the OSI and TCP/IP Models:

The OSI and TCP/IP models both describe how network communication works, but they differ in the number of layers and level of detail. The table below summarises the key similarities and differences.

Table 1.37: OSI vs TCP/IP , Comparison

Aspect	OSI Model	TCP/IP Model
Number of layers	7	4
Developed by	ISO (1984)	US Department of Defense (1970s)
Primary use	Reference model for learning and design	Practical model used on the Internet
Application layer	Three separate layers: Application, Presentation, Session	Combined into one Application layer
Transport layer	Layer 4 (Transport)	Transport layer (TCP/UDP)
Network layer	Layer 3 (Network)	Internet layer (IP)
Lower layers	Two separate layers: Data Link, Physical	Combined into Network Access layer
Protocol independence	Model is defined first; protocols fit into it	Model was built around existing protocols
Still used today?	Used for teaching and troubleshooting	Yes , powers the Internet

OSI Model

VS

TCP/IP Model



Both models share the same Transport and Network (Internet) concepts. The main difference is that TCP/IP collapses the top three OSI layers into one Application layer, and the bottom two OSI layers into one Network Access layer.



Career Opportunities

This unit covers digital logic, logic gates, the software development life cycle (SDLC), and how computers communicate through networks and the OSI model. Mastery of these foundational concepts opens the door to a wide range of technical careers in hardware design, software engineering, and networking.

Hardware / Digital Logic Engineer

Designs and tests digital circuits and logic gates used inside computers, calculators, and embedded devices.

Software Engineer / Developer

Applies SDLC models such as Waterfall and Agile to design, build, and maintain reliable software applications.

Network Engineer / Administrator

Designs, configures, and maintains computer networks using the OSI and TCP/IP models to keep organizations connected.

Systems Analyst

Studies business needs and translates them into technical requirements using structured development methodologies.

Embedded Systems Engineer

Builds the logic and firmware that power smart appliances, vehicles, and IoT devices.

IT Support / Network Technician

Installs, troubleshoots, and maintains hardware, cabling, and networked systems for businesses and institutions.



Summary

- ◆ **Discrete signals** have countable values, while **continuous signals** can take any value within a range.
- ◆ **Digital systems** use only binary values (0 and 1) and form the basis of modern electronics.
- ◆ **Analog signals** vary continuously, whereas **digital signals** use discrete values.
- ◆ The **binary number system** uses only the digits 0 and 1.
- ◆ **Boolean algebra** performs operations using two logical values: TRUE (1) and FALSE (0).
- ◆ The basic **Boolean operations** are AND, OR, and NOT.
- ◆ **Logic gates** are electronic circuits that implement Boolean operations.

- ◆ **Truth tables** show all possible input and output combinations of a logic circuit.
- ◆ **Logic expressions** represent logical operations using variables and operators.
- ◆ **Logic diagrams** provide a graphical representation of digital circuits.
- ◆ **Software** is a set of instructions that directs a computer to perform tasks.
- ◆ The **Software Development Life Cycle (SDLC)** is a structured process for planning, developing, testing, deploying, and maintaining software.
- ◆ Common **SDLC models** include the **Waterfall model**, which follows a sequential approach, and the **Agile model**, which emphasizes iterative development and continuous improvement.
- ◆ **Software testing** includes **Black Box Testing** (external testing) and **White Box Testing** (internal testing).
- ◆ **Software maintenance** involves corrective, adaptive, perfective, and preventive updates after deployment.
- ◆ In the **communication model**, data is transmitted from a sender to a receiver through a communication channel.
- ◆ The **OSI model** consists of seven layers: Physical, Data Link, Network, Transport, Session, Presentation, and Application.
- ◆ The **TCP/IP model** consists of four layers: Network Access, Internet, Transport, and Application.
- ◆ **Encapsulation** is the process of adding protocol headers to data as it moves down the network layers.
- ◆ **Decapsulation** is the process of removing headers as data moves up the receiving layers.
- ◆ The **Physical layer** transmits raw bits over communication media.
- ◆ The **Data Link layer** handles MAC addressing, framing, error detection, and hardware communication.
- ◆ The **Network layer** performs IP addressing and routing.
- ◆ The **Transport layer** ensures end-to-end communication using **TCP** or **UDP**.
- ◆ The **Session layer** establishes, manages, and terminates communication sessions.
- ◆ The **Presentation layer** translates, encrypts/decrypts, and compresses/decompresses data.
- ◆ The **Application layer** provides network services for applications such as HTTP, HTTPS, FTP, SMTP, and DNS.
- ◆ **TCP** is connection-oriented and reliable, while **UDP** is connectionless and faster with lower overhead.
- ◆ An **IP address** is a logical address used to identify devices on a network.
- ◆ A **MAC address** is a unique physical hardware address permanently assigned to a network interface.



Exercise

1. Encircle the correct answer:

- i) A digital signal has:
(a) Infinite values (b) Only two values
(c) Three values (d) Ten values
- ii) Which gate gives output 1 only when all inputs are 1?
(a) OR gate (b) AND gate
(c) NOT gate (d) NOR gate
- iii) XOR gate gives output 1 when inputs are:
(a) Same (b) Different
(c) Both 0 (d) Both 1
- iv) Which phase comes first in SDLC?
(a) Testing (b) Requirement Analysis
(c) Coding (d) Design
- v) In Agile, a short development cycle is called:
(a) Phase (b) Sprint
(c) Module (d) Release
- vi) Which testing type does NOT require knowledge of code?
(a) White box testing (b) Black box testing
(c) Unit testing (d) Code review
- vii) What device operates at the Network layer (Layer 3) of the OSI model?
(a) Hub (b) Switch
(c) Router (d) Repeater
- viii) How many layers does the TCP/IP model have?
(a) 3 (b) 4
(c) 5 (d) 7
- ix) UDP is preferred over TCP for video streaming because:
(a) UDP is more reliable (b) UDP is slower but more accurate
(c) UDP is faster and has lower delay (d) UDP guarantees delivery
- x) Which OSI layer is responsible for routing packets between networks?
(a) Physical (b) Data Link
(c) Network (d) Transport

2. Fill in the Blanks:

- i) An _____ gate gives output 1 only when both the inputs are same.
- ii) The _____ gate is also called an inverter.
- iii) In Agile, a short development cycle is called a _____.

- iv) SRS stands for _____.
- v) UDP stands for _____.

3. Provide short descriptive answers of the following questions.

- i) Define digital logic and give two examples of digital devices.
- ii) What is the difference between analog and digital signals?
- iii) Compare Waterfall and Agile models. When would we use each?
- iv) What is the difference between an IP address and a MAC address?
- v) Explain the difference between TCP and UDP. Give one real-life example of each.

4. Provide detailed answers of the following questions.

- i) Explain all seven logic gates with their symbols, Boolean expressions, and truth tables.
- ii) Draw the truth table for the expression: $Y = A' \cdot B + C$
- iii) Explain all phases of SDLC in detail with activities and deliverables.
- iv) Explain each layer of the OSI model in detail. For each layer, state its main function, one real-life analogy, and two protocols used.
- v) Describe the four types of software maintenance with examples.



Activities

Activity 1. Create a truth table for $Y = A \cdot B \cdot C + A' \cdot B$

Activity 2. Design a logic circuit for a simple burglar alarm that sounds when EITHER the door OR window is opened.

Activity 3. Use an online LogiSim to verify your truth tables.

Activity 4. Create an SRS document for a simple calculator app.

Activity 5. Draw a flowchart for a program that checks if a number is even or odd.

Activity 6. Open the command prompt (Windows) or terminal (Mac/Linux) and type: `ping www.google.com`. Record the results. Which OSI layer and protocol is being used? What does the output tell you about network connectivity?

Activity 7. Draw a diagram showing the seven layers of the OSI model. For each layer, write its name, main function, and at least one protocol. Use the mnemonic "Please Do Not Throw Sausage Pizza Away" to help you remember the order.

Activity 8. Create a comparison table of TCP and UDP, listing at least five differences. For each protocol, give two examples of applications that use it and explain why.

Unit

2

Computational Thinking & Algorithm

Student Learning Outcomes:

After the completion of this unit students will be able to:

- ◆ Analyze real-world problems and apply computational thinking to solve them.
- ◆ Identify and analyze real-world problems.
- ◆ Understand the importance of Algorithms in problem-solving.
- ◆ List the properties of a Good Algorithm.
- ◆ Illustrate Counting and Simple Computational Problems.
- ◆ Logical Reasoning for Algorithmic Thinking.
- ◆ Understand and apply common Search Algorithms (any two).
- ◆ Understand and apply common Sorting Algorithms (any two).
- ◆ Understand when and how to apply specific algorithms to solve problems.
- ◆ Applying algorithmic strategies (abstraction, decomposition, pattern recognition).



2.1. Introduction to Unit:

Computational Thinking is a systematic approach to solving problems, involving breaking them down, identifying patterns, abstracting details, and developing algorithms. In this unit, you will learn about problem solving, i.e., analyzing a problem and designing algorithms.



Pseudocode is a human-readable representation of an algorithm using plain English and structured programming keywords.

2.1 Algorithm:

An **algorithm** is a step-by-step logical procedure used to solve a problem. It is conceptual in nature and completely independent of any programming language.

2.1.1 Importance of Algorithm:

Algorithms are vital because they provide clear and efficient solutions. Without algorithms, we would be forced to reinvent the wheel every time we faced a challenge.

- Algorithms provide a structured approach to problem-solving, making complex tasks easier to understand and execute.
- Algorithms help in selecting solutions that optimize time and memory usage, which is crucial for large-scale applications.
- Once developed, an algorithm can be reused for similar problems, saving both time and effort.
- Algorithms ensure accuracy and consistency, producing correct results when followed properly and reducing the likelihood of human error.
- Every computer program is built upon algorithms; they serve as a blueprint before actual code implementation.

2.1.2 Difference between Algorithm and Pseudocode:

Algorithm	Pseudocode
A step-by-step logical procedure to solve a problem.	A human-readable representation of an algorithm using plain English and structured programming keywords.
Conceptual and language-independent.	Semi-formal and structured but not executable code.
To define what needs to be done logically.	To present how it can be implemented in a readable format before actual coding.

Usually described in plain steps, bullets, or numbered points.	Written using structured constructs (IF, FOR, WHILE) like programming but without syntax.
Focuses on the logic and steps of the problem.	Focuses on how to represent those steps clearly before coding.
Anyone for trying to understand the logic, including non-programmers.	Mainly for programmers or students preparing for actual coding.
No fixed format; can be informal.	Follows standard pseudocode conventions for clarity.

2.2 Algorithmic Strategies for Problem Solving:

2.2.1 Decomposition:

It is a problem-solving technique in which a complex task or system is broken down into smaller, manageable parts. This approach is a key concept in computational thinking and software development, and it is equally useful in real-life learning situations. By dividing a big problem into simpler steps, it becomes easier to understand, develop, and solve effectively.

Real-Life Example: Learning a New Language:

When a person learns a new language (e.g., Spanish, French, English), they do not learn everything at once. They **break the process into parts**, such as:

Break it into Subtasks:

Subtask	Description
Learn Vocabulary	Learn character sets for people, actions, things (e.g., "I", "eat", "apple")
Understand Grammar Rules	Learn structure: Subject → Verb → Object
Learn Tenses	Present, past, future (e.g., "I eat" vs. "I ate")
Practice Sentence Building	Combine words using rules to create valid sentences
Listen & Imitate	Observe native speakers to improve accuracy
Get Feedback	Check if sentence makes sense (from a teacher or app)

Why Decomposition Is Useful:

Benefit	Description
Reduces complexity	Focus on one piece at a time
Makes learning manageable	Easier to practice in steps
Helps in debugging	Find exactly where the mistake is (e.g., verb tense or word order)
Promotes reusability	Once you learn how to form a sentence, you can apply it to new words

2.2.2 Pattern Recognition:

It is the process of identifying regularities, repetitions, or rules in data. In programming and algorithm design, this helps you to

- Discover meaningful relationships among elements.
- Reduce overall algorithm complexity
- Develop generic and reusable solutions

This idea forms the foundation of concepts such as loops, conditions, and data structure design.

How It Works:

You observe input data and ask:

- What is repeating?
- What changes in a predictable way?
- Can I generalize this?

Example:

Consider the triangle of stars, where the first row shows 1 star, the 2nd row shows 2 stars, the 3rd row shows 3 stars, and so on.

```

*
**
***
****
*****

```

In the above example, the pattern shows that each row adds 1 star to the previous row's count.

2.2.3 Abstraction:

Abstraction means hiding unnecessary features and focusing only on the essential ones needed to solve a problem. In simple terms, it reduces algorithm complexity by ignoring what is not important and concentrating on what truly matters.

Example:

Let's apply abstraction to "Making Tea" by identifying:

1. What is necessary?
2. What can change based on situation?
3. What can we ignore?

Necessary Steps of making tea:

These are the minimum steps required, regardless of tea type:

1. Boil water
2. Add tea leaves or tea bag
3. Steep for a few minutes
4. Pour into cup
5. Serve

This is the abstracted process; the key logic of "making tea" in all cases.



Figure No. 2.1: Steps of making tea

Changes based on situation:

These details can vary depending on:

Ingredients: Black tea, Green tea, Herbal, Milk tea

Additions: Sugar, honey, milk, lemon

Tools: Electric kettle, stove, Tea pot

Time: Boiling time, steeping time

These are the inputs or parameters in an algorithm.

Abstraction items:

These are not important for understanding the main logic:

- Brand of kettle
- Cup color or design
- Where the tea is being made (kitchen, office, etc.)
- Type of mug (glass, ceramic)

These details add noise and are excluded from abstraction.

2.3 Sorting and Searching Algorithms:

Both are the fundamental techniques of computer science and data analysis. Both work on lists of the same data type, such as a student's name, father's name, or age.

Sorting is the process of arranging a collection of items or data. It may be ascending (lowest to highest → 0-9, a-z, A-Z), descending (highest to lowest → 9-0, z-a, Z-A) or chronological order. It is used to organize and make data more accessible for searching, retrieval, and analysis.

Searching algorithms are designed to locate a specific element within list of data, like student's name in the class list. These algorithms help to find the position, existence, or value of a target element in a more efficient and systematic manner.

Both are used in real-life computational problems such as:

- Database queries
- Online shopping filters
- Leaderboards
- File organization
- Recommendation systems

2.3.1 Sorting Algorithms:

a. Bubble Sort:

Bubble sort is an elementary sorting algorithm used to arrange elements in a list or array in either ascending or descending order. It is also known as ripple sort. It works

Unit 2: Computational Thinking & Algorithm

by repeatedly comparing adjacent elements and swapping them if they are in the wrong order. This process is repeated until the entire list is sorted.

Algorithm:

```
Step 1: Start
Step 2: Set DATA := {8, 4, 1, 9, 3}. [Any Input array data]
Step 3: Set N := 5. [Total Number of items in DATA array]
Step 4: Repeat Step 5 and Step 6 for K = 1 to N-1.
Step 5:   Set PTR := 1. [Initializes pass pointer PTR]
Step 6:   Repeat while PTR ≤ N-K: [Executes pass]
           a: If DATA[PTR] > DATA[PTR + 1], then:
               1. Set temp := DATA[PTR]
               2. Set DATA[PTR] := DATA[PTR + 1]
               3. Set DATA[PTR + 1] := temp
           [End of If structure]
           b: Set PTR := PTR + 1.
           [End of Step 6 inner loop]
       [End of Step 4 outer loop]
Step 7: End
```

Pseudocode:

```
Start
DATA = {8, 4, 1, 9, 3} /* Array of an Integer data */
N = 5 /* Total Number of Items in Array */
For outerloop = 0 to N-1 /* Outer Loop */
    For innerloop = 0 to N-2-outerloop /* Inner Loop */
        IF DATA[innerloop] > DATA[innerloop + 1] then
            /* Swap Data */
            temp = DATA[innerloop]
            DATA[innerloop] = DATA[innerloop + 1]
            DATA[innerloop + 1] = temp
        End If
    End For /* End of Inner Loop */
End For /* End of Outer Loop */
End
```

Example:

Unsorted List: 8, 4, 1, 9, 3

First Step:

8	4	1	9	3	1 st (8) is greater than 2 nd (4) → Swap the data.
4	8	1	9	3	2 nd (8) is greater than 3 rd (1) → Swap the data.
4	1	8	9	3	3 rd (8) is not greater than 4 th (9) → No change
4	1	8	9	3	4 th (9) is greater than 5 th (3) → Swap the data.
4	1	8	3	9	(9) is at its place.

Second Step:

4	1	8	3	9	1 st (4) is greater than 2 nd (1) → Swap the data.
1	4	8	3	9	2 nd (4) is not greater than 3 rd (8) → No Change.
1	4	8	3	9	3 rd (8) is greater than 4 th (3) → Swap the data.
1	4	3	8	9	(8) is at its place.

Third Step:

1	4	3	8	9	1 st (1) is not greater than 2 nd (4) → No Change
1	4	3	8	9	2 nd (4) is greater than 3 rd (3) → Swap the data.
1	3	4	8	9	(4) is at its place.

Fourth Step:

1	3	4	8	9	1 st (1) is not greater than 2 nd (3) → No Change
1	3	4	8	9	(3) is at its place

Sorted List: 1, 3, 4, 8, 9

Advantages:

- It is the easiest sorting algorithm to visualize and explain. If you can understand "swapping two cards in your hand," you understand Bubble Sort.
- It does not need to create a copy of the list or use extra RAM to do its work. It just swaps elements within the original array.
- This means if you have two items with the same value, they stay in their original relative order. This is actually pretty important for certain types of database work.

Disadvantages:

- It uses nested loop, the time it takes to sort grows exponentially. If you double the number of items, it takes four times longer to sort.
- Every swap requires the computer to perform three distinct operations. Bubble Sort performs a massive amount of swaps compared to other sorting algorithms.
- While it works fine for a list of 10 or 20 items in a classroom exercise, it is completely useless for real-world "Big Data."

b. Selection Sort:

Selection sort is a simple sorting algorithm used to arrange elements in a list or array in either ascending or descending order. It is a basic sorting algorithm that works by repeatedly selecting the smallest element from the unsorted portion of a list and moving it to its correct position.

Algorithm:

```

Step 1: Start
Step 2: Set DATA := {8, 4, 1, 9, 3}. [Any Input array data]
Step 3: Set N := 5. [Total Number of items in DATA array]
Step 4: Repeat Step 5 and Step 6 for K = 1 to N.
Step 5:   Set PTR := K+1. [Initializes pass pointer PTR]
Step 6:   Repeat while PTR ≤ N: [Executes pass]
           a: If DATA[K] > DATA[PTR], then:
               1. Set temp := DATA[K]
               2. Set DATA[K] := DATA[PTR]
               3. Set DATA[PTR] := temp
           [End of If structure]
           b: Set PTR := PTR + 1.
       [End of Step 6 inner loop]
   [End of Step 4 outer loop]
Step 7: End
    
```

Pseudocode:

```

Start
  DATA = {8, 4, 1, 9, 3} /* Array of an Integer data */
  N = 5 /* Total Number of Items in Array */
  For outerloop = 0 to N-1 /* Outer Loop */
    For innerloop = outerloop+1 to N-1 /* Inner Loop */
      IF DATA[outerloop] > DATA[innerloop] then
        /* Swap Data */
        Temp = DATA[outerloop]
        DATA[outerloop] = DATA[innerloop]
        DATA[innerloop] = Temp
      End If
    End For /* End of Inner Loop */
  End For /* End of Outer Loop */
End
    
```

Example:

Unsorted List: 8, 4, 1, 9, 3

First Step:

8	4	1	9	3	1 st (8) is greater than 2 nd (4) → Swap the data.
4	8	1	9	3	1 st (4) is greater than 3 rd (1) → Swap the data.
1	8	4	9	3	1 st (1) is not greater than 4 th (9) → No Change.
1	8	4	9	3	1 st (1) is not greater than 5 th (3) → No Change.
1	8	4	9	3	(1) is at its place.

Second Step:

1	8	4	9	3	2 nd (8) is greater than 3 rd (4) → Swap the data.
1	4	8	9	3	2 nd (4) is not greater than 4 th (9) → No Change.
1	4	8	9	3	2 nd (4) is greater than 5 th (3) → Swap the data.
1	3	8	9	4	(3) is at its place.

Third Step:

1	3	8	9	4	3 rd (8) is not greater than 4 th (9) → No Change.
1	3	8	9	4	3 rd (8) is greater than 5 th (4) → Swap the data.
1	3	4	9	8	(4) is at its place.

Fourth Step:

1	3	4	9	8	4 th (9) is greater than 5 th (8) → Swap the data.
1	3	4	8	9	(8) is at its place

Sorted List: 1, 3, 4, 8, 9

Advantages:

- Like Bubble Sort, it is very easy to code and understand. It is basically just two loops: one to keep track of the "sorted" boundary and one to find the minimum value.
- This is where it beats Bubble Sort. In every pass, it performs at most one swap. If you are sorting items on a system where "writing" or "moving" data is very expensive, Selection Sort is actually quite efficient.
- It does not need extra memory.
- It does not matter if the list is already sorted, reversed, or totally random; it will always take the same amount of time because it always scans the entire remaining list.

Disadvantages:

- Just like Bubble Sort, it has nested loops. As the list grows, the time it takes to finish grows very fast.
- Unlike Bubble Sort, Selection Sort must finish every single loop. It is "blind" to the fact that a list might already be in order.
- This is a big one for data integrity. If you have two "5s" in your list, Selection Sort might swap their relative positions.
- Because of the speed, it is strictly for small datasets. You would not use this to sort a database of customers.

Sorting 1 million items with Bubble Sort could take *hours*, while an algorithm like **Merge Sort** would finish in *seconds*.

2.3.2 Searching Algorithms:

a. Linear Search

Linear search also known as sequential search, it is a simple searching algorithm used to find a specific element within a list. It works by examining each element in the list one by one until the target element is found or until all elements have been checked.

It is simple and easy to implement and work on sorted and unsorted data list.

Algorithm:

```
Step 1: Start
Step 2: Set DATA := {8, 3, 6, 1, 7, 2, 4, 5} [Array Data].
Step 3: Set N := 8. [ Total Number of Elements ].
Step 4: Set FindValue := 7. [ Searching Value ].
Step 5: Set ValueFound := 0. [For Checking 0 = Not Found:1= Found ].
      [ Start Loop for find text in Array ].
Step 6: Repeat Step 7 for PTR = 1 to N.
      [Check Current Array Value with Searching value, If found Set
      Value Found Value].
Step 7: If DATA[PTR] = FindValue then
      a. ValueFound := 1.
      b. Goto Step 8.
      [ End of If Structure ]
      [ End of For Loop ]
Step 8: If ValueFound = 1 then
      Print DATA[PTR]. [ Show Data ].
      Else
      Print "Data not Found"
      [ End of If Structure ]
Step 9: End
```

Pseudocode:

```
Start
  DATA = {8, 3, 6, 1, 7, 2, 4, 5} /* Array Data */
  N = 8 /* Total Item in Array */
  FindValue = 7 /* Searching Item */
  ValueFound = 0 /* For Checking 0 = Not Found | 1 = Found */
  For PTR = 0 to N-1 /* Start Loop */
    If DATA[PTR] == FindValue then /* Check Data Found */
      ValueFound := 1 /* Set as Data is Found */
      Exit /* Exit from Loop */
    End If
  End For
  If ValueFound == 1 then /* Check ValueFound Data */
    Print DATA[PTR]
  Else
    Print "Data Not Found"
  End If
End
```

Example:

Data List: 8, 3, 6, 1, 7, 2, 4, 5

Searching Item: 7

Step by step process:

1. Check the 1st element → 8 → Not equal to 7 → Not Found
2. Check the 2nd element → 3 → Not equal to 7 → Not Found
3. Check the 3rd element → 6 → Not equal to 7 → Not Found
4. Check the 4th element → 1 → Not equal to 7 → Not Found
5. Check the 5th element → 7 → Equal to 7 → Found

Advantages:

- This is the biggest plus. Linear Search does not care if your data is scrambled, alphabetical, or numerical. It works on any list.
- The code is literally just one for loop and an if statement. It is very hard to mess up.
- If you only have 10 or 20 items, Linear Search is actually faster than more complex algorithms because there is zero "setup" time.
- It does not require any extra space.

Disadvantages:

- If you have 1 million items and your target is at the very end, the algorithm has to perform 1 million checks.
- The time taken grows linearly with the number of items. If you double the size of the list, you double the search time.
- If you need to search the same list thousands of times, it is much smarter to sort it once and use Binary Search instead of doing a Linear Search every time.

b. Binary Search:

It is a highly efficient search algorithm that works by repeatedly dividing the search interval in half and comparing the middle element to the target element. If the target is equal to the middle element, the search is successful. If the target is smaller, the search continues on the left half of the list; if larger, it continues on the right half. This process is repeated until the target element is found or it is determined that the element does not exist in the list.

The only limitation of Binary Search is that it only works on a sorted list.

Algorithm:

```
Step 1: Start
Step 2: Set DATA := {1, 2, 3, 4, 5, 6, 7, 8, 9}. [ Sorted Array Data ].
Step 3: Set BEG := 1. [ Set Beginning Value ]
Step 4: Set END := 9. [ Set End Value ].
Step 5: Set MID := INT((BEG + END) / 2). [ Set Middle Value ].
Step 6: Set Item := 7. [ Searching Item ].
Step 7: Set Loc := 0. [ Set Location value is 0 or NULL ].
      [ Check Beginning Value is Less than End Value or
      Data of Middle Location is equal to finding item ]
Step 8: Repeat Step 9 and 10 While BEG ≤ END and DATA[MID] ≠ Item.
Step 9:   If Item < DATA[MID] then:
          Set END := MID - 1. [ Set End Value ]
        Else
          Set BEG := MID + 1. [ Set Beginning Value ]
        [ End of If Structure ]
Step 10:   Set MID := INT((BEG + END) / 2) [ Set Middle Value ]
          [ End of While Loop ]
          [ Check Item is Found or Not ]
Step 11:   If DATA[MID] = Item then:
          Set LOC := MID
        Else
          Set LOC := 0
        [ End of If Structure ]
        [ Display Item is Found otherwise Not Found ]
Step 12:   If LOC > 0 then
          Print DATA[LOC]
        Else
          Print "Data Not Found"
        [ End of If Structure ]
Step 13: End
```

Pseudocode:

```
Start
DATA = {1,2,3,4,5,6,7,8,9} /* Sorted Array List */
BEG = 1 /* Beginning Array Pointer */
END = 9 /* End Array Pointer */
MID = INT((BEG + END) / 2) /* Middle Array Pointer */
Item = 7 /* Searching Value */
LOC = NULL /* Pointer Location Null means No Value */
/* Check Beginning Value is Less than End Value or
Data of Middle Location is equal to finding item */
While BEG ≤ END AND DATA[MID] ≠ Item
    IF Item < DATA[MID] then /* Check Item is less then
Middle Data */
        END = MID - 1
    Else
        BEG = MID + 1
```



```

        End If
        MID = INT( (BEG + END) / 2)
    End While
    If DATA[MID] = Item then /* Set Location value according to
Data Found */
        LOC = MID
    Else
        LOC = NULL
    End If
    If LOC >= 0 then /* If Data Found Show otherwise Not Found */
        Print DATA[MID]
    Else
        Print "Data Not Found"
    End If
End

```

Example:

Data List: 8, 3, 6, 1, 7, 2, 4, 5

Sorted List: 1, 2, 3, 4, 5, 6, 7, 8

Searching Item: 3

Middle Item: Total Item / 2 $\rightarrow 8 / 2 \rightarrow 4$

Left (Less than Middle)			Middle	Right (Greater than Middle)			
1	2	3	4	5	6	7	8

Middle Item: 4 $\rightarrow$ Not Found

Apply same logic for Left part because searching item (3) is less than middle item (4).

Searching Item: 3

Middle Item: Total Item / 2 $\rightarrow 3 / 2 \rightarrow 1.5 \rightarrow 2$

Note: 1.5 is a decimal number now we change it into integer 2 because list element number is always in integer.

Left (Less than Middle)	Middle	Right (Greater than Middle)
1	2	3

Middle Item: 2 $\rightarrow$ Not Found

Apply same logic for Right part because searching item (3) is greater than middle item (2).

Searching Item: 3

Middle Item: Total Item / 2 $\rightarrow 1 / 2 \rightarrow 0.5 \rightarrow 1$

Note: 0.5 is a decimal number now we change it into integer 1 because list element number is always in integer.

Middle
3

Middle Item: 3 → Found

Advantages:

- This is the biggest advantage. Because it cuts the search area in half every single step, it can find an item in a list of 1 million elements in just 20 steps. Linear Search would take 1 million.
- It is the go-to for "Big Data." Whether the list has 1,000 items or 1 billion, Binary Search stays incredibly fast.
- If you have a static list (like a dictionary or a phone book) that you need to search thousands of times, the time saved is massive.

Disadvantages:

- This is the "catch." Binary Search only works if the data is already in order (numerical or alphabetical). If the list is scrambled, the logic fails completely.
- If your list is not sorted, you have to sort it first. Sorting takes time, so if you are only searching once, it might actually be faster to just use a Linear Search.
- You need to be able to "jump" to the middle index ($\text{mid} = (\text{low} + \text{high}) / 2$). This means it works great for Arrays.
- While not super hard, it is easier to make a "Logic Error" (like an infinite loop or an "off-by-one" error) with Binary Search than with Linear Search.

Linear Search is used for unsorted and small datasets.

Binary Search is used for sorted and large datasets that also require fast searching.

2.4 Evaluation Criteria of Algorithm:

The following are the most important criteria used to assess the quality of an algorithm.

2.4.1 Correctness:

An algorithm is considered correct if its output matches the expected outcomes for the given input.

- Whether the algorithm solves the problem exactly as defined in the requirements?

- Whether the logic handles typical inputs gracefully, such as negative numbers, null values, empty strings, or maximum-capacity data?
- Whether the algorithm is guaranteed to finish its execution? It must reach a logical conclusion and avoid falling into an infinite loop.

In the following example, a number is input and checked: if the remainder when divided by 2 is 0, display “Even”; otherwise display “Odd”.

Example:

1. INPUT number
2. IF number MOD 2 = 0 THEN
3. DISPLAY "Even"
4. ELSE
5. DISPLAY "Odd"
6. END IF

Trace Table:

Input (number)	Operation (Divide by 2 and find remainder)	Remainder is 0	Result
3	$3 / 2 = (\text{Quotient} = 1, \text{Remainder} = 1)$	No	Odd
6	$6 / 2 = (\text{Quotient} = 3, \text{Remainder} = 0)$	Yes	Even
9	$9 / 2 = (\text{Quotient} = 4, \text{Remainder} = 1)$	No	Odd
12	$12 / 2 = (\text{Quotient} = 6, \text{Remainder} = 0)$	Yes	Even
15	$15 / 2 = (\text{Quotient} = 7, \text{Remainder} = 1)$	No	Odd

This algorithm is **correct** because it always determines even / odd properly.

2.4.2 Efficiency:

We measure any algorithm efficiency through time and space or speed and memory as resources. In simple words we find how fast it runs as input size grows and how much memory it uses. For this we check.

- Does the algorithm scale well with larger input sizes?
- Is there a way to make it faster or use less memory?
- Is the algorithm optimized for real-world use?

Example:

A linear search is less efficient than binary search when dealing with large sorted data sets.

2.4.3 Clarity (Readability & Understandability):

Algorithm must be easy to read, understandable and easy to modify by other users. For clarity the algorithm should follow the following principles.

- Steps are clearly written.
- Variable names are meaningful.
- The logic structure is coupled with indentation and comments.
- The flow of control (IF, LOOP, etc.) are easy to follow.

2.5 How to Decide Which Algorithm to Use:

It can be difficult to decide which algorithm to use. Before deciding which algorithm is best for our problem, we must consider the following questions.

- What is the type of problem?
- What is input size?
- Is the data sorted?
- Do we need Speed and Memory efficiency?

2.5.1 Simple Decision Guide:

Now we have some question and also have some problem to decide what the best algorithm for specific problem is.

- If searching on unsorted data then use Linear Search algorithm.
- If searching on sorted data then use Binary Search algorithm.
- If data needs ordering then use Sorting algorithm.
- If problem repeats many times then use Dynamic Programming algorithm.

Example:

Find a number in a sorted list

- Is the data sorted?
- Is a fast result needed?

Choose: Binary Search



Career Opportunities

This unit builds the ability to break problems into smaller parts, recognize patterns, and design efficient algorithms for searching and sorting. These computational thinking skills are the backbone of nearly every technology career, especially those focused on software and data.

Software Developer

Designs step-by-step algorithms and turns them into working, efficient computer programs.

Data Scientist / Analyst

Uses algorithmic thinking to clean, sort, and analyze large datasets to uncover patterns and insights.

Algorithm / Research Engineer

Develops and optimizes search, sorting, and decision-making algorithms for companies and research labs.

Competitive Programmer

Solves complex logical and algorithmic problems, often a stepping stone to top tech-industry roles.

AI / Machine Learning Engineer

Applies computational thinking to design intelligent systems that learn from data and improve over time.

Systems/Business Analyst

Breaks down real-world organizational problems and designs logical, algorithmic solutions.



Summary

- ◆ Computational Thinking is a systematic approach to solving problems that involves breaking them down into smaller parts, identifying patterns, abstracting unnecessary details, and developing algorithms.
- ◆ An algorithm is a step-by-step logical procedure used to solve a problem. It is conceptual in nature and independent of any programming language.
- ◆ Decomposition is a problem-solving technique in which a complex problem or task is divided into smaller, manageable parts.
- ◆ Pattern Recognition is the process of identifying similarities, regularities, or repeated patterns in data or problems.
- ◆ Abstraction is the process of hiding unnecessary details and focusing only on the essential features needed to solve a problem.
- ◆ Sorting is the process of arranging a collection of data or items in a specific order, such as ascending (lowest to highest) or descending (highest to lowest).
- ◆ Common sorting algorithms include:
 - ◆ Bubble Sort
 - ◆ Selection Sort

- ◆ Searching is the process of locating a specific element within a collection of data.
- ◆ Common searching algorithms include:
 - ◆ Linear Search
 - ◆ Binary Search
- ◆ Correctness is the property of an algorithm that ensures it produces the expected output for every valid input.
- ◆ Efficiency of an algorithm is measured by the amount of time it takes to execute (time complexity) and the amount of memory it uses (space complexity).
- ◆ Clarity refers to how easy an algorithm is to read, understand, and modify by others.



1. Encircle the correct answer

- i) Which strategy involves ignoring the brand of a kettle when learning how to make tea?
 - (a) Decomposition
 - (b) Pattern Recognition
 - (c) Abstraction
 - (d) Sorting
- ii) Bubble Sort is also commonly known as:
 - (a) Linear Sort
 - (b) Ripple Sort
 - (c) Selection Sort
 - (d) Binary Sort
- iii) Which searching algorithm requires the data to be sorted before it can be used?
 - (a) Linear Search
 - (b) Sequential Search
 - (c) Binary Search
 - (d) Bubble Search
- iv) What is the main disadvantage of Selection Sort?
 - (a) It uses extra RAM
 - (b) It is difficult to code
 - (c) It must finish every loop even if the list is sorted
 - (d) It only works on alphabetical data
- v) An algorithm is considered "correct" if:
 - (a) It runs very fast
 - (b) Its output matches expected outcomes for given inputs
 - (c) It uses minimal memory
 - (d) It is written in a specific programming language
- vi) What does the abbreviation CT stand for?
 - (a) Computer Technology
 - (b) Computational Thinking
 - (c) Central Terminal
 - (d) Coding Technique
- vii) Which operation is represented by MOD?
 - (a) Multiplication
 - (b) Division
 - (c) Subtraction
 - (d) Finding the remainder after division

- viii) In algorithms, what is the purpose of a PTR?
- (a) To store text data (b) To perform calculations
(c) To track positions in an array (d) To display output
- ix) What does INT represent in an algorithm?
- (a) Internet (b) Integer
(c) Interval (d) Input
- x) In the Binary Search algorithm, the abbreviations BEG, END, and MID represent:
- (a) Input, Process, and Output (b) Beginning, End, and Middle positions
(c) First, Last, and Next values (d) Start, Stop, and Result

2. Fill in the Blanks

- i) _____ is a step-by-step logical procedure to solve a problem.
- ii) Sorting may be Ascending or _____.
- iii) Algorithm is _____ and language-independent.
- iv) _____ algorithm is designed to locate a specific element within list of data.
- v) We measure any algorithm efficiency through _____ and _____ as resources.

3. Provide short answers to the following questions.

- i) Define pattern recognition.
- ii) Explain the concept of decomposition.
- iii) What is meant by correctness in problem-solving?
- iv) Why is efficiency important in problem-solving?
- v) What do you understand by clarity in problem-solving?

4. Provide detailed answers to the following questions.

- i) What is the difference between an algorithm and pseudocode?
- ii) What is abstraction in an algorithm, and why is it used?
- iii) Write the steps of the Selection Sort algorithm.
- iv) Write the steps of the Binary Search algorithm.
- v) Which searching algorithm is best for large datasets? Explain your answer.



Activities

- i) Arrange the assembly queue according to students' heights.
- ii) Arrange the names in the attendance register in descending alphabetical order.
- iii) Find the name of your class monitor in the attendance register using the binary search technique.

Unit

3

Programming Fundamentals

Student Learning Outcomes:

After the completion of this unit students will be able to:

- ◆ Understand the concept of programming
- ◆ Identify real-world applications of Python
- ◆ Use and explore the elements of IDE
- ◆ Understand the basic concept of Python
- ◆ Understand the use of selection and repetition control constructs in Python
- ◆ Write and execute program in Python
- ◆ Use built-in and third-party libraries in Python
- ◆ Debug code in Python

Introduction to Unit:

In this unit, students will learn about the basics of Python programming and how it is used in real-world applications. Students will learn to write, run and debug simple programs while understanding variables, data-types, operators and coding practices. This unit also covers decision making and loops to solve problems logically, along with the use of common Python libraries to build more practical and useful programs.



3.1 Programming:

Computers are dumb machines. They cannot think and do anything by themselves without instructions. A set of instructions is called program and the language through which we give instructions to computer is called programming language. It uses special words and writing rules to create programs that tell the computer what to do. These instructions can make the computer perform tasks like solving problems, playing games, or running apps.

3.1.1 Programming Language Types:

Programming languages are classified into the following types.

- i. **Low Level Language:** are closer to the native language of computer (binary 0s & 1s) and difficult to understand by human.
- ii. **Mid-Level Language:** bridges the gap between low-level and high-level languages it has both hardware access capabilities and easier to read and write than low-level code.
- iii. **High-Level Language:** are designed to be easy for humans to read and write codes.

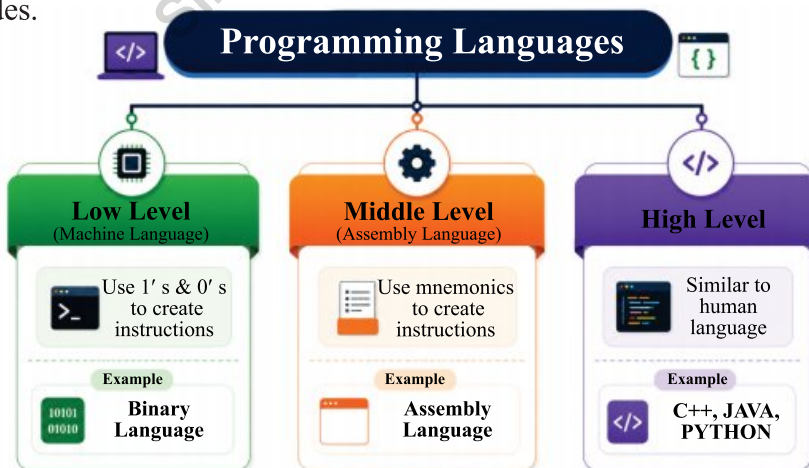


Figure 3.1 Programming Language Type

Programming is not just about code; it is about logic, structure, and turning ideas into something a computer can execute. Different languages are designed for different purposes. The following are some common programming languages:

- **C++** is a powerful programming language that is closer to the hardware compared to Python and Java. It is often used to develop system software, operating systems, and computer games. C++ also allows programmers to control memory and performance, making programs run faster.
- **Java** is a language that is widely used in professional software development. It follows the principle “write once, run anywhere,” meaning a Java program can run on different platforms without changes. Java is commonly used to develop large applications, desktop software, Android mobile apps, web-based programs, scripts and server programming.
- **Python** is a high-level programming language known for its simple and readable syntax. It is easy to learn, which makes it very popular for beginners and students. It is used to create simple programs, games, and graphical drawings and also widely used in data analysis, artificial intelligence, and web development.



3.2 Why Learn Python:

Python is one of the most popular programming languages in the world. It is widely used because of its simple and easy-to-read syntax, making it an excellent choice for beginners as well as experienced programmers. Python allows developers to write programs using fewer lines of code, which makes programming faster and more efficient.

Python is a versatile language that supports multiple programming paradigms, including procedural, object-oriented, and functional programming. It is platform-independent, meaning that Python programs can run on different



operating systems, such as Windows, Linux, and macOS, with little or no modification.

Python is used in a wide range of applications, including web development, artificial intelligence (AI), machine learning, data science, automation, cybersecurity, desktop applications, and game development. Due to its extensive libraries and strong community support, Python has become one of the most preferred programming languages for education, research, and industry. For these reasons, Python is an ideal language for learning programming concepts and developing practical problem-solving skills.



Activity

Demonstrate the different frameworks of Python.

3.3 Integrated Development Environment (IDE):

An **IDE (Integrated Development Environment)** is a software tool that helps write, run, and manage code in one place. An IDE is a complete workspace for programming. Following are the key features of an IDE

- **Code Editor:** write and edit code
- **Syntax Highlighting:** makes code easier to read
- **Auto completion:** suggest code while typing
- **Run/Execute Tool:** execute program
- **Debugger:** find and fix errors.

3.3.1 Visual Studio Code:

Visual Studio Code (VS Code) is a free, lightweight and powerful code editor widely used for Python, JavaScript, C#, and many other languages. It is not a full IDE by default, but can turn it into one by installing extensions. VS code can be downloaded from following link



Visual Studio Code



<https://code.visualstudio.com/Download>

Explore User Interface:

After installing the VS Code from above link. Open VS Code by double clicking on the icon  from computer desktop. A welcome screen will appear as shown

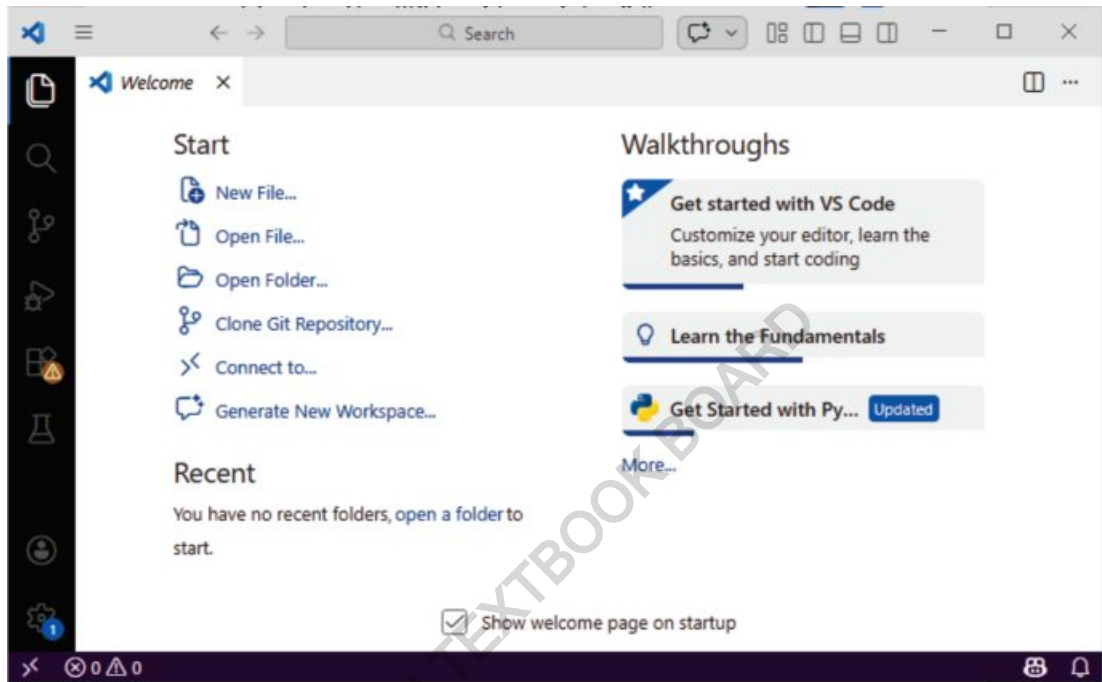


Figure 3.2 VS Code Welcome Screen

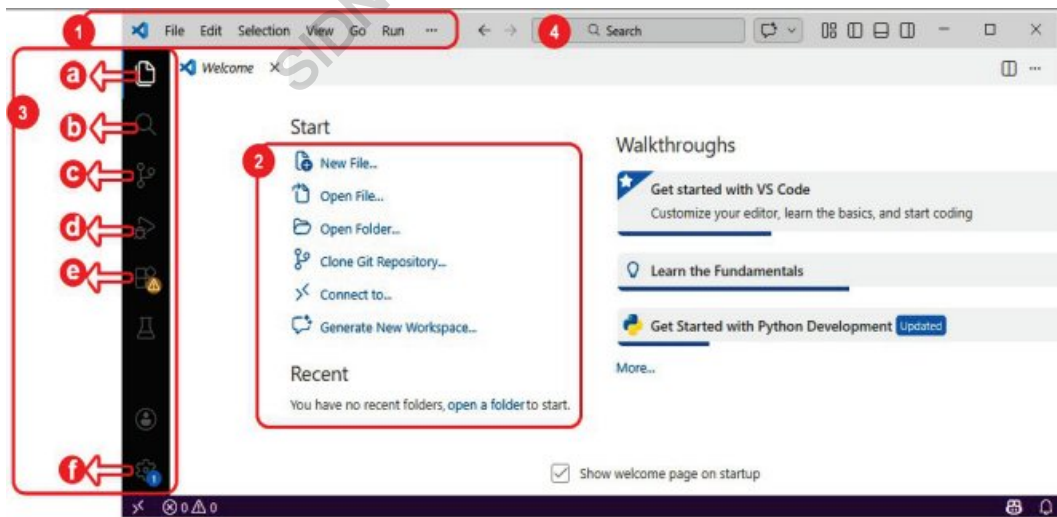


Figure 3.3 VS Code Interface Overview

1. **Top Menu Bar:** gives **complete control** over the files, code editing, navigation, debugging, and workspace all organized into logical groups for easy access.
2. **Start Section:** provides option to get start quickly New File, Open File, Open Folder, Recent Section (quickly jump back into previous projects with one click).
3. **Side Bar:** contains Explorer, Search, Source control, Run & Debug, Extension and Manage configuration.
 - a. **Explorer:** Shows project files and folders, it helps to create, open and manage files.

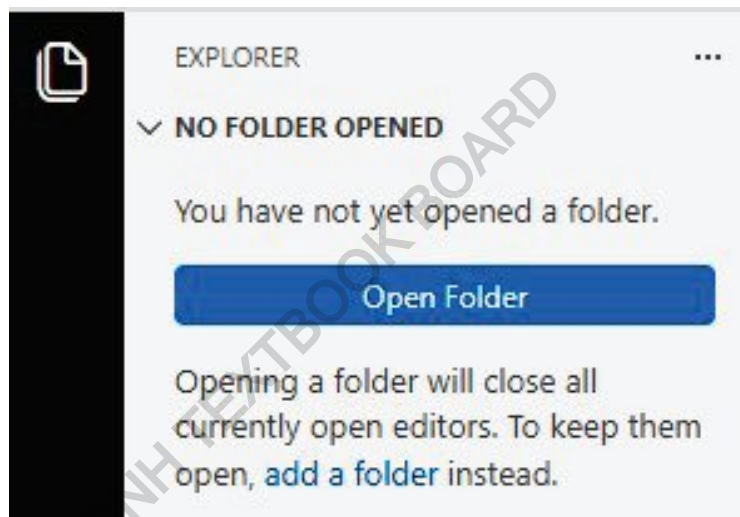


Figure 3.4 VS Code Explorer Overview

- b. **Search:** Provides the searching and replacing any text in the currently open file.

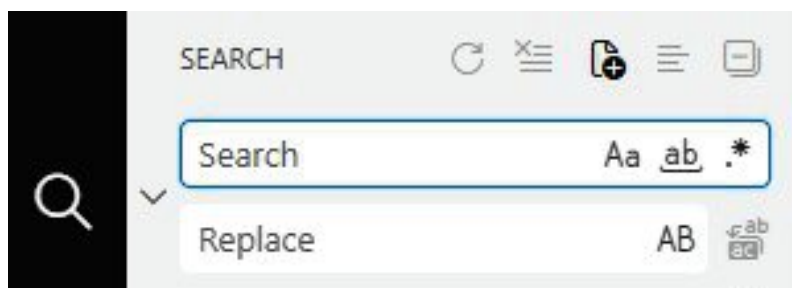


Figure 3.5 VS Code Search Overview

- c. **Source Control:** Connects with Git. Git is a free, open-source version control system that tracks and records every change made to the code over time, so that the user can save progress, undo mistakes, and collaborate with other developers easily.

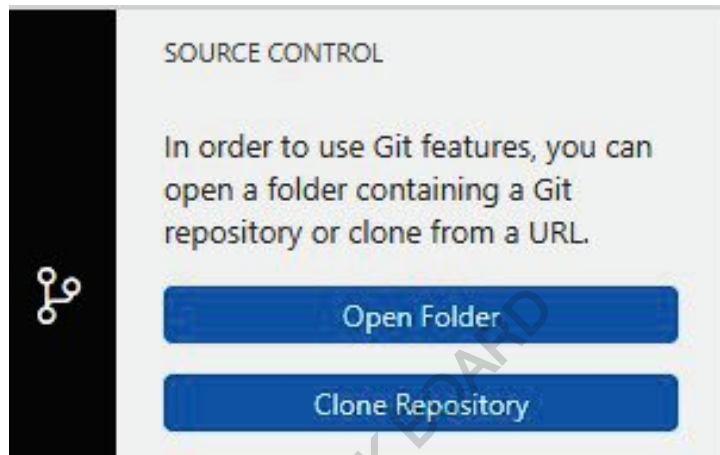


Figure 3.6 VS Code Source Control Overview

- d. **Run & Debug:** Allows to execute code, find errors, pause at specific lines, and inspect variables.



Figure 3.7 VS Code Run & Debug Overview

- e. **Extensions:** are like apps for VS Code; users can install them to add new features, language support, and tools that are not built-in.

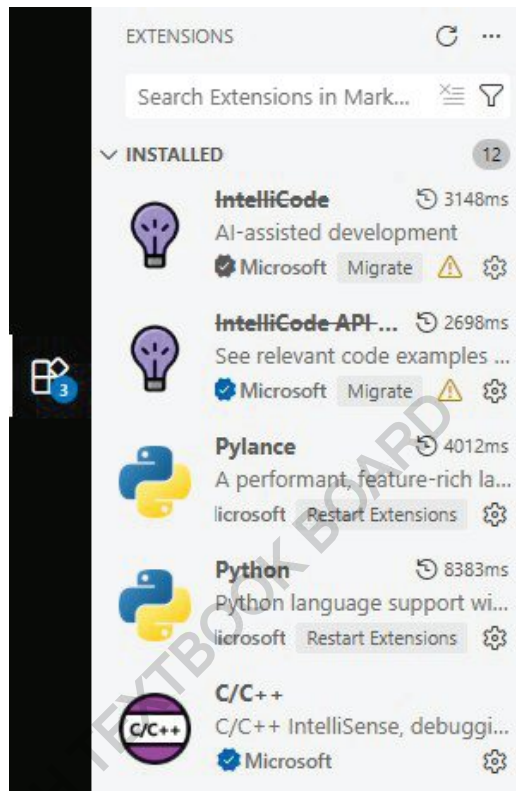


Figure 3.8 VS Code Extension Overview

- * **Recommended Extensions for Python Programming:** search the following extensions and install it in vs code.

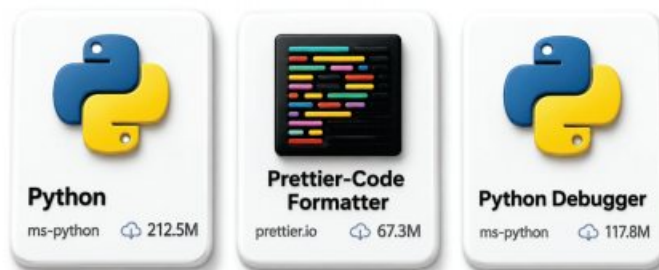


Figure 3.9 VS Code Recommended Extensions

- f. **Manage Setting:** provides quick access to all settings, configurations, themes, keyboard shortcuts, and account options and acting as the main control center for the entire VS Code environment.

4. **Search Bar:** is the master navigator of VS Code. It instantly finds and jumps to anything inside project with just a few keystrokes.

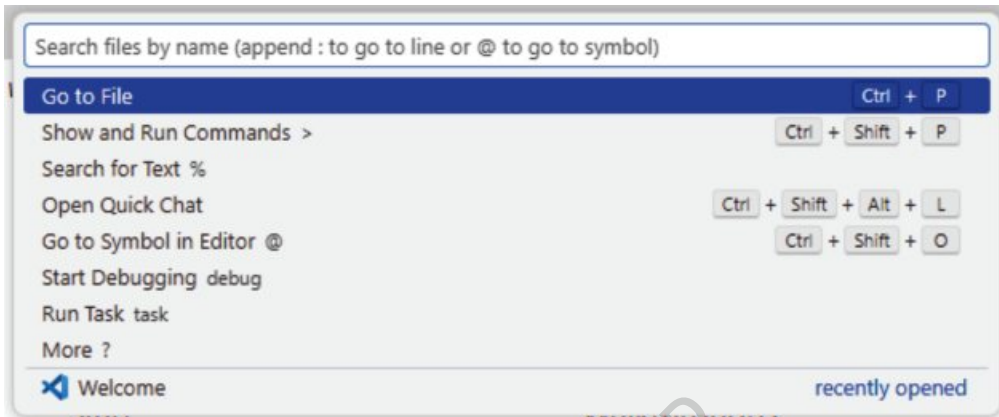


Figure 3.10 VS Code Search Bar Overview

3.4 Fundamentals of Python Programming:

The basic structure of a Python program is simple and clear. A Python program may start with comments, which are written using the # symbol. Comments are lines ignored by the Python interpreter and are used for documentation and explanation within the code. The program processes the data using statements and expression, and finally shows the output. Python uses indentation (spaces at the beginning of a line) to define blocks of code, instead of curly brackets. Indentation is mandatory in Python, standard indentation = **4 spaces** per level (also use 1 tab).



Figure 3.11 Simple Python Program

In this program, the first line is a comment. The second, third and fourth lines displays output. Python programs run from top to bottom, making them easy to understand and follow.

3.4.1 Variables and Data Types:

A **variable** is a name space of computer memory used to store data in a program. Think of it as a container that holds information which can change during program execution. In Python, there is no need to declare the type of a variable explicitly. Python automatically determines it based on the assigned value. Following are some rules for naming variables in Python:

- can contain letters, numbers and underscores (_).
- must start with a letter or underscore, not a number.
- are case-sensitive (age and Age are different).
- Avoid using keywords such as if, while, or for as variable names.

```
name = "Ali"
age = 16
height = 5.7
is_student = True
```

Figure 3.12 Variable Naming Example

Data types specify the kind of data a variable holds. Python has several built-in data types which are as follows:

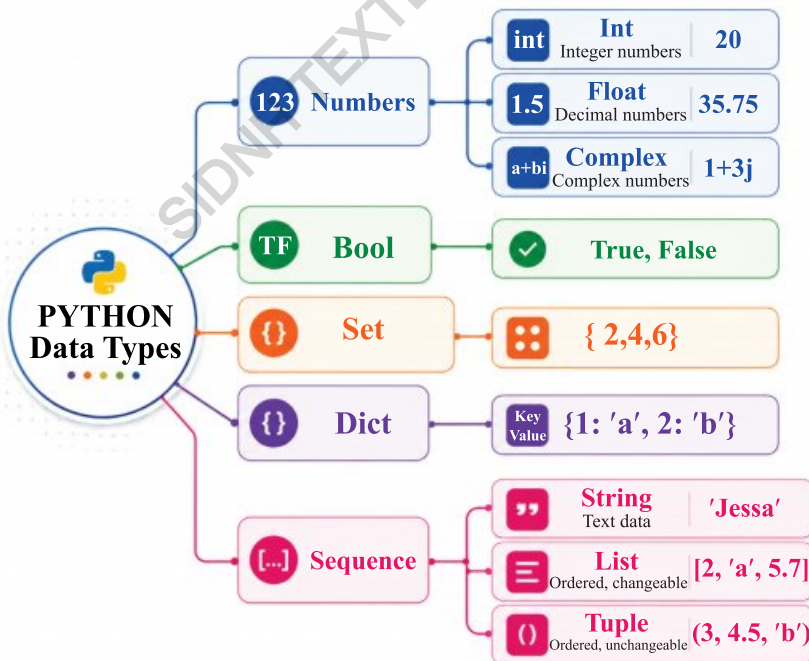


Figure 3.13 Data Types in Python

Name	Type	Description
Integers	int	Whole numbers, such as 3 300 200
Floating Point	float	Number with a decimal point: 2.3 4.6 100.56
Strings	str	Text or sequence of characters: "hello" 'Sammy' "2026"
Boolean	bool	Logical value indicating True or False
Lists	list	Ordered changeable collection [1,2,3,4,5,6]
Dictionaries	dict	Unordered Key:Value pairs: {"name": "Ahmed", "age": 16 }
Tuples	tup	Ordered unchangeable sequence of object (10, "hello", 200.3)
Sets	set	Unordered collection of unique objects: { "p","d","f" }

3.4.2 Input Output Handling:

For a program to be useful, it usually needs to communicate with the outside world. Information provided by the user to the computer is called input, and the information displayed on the screen after processing that input is called output. Input and output are essential for interactive programs.

Python uses the **print()** function to display information on the screen. User can print strings, numbers, variables, or combinations of them.

print() function has following syntax too.



Figure 3.14 Input Output Handling

```
# following print function takes only string message
print("message")

# following print function takes string message and variable
print("message",variable)

# following print function takes multiple string message and variable
print("message",variable1,"message",variable2)

# following print function takes string message with variable names|
print(f"message {variablename} message {variablename}")
```

Figure 3.15 print() function syntax

Example 1: Printing multiple values:

```
name = "Ali"
age = 16
print("Name:", name, "Age:", age)
```

**Example 2: Advance output using f-strings:**

f-strings make output more readable and easier to format.

```
name = "Ali"
age = 16
print(f"My name is {name} and I am {age} years old.")
```

Output:

```
My name is Ali and I am 16 years old.
```

Python uses the **input()** function to take input from the user while program is running. It allows the program to pause, wait for the user to enter some data, and then save that data into a variable for further processing. Whatever the user types is always treated as a **string** (data type), Unless converted into another data type.

Basic Syntax: **input(prompt)**

* **prompt(optional)** → a message shown to the user before taking input

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

Example 1 - taking name as input and print:

```
name = input("Enter your name: ")
print("Hello,", name)
```

The code prompts the user to input their name, stores it in the variable "name" and then prints a greeting message addressing the user by their entered name.

Example 2 - taking numeric input and print:

```
age = int(input("Enter your age: ")) # converting input to integer
height = float(input("Enter your height in meters: ")) # converting input to float

print("Your age is", age)
print("Your height is", height)
```

The code prompts the user to input their age and height, stores it in the variable "age", "height" and then prints a message addressing the user by their entered age and height.

Since age is always expressed as a whole number, it is stored in an **integer (int)** variable. On the other hand, height is typically expressed as a decimal or fractional value, therefore it is stored in a **floating-point (float)** variable. This ensures accurate representation and efficient use of memory in a program.



Instructions for Teachers

Teacher should demonstrate the conversion from string to other datatypes in Python.

Activity:

Writing and Executing Python Program using IDE (VS Code)

Step 1: Open Visual Studio Code

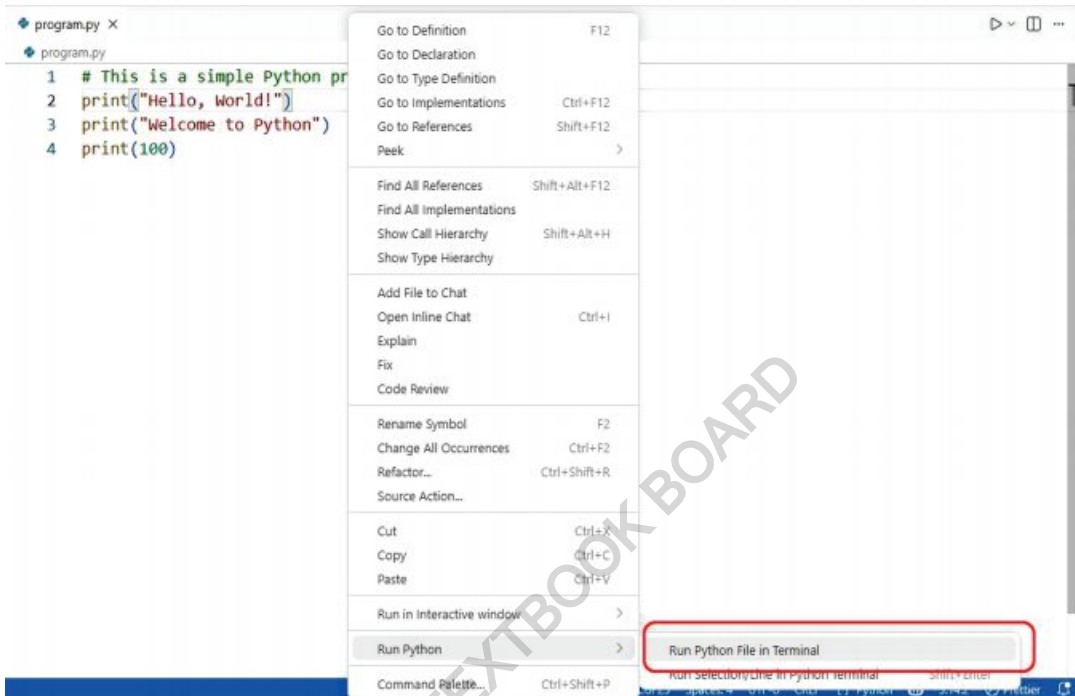
Step 2: Create Python File

- Open VS Code, go to File menu and click on New File
- Save the file as program.py
- Write the following code as shown in the image

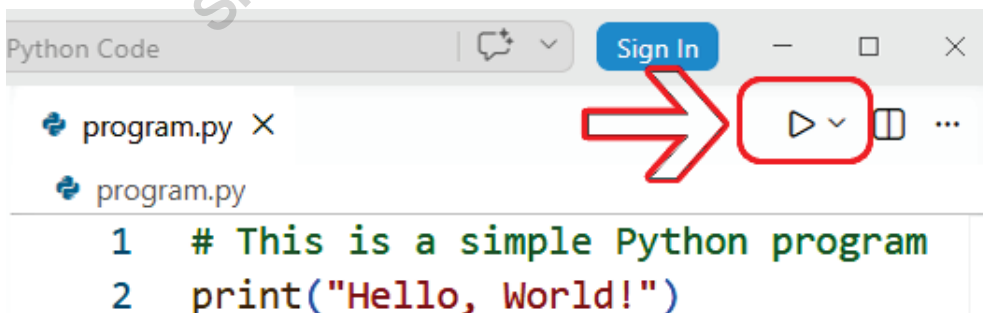
```
1 # This is a simple Python program
2 print("Hello, World!")
3 print("Welcome to Python")
4 print(100)
```

- Following are two options to execute the program, programmer can choose either of them.

1) Right click in the file and select → **Run Python File in Terminal**

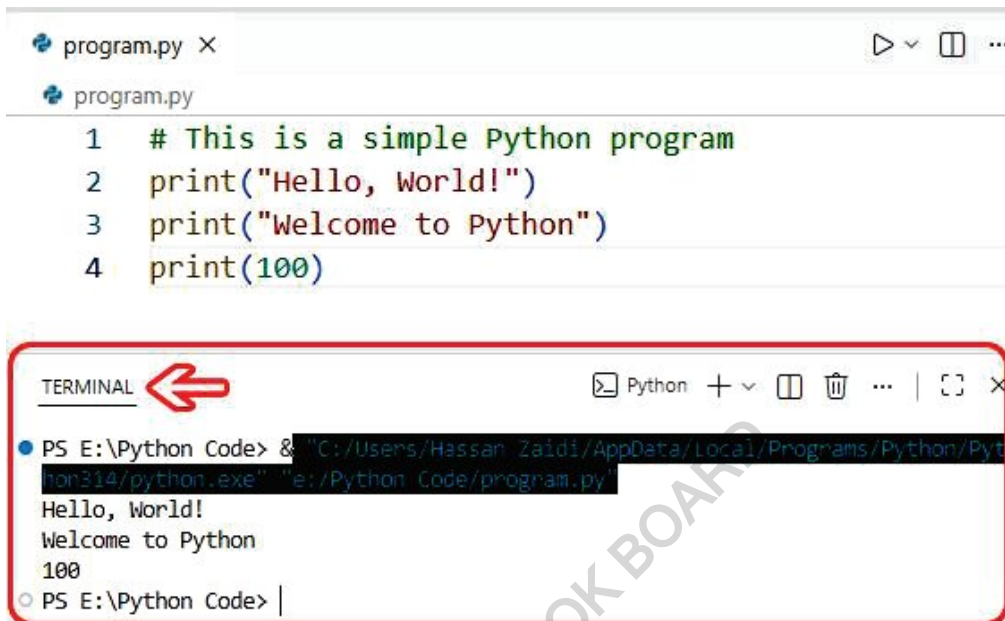


- 2) Click the **Play** button located at the top-right corner of the VS Code window to execute the program.



- After clicking on the play button, the code will execute and a terminal window is open where the execution/output of the program will show.

- Output of the program



The screenshot shows a Python IDE with a file named 'program.py' containing the following code:

```
1 # This is a simple Python program
2 print("Hello, World!")
3 print("Welcome to Python")
4 print(100)
```

Below the code editor is a terminal window titled 'TERMINAL' with a red arrow pointing to it. The terminal shows the command to run the program and its output:

```
PS E:\Python Code> & "C:/Users/Hassan_Zaidi/AppData/Local/Programs/Python/Python314/python.exe" "e:/Python Code/program.py"
Hello, World!
Welcome to Python
100
PS E:\Python Code> |
```

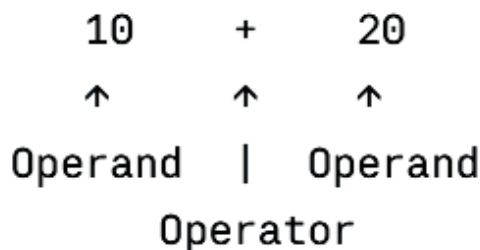


Activity

1. Write a program that takes input and display it.
2. Write a program that takes student bio data and display with welcome message.

3.4.3 Operators and Operands:

An **Operand** is the **value**, or a **variable** on which the operation is performed. An **Operator** is a symbol that tells Python to perform an operation on one or more operand.



Operators are building blocks used to calculate results, compare values, combine conditions, manipulate data and control the program flow. Some of the operators are as follows:

- **Arithmetic Operators:** These operators are used for mathematical calculations.

Operator	Purpose	Example	Result
+	Addition	5 + 3	8
-	Subtraction	10 - 4	6
*	Multiplication	4 * 2	8
/	Division	9 / 2	4.5
//	Floor division	9 // 2	4
%	Modulus (remainder)	9 % 2	1
**	Exponentiation	2 ** 3	8

Code Example:

ArithmeticOperatorProgram.py X

ArithmeticOperatorProgram.py > ...

```

1  # Simple Arithmetic Operation Program
2  a = 10
3  b = 3
4
5  print(f"Addition: {a} + {b} =", a + b)      # 13 → Addition
6  print(f"Subtraction: {a} - {b} =", a - b)    # 7 → Subtraction
7  print(f"Multiplication: {a} * {b} =", a * b) # 30 → Multiplication
8  print(f"Division: {a} / {b} =", a / b)        # 3.33 → Division
9  print(f"Floor Division: {a} // {b} =", a // b) # 3 → Floor Division
10 print(f"Modulus: {a} % {b} =", a % b)         # 1 → Modulus
11 print(f"Exponent: {a} ** {b} =", a ** b)      # 1000 → Exponent

```

TERMINAL

```

PS D:\Python Code> & D:\Programs\Python\Python314\python.exe "d:/Python Code/ArithmeticOperatorProgram.py"
Addition: 10 + 3 = 13
Subtraction: 10 - 3 = 7
Multiplication: 10 * 3 = 30
Division: 10 / 3 = 3.3333333333333335
Floor Division: 10 // 3 = 3
Modulus: 10 % 3 = 1
Exponent: 10 ** 3 = 1000

```

- **Assignment Operators:** These operators assign values to variables.

Operator	Example	Description
=	x = 5	Assign 5 to x
+=	x += 2	Add 2 to x value and assign it to x x = x + 2
-=	x -= 1	Subtract 1 from x value and assign it to x x = x - 1
*=	x *= 3	Multiply 3 to the value of x and assign it to x. x = x * 3
/=	x /= 2	Divide 2 to the value of x and assign it to x. x = x / 2

Code Example:

```

AssignmentOperatorProgram.py X
AssignmentOperatorProgram.py > ...
1  # Simple Assginment Operation Program
2  a = 10          # Assign 10
3  print(a)
4  a += 5          # a = a + 5 → 15
5  print(a)
6  a -= 3          # a = a - 3 → 12
7  print(a)
8  a *= 2          # a = a * 2 → 24
9  print(a)
10 a /= 4          # a = a / 4 → 6.0
11 print(a)
12 a //= 2         # a = a // 2 → 3.0
13 print(a)
14 a **= 3         # a = a ** 3 → 27.0
15 print(a)
16 a %= 5         # a = a % 5 → 2.0
17 print(a)
# 2.0

TERMINAL
10
15
12
24
6.0
3.0
27.0
2.0
  
```

- **Relational Operators:** These operators compare two values and return either True or False

Operator	Description	Example	Result
==	Equal to	10 == 10	True
!=	Not equal to	10 != 5	True
>	Greater than	10 > 5	True
<	Less than	10 < 5	False
>=	Greater than or equal to	10 >= 10	True
<=	Less than or equal to	10 <= 9	False

Example:

```

ComparisonOperatorProgram.py X
ComparisonOperatorProgram.py > ...

1  # Simple Comparison Operation Program
2  a = 10
3  b = 20
4
5  print(a == b)    # False → Equal to
6  print(a != b)    # True → Not equal to
7  print(a > b)     # False → Greater than
8  print(a < b)     # True → Less than
9  print(a >= 10)   # True → Greater or equal
10 print(a <= 9)    # False → Less or equal

TERMINAL
False
True
False
True
True
False

```


Unit 3: Programming Fundamentals

- **Logical Operators:** Logical operators are used to combine conditions.

Operator	Description
and	True if both conditions are true
or	True if at least one condition is true
not	Reverses the result

Example:

```
LogicalOperatorProgram.py X
LogicalOperatorProgram.py > ...
1 # Simple Logical Operator Program
2 a = True
3 b = False
4
5 print(a and b)      # False → Both must be True
6 print(a or b)       # True → At least one True
7 print(not a)        # False → reverse the result of variable a
8 print(not (a or b)) # False → reverse the result of (a or b)

TERMINAL
False
True
False
False
```

- **Membership Operators:** Used to check if a value exists in a sequence.

Operator	Description	Example
in	True if value exists	"a" in "apple"
not in	True if value does not exist	"z" not in "apple"

Example:

```
MembershipOperatorProgram.py X
MembershipOperatorProgram.py > ...
1 # Simple Membership Operator Program
2 name = "Python"
3 print("P" in name)      # True
4 print("z" not in name)  # True

TERMINAL
True
True
```

- **Bitwise Operators:** A **Bit** is the smallest unit of data in a computer, it can only be **0** or **1**. Bitwise operators treat numbers as binary digits (bits) and perform operations bit by bit like a switch that turns each 0 and 1 ON or OFF. Python uses **bin()** function to convert number to binary.

Operator	Symbol	Name	Description
AND	&	Bitwise AND	1 only if both bits are 1
OR	 	Bitwise OR	1 if at least one bit is 1
XOR	^	Bitwise XOR	1 only if bits are different
NOT	~	Bitwise NOT	Flip or inverts all bits
Left Shift	<<	Left Shift	Shifts bits to the LEFT
Right Shift	>>	Right Shift	Shifts bits to the RIGHT

1) Bitwise AND (&):

Returns **1** only when **BOTH** corresponding bits are **1** otherwise returns **0**.

Truth Table		
Bit A	Bit B	A & B
0	0	0
0	1	0
1	0	0
1	1	1

Bitwise AND Program.py

```

1 # Bitwise AND Operator Program
2 a = 10 # Binary: 1010
3 b = 6  # Binary: 0110
4 print("Binary of a ",bin(a))
5 print("Binary of b ",bin(b))
6
7 result = a & b
8
9 print("Result = ",result) # 2
10 print("Binary of Result =",bin(result)) # 0b0010

```

TERMINAL

```

• Binary of a  0b1010
  Binary of b  0b110
  Result = 2
  Binary of Result = 0b10

```

Step by step explain

```

a = 10 → 1 0 1 0
b = 6  → 0 1 1 0
-----
a & b → 0 0 1 0 = 2

```

* Compare each bit and give 1 where both are 1.

2) Bitwise OR (|):

Returns 1 when **AT LEAST ONE** of the corresponding bits is 1 returns 0 only when **BOTH** are 0.

Truth Table		
Bit A	Bit B	A B
0	0	0
0	1	1
1	0	1
1	1	1

Bitwise OR Program.py

```

1  # Bitwise OR Operator Program
2  a = 10      # Binary: 1010
3  b = 6       # Binary: 0110
4
5  result = a | b
6
7  print(result)      # 14
8  print(bin(result)) # 0b1110

```

14

0b1110

14

0b1110

14

0b1110

14

0b1110

Step by step explain

```

a = 10 → 1 0 1 0
b =  6 → 0 1 1 0
          _____
a | b → 1 1 1 0 = 14

```

3) Bitwise XOR (^):

Returns **1** only when the **two corresponding bits are DIFFERENT**, returns **0** when bits are the **SAME**.

Truth Table		
Bit A	Bit B	A B
0	0	0
0	1	1
1	0	1
1	1	0

Bitwise XOR Program.py ×

▶ ▢ ...

Bitwise XOR Program.py > ...

```

1  # Bitwise XOR Operator Program
2  a = 10      # Binary: 1010
3  b = 6       # Binary: 0110
4
5  result = a ^ b
6
7  print(result)      # 12
8  print(bin(result)) # 0b1100

```

TERMINAL

```

12
0b1100

```

Step by step explain

a = 10	→	1 0 1 0	
b = 6	→	0 1 1 0	
—————			
a ^ b	→	1 1 0 0	= 12

4) Bitwise NOT (~):

It inverts/flips **ALL** bits of the number, every **0** becomes **1** and every **1** becomes **0**. It is also called ones complement.

Truth Table	
Bit A	$\sim A$
0	1
1	0

```
Bitwise NOT Program.py X
Bitwise NOT Program.py > ...

1  # Bitwise NOT Operator Program
2  a = 10      # Binary: 0000 1010
3
4  result = ~a
5
6  print(result)      # -11
7  print(~5)          # -6
8  print(~0)          # -1
9  print(~(-1))       # 0

TERMINAL
-11
-6
-1
0
```

5) Bitwise Shift LEFT (<<):

Shifts all bits to the **LEFT** by a specified number of positions, empty positions on the right are filled with **zeros**.

```
Bitwise Shift left Program.py X
Bitwise Shift left Program.py > ...

1 # Bitwise Shift LEFT Operator Program
2 a = 5      # Binary: 0000 0101
3
4 print(a << 1)   # 10   → 0000 1010
5 print(a << 2)   # 20   → 0001 0100
6 print(a << 3)   # 40   → 0010 1000
```

TERMINAL

```
10
20
40
```

Step by step explain

a = 5 → 0 0 0 0 0 1 0 1

a << 1 → 0 0 0 0 1 0 1 0 = 10

↑

(0 added on right)

a << 2 → 0 0 0 1 0 1 0 0 = 20

↑ ↑

(two 0s added on right)

6) Bitwise Shift RIGHT (>>):

Shifts all bits to the **RIGHT** by a specified number of positions, bits shifted out are discarded and empty positions on the left are filled with **zeros**.

```
Bitwise Shift Right Program.py X
Bitwise Shift Right Program.py > ...

1 # Bitwise Shift RIGHT Operator Program
2 a = 20 # Binary: 0001 0100
3
4 print(a >> 1) # 10 → 0000 1010
5 print(a >> 2) # 5 → 0000 0101
6 print(a >> 3) # 2 → 0000 0010
```

Step by step explain

```

a = 20 → 0 0 0 1 0 1 0 0

a >> 1 → 0 0 0 0 1 0 1 0 = 10
           ↑
        (0 added on left)

a >> 2 → 0 0 0 0 0 1 0 1 = 5
           ↑ ↑
        (two 0s added on left)
```



Test Yourself

Here are some logical tests using python operators which are true and which are false?

A. $4 == 5-1$ B. $55 \geq 11 * 5$ C. $22 != 23 -1$

3.5 Control Structures:

Control structures are used to decide the order in which statements are executed in a program. Almost every program uses three basic structures: **sequence**, **selection**, and **repetition**.

3.5.1 Sequence:

Sequence means that statements in a program are executed **one after another**, from top to bottom, in the order they are written. This is the default behavior of a Python program.

```
print("Enter two numbers")
a = int(input("Enter first number: "))
b = int(input("Enter second number: "))
sum = a + b
print("Sum =", sum)
```

Understanding:

- Input is taken first
- The **input()** function in Python always returns the entered value as a string data type. Therefore, in order to perform numerical operations, it is necessary to convert it into an integer data type by using the **int()** function.
- Processing is done by adding the values taken by user in variable “a” and “b” after that assign the result in a variable named “sum”.
- Finally, the Output is displayed.

Each line runs only after the previous one is finished. This simple flow is called Sequence.

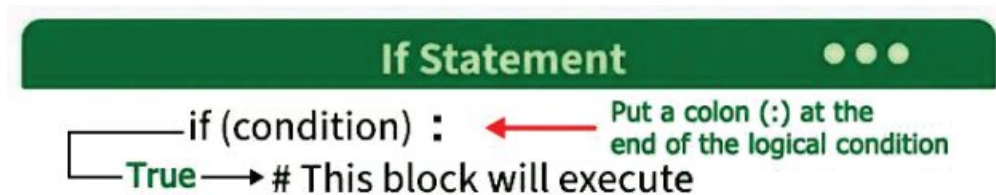
3.5.2 Selection Statement:

Selection allows to alter the flow of a program based on decisions. Based on a condition, the program chooses which block of code to execute. Selection helps programs respond intelligently to different situations. Following are the types of selection statement.

i. **if Statement:**

The **if statement** is the simplest selection statement; it executes a block of code **only when the given condition is TRUE**. If the condition is FALSE, the block is completely skipped.

Syntax:



All the commands that follow the colon will be indented. That means they will be set in a block of code (spaces/ 1 tab at the beginning of a line). The computer will only carry out the indented commands if the test is True.

```

if statement program.py ×
if statement program.py > ...

1  # This is a if statement program
2  age = int(input("Enter your age: "))
3
4  if age >= 18:
5      print("You are an Adult.")
6      print("You are eligible to vote.")
7  print("Program continues...")

TERMINAL
Python + v [ ] [ ] ... | [ ] ×

● PS E:\Python Code> & "python.exe" "e:/Python Code/if statement program.py"
Enter your age: 18
You are an Adult and eligible to vote.
You are eligible to vote.
Program continues...
Condition is true

● PS E:\Python Code> & "python.exe" "e:/Python Code/if statement program.py"
Enter your age: 10
Program continues...
Condition is false

○ PS E:\Python Code> |

```


ii. **if-else Statement:**

The **if-else** statement provides two paths, if the condition is **TRUE**, the if block runs; if the condition is **FALSE**, the else block runs. One of the two blocks always executes. In simple words if-else statement is defined as IF this is **true** DO this, ELSE DO that.

```
if condition:
    # runs when condition is True
else:
    # runs when condition is False
```



```
if else statement program.py X
if else statement program.py > ...

1  # This is a if-else statement program
2  age = int(input("Enter your age:"))
3
4  if age >= 18:
5      print("You are an Adult.")
6      print("Apply for CNIC.")
7  else:
8      print("You are a Minor.")
9      print("Apply for CRC")
10
11 print("Check complete.")
```

TERMINAL

Python + v [] [] ... | [] [] X

- PS E:\Python Code> & "C:/python.exe" "e:/Python Code/if else statement program.py"
Enter your age:18
You are an Adult.
Apply for CNIC.
Check complete.
Condition True
- PS E:\Python Code> & "C:/python.exe" "e:/Python Code/if else statement program.py"
Enter your age:12
You are a Minor.
Apply for CRC
Check complete.
Condition False
- PS E:\Python Code> |

iii. **if-elif-else Statement:**

The **if-elif-else** statement handles **multiple conditions**, it checks conditions one by one from top to bottom and executes the **first block whose condition is TRUE**. If none match, the else block runs.

```
if condition1:
    # runs if condition1 is True
elif condition2:
    # runs if condition2 is True
elif condition3:
    # runs if condition3 is True
else:
    # runs if ALL conditions are False
```

if elif else statement program.py X

▶ ▢ ...

if elif else statement program.py > ...

```
1 # This is a if-elif-else statement program
2 age = int(input("Enter your age: "))
3
4 if age <= 0:
5     print("Invalid age!")
6 elif age <= 12:
7     print("Child")
8 elif age <= 17:
9     print("Teenager")
10 elif age <= 35:
11     print("Young Adult")
12 elif age <= 60:
13     print("Middle-Aged Adult")
14 else:
15     print("Senior Citizen")
```

TERMINAL

Python + ▢ ▢ ... | ⌂ X

- PS E:\Python Code> & "C:/Hassan Zaidi/Python/Python314/python.exe" "e:/Python Code/if elif else statement program.py"
 - Enter your age: 17
 - Teenager
- PS E:\Python Code>

iv. Nested if Statement:

A **Nested if Statement** is an **if** statement placed inside another **if** statement. It is used when a decision depends on multiple levels of conditions.

```
if condition1:
    # outer if block
    if condition2:
        # inner if block
        if condition3:
            # deeply nested block
        else:
            # inner else
    else:
        # inner else
else:
    # outer else
```

The screenshot displays a Python IDE window titled "Nested if statement program.py". The code defines a program where age and salary are input by the user. It uses nested if statements to check if the age is 18 or older and if the salary is at least 30,000. If both conditions are met, it prints "Age requirement met", "Salary requirement met", and "Loan APPROVED!". Otherwise, it prints "Salary too low" or "Age requirement not met", followed by "Loan REJECTED!".

```
1 # This is a nested if statement program
2 age = int(input("Enter your age:"))
3 salary = int(input("Enter your monthly Salary:"))
4
5 if age >= 18:
6     print("Age requirement met")
7     if salary >= 30000:
8         print("Salary requirement met")
9         print("Loan APPROVED!")
10    else:
11        print("Salary too low")
12        print("Loan REJECTED!")
13 else:
14     print("Age requirement not met")
15     print("Loan REJECTED!")
```

The TERMINAL panel shows two execution runs:

- Run 1: PS E:\Python Code> python.exe "C:/Hassan_Zaidi/AppData/Local/Programs/Python/Python314/python.exe" "e:/Python Code/nested if statement program.py"
Enter your age:24
Enter your monthly Salary:35000
Age requirement met
Salary requirement met
Loan APPROVED!
- Run 2: PS E:\Python Code> python.exe "C:/Hassan_Zaidi/AppData/Local/Programs/Python/Python314/python.exe" "e:/Python Code/nested if statement program.py"
Enter your age:24
Enter your monthly Salary:18000
Age requirement met
Salary too low
Loan REJECTED!

3.5.2.1 Differentiate between if and if-else:

Features	If Statement	if-else Statement
Definition	Executes a block only when condition is TRUE	Executes one of two blocks based on condition
Syntax	if condition :	if condition : else :
Number of Blocks	1 block only	2 blocks (if + else)
When Condition is TRUE	Execute if block	Execute if block
When Condition is FALSE	Does nothing, skips silently	Execute else block
Action Guaranteed	No, may perform nothing	Yes, always perform something
Decision Type	Optional action	Mandatory choice
Example Scenario	“if score is 100 → award trophy”	“if pass → promote, else → fail”
Code Example	Code should indent if marks >=50: print(“Pass”)	Code should indent if marks >=50: print(“Pass”) else: print(“Fail”)

3.5.3 Repetition Statement:

Repetition statement (also called a **loop**) is a control structure that allows a block of code to be executed **repeatedly** based on a condition or a sequence. Loop reduces the repetitive code. Python has two primary types of loops:

1) **for Loop:**

Used when the number of repetitions/iterations is known. It iterates over a **sequence** (string, range, list, tuple, etc.)

```
for variable in sequence:
    # body of loop
```

All commands that need to be iterated should come after the colon, and must be indented (remember topic 3.5.2 if-statement).

Unit 3: Programming Fundamentals

```
FOR Loop program.py X
FOR Loop program.py > ...
1 # This is a For Loop statement program
2
3 for i in range(5):
4     print (i)
```

TERMINAL Python + v [] [] ... | [] X

```
PS E:\Python Code> & "C:/Python/Python314/python.exe" "e:/Python Code/FOR Loop program.py"
0
1
2
3
4
```

a) range() Function:

The range() function is a built-in Python function that generates a sequence of numbers. It is most commonly used with for loops to repeat a block of code a specific number of times. The range() function has following three forms

range(stop)

range(start, stop)

range(start, stop, step)

start: The beginning value of the sequence. Default value is 0

stop: The ending value

step: The increment between each number. Default is 1.

i) range (stop):

Generates numbers from 0 up to (but not including) stop.

```
for i in range(5):
    print(i)
```

Output: 0 1 2 3 4

ii) range (start , stop):

Generates numbers from start up to (but not including) stop.

```
for i in range(2, 8):
    print(i)|
```

Output: 2 3 4 5 6 7

iii) **range** (start , stop , step):

Generates numbers from start to stop, incrementing by step.

(1) Positive step (count up)

```
for i in range(0, 10, 2):
    print(i)
```

```
# Output: 0 2 4 6 8
```

(2) Negative step (count down):

```
for i in range(10, 0, -2):
    print(i)
```

```
# Output: 10 8 6 4 2
```

Example: Even & Odd number program

```
FOR Loop program2.py ×
FOR Loop program2.py > ...

1 # This is a For Loop statement program
2
3 for i in range(0,10,2):
4     print ("Even ",i)
5
6 for i in range(1,10,2):
7     print ("Odd ",i)
```

TERMINAL

```
Python + - [ ] [ ] ... [ ] [ ]
Even 0
Even 2
Even 4
Even 6
Even 8
Odd 1
Odd 3
Odd 5
Odd 7
Odd 9
```



Activity

1. Write a program that prints even and odd number between 1 to 20 by using arithmetic operator.
2. Take the input from user and print the multiplication table.

2) while Loop:

Repeats a block of code **as long as a condition is True**. Used when the number of iterations is **not known in advance**. Condition controls how many times the loop runs.

```
while condition:  
    # body of loop
```

Example:

```
count = 1  
while count <= 5:  
    print(count)  
    count += 1  
# Output: 1 2 3 4 5
```

❖ Loop Control Statements:

Following are the statements that alter the flow of Loop.

- **break:** Exits the loop immediately. Following is the program that stops the loop when 3 is found.

```
for i in range(6):  
    if i == 3:  
        break  
    print(i)  
# Output: 0 1 2
```

- **continue:** Skips the current iteration and moves to the next. Following is the program that skips the even number and prints only odd number.

```
for i in range(6):  
    if i % 2 == 0:  
        continue  
    print(i)  
# Output: 1 3 5
```

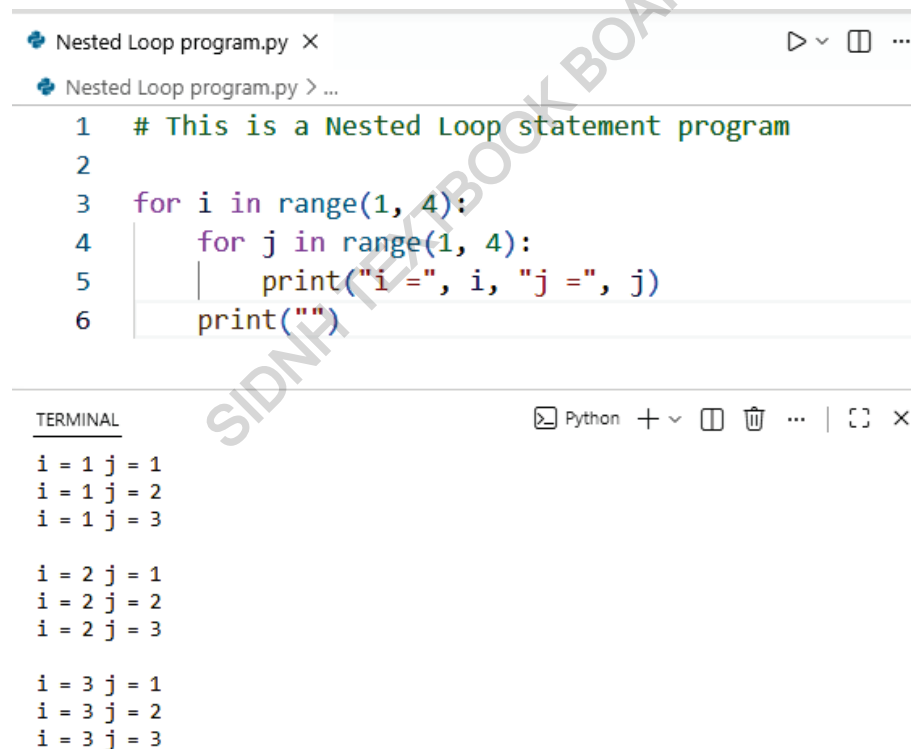
❖ **Nested Loop:**

A **nested loop** is a loop inside another loop. The inner loop runs completely for every single iteration of the outer loop. Following is the syntax of nested loop

```
for outer in range(...):
    for inner in range(...):
        # code block
```

The outer loop runs first, and for each of its iterations, the inner loop executes completely; once the inner loop finishes, the outer loop moves to the next iteration and the process repeats.

A Nested loop is used when a task needs to be repeated inside another repetition. It helps solve problems involving combinations, patterns, and multi-step repetition. However, common mistakes include incorrect indentation and forgetting that the inner loop restarts and runs completely for each iteration of the outer loop.



```
Nested Loop program.py X
Nested Loop program.py > ...

1  # This is a Nested Loop statement program
2
3  for i in range(1, 4):
4      for j in range(1, 4):
5          print("i =", i, "j =", j)
6      print("")

TERMINAL
i = 1 j = 1
i = 1 j = 2
i = 1 j = 3

i = 2 j = 1
i = 2 j = 2
i = 2 j = 3

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

**Instructions for Teachers**

Teacher should demonstrate the Nested Loop example using mix up the **for** and **while**.

❖ Differentiate between for Loop vs while Loop:

for Loop	while Loop
When number of iterations is known	When number of iterations is unknown but based on condition
Used to iterate over a fixed sequence of items.	Used for more dynamic scenarios where conditions dictate loop continuation.
Cleaner and shorter for counting tasks	Requires manual update of condition

Activity: Program that uses Sequence, Selection and Repetition

1. Program Check Even Odd number:

The following program shows that the loop runs three times, takes input from the user three times, and checks each time whether the entered number is even or odd.

```
for i in range(3):                                # Repetition
    number = int(input("Enter a number: "))        # Sequence
    if number % 2 == 0:                            # Selection
        print("Even")
    else:
        print("Odd")
```

2. Program Shopping Bill Calculator:

Combine program.py ×

Combine program.py > ...

```

1  # Shopping Bill Calculator
2
3  print("===== SHOPPING BILL CALCULATOR =====")
4  total    = 0.0    # float - total bill amount
5  count    = 0      # int   - number of items
6  shopping = True   # bool  - controls the loop
7
8  # REPETITION: Keep adding items
9  while shopping:
10     item = input("\nEnter Item Name (or 'done' to finish): ")
11     # SELECTION: Check if user wants to stop
12     if item.lower() == "done":
13         shopping = False
14     else:
15         price = float(input(f"Enter Price of {item}: Rs. "))
16         # SELECTION: Validate price
17         if price <= 0:
18             print("Invalid price! Try again.")
19         else:
20             total = total + price
21             count = count + 1
22             print(f"{item} added - Rs. {price:.2f}")
23
24  # — Summary —
25  print("\n===== BILL SUMMARY =====")
26  if count == 0:
27      print("No items purchased.")
28  else:
29      print(f"Total Items   : {count}")
30      print(f"Total Bill    : Rs. {total:.2f}")
31
32      # SELECTION: Apply discount based on total
33      if total >= 5000:
34          discount = total * 0.20
35          label    = "20% Discount"
36      elif total >= 2000:
37          discount = total * 0.10
38          label    = "10% Discount"
39      else:
40          discount = 0
41          label    = "No Discount"
42
43      final = total - discount
44      print(f"Discount      : {label} (-Rs. {discount:.2f})")
45      print(f"Final Bill    : Rs. {final:.2f}")
46  print("=====")

```

3.6 Libraries In Python:

Libraries are pre-written code that offer helpful functions and modules to complete specific tasks. By using libraries, programmers can build programs more efficiently and with fewer errors, as they do not have to write the same code by themselves repeatedly. Libraries make Python powerful and flexible. They are used to:

- Perform mathematical calculations
- Work with files and data
- Create graphics and charts
- Build web and desktop applications
- Automate tasks

Instead of writing complex code from scratch, programmers use libraries that already solve common problems. There are mainly 5 ways to import a library into a program.



1. **import module:**

Import the **entire module**. Access functions using `module.function()`

```
import math
```

```
result = math.sqrt(25)
```

```
print(result)          # Output: 5.0
```

```
result = math.pi
```

```
print(result)          # Output: 3.141592653589793
```

2. **import module as alias:**

Import module with a short nickname (alias).

```
import math as m
```

```
print(m.sqrt(16))      # Output: 4.0
```

```
print(m.pi)           # Output: 3.141592653589793
```

3. from module import function:

Import only a specific function from a module.

```
from math import sqrt

print(sqrt(36))      # Output: 6.0
# No need to write math.sqrt()
```

4. from module import function 1, function 2:

Import multiple specific functions at once.

```
from math import sqrt, pi, factorial

print(sqrt(49))      # Output: 7.0
print(pi)            # Output: 3.141592653589793
```

5. from module import *:

Import everything from a module.

```
from math import *

print(sqrt(64))      # Output: 8.0
print(pi)            # Output: 3.141592653589793
print(ceil(4.3))     # Output: 5
```

There are two types of libraries built-in and third-party libraries.

3.6.1 Built-in Libraries:

These libraries come with Python and do not need to be installed; users can readily use them by importing them into the program. Some of the most common libraries are listed below.

Unit 3: Programming Fundamentals

Examples:

- **math:** library provides mathematical operations and constants. It includes functions for trigonometry (like $\sin \theta$, $\cos \theta$), logarithms, exponentiation and constants like π .

```
Math Library program.py ×
Math Library program.py
1  import math
2
3  print(math.pi)           # Pi value      → 3.14159...
4  print(math.ceil(4.3))    # Round up    → 5
5  print(math.floor(4.9))   # Round down   → 4
6  print(math.pow(2, 4))    # Power 2^4     → 16.0
7  print(math.sqrt(81))     # Square root  → 9.0

TERMINAL
Python + v [ ] [ ] ... | [ ] ×
PS E:\Python Code> "C:/Python/Python314/python.exe" "e:/Python Code/Math Library program.py"
3.141592653589793
5
4
16.0
9.0
PS E:\Python Code> |
```

- **random:** library is used for generating random numbers or making random selections.

```
random program.py ×
random program.py > ...
2  import random
3
4  # Generating a random number between 1 to 10
5  random_num = random.randint(1,10)
6  print("Random Number: ",random_num)
7  print(random.random())    # Random float
8  print(random.uniform(1.5, 9.5)) # Random float in range
9

TERMINAL
Python + v [ ] [ ] ... | [ ] ×
PS E:\Python Code> "C:/Python/Python314/python.exe" "e:/Python Code/random program.py"
Random Number:  8
0.5689448558030233
4.15758310615087
PS E:\Python Code> |
```

- **datetime**: library provides classes for working with dates and time.

• datetime prog.py ×

▶ ▾ □ ...

C: > Users > Hassan Zaidi > Python Program > • datetime prog.py > ...

```
1  import datetime
2
3  # Getting current date and time
4  current_datetime = datetime.datetime.now()
5  print("Current Date and Time: ", current_datetime)
6
7
8  formatted_date = current_datetime.strftime("%d-%m-%Y")
9  print ("Formatted Date: ",formatted_date)
10
```

Formatting a date
%Y means year part,
%m means month part
%d means date part

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Hassan Zaidi> python.exe "c:/Users/Hassan Zaidi/Python Program
/datetime prog.py"
Current Date and Time: 2026-01-19 21:47:00.099923
Formatted Date: 19-01-2026
PS C:\Users\Hassan Zaidi> █
```

Activity: Program that generate student report card by using library functions

• student report program.py ×

• student report program.py > ...

```
1  # Student Report Card Generator
2  # Libraries: math, random, datetime
3
4  import math          # For mathematical calculations
5  import random        # For generating roll number
6  import datetime      # For report date and time
7
8  print("      STUDENT REPORT CARD GENERATOR")
9  # — datetime library —
10 # Get current date and time for the report card
11 now = datetime.datetime.now()
12 date = now.strftime("%d-%m-%Y") # Format: Day-Month-Year
13 time = now.strftime("%I:%M %p") # Format: Hour:Min AM/PM
14 print(f>Date : {date} Time : {time}")
15
16 # — random library —
17 # Generate a random Roll Number
18 roll_no = random.randint(100, 999)
19
```


3.6.2 Third Party Libraries:

A **third-party library** is not included with Python by default. It is installed separately and adds extra features, such as data handling, plotting, or web access. Before using any third-party library, it must be installed using:

```
pip install library_name
```

Examples:

- **Numpy:** numerical computing
- **Pandas:** data analysis, use to handle table like data.
- **Matplotlib:** graphs and charts



3.7 Debugging Code in Python:

Debugging is the process of identifying and resolving issues within a program to ensure it functions as intended. This involves locating the source of errors, understanding the underlying causes, and implementing the necessary corrections.

Bugs or errors can cause a program to crash, produce incorrect results, or behave unexpectedly. These errors can be broadly categorized into three main types



- **Syntax Errors:** Occur when the rules of Python are not followed (any writing mistake in keywords). It prevents the program from running at all.

```
print("Hello World"  # Missing closing parenthesis
```

- **Runtime Errors (Exceptions):** Occur while the program is running. The program starts but fails during execution.

```
x = 10 / 0  # Division by zero
```

- **Logical Errors:** Occur when the program runs without crashing but produces the wrong output. It is harder to detect because there is no error message.

```
i = 1
while i < 6:
    print(i)
    i -= 1  #logical error: - minus should be replace with + plus sign
```


Unit 3: Programming Fundamentals

There is a **logical error** in the above code, that is instead of increasing “i” to eventually reach 6, it is decreasing. This creates an infinite loop.

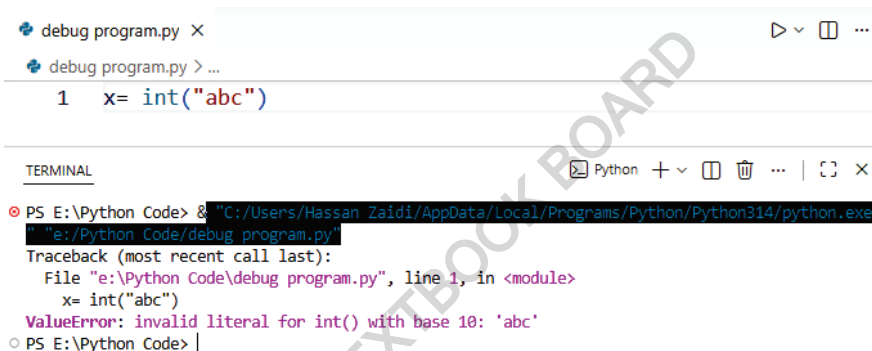
Following are some multiple ways to debug code from simple techniques.

➤ Reading Error Messages (Tracebacks):

When an error occurs in a Python program, Python provides a **detailed error message** to help the user understand what went wrong. These messages inform the user:

- ✓ What type of error occurred
- ✓ Where the error happened
- ✓ Which line caused the problem

Example 1:

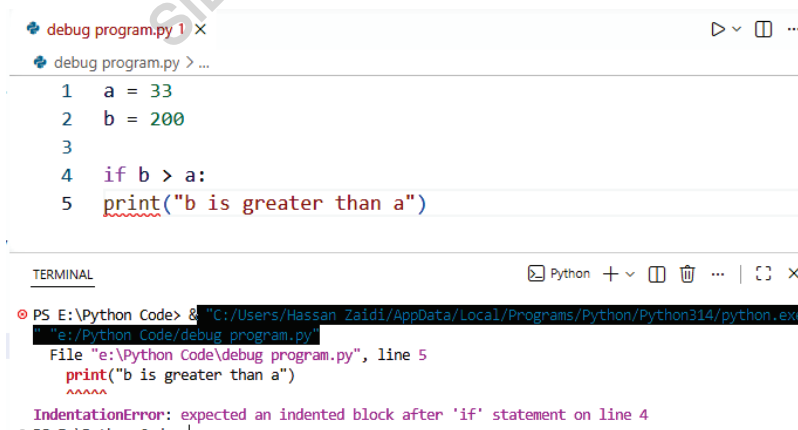


```
debug program.py ×
debug program.py > ...
1 x= int("abc")

TERMINAL
PS E:\Python Code> & "C:/Users/Hassan_Zaidi/AppData/Local/Programs/Python/Python314/python.exe"
"e:/Python Code/debug program.py"
Traceback (most recent call last):
  File "e:/Python Code/debug program.py", line 1, in <module>
    x= int("abc")
ValueError: invalid literal for int() with base 10: 'abc'
PS E:\Python Code> |
```

This clearly indicates that Python cannot convert text into a number. Error messages often point directly to the problem line.

Example 2:



```
debug program.py 1 ×
debug program.py > ...
1 a = 33
2 b = 200
3
4 if b > a:
5 print("b is greater than a")

TERMINAL
PS E:\Python Code> & "C:/Users/Hassan_Zaidi/AppData/Local/Programs/Python/Python314/python.exe"
"e:/Python Code/debug program.py"
File "e:/Python Code/debug program.py", line 5
    print("b is greater than a")
    ~~~~~
IndentationError: expected an indented block after 'if' statement on line 4
PS E:\Python Code> |
```

This clearly indicates that at line no 5 the indentation is expected after if statement.



Career Opportunities

This unit introduces Python programming, including variables, control structures, loops, and the use of libraries to build practical programs. Programming skills are among the most in-demand abilities in the modern job market and open doors across nearly every industry.

Python / Software Developer

Writes, tests, and debugs applications ranging from desktop tools to large-scale enterprise systems.

Web Developer

Builds websites and web applications using Python frameworks alongside HTML, CSS, and JavaScript.

Data Analyst

Uses Python libraries to process, visualize, and interpret data for business decision-making.

Automation / Scripting Engineer

Writes Python scripts to automate repetitive tasks and improve workflow efficiency.

Mobile/App Developer

Applies programming logic learned in Python to design and build mobile and desktop applications.

Game Developer

Uses programming fundamentals and libraries to design logic, mechanics, and interactivity in games.



Summary

- ◆ In Python, the **input()** function allows to capture user input from the keyboard, while the **print()** function is to display output to the console.
- ◆ **input()** always returns a string, convert it to **int** or **float** data type if needed.
- ◆ **Variable** is a named space of a memory.
- ◆ A **variable** is like container or storage box that holds a value.
- ◆ The data type of a variable can **change dynamically** when a new value is assigned.

- ◆ An **IDE** is a complete workspace that includes a Code Editor, Syntax Highlighting, Auto Completion, Run/Execute Tool, and Debugger.
- ◆ **VS Code** is a free, lightweight code editor that becomes a full IDE by installing extensions.
- ◆ Python automatically detects the data type, so do not have to declare it.
- ◆ **Arithmetic Operators** perform mathematical calculations (+, -, *, /, //, %, **).
- ◆ **Assignment Operators** assign values to variables (=, +=, -=, *=, /=).
- ◆ **Relational Operators** compare two values and return True or False (==, !=, >, <, >=, <=).
- ◆ **Logical Operators** combine conditions (and, or, not).
- ◆ **Membership Operators** check if a value exists in a sequence (in, not in).
- ◆ **Bitwise Operators** operate on binary digits bit by bit (&, |, ^, ~, <<, >>).
- ◆ **Sequence, Selection and Repetition controls** together form the foundation of all Python programs.
- ◆ **break** exits the loop immediately, while **continue** skips the current iteration.
- ◆ **Libraries** provide pre-made code for common programming tasks.
- ◆ **Built-in libraries** (math, random, datetime) come with Python and do not need installation.
- ◆ **Third-party libraries** (NumPy, Pandas, Matplotlib) must be installed separately using **pip install**.
- ◆ **Debugging** is the process of identifying and fixing errors in a program.
- ◆ There are three types of errors: **Syntax Errors**, **Runtime Errors**, and **Logical Errors**.
- ◆ Python provides **detailed error messages (Tracebacks)** that tell the type, location, and line of the error.
- ◆ Logical errors are the hardest to detect because the program runs but produces **wrong output**.



Exercise

1. Encircle the correct answer:

- i) Which is the correct statement to insert Comments in Python code?
 - (a) // This is comment
 - (b) /* This is comment*/
 - (c) # This is comment
 - (d) - - This is comment
- ii) Which one is not a legal variable name?
 - (a) _myvar
 - (b) my_var
 - (c) Myvar
 - (d) my-var
- iii) Which one is the correct way to write if statement in Python?
 - (a) if (condition):
 - (b) if(condition);
 - (c) if (1):
 - (d) none of these
- iv) ____ is the operator can be used to compare two values?
 - (a) < >
 - (b) ==
 - (c) ><
 - (d) =
- v) ____ loop is used when the number of iterations is known in advance
 - (a) While
 - (b) Do while
 - (c) For
 - (d) For each
- vi) Which programming language is closest to binary (0s and 1s)?
 - (a) High Level language
 - (b) Mid-Level Language
 - (c) Low Level language
 - (d) Script Language
- vii) Standard indentation in Python?
 - (a) 2 spaces
 - (b) 4 spaces
 - (c) 6 spaces
 - (d) 8 spaces
- viii) ____ data type does the input() function always return.
 - (a) Int
 - (b) Float
 - (c) String
 - (d) Bool
- ix) ____ statement exits a loop immediately
 - (a) Pass
 - (b) Continue
 - (c) Break
 - (d) Exit
- x) ____ bitwise operator returns 1 only when BOTH bits are 1?
 - (a) AND
 - (b) OR
 - (c) NOT
 - (d) XOR

2. Fill in the Blanks:

- i) Remarks or notes added for the developer in the source code are called _____.
- ii) An identifier whose value can change during program execution is called a _____.
- iii) A _____ cannot be used as a variable name in Python.
- iv) The extension of Python source code files is _____.
- v) The statement used to execute different blocks of code based on a condition in Python is called the _____ statement.

3. Provide short descriptive answers of the following questions.

- i) What is a selection statement? Differentiate between if, elif, and else with an example.
- ii) What is the difference between a for loop and a while loop?
- iii) What are the three types of errors in Python? Define each in one line.
- iv) What is debugging? Explain the techniques used for debugging.
- v) Define following terms:
 - (a) Sequence
 - (b) Datatype
 - (c) Repetition
 - (d) Variable
 - (e) Arithmetic Operator

4. Provide detailed answers of the following questions.

- i) What are control structures in Python? Explain Sequence, Selection (if, if-else, if-elif-else, Nested if), and Repetition (for loop, while loop, Nested loop) with syntax and code examples for each.
- ii) Explain variables and data types in Python. List all built-in data types with their names, type keywords, descriptions, and examples in the form of a table.
- iii) Describe input and output handling in Python. Explain the print() and input() functions with their syntax, purpose, and at least two code examples each.
- iv) Explain all Arithmetic and Assignment Operators in Python with their symbols, purpose, examples, and results in tabular form. Also write a code example for each.
- v) What are the different types of error in Python? Explain each type in detail with code examples for each type. Also describe the technique of reading error messages (Trackbacks).



Activities

1. Write a program of arithmetic calculator that takes two integer input from user also take a choice for arithmetic calculation. Display proper messages for user.
2. Create a “guess the number” game where the player has only one chance to guess the correct number. Use Python’s random module to generate a random number and compare it with the user’s input. If the guess matches the number, print “You win”; otherwise, print “You lose.”
3. Write a Python program that simulates a basic ATM Machine using simple data types (int, float, str, bool). The program should combine Selection (if/elif/else) and Repetition (while loop) statements to perform real-world banking operations such as PIN verification, balance inquiry, cash deposit, and cash withdrawal. The output of the program should look like as following.

===== WELCOME TO ATM =====

Enter PIN: 0000
Wrong PIN! 2 attempt(s) left.
Enter PIN: 1234

Login Successful!

----- ATM MENU -----

1. Check Balance
2. Deposit
3. Withdraw
4. Exit

Choose (1-4): 1
Your Balance: Rs. 10000.00

Choose (1-4): 2
Enter deposit amount: Rs. 5000

Deposited Rs. 5000.00
New Balance: Rs. 15000.00

Choose (1-4): 3
Enter withdrawal amount: Rs. 3000

Withdrawn Rs. 3000.00
Remaining Balance: Rs. 12000.00

Choose (1-4): 4
Thank you! Goodbye.

=====

Data and Analysis

Student Learning Outcomes:

After the completion of this unit students will be able to:

- ◆ Explain the concepts of a database and its elements.
- ◆ Describe basic types of relationship (1:1, 1:M, M:N)
- ◆ Explain the concept of referential integrity (cascade update/delete)
- ◆ Develop relational schema.
- ◆ Develop Entity-Relationship Models using real life examples.
- ◆ Use Database Objects (Tables, Forms, Queries and Reports)
- ◆ Use database objects (tables, forms, and queries)
- ◆ Create tables using appropriate datatypes.
- ◆ Create queries based on different criteria
- ◆ Design reports based on data selection
- ◆ Analyze existing datasets to create summary statistics and data visuals using relational DBMS.



Introduction to Unit:

This unit helps students understand how data is organized, analyzed, and used to make informed decisions, covering key concepts, real-world examples, and digital tools.

Data is generated all around us through smartphones, online shopping, social media, and smart cities. Organizations across every sector rely on it to plan and improve services. However, data alone means nothing; it becomes knowledge only when analyzed.

For Example: A month of recorded temperatures is raw data but calculating averages or spotting trends turns it into insight. That is data analysis.

In this unit, students will explore database concepts, real-world applications, and how digital tools transform raw data into meaningful decisions.

4.1 Concepts of Database and Elements:

4.1.1 Data, Information and Database:

- **Data:**

Data are unprocessed facts, figures, or observations collected from different sources. It can be quantitative (numerical) or qualitative (descriptive).

Examples of data include:

- The number of students in a class (quantitative).
- The color of cars in a parking lot (qualitative).
- Daily temperature readings (quantitative).
- Feedback about a product (qualitative).

- **Information:**

When we organize, process, and interpret this data, it becomes information which provides meaning and supports decision-making.

Example 1:

Raw Data: 80, 75, 90, 70, 85

Processed Information: The average score of the class is 80%.

Hence, Data → Processing → Information

The Table 4.1 highlights the difference between Data and Information

Table 4.1: Difference between Data and Information

Aspect	Data	Information
Definition	Raw facts and figures	Processed and Meaningful data
Example	85,90,75	Average Marks = 83.3%
Usefulness	Less useful on its own	Useful for decision making
Format	Numbers text, images, symbols, videos, etc.	Reports, Summaries, Charts, etc.

- **Database**

A database is an organized collection of related data that is stored in a structured manner so that it can be easily accessed, managed, and updated. In traditional systems, data was stored in files or books, which made it difficult to search, update, or maintain consistency. A database solves these problems by providing a systematic way to store and retrieve data.

4.1.2 Database Management System:

A Database Management System (DBMS) is software that allows users to create, manage, and interact with databases. It acts as a bridge between the user and the database. Instead of directly handling data files, users interact with DBMS, which performs operations such as inserting, updating, deleting, and retrieving data.

For example, in a school environment, a DBMS can be used to manage student data. Instead of writing information in books, the school can store all student data in a database. Teachers can quickly search for a student's data, update marks, or generate reports using the DBMS. DBMS provides the following advantages:

- i) **Less Redundancy:** Which means the same data is not stored multiple times unnecessarily
For example, instead of storing a student's department name in every table (Student table, Exam table, Fee table), the department is stored once in the Department table, and other tables refer to it using a DepartmentID.
- ii) **Improved Data Consistency:** which ensures that all users see the same correct data
For example, if a student's contact number is updated in the Student table, every query, form, or report that accesses this table will automatically show the updated contact number, so all users see the same information.
- iii) **Improved Security:** Allowing only authorized users to access sensitive information.
For example, only the admin or teacher can view students' exam marks, while general staff can only view names and contact details.
- iv) **Data integrity:** It ensures that the data stored in the database is accurate and reliable.
For example, a student's roll number must be unique, and the system will prevent duplicate entries.

Common examples of DBMS include Microsoft Access, MySQL, Oracle, and SQL Server. These systems are widely used in educational institutions, businesses, banks, and government organizations.

4.2 Database Elements:

A database is made up of several important components that help organize data effectively. The most basic structure of a database is a table. A table is used to store data in rows and columns, similar to a spreadsheet in MS Excel. Each table is designed to store information about a specific entity, such as students, teachers, or courses. Each table will be stored as a separate file.

The rows in a table are called records. Each record represents a single entry or instance of data. For example, in a student table, each row represents one student and contains all the information related to that student, such as name, roll number, class, and marks.

The columns in a table are called fields. Each field represents a specific type of data or attribute. For example, in a student table, fields may include StudentID, Name, Class, and Marks. Each field contains values of a particular type, such as text, numbers, or dates. Fig. 4.1 shows an illustration of these components.

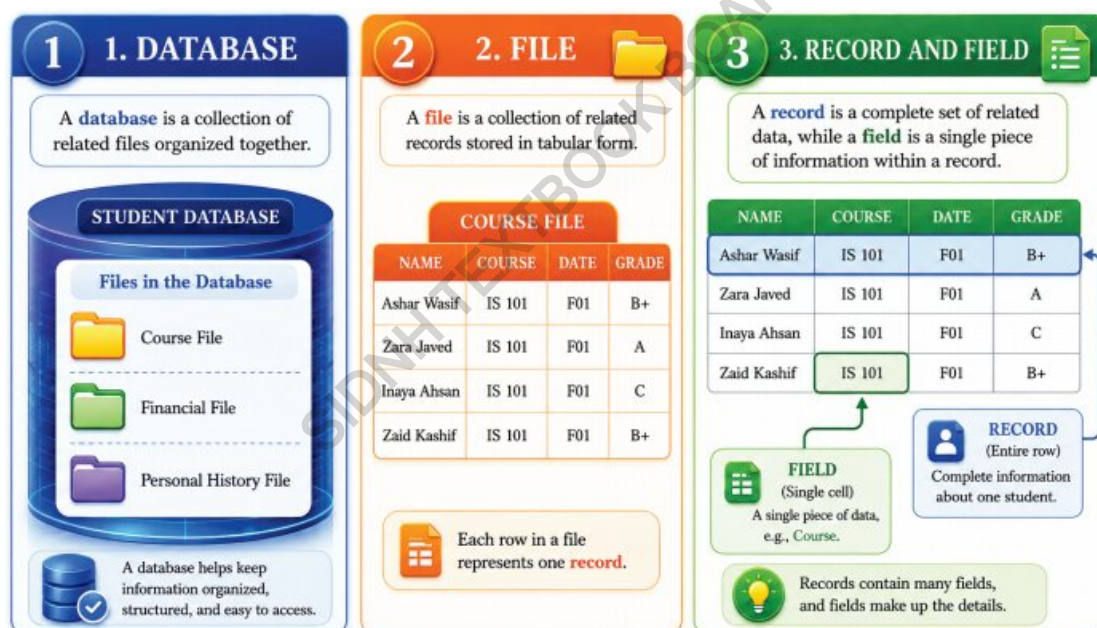


Figure 4.1: Database Components

4.2.1 Keys and Integrity Constraints:

Another important concept related to tables is the use of key fields. A key field is an attribute (or a set of attributes) that helps uniquely identify records and link tables together. Understanding different types of keys is essential for designing efficient and reliable database systems. The following types of key fields can exist in the database.

- i) **Candidate Key:** A candidate key is any field (or set of fields) that can uniquely identify a record in a table. A table may have multiple candidate keys, but only one of them is selected as the primary key.

Characteristics:

- Must contain unique values
- Cannot contain NULL values
- There can be multiple candidate keys

For example, in an Emp table, EmpID, EmpLicense#, and EmpPassport can be candidate keys as they uniquely identify each record.

- ii) **Primary Key:** A primary key is a field (or combination of fields) that uniquely identifies each record in a table. It ensures that no two records have the same value and that no value is left null.

Characteristics:

- Must contain unique values
- Cannot contain NULL values
- Each table has only one primary key

For example, in an Emp Table, EmpID is the primary key because it uniquely identifies each employee record.

- iii) **Alternate Key:** An alternate key is a candidate key that is not selected as the primary key. These keys still uniquely identify records but serve as alternatives.

Characteristics:

- Derived from candidate keys
- Not chosen as primary key
- Contains unique values

For example, in an Emp Table, EmpLicense# and EmpPassport will serve as alternate keys.

- iv) **Foreign Key:** A foreign key is a field in one table that refers to the primary key in another table. It is used to create relationships between tables and maintain referential integrity.

Characteristics:

- Can contain duplicate values
- Can contain NULL (in some cases)
- Establish relationships between tables

For example, “Did” will serve as Foreign Key between employee and department table as shown in figure 4.2.

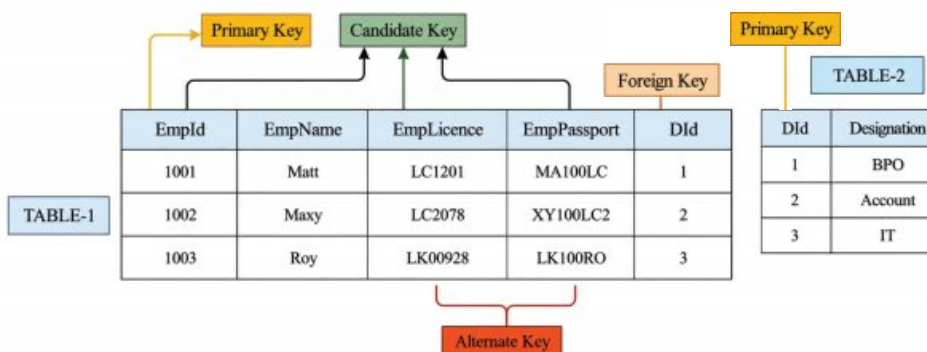


Figure 4.2: Key fields in a database

4.3 Relational Database Management System (RDBMS):

A Relational Database Management System (RDBMS) is one of the most widely used database models in modern computing. A Relational Database Management System (RDBMS) stores data in tables and uses keys and constraints to maintain relationships and ensure data integrity. It is a powerful and structured way of managing data, widely used in applications such as banking systems, university databases, and online services. RDBMS offers the following advantages:

- Ensures data consistency and accuracy
- Reduces data redundancy
- Supports data sharing and multi-user access
- Provides security and access control
- Easy to query using SQL

4.4 Entity Relationship Model (ER-Model):

The Entity Relationship (ER) Model is a conceptual tool used to design databases before they are created. It helps in understanding what data needs to be stored and how different pieces of data are related to each other. Designing a database without proper planning can lead to errors, duplication, and inefficiency. Therefore, the ER model plays a very important role in database development.

An ER model consists of three main components:

1. Entities:

An entity is any real-world object or concept about which data is stored. For example, in a school database, entities can include Student, Teacher, and Course. These are things that exist independently and have their own identity.

2. Attributes:

Attributes are the properties or characteristics of an entity. For example, the Student entity may have attributes such as StudentID, Name, Class, and Marks. These attributes provide detailed information about the entity.

3. Relationships:

Entities in a database are connected through relationships, which define how data is stored and linked between tables in the ER model. Understanding these relationships is essential for designing efficient and accurate databases.

There are three main types of relationships:

- **One-to-One (1:1):** One record is related to only one record of another entity (e.g., one student has one ID card). This is rare and used for separation or security.
- **One-to-Many (1:M):** One record is related to multiple records of another entity (e.g., one teacher teaches many students). This is the most common type.
- **Many-to-Many (M:N):** Multiple records of one entity relate to multiple records of another (e.g., students enroll in many classes, and classes have many students).

Since relational databases do not directly support many-to-many relationships, a junction (bridge) table is used to break them into two one-to-many relationships.

For example, many students can enroll in many classes; therefore, this many-to-many relationship is represented using a junction table (Enrolment), which stores StudentID and ClassID to link students with their respective classes.

Properly defining relationships ensures logical data organization, avoids redundancy, and improves data accuracy and retrieval.



Activity

Teacher must give students a case study (e.g., library system).
Ask them to list relationships between entities and assign types.

The ER model is usually represented using an ER diagram. In ER diagram, entities are shown as rectangles, attributes as ovals, and relationships as diamonds. For example, A school maintains information about Students, Teachers, and Classes.

- Each student is identified by a unique ID and name and can enroll in multiple classes.
- Each teacher also has a unique ID and name and may teach several classes.
- Every class has a subject and room number, is taught by one teacher, and can include many students.
- To manage student participation in classes, an Enrolment entity is used to record which student is registered in which class.

The ER diagram for this scenario is shown in Fig 4.3. It provide a clear visual representation of the database structure, making it easier to understand and communicate.

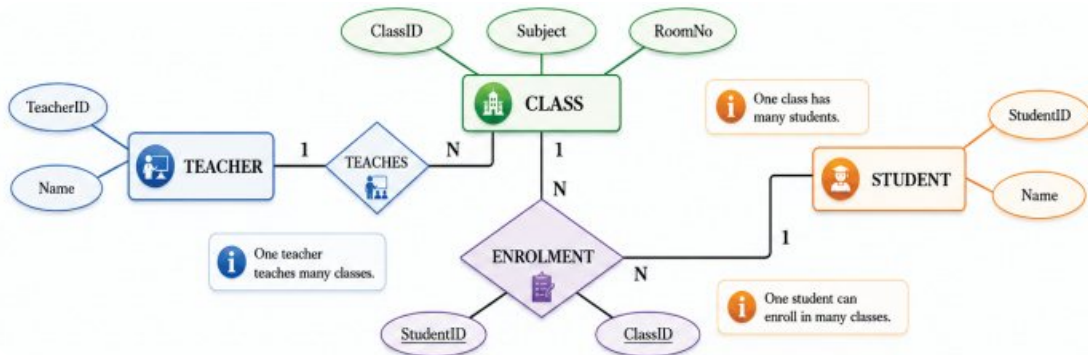


Figure 4.3: An Example ER Diagram

Designing a database for a college, the ER model helps identify all required data and their connections before implementation. This reduces errors and ensures that the database is efficient and well-structured.

In conclusion, the ER model is an essential step in database design. It provides a clear blueprint that guides the creation of a reliable and efficient database system.



Activity

Teacher must ask students to pick a real-world scenario (school, library, hospital, e-commerce store). Also, list all entities they can identify (e.g., Student, Teacher, Course, Book, Patient).

4.5 Referential Integrity and Its Importance:

Referential integrity is a fundamental concept in relational databases that ensures the accuracy and consistency of data across related tables. When multiple tables are connected using foreign keys, it is important that the relationships between them remain valid at all times. Referential integrity ensures that a foreign key in one table always refers to an existing and valid record in another table.

To understand this concept, consider a simple example of a school database with two tables: a Student table and a Result table. The Student table contains student details such as StudentID and Name, while the Result table contains marks along with StudentID as a foreign key. Referential integrity ensures that every StudentID in the

Result table must exist in the Student table. If a StudentID appears in the Result table but does not exist in the Student table, it creates inconsistency and confusion.

There are also operations related to referential integrity, such as cascade update and cascade delete. In cascade update, if a primary key value is changed in the parent table, the change is automatically updated in all related tables. In cascade delete, if a record is deleted from the parent table, all related records in child tables are also deleted automatically. This helps maintain consistency without manual effort. Fig. 4.4 illustrates the effects of cascade update and cascade delete operations.

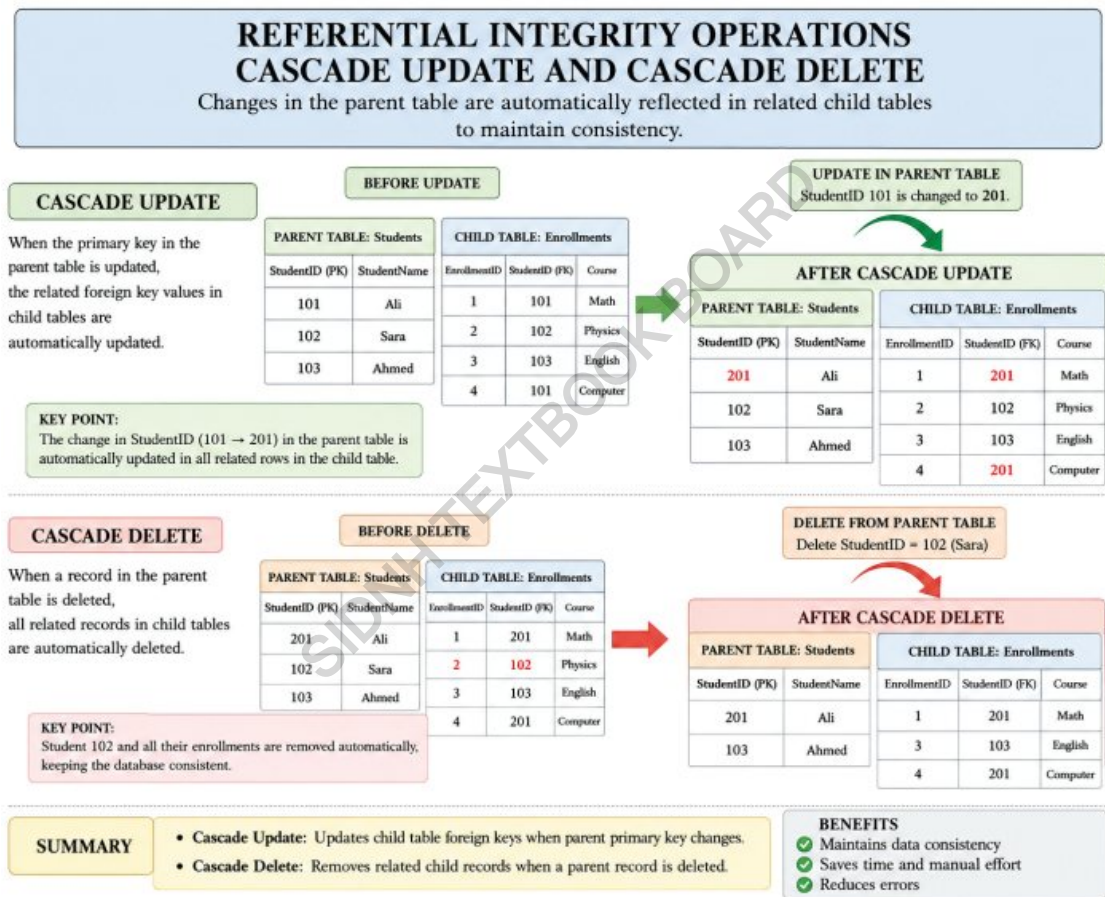


Figure 4.4: Cascade Update and Cascade Delete Illustration

In real-life systems, such as schools, colleges and university databases, referential integrity is very important. For example, if a student is removed from the system, all related records such as attendance, results, and fee records should also be handled properly. Without referential integrity, the database may contain incomplete or incorrect data.

4.6 Relational Schema Development:

A relational schema is a formal representation of the structure of a database. It defines how data is organized into tables, what fields each table contains, and how different tables are related to each other. In simple terms, a schema acts as a design of the database before it is implemented.

Developing a relational schema is an important step in database design because it helps in organizing data logically and efficiently. A well-designed schema reduces redundancy, improves data integrity, and makes the database easier to manage.

To understand relational schema development, consider a school database system.

- Initially, we identify the main entities such as Student, Course, and Teacher. Each of these entities will be represented as a table in the database. For example, the student table may contain fields such as StudentID, Name, Class, and Address. Similarly, the Course table may include CourseID and CourseName.
- After identifying tables, the next step is to define relationships between them. For example, students enroll in courses. Since this is a many-to-many relationship, we create an additional table called Enrollment. This table contains StudentID and CourseID as foreign keys, linking the Student and Course tables.
- Another important aspect of schema design is defining keys. Each table must have a primary key that uniquely identifies its records. Foreign keys are used to establish relationships between tables.

4.7 Case Study (Library Management System):

This example demonstrates the step-by-step development of an Entity-Relationship (ER) schema, starting from requirement analysis to the final ER diagram. In this example, database schema for a Library Management System will be developed. This system will be used to organize and manage books, members, and borrowing activities in a library.

4.7.1 Step-1: Requirement Analysis

Before designing the database, the system requirements are carefully analyzed. Suppose a college library needs to maintain the following information:

- Each book has a unique ID, title, author, and publisher.
- Each member (student or teacher) has a unique ID, name, and contact information.
- A member can borrow multiple books, and a book can be borrowed by different members over time.
- The system should record the issue date and return date of each borrowed book.

4.7.2 Step-2: Identification of Entities

Entities represent real-world objects or concepts that can be distinctly identified. From the requirements, the following entities are identified:

- Book
- Member

Each Entity represents a separate table in the database.

4.7.3 Step-3: Identification of Attributes

Attributes describe the properties or characteristics of an entity. The selection of appropriate attributes is essential to ensure accuracy, completeness, and efficient data retrieval. For this particular application the following attributes are identified for each entity:

- **Book Entity Attributes:**

The Book entity represents all books available in the library. Its attributes are defined as follows:

- BookID (Primary Key):
 - A unique identifier assigned to each book. This ensures that every book can be distinctly recognized in the system, even if multiple books have similar titles.
- Title:
 - The name of the book. This attribute helps users easily identify and search for books in the library.
- Author:
 - The name of the person who wrote the book. This attribute is useful for categorization and retrieval based on authorship.
- Publisher:
 - The organization or company responsible for publishing the book. This information may be useful for reference and classification purposes.

- **Member Entity Attributes:**

The Member entity represents individuals who are registered to use the library services. Its attributes include:

- MemberID (Primary Key):
 - A unique identifier assigned to each member. It ensures that every member can be uniquely distinguished within the system.
- Name:
 - The full name of the member. This attribute is used for identification and record-keeping.

- ContactInfo:
 - The contact details of the member, such as phone number or email address. This attribute allows the library administration to communicate with members when necessary.

4.7.4 Step-4: Identification of Relationships:

Identifying relationships is an essential step in ER modeling, as it helps in understanding how data is linked and how different entities interact with each other. In the given scenario, the primary interaction occurs between the Member and Book entities. This interaction can be described as:

- A Member borrows a Book

This relationship indicates that members of the library are able to borrow books available in the collection. It reflects a real-world activity and establishes a meaningful connection between the two entities.

4.7.5 Step-5: Determine Cardinality

It is a crucial aspect of ER modeling because it clarifies the nature of relationships and guides the structure of the resulting database design. In the context of the Library Management System, the relationship between the Member and Book entities is carefully analyzed:

- A member can borrow many books over a period of time. This means a single member is not limited to borrowing just one book; instead, they may borrow multiple books depending on library policies.
- A book can be borrowed by many members. Although a book may be issued to one member at a specific moment, over time it can be borrowed by different members.

This analysis shows that multiple instances of Member can be associated with multiple instances of Book, and vice versa. Therefore, the relationship is classified as:

- Many-to-Many (M:N)

4.7.6 Step-6: Resolving Many to Many Relationship

Relational database systems are designed to handle relationships primarily through one-to-one and one-to-many associations. They do not directly support many-to-many (M:N) relationships because such relationships can lead to data redundancy

and complexity in data management. Therefore, an M:N relationship must be transformed into a structure that the relational model can efficiently implement. To resolve this issue, a new entity is introduced, commonly referred to as a junction entity (or bridge/associative entity). This entity serves as an intermediary that breaks the many-to-many relationship into two one-to-many relationships.

In the Library Management System, the relationship between Member and Book is resolved by creating a new entity:

- BorrowRecord
 - This entity represents each individual instance of a book being borrowed by a member. Instead of directly linking Member and Book in a many-to-many manner, each borrowing activity is recorded as a separate entry in the BorrowRecord entity.
- BorrowRecord Entity Attributes
 - BorrowID (Primary Key):
 - A unique identifier assigned to each borrowing record. It ensures that every transaction is uniquely distinguishable within the system.
 - MemberID (Foreign Key):
 - This attribute references the primary key of the Member entity. It identifies which member is associated with a particular borrowing record.
 - BookID (Foreign Key):
 - This attribute references the primary key of the Book entity. It specifies which book is involved in the borrowing record.

4.7.7 Step-7: Final ER Schema

Finally, the ER diagram reflects the following as shown in Fig. 4.5:

- Member → BorrowRecord (One-to-Many):
 - One member can be associated with multiple borrowing records. Each record in BorrowRecord corresponds to a specific instance of borrowing by that member.

- Book → BorrowRecord (One-to-Many):
 - One book can appear in multiple borrowing records over time. Each record represents a separate instance in which the book is borrowed.

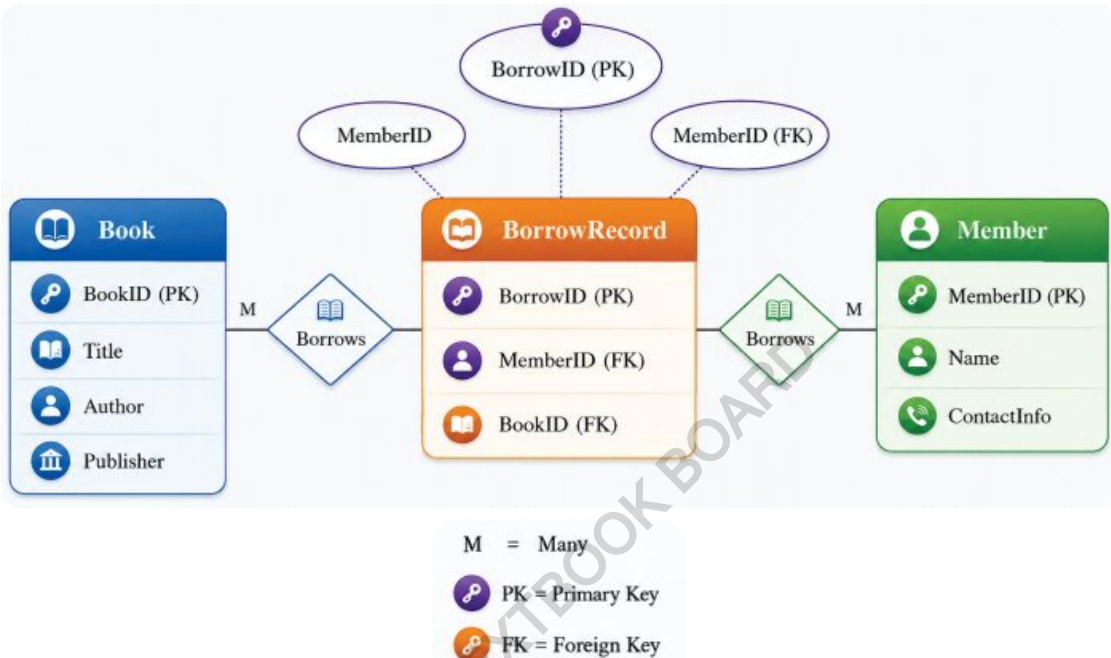


Figure 4.5: ER Diagram of Library Management System

4.8 Different Database Objects:

In a Database Management System (DBMS), data is not only stored but also managed and presented using different tools known as database objects. These objects help users interact with the database in an easy and efficient way. The four main database objects are:

- Tables
- Forms
- Queries, and
- Reports.

1. **Table:** The most important object in the database paradigm is the table, which is used to store data. A table is organized into rows and columns, where each row represents a record and each column represents a field as shown in Fig. 4.6. For example, in a student database, a table may store information such as

StudentID, Name, Class, and Marks. Tables are the foundation of a database because all other objects depend on them.

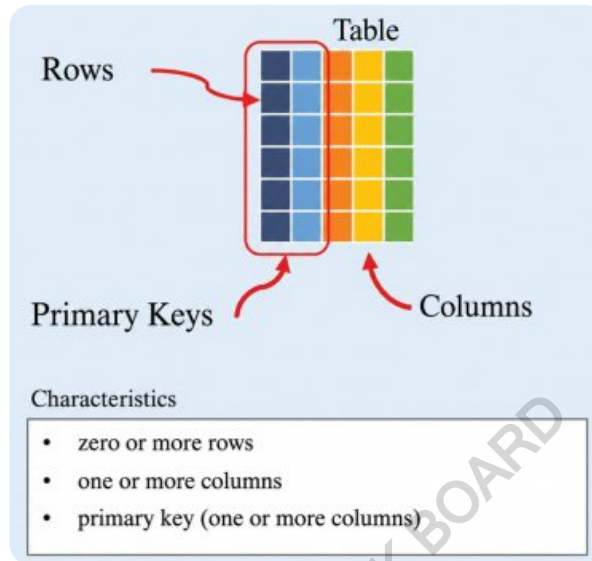


Figure 4.6: Anatomy of the Table

- Form:** The second important object is the form. A form provides a user-friendly interface for entering and updating data. Instead of directly typing data into tables, users can use forms to input information easily. For example, in a school system, an admission form can be designed where users enter student details such as name, date of birth, and class. Forms reduce errors and make data entry simple, especially for users who are not experts in databases. A database form is made up of several interactive and visual components that help users enter and manage data efficiently. These components act as the building blocks of a form in Database Management Systems. Figure 4.7 shows some of these components.
- Query:** The third object is the query, which is used to retrieve specific data from the database. Queries allow users to ask questions from the database and get relevant results. For example, a query can be used to find all students who scored more than 80 marks or to count the total number of students in a class. Queries are very powerful because they help filter, sort, and analyze data.
- Report:** The fourth object is the report, which is used to present data in a structured and formatted way. Reports are often used for printing or sharing information. For example, a school may generate a report showing student results, attendance summaries, or fee records.

In conclusion, database objects make it easier to store, manage, retrieve, and present data. Tables store data, forms help in data entry, queries retrieve information, and reports display results in a readable format.

STUDENT REGISTRATION FORM
Enter student details in the form below

Labels: Descriptive text placed next to input fields. They tell the user what information is required.

Text Boxes: Used to enter text or numeric data.

Drop Down List (Combo List): Provides a list of predefined options.

Radio Buttons: Allows the user to select only one option from the group.

Check Boxes: Allow multiple selections from a list.

Text Area: Used to enter a large amount of textual data, usually multi-line data.

Data Picker: Helps select a date easily. Avoids incorrect date format.

Buttons (Action Buttons): Used to perform specific actions.

Action	Button Label	Description
Submit	SUBMIT	Saves new record
Reset	RESET	Clear all fields
Update	UPDATE	Update selected record
Delete	DELETE	Delete selected record

Submit Button: Saves the entered information as a new record.

Reset Button: Clears all the entered fields.

Update Button: Updates the currently selected record.

Delete Button: Deletes the currently selected record.

This form includes all the essential components used in a database form for data entry and management.

Figure 4.7: Example of Database Form

4.9 Creation of Tables in Microsoft Access:

Creating tables is the fundamental step in developing a database system, as tables serve as the primary structure for storing and organizing data. A carefully designed table not only ensures data accuracy and consistency, but also improves query performance, storage efficiency, and ease of data retrieval. Poor table design, on the other hand, can lead to redundancy, inconsistency, and difficulty in maintaining the database.

4.9.1 Planning the Table Structure:

Before physically creating a table in Microsoft Access, it is important to perform proper planning. This step ensures that the table structure aligns with the requirements of the system and avoids future modifications.

Key planning activities include:

- **Identifying the Purpose of the Table:** Clearly define what information the table will store and how it will be used.
- **Listing Required Fields:** Determine all attributes necessary to represent the entity completely.
- **Selecting Appropriate Data Types:** Choose data types that match the nature of the data (e.g., text, numeric, date).
- **Determining the Primary Key:** Identify a field (or combination of fields) that uniquely identifies each record.

Example: Student Table Design

- | | |
|---------------------------|--------------|
| • StudentID (Primary Key) | - AutoNumber |
| • Name | - Short Text |
| • Department | - Short Text |
| • Age | - Number |
| • Email | - Short Text |

A well-planned table minimizes redundancy and supports efficient database operations.

4.9.2 Create Table using Datasheet View:

The Datasheet View in Microsoft Access provides a straightforward and intuitive method for creating tables, particularly suitable for beginners.

Procedure:

1. Launch Microsoft Access and open a database.
2. Navigate to the **Create** tab and select **Table**.
3. A new table appears in **Datasheet View**, resembling a spreadsheet interface.
4. Enter data directly into the cells. MS Access automatically assigns default field names (e.g., Field1, Field2).

5. Rename fields by double-clicking on the column headers and entering meaningful names.

This method allows users to quickly create a table without defining the structure in advance. However, it offers limited control over advanced settings such as field properties and data validation. Figure 4.8 illustrates this entire process of table creation in MS. Access.

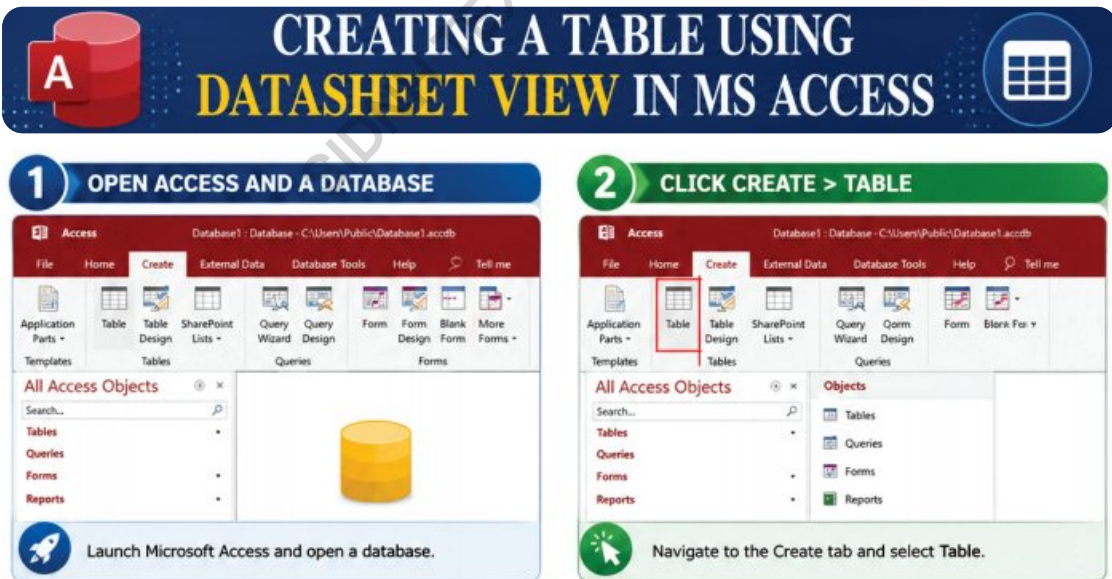
4.9.3 Create Table using Design View:

Design View provides a more structured and flexible approach for creating tables, allowing users to define all aspects of the table before entering data.

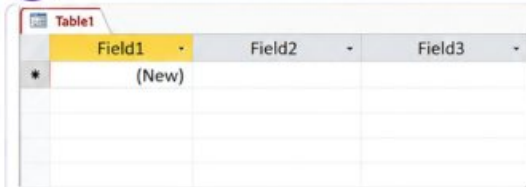
Procedure:

1. Go to the Create tab and select Table Design.
2. Enter the Field Name for each attribute.
3. Specify the Data Type corresponding to each field.
4. Optionally, provide a Description to document the purpose of the field.
5. Select a field and assign it as the Primary Key using the toolbar or right-click option.
6. Save the table with an appropriate name.

Design View is preferred in professional database development because it provides complete control over the structure and constraints of the table as shown in Fig 4.9.



3 DATASHEET VIEW OPENS

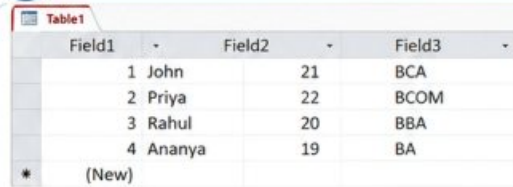


Field1	Field2	Field3
(New)		



A new table appears in Datasheet View, resembling a spreadsheet interface.

4 ENTER DATA



Field1	Field2	Field3
1 John	21	BCA
2 Priya	22	BCOM
3 Rahul	20	BBA
4 Ananya	19	BA



Enter data directly into the cells. Access automatically assigns default field names (e.g., Field1, Field2).

5 RENAME FIELDS



ID	Name	Age	Course
1	John	21	BCA
2	Priya	22	BCOM
3	Rahul	20	BBA
4	Ananya	19	BA



Rename fields by double-clicking on the column header and typing a new name.



KEY POINTS

- ☒ Fast and easy method for beginners.
- ☒ Data is entered directly in a spreadsheet-like grid.
- ☒ Field names are assigned automatically (Field1, Field2, ...).
- ☒ Limited control over field properties and validation rules.



Datasheet View provides a quick and simple way to create a table and start entering data in Microsoft Access.



Figure 4.8: Illustration of creating table in Datasheet view

CREATING A TABLE USING DESIGN VIEW IN MS ACCESS: STEP-BY-STEP



Design View allows you to define the structure of the table before entering data, giving you full control over field names, data types, properties, and the primary keys.

1 OPEN MICROSOFT ACCESS

Launch Access and open or create a database.

2 GO TO CREATE TAB → TABLE DESIGN

Click the Create tab, then select Table Design.

A new table opens in Design View.

3 DEFINE FIELD NAMES AND DATA TYPES

Enter each field name and choose an appropriate data type.

Field Name	Data Type	Description (Optional)
StudentID	AutoNumber	Unique student ID
Name	Short Text	Student full name
Department	Short Text	Student department
Age	Number	Student age in years
Email	Short Text	Student email address

Choose the right data type for each field.

4 SET THE PRIMARY KEY

Select the field that uniquely identifies each record, right-click, and choose Primary Key.

Field Name	Data Type	Description (Optional)
StudentID	AutoNumber	Unique student ID
Name	Short Text	Student full name
Department	Short Text	Student department
Age	Number	Student age in years
Email	Short Text	

A key icon appears next to the primary key field.

5 SET FIELD PROPERTIES (OPTIONAL BUT RECOMMENDED)

Select a field, then set its properties in the Field Properties pane to control how data is stored and validated.

Field Properties	Value
Field Size	255
Format	
Input Mask	
Caption	Student Name
Default Value	
Validation Rule	Is Not Null
Validation Text	Name is required.
Required	Yes
Allow Zero Length	No
Indexed	No
Unicode Compression	Yes
IME Mode	No Control
IME Sentence Mode	None

Common Properties:

- Field Size
- Format
- Default Value
- Validation Rule
- Required
- Input Mask

6 SAVE THE TABLE

Click the Save button (Ctrl + S), enter a meaningful table name, and click OK.

Your table is now saved and ready to use.

★ KEY BENEFITS OF DESIGN VIEW

- Define structure before entering data
- Set data types and rules in advance
- Ensures data integrity and consistency
- Better control over field properties
- Easier to manage and maintain tables

📋 BEST PRACTICE

Always plan your table structure, choose the correct data types, set a primary key, and define field properties to maintain clean and reliable data.

Figure 4.9: Illustration of creating table in Design View

4.9.4 Selecting appropriate data types:

Choosing the correct data type for each field is a critical design decision. Data types define the kind of values a field can store and influence both storage efficiency and data validation.

Common data types in Microsoft Access include:

- **Short Text:** Used for alphanumeric data such as names, addresses, and email IDs.
- **Number:** Suitable for numeric values used in calculations.
- **Date/Time:** Stores date and time information.
- **Currency:** Designed for financial values with precision.
- **AutoNumber:** Automatically generates unique values, often used for primary keys.

Selecting inappropriate data types can lead to data inconsistencies, increased storage requirements, and limitations in performing operations.

4.10 Designing Forms for Data Manipulation:

Forms in Microsoft Access serve as an essential interface for data manipulation, enabling users to efficiently enter, modify, and view records stored in database tables. Unlike direct interaction with tables, forms provide a structured, user-friendly environment that minimizes input errors, enforces validation rules, and enhances overall usability. Designing appropriate forms is therefore a critical step in developing a reliable and efficient database system.

4.10.1 Understanding the purpose of forms:

Before designing a form, it is important to clearly define its purpose. Forms are primarily used for:

- **Data Entry:** Allowing users to input new records
- **Data Modification:** Updating existing records
- **Data Viewing:** Displaying information in an organized layout

A well-designed form ensures that users interact with the database efficiently and accurately, without needing to understand the underlying table structure.

4.10.2 Identifying the Data Source:

Every form must be linked to a data source, which can be either a table or a query.

- A table-based form is suitable for simple data entry tasks
- A query-based form is useful when data from multiple tables needs to be displayed

Proper selection of the data source ensures that the form retrieves and updates data correctly.

4.10.3 Designing forms in Form Wizard:

The Form Wizard in Microsoft Access provides a structured and guided approach to creating forms. It is especially useful for beginners and for quickly generating forms for data entry, viewing, and modification without requiring manual design.

In this section, a sample form (Student Registration Form) will be created step by step using the Form Wizard.

Before creating a form, ensure that a table with following structure already exists.

Example Table: Student

- StudentID
- StudentName
- FatherName
- CNIC
- Department
- ContactNumber

This table will act as the data source for the form. Now follow the following list of steps to create a form

- Open your database in Microsoft Access
- Click on the **Create** tab in the Ribbon
- Select **Form Wizard**, this will launch a step-by-step wizard to guide form creation as shown in the Fig. 4.10.

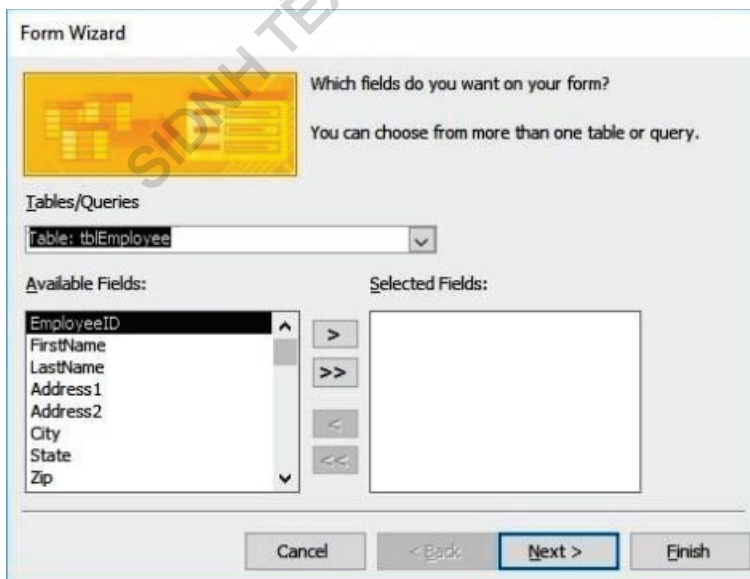


Figure 4.10: Form Wizard

- Choose the Student table from the dropdown as the main Data Source.

- Select required fields:
 - StudentID
 - StudentName
 - FatherName
 - CNIC
 - Department
 - ContactNumber
- Or, Click the >> (Add All) button to include all fields
- Click Next
- From the List of Available Layouts, choose a form Layout. MS Access supports the following form layouts.
 - Columnar → One record at a time (best for data entry)
 - Tabular → Multiple records in rows
 - Datasheet → Spread-sheet like format
 - Justified → Compact arrangement
- For Student registration form layout, we select columnar layout.
- Enter a meaningful name, e.g., StudentRegistrationForm
- Choose:
 - Open the form to view or enter information
- Click Finish

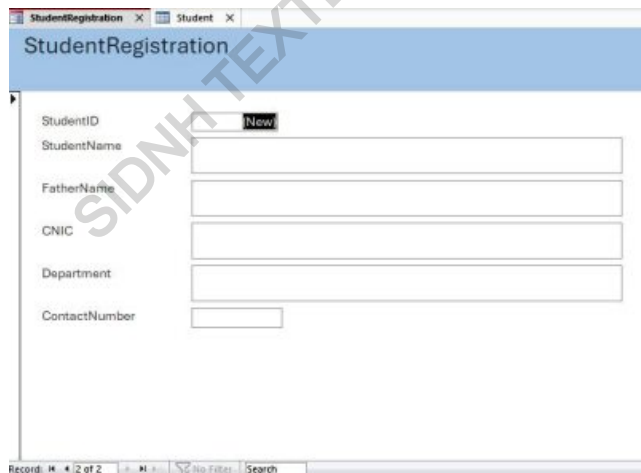


Figure 4.11: Student Registration Form

The Form Wizard in Microsoft Access provides a systematic and efficient method for creating forms. By following these steps, users can quickly generate a functional form for managing student data as shown in Fig. 4.11. Although the wizard automates the process, further customization using Design View can significantly enhance the usability, appearance, and functionality of the form.

4.11 Creating Queries in MS Access:

Queries are one of the most powerful features of Microsoft Access. They allow users to retrieve, filter, and analyze data stored in tables based on specific conditions or criteria. In a student database system, queries are particularly useful for extracting meaningful information such as lists of students from a specific department, students enrolled in a particular year, or records matching certain conditions.

In this section, we will focus on basic queries using the previously created Student table.

4.11.1 Introduction to Queries:

A query is a request to retrieve specific data from one or more tables based on defined criteria. Instead of displaying all records, queries allow users to focus only on relevant data.

Purpose of Queries:

- Filter records based on conditions
- Sort data in ascending or descending order
- Display selected fields only
- Perform simple calculations (in advanced cases)

4.11.2 Creating a simple SELECT Query:

For demonstration purpose of query creation, we will consider the Student Table created previously. The query creation will follow the following steps.

1. Open the database in Microsoft Access
2. Go to the **Create** tab
3. Click on **Query Design**
4. Add the **Student** table
5. Select required fields (e.g., StudentName, Department)
6. Click **Run (!)** to execute the query

The process of creating a simple SELECT query is elaborated in Fig. 4.12.

4.11.3 Filtering Data:

When a query is created in Microsoft Access by selecting all or some of the fields, by default, the query retrieves all the records from the table that contain those fields. This means that every row stored in the table will appear in the query results, which can be overwhelming if the table contains a large amount of data. Often, we are interested in only a specific subset of the data, such as students from a particular department, students who scored above a certain mark, or records created within a specific time period. To achieve this, we use data filtering techniques. Filtering allows us to define criteria or conditions so that the query displays only the records that match

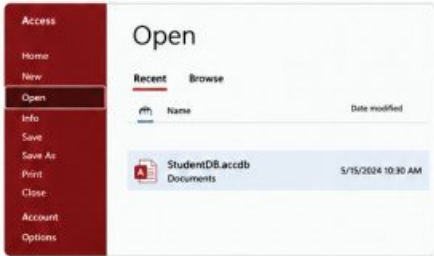
Unit 4: Data and Analysis

those conditions. By applying filters, we can focus on relevant information, make data analysis easier, and create more meaningful reports without modifying the underlying table.

CREATING A SIMPLE SELECT QUERY IN MS ACCESS: STEP-BY-STEP

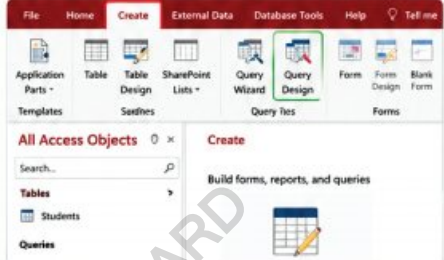
- 1 Open the database in Microsoft Access**

Launch Access and open your database (e.g., StudentDB.accdb).



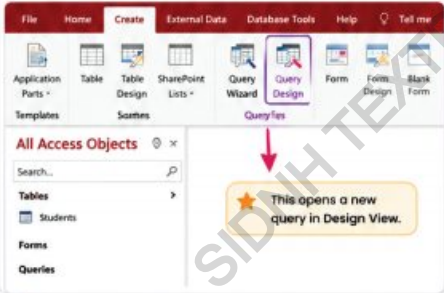
Open an existing database that contains the tables you want to query.
- 2 Go to the Create tab**

On the Ribbon, click the Create tab.



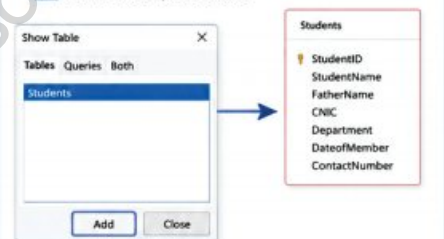
We will use Query Design to create a new query.
- 3 Click on Query Design**

In the Queries group, click Query Design.



This opens a new query in Design View.
- 4 Add the Student table**

In the Show Table dialog box, select Students and click Add, then Close.



The Students table is now added to the query design.
- 5 Select required fields**

Double-click the fields you want (e.g., StudentName, Department). They appear in the design grid.

Students

 - StudentID
 - StudentName
 - FatherName
 - CNIC
 - Department
 - DateofMember
 - ContactNumber

Only selected fields will appear in the results.

Field:	StudentName	Department	
Table:	Students	Students	
Sort:			
Show:	☑	☑	
Criteria:			

The design grid contains only the selected fields.
- 6 Click Run (!) to execute the query**

Click the Run button to view the results.



The query runs and shows only the selected fields from the Students table.

StudentName	Department
All Raza	Computer Science
Sara Khan	Software Engineering
Ahmed Ali	Computer Science
Fatima Noor	Information Technology
Usman Tariq	Software Engineering

Figure 4.12: Illustration of Query Design

i) Filtering by a Single Criteria:

Example 1:

In the query design grid, under the Department column, enter the criterion: “Computer Science” and Click Run (!). Only students from the Computer Science department are displayed.

Example 2:

In the query design grid, under the Contact Number column, enter the criterion: “0223865856” and Click Run (!). Only students with the given contact number will be displayed.

ii) Filtering by multiple Criteria:

Queries can often be written with multiple conditions. The effects of these conditions can be combined using AND or OR operators, which are known as logical operators. Their working principles are as follows:

- AND: All conditions must be true.
- OR: Any one of the conditions can be true.

For Example, to display students from either IT or Computer Science department, we need to Mention “IT” and “Computer Science” in the criteria under department column in the query design wizard as shown below:

Field:	Department					
Table:	Student					
Sort:						
Show:	☒	☐	☐	☐	☐	☐
Criteria:	"IT"					
or:	"Computer Science"					

Figure 4.13: Multiple Criteria using OR

Similarly, if we want to display all the students having StudentName “Aslam” and are from “IT” Department, we need to mention the criteria on the same row as shown in the figure 4.14.

Field:	Department	Student				
Table:	Student	Student				
Sort:						
Show:	☒	☒	☐	☐	☐	☐
Criteria:	"IT"	'Aslam'				
or:						

Figure 4.14: Multiple Criteria using AND

Whereas, for applying OR Operation, we need to specify criteria on different rows as shown in figure 4.15.

Field:	Departn	Student				
Table:	Student	Student				
Sort:						
Show:	☒	☒	☐	☐	☐	☐
Criteria:	"IT"					
or:		"Aslam"				

Figure 4.15: Multiple Criteria using OR

4.11.4 Data Summarization and Grouping in MS Access:

Data summarization is a fundamental operation in relational databases that enables users to condense large datasets into meaningful aggregates. In Microsoft Access, this is typically achieved through aggregate functions and grouping queries.

Grouping Queries allows records with similar values in a selected field to be combined into a single category, enabling the calculation of summary values for each group.

For Example:

Group students in the Department to determine:

- Total number of students
- Highest marks in each department
- Total marks per department

The following aggregate functions are widely used in MS Access for grouping:

- COUNT() → Counts number of records
- SUM() → Calculates total values

- MIN() → Finds the smallest value
- MAX() → Finds the largest value
- AVG() → Computes average values

The following list of steps allows us to create a grouping query in MS Access.

1. Open the database in Microsoft Access
2. Navigate to Create → Query Design
3. Add the relevant table (e.g., Student)
4. Select fields such as Department and StudentID
5. Click the Totals (Σ) button to enable grouping
6. In the Total row:
 - Set Department → Group By
 - Set StudentID → Count
7. Run the query

This Query Lists the number of students registered in each department.

4.11.5 Statistical Analysis using Average and Related Measures:

In Microsoft Access, statistical aggregation functions can be applied through Query Design View using the Totals (Σ) option. Common functions include:

- **Avg** for calculating mean values
- **Mode** can be determined by grouping and identifying the most frequently occurring value
- **Mod** operations can be used in calculated fields for remainder-based analysis.

One of the most important measures is the average (mean), which provides a representative value of a dataset. The AVG() function is used to compute the arithmetic mean of a numeric field. For example, the query to obtain the average marks of each student, grouped by department, can be created as follows:

1. Open the database in Microsoft Access
2. Navigate to Create → Query Design
3. Add the relevant table (e.g., Student)
4. Select fields such as Department and Marks
5. Click the Totals (Σ) button to enable grouping
6. In the Total row:
 - Set Department → Group By
 - Set Marks → Avg
7. Run the query

Department	Ave Of Marks
IT	55
Science	80
Systems	80

The output shows average performance per department and Helps identify:

- High-performing groups
- Underperforming groups
- Trends across categories

4.11.6 Data Visualization in MS Access:

While summary statistics provide numerical insights, data visualization transforms these insights into graphical representations, making patterns easier to understand. Microsoft Access offers integrated tools to generate charts directly from queries or tables.

The following type of charts can be generated using MS Access.

1. Bar Chart:

A Bar Chart represents data using horizontal bars, where the length of each bar corresponds to the value it represents. It is particularly effective for comparing discrete categories.

Bar charts are useful when categories have long names or when there are many categories to compare. They emphasize differences in magnitude and are easy to interpret visually.

For Example, the average score obtained by students in each subject as shown in Fig. 4.16.

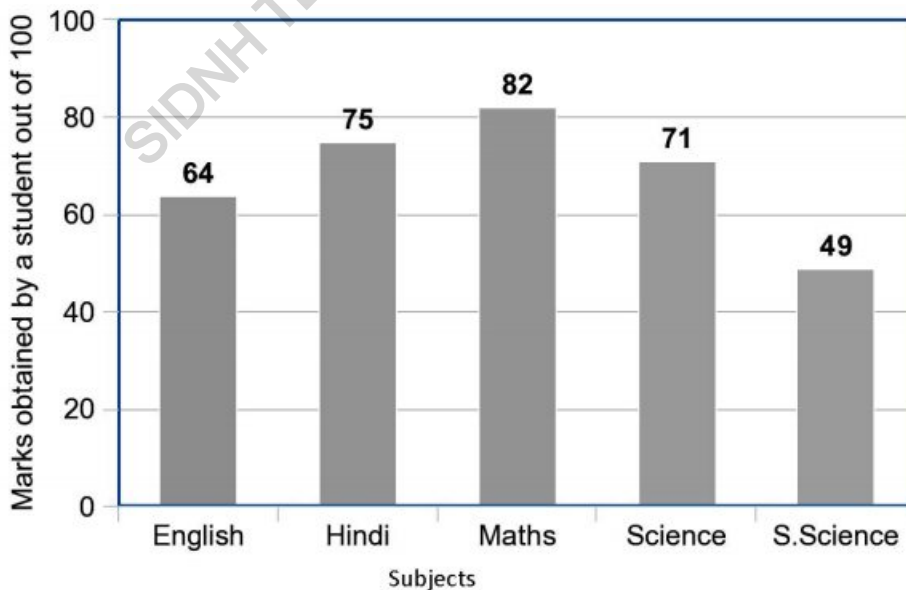


Figure 4.16: Example of Bar Chart

2. Column Chart:

A Column Chart is similar to a bar chart but uses vertical bars. It is widely used to show comparisons across categories and is often preferred for time-based or ordered data. Column charts highlight variation across categories and are visually intuitive when the number of categories is limited. They are commonly used in academic reports and dashboards as shown in Fig. 4.17.

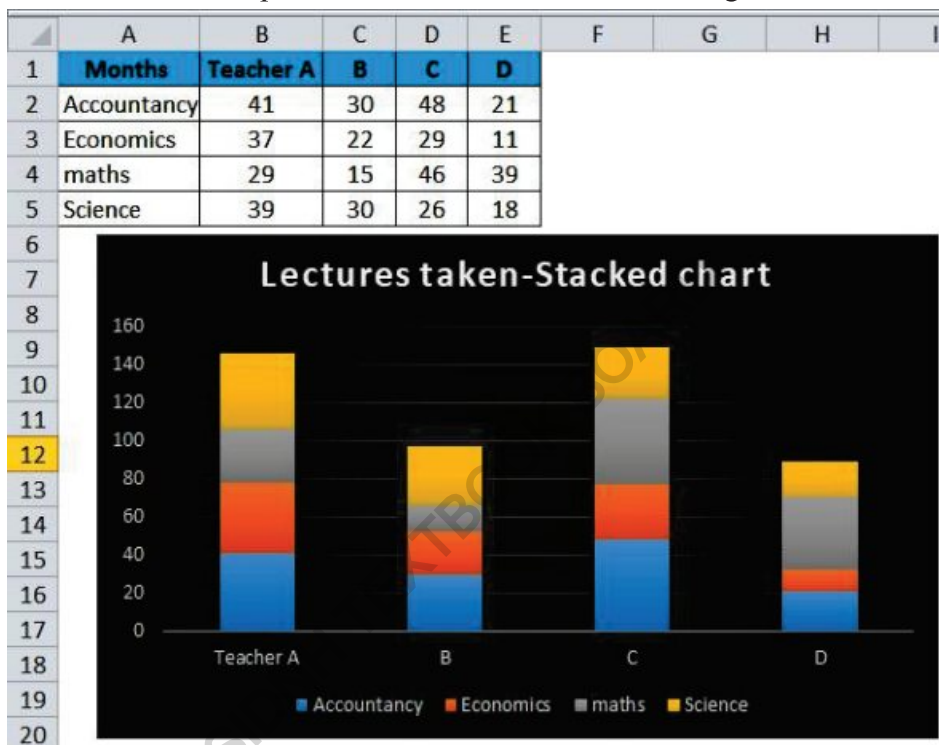


Figure 4.17: Example of Column Chart

3. Pie Chart:

A Pie Chart divides a circle into slices, where each slice represents a proportion of the whole. Pie charts are most effective when illustrating percentage or proportional relationships. However, they are less suitable when there are too many categories or when values are very similar. For example, pie charts can be used to show the distribution of students across departments as shown in Fig. 4.18.

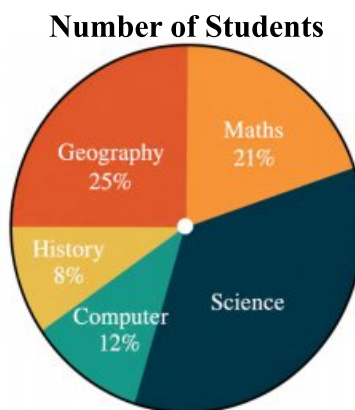


Figure 4.18: Example of Pie Chart

4. Line Chart:

A Line Chart connects data points with lines to show trends over time or ordered sequences. Line charts are particularly useful for analyzing temporal changes, such as performance trends, growth patterns, or progress over multiple observations. For example, the average number of website users per month as shown below:

Website Users by Months



Figure 4.19: Example of Line Chart

The following sequence of steps can be followed to create charts in MS Access.

1. Prepare a query with summarized data
2. Go to Create → Form Design or Report Design
3. Insert a Chart control
4. Select the query as the data source
5. Choose chart type (bar, pie, line, etc.)
6. Assign fields to axes and values
7. Format and customize the chart



Career Opportunities

This unit covers how data is organized, related, and managed using relational database concepts such as entities, relationships, referential integrity, tables, queries, and reports. These skills are essential for careers that involve storing, securing, and making sense of organizational data.

Database Administrator (DBA)

Designs, secures, and maintains databases to ensure accurate, reliable, and available data for organizations.

Data Analyst

Writes queries and builds reports to turn raw data into meaningful business insights.

Data Engineer

Builds and maintains the systems and pipelines that collect, store, and organize large volumes of data.

Business Intelligence (BI) Analyst

Designs dashboards and reports from relational databases to support management decision-making.

Backend / Full-Stack Developer

Designs relational schemas and connects applications to databases to store and retrieve information.

Data Scientist

Uses structured datasets from relational databases to build models and generate predictive insights.



Summary

- ◆ Data plays a crucial role in decision-making in the modern world, influencing business, science, education, and governance.
- ◆ Data consists of raw facts and figures (such as text, numbers and observations)
- ◆ When data is processed, organized and interpreted it becomes information
- ◆ As data grows in volume and complexity in modern environments (schools, businesses, governments), there arises a need to store it in an organized manner.
- ◆ A database fulfills this need by providing a structured collection of related data that can be easily accessed, updated, and managed.
- ◆ To efficiently interact with databases, specialized software known as a DBMS (e.g., Microsoft Access) is used.

- ◆ A database is composed of tables, which store data in rows and columns.
- ◆ Each row is a record representing a single entity instance, and each column is a field representing an attribute.
- ◆ The primary key is used to uniquely identify each record, ensuring accuracy and preventing duplication.
- ◆ The Entity-Relationship (ER) Model provides a way to represent real-world systems using entities (objects), attributes (properties), and relationships (associations).
- ◆ ER diagrams visually illustrate how data elements are connected.
- ◆ Relationships can be:
 - One-to-One (1:1)
 - One-to-Many (1:M)
 - Many-to-Many (M:N) (resolved using a junction table)
- ◆ To maintain correctness across related tables, the concept of referential integrity is introduced.
- ◆ Referential integrity ensures that relationships remain valid by enforcing rules on foreign keys.
- ◆ Mechanisms such as cascade update and cascade delete automatically maintain consistency across tables.
- ◆ A relational schema acts as a blueprint of the database.
- ◆ Forms provide a user-friendly interface for entering, updating, and viewing data without directly interacting with tables.
- ◆ Queries are used to retrieve, filter, and analyze specific data from one or more tables based on defined criteria.
- ◆ Reports are used to present data in a formatted and organized manner, suitable for printing and sharing.



Exercise

- 1. Encircle the correct answer:**
- i) Which of the following is a DBMS?
 - (a) MS Word
 - (b) MS Excel
 - (c) MS Access
 - (d) MS Paint
 - ii) In an ER Model, a rectangle represents:
 - (a) Attribute
 - (b) Relationship
 - (c) Entity
 - (d) Key
 - iii) A one-to-many relationship means
 - (a) One record relates to many
 - (b) Many to one only
 - (c) One to One
 - (d) None

- iv) Which object is used to retrieve data?
 - (a) Table
 - (b) Form
 - (c) Query
 - (d) Report
- v) Which function calculates average?
 - (a) Sum()
 - (b) Count()
 - (c) Avg()
 - (d) Max()
- vi) Which key uniquely identifies each record in a table?
 - (a) Foreign Key
 - (b) Primary Key
 - (c) Alternate Key
 - (d) Candidate Key
- vii) Which relationship requires a junction table?
 - (a) One-to-One
 - (b) One-to-Many
 - (c) Many-to-Many
 - (d) None
- viii) Which database objects provides a user friendly interface for data entry?
 - (a) Query
 - (b) Table
 - (c) Report
 - (d) Form
- ix) Which chart is best used to show trends over time?
 - (a) Pie Chart
 - (b) Bar Chart
 - (c) Line Chart
 - (d) Column Chart
- x) Which query operation combines records into categories for summarization?
 - (a) Sorting
 - (b) Filtering
 - (c) Group by
 - (d) Validation

2. Fill in the Blanks:

- i) In a relational database, a _____ key is used to establish a link between two tables by referring to the primary key of another table.
- ii) If two entities are connected through an intermediate entity instead of directly, this usually indicates the resolution of a _____ relationship.
- iii) When a record in one table is deleted, and related records in another table must also be removed automatically, this is controlled through _____ rules.
- iv) A table created to resolve a many-to-many relationship is commonly referred to as a _____ table.
- v) A candidate key that is not chosen as the primary key is known as _____.

3. Provide short descriptive answers of the following questions.

- i) Differentiate between a field and a record.
- ii) What is referential integrity?
- iii) Explain the role of a Database Management System (DBMS).
- iv) Explain a one-to-many relationship with an example.
- v) Define data visualization and give one example.

4. Provide detailed answers to the following questions.

- i) Define data and information with examples. Also explain the concept of a database and its advantages.
- ii) What is a Database Management System (DBMS)? Explain its functions and advantages.
- iii) Explain the Entity-Relationship (ER) Model. Define entities, attributes, and relationships with suitable examples.



Activities

1. Create a Student table in MS Access with at least five fields and assign an appropriate primary key.
2. Design an Entity-Relationship (ER) diagram for a school database showing the entities Student, Course, and Teacher along with their relationships.
3. Create a data entry form for the Student table using the Form Wizard, and use it to enter student records.
4. Create queries to retrieve student records by displaying selected fields, applying filters, and using both AND and OR conditions.
5. Generate a grouped query to summarize the number of students in each department, and present the results using a suitable chart (bar or pie chart).

Unit

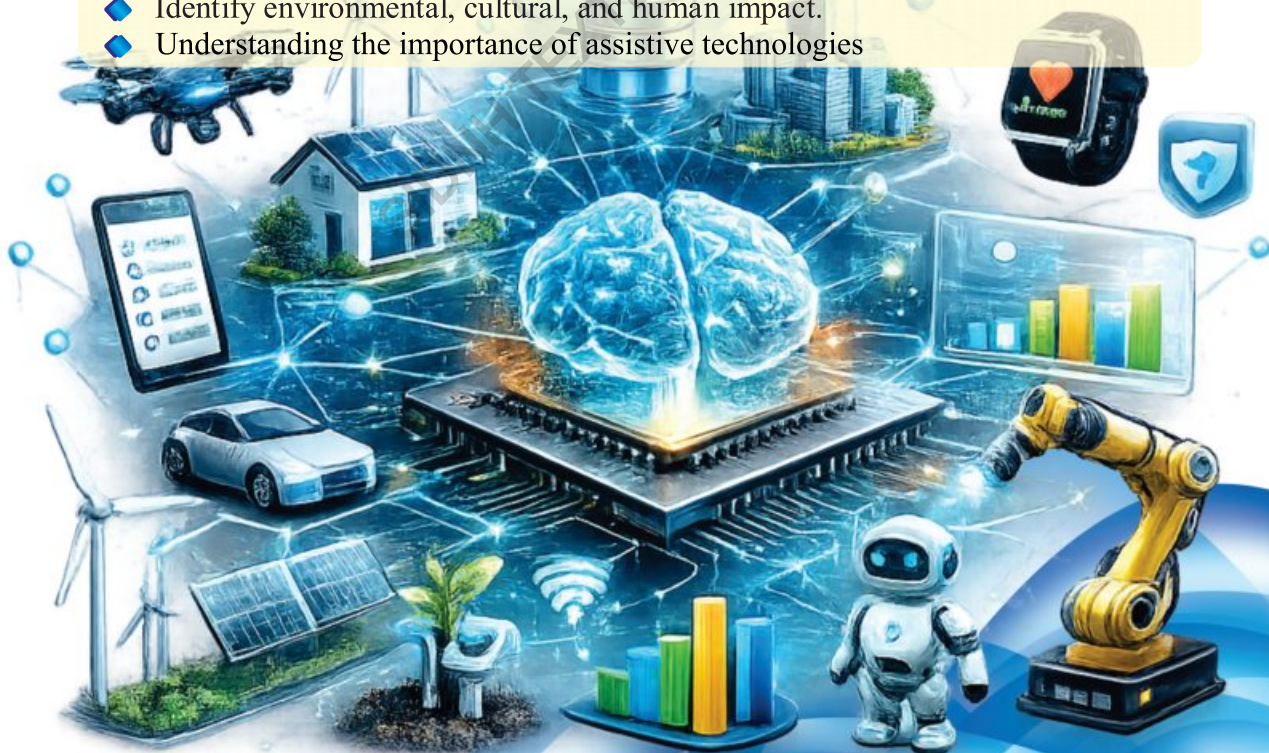
5

Application and Impacts of Computing

Student Learning Outcomes:

After the completion of this unit students will be able to:

- ◆ Describe the components of the Internet of Things and Data Analytics.
- ◆ Identify uses of Internet of Things in smart cities, industries, healthcare, education, automation, smart grid, Smart farming, wearable, etc.
- ◆ Identify reliable and unreliable information sources.
- ◆ Identify environmental, cultural, and human impact.
- ◆ Understanding the importance of assistive technologies



Introduction to Unit:

Computing is now an important part of our daily lives. We use computers, mobile phones, the internet, smart devices, online learning systems, digital banking, and many other technologies.

In Pakistan, it is also becoming important in schools, colleges, offices, hospitals, banks, farms, and industries. Students use mobile phones and computers for learning. Teachers use online resources and Learning Management Systems. Hospitals use digital records. Banks use mobile apps and ATM machines. Farmers can use weather information and smart irrigation systems. These examples show that computing is not limited to computer labs only; it is used in almost every field of life.

5.1.1 Artificial Intelligence:

AI is called “**artificial**” because it is created by humans, and “**intelligence**” means it can perform tasks that require thinking. AI systems use data to make decisions. For example, when a mobile phone recognizes a face to unlock the screen, it uses AI. When YouTube suggests videos according to a user’s interests, it also uses AI. When a chatbot answers simple questions on a website, it is using AI.

AI in Smart Systems: Artificial Intelligence often works together with other technologies like the Internet of Things (IoT) and Data Analytics. IoT devices gather data from the real world. Data Analytics sorts and studies this information. Then, AI uses the results to make decisions.

For example, in a smart home, a temperature sensor collects data about the room temperature. The system analyzes the data. If the room is hot, AI can decide to turn on the fan or air conditioner. In this way, AI acts like the “brain” of a smart system.

5.1.2 Internet of Things:

The Internet of Things is a system in which devices are connected to the Internet and can collect and share data to perform tasks automatically. IoT devices include sensors, smart watches, smart cameras, smart meters, smart appliances, machines, vehicles, and other connected devices. **For example**, a smartwatch collects data about steps, heart rate, and sleep. It sends this data to a mobile phone. The mobile phone shows the information to the user.

IoT relies on several foundational components that have been strengthened by recent technological advancements.

- i. **Sensors:** Sensors are a vital component of IoT systems. Sensors are devices that collect information from the physical world. They can detect temperature, light, sound, movement, pressure, humidity, or other conditions. The following sensor technologies are implemented in devices:

- Temperature sensor
- Motion sensor
- Fingerprint sensor
- Camera sensor
- Smoke sensor
- Soil moisture sensor

For example, a fingerprint sensor in mobile phones is a biometric security device that identifies and verifies a user based on unique fingerprint patterns. When a user places a finger on the sensor, it captures a digital image, converts it into a template, and compares it with stored data to verify identity.



Figure 5.1: Sensor and Devices

- ii. Actuators:** Actuators are devices that perform actions in response to system commands. They convert digital signals into physical actions, such as movement or switching.

For example, in an automatic door system, a motion sensor detects movement and sends a signal to the control unit. The control unit processes this signal and sends a command to the actuator (motor), which opens the door smoothly.

The following actuator technologies are implemented in devices.

- Motor
- Electric switch
- Automatic door opener
- Water valve
- Fan controller
- Alarm system

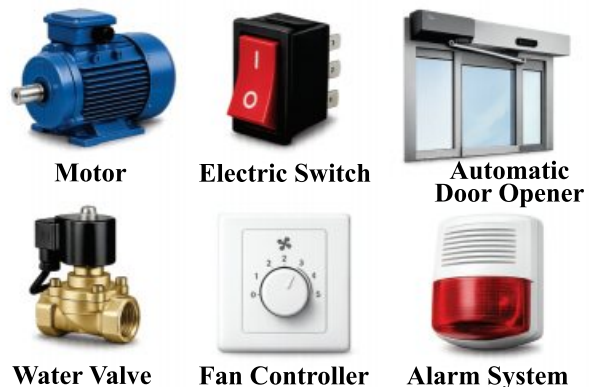


Figure 5.2: Actuator and Devices

- iii. Connectivity:** Connectivity means the connection between devices. It allows IoT devices to send and receive data. Without connectivity, sensors cannot send data and devices cannot receive commands.

Common Connectivity Technologies:

- Wi-Fi
- Bluetooth

- ZigBee
- LoRaWAN
- NB-IoT
- 4G/5G

If the connection is weak or lost, data cannot be transmitted, and devices cannot receive commands.

For example, a smartwatch connects to a mobile phone via Bluetooth and sends health data, such as heart rate and steps, for analysis and display.

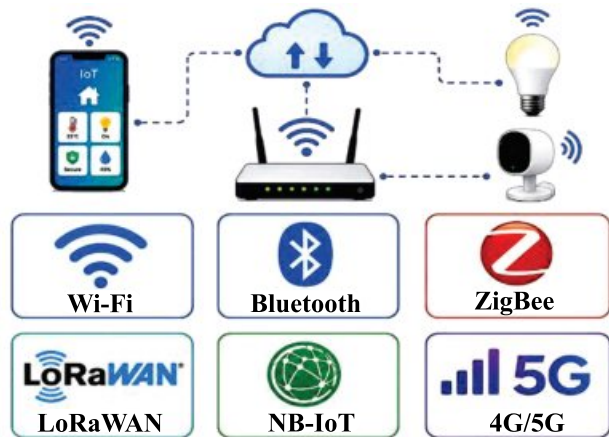


Figure 5.3: Connectivity Technologies

- iv. Data Processing:** Data processing means changing raw data into useful information. In IoT systems, data collected by sensors is processed to make decisions.

Steps in Data Processing:

- Collecting data
- Organizing data
- Analyzing data
- Making a decision

For example, a temperature sensor sends data to a smart home system. The system checks whether the room is hot or cold. If the room is hot, it turns on the fan or air conditioner.

- v. Cloud Computing:** It allows users to store data and use services over the internet instead of relying solely on their own computers or mobile phones. Cloud computing supports IoT because IoT devices often collect large amounts of data. This data can be stored and processed on cloud servers. Cloud computing provides different types of services:

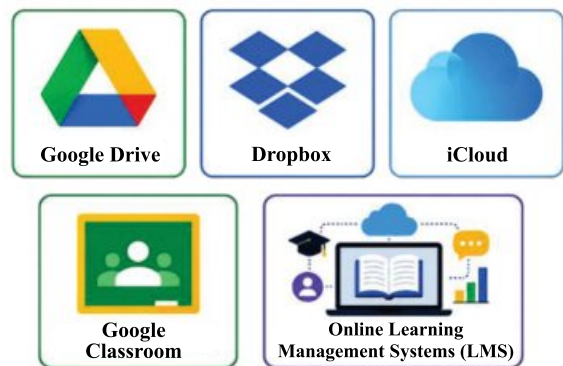


Figure 5.4: Cloud Services

- Google Drive
- Dropbox

- iCloud
- Google Classroom
- Online Learning Management Systems (LMS)

For example, students and teachers in Pakistan can use cloud services to share notes, assignments, lectures, and educational material. This is especially helpful when students cannot attend physical classes regularly.

vi. User Interface (UI): A UI is the component of a system that enables interaction between the user and a device or applications. Its main functions are:

- Displaying system data
- Allowing user commands
- Monitoring system activity

It helps users see information and give commands. The following devices or applications are examples of UI technologies.

- Mobile app
- Touch screen
- Buttons
- Dashboard
- Voice command system



Figure 5.5: User Interfaces' examples

5.1.3 Data Analytics:

Data Analytics is the process of collecting, organizing, cleaning, studying, and presenting data to find useful information. Data Analytics helps people make better decisions. It is used in education, business, healthcare, government offices, banks, agriculture, and many other fields.

For example, a college prepares examination results. Teachers collect marks, correct mistakes, calculate percentages, and check which students need help. It comprises five key components; each component plays a vital role in making an accurate decision.

i. Data Collection: This is the first step in the data analytics process. It collects raw information from multiple sources, such as websites, apps, surveys, or machines.

Examples of Data Sources:

- School attendance records
- Examination marks
- Surveys
- Mobile apps

- Websites
- Sensors
- Hospital records
- Bank transactions



Figure 5.6: Data Sources examples

ii. Data Storage: After collection, data must be stored safely and properly. Data may be stored in files, databases, spreadsheets, or in the cloud.

For example, a college stores students' admission records, attendance, and examination results in a computer system or online storage facilities.

iii. Data Cleaning: Data cleaning involves identifying and correcting errors in data. Raw data may contain errors, duplicate records, missing values, or wrong formats. The following issues may arise during data collection and must be identified and addressed.

- Duplication of files or records.
- Incorrect values, where data entries do not accurately reflect the intended information.
- Missing files or records.
- The data is organized in an improper format.

For example, during result preparation, a teacher may find that one student's marks are missing or a roll number is typed incorrectly. These mistakes must be corrected before final results are prepared.

iv. Data Analysis: After cleaning, the data is analyzed to identify patterns, trends, and relevant information for problem-solving and decision-making. The main objective is to obtain useful and meaningful insights from the data.

For example, teachers analyze exam results to determine:

- Subjects in which students performed poorly.
- Students who require additional support.

- v. Data Visualization:** It involves creating visual representations of data, such as plots, charts, and graphs, to analyze patterns and trends and provide valuable insights. By comparing and analyzing datasets, data analysts extract meaningful information from raw data.

For example, schools will show their performance data by using the following methods:

- Bar charts to display marks by subject.
- Pie charts to illustrate pass and fail percentages.

Differentiate Internet of Things (IoT), Artificial Intelligence (AI), and Data Analytics

Aspect	Internet of Things (IoT)	Artificial Intelligence (AI)	Data Analytics
Definition	A network of physical devices connected to the internet that collect and share data	Technology that enables machines to think, learn, and make decisions like humans	Process of examining data to find patterns and useful information
Main Purpose	To collect real-world data using sensors and devices	To make intelligent decisions and predictions	To analyze data and extract meaningful insights
Key Function	Data collection	Decision making and automation	Data analysis and interpretation
Components	Sensors, actuators, connectivity (Wi-Fi, Bluetooth)	Algorithms, models, machine learning	Data collection, cleaning, processing, visualization
Type of Work	Works with physical devices	Works with smart software systems	Works with data (numbers, records)
Example	Smart home system controlling lights and temperature	Voice assistants like Siri or Google Assistant	Analyzing student exam results to find performance trends
Output	Raw data from environment	Smart decisions or actions	Reports, graphs, and insights
Dependency	Generates data for AI and analytics	Uses data from IoT and analytics	Uses data collected from IoT or other sources

5.2 Uses of IoT In Different Areas:

The Internet of Things (IoT) has become an essential part of modern life. It is widely used in homes, industries, healthcare, education, and agriculture. IoT systems improve efficiency, save time, reduce human effort, and enhance the overall quality of life. It plays a significant role in the development of smart cities by improving urban infrastructure and public services through automation and real-time monitoring.

i. **Smart Cities:** Various applications are being implemented in different cities.

- **Smart Traffic Management:** Sensors and cameras monitor traffic flow and adjust traffic signals automatically to reduce congestion, E-Challan.
- **Smart Street Lighting:** Street lights turn ON/OFF automatically based on movement or daylight, saving energy.
- **Smart Parking Systems:** Sensors detect empty parking spaces and guide drivers accordingly.
- **Waste Management:** Smart bins send alerts when they are full, ensuring timely collection.

For example: In large cities such as Karachi, Hyderabad, Lahore, and Islamabad, smart traffic monitoring can help reduce traffic problems and improve road safety.



1. Smart Traffic Management



2. Smart Street Lighting



3. Smart Parking Systems



4. Smart Waste Management

Figure 5.7: Smart Cities

ii. **Industrial Automation (Smart Industry):** IoT is widely used in industries to automate processes and improve productivity.

- Sensors can check machine temperature.
- Vibration sensors can detect machine problems.
- Automated machines can perform repeated tasks.
- Quality control sensors can check products during manufacturing.

For example: In textile industries of Sindh and Punjab, IoT systems can help monitor machines and reduce production delays.



Figure 5.8: Industrial Automation

iii. **Healthcare (Smart Health Systems):** IoT improves healthcare services by enabling continuous monitoring and timely medical response.

- Smartwatches can measure heart rate.
- Remote monitoring devices can send patient data to doctors.
- Emergency alert devices can inform family members or hospitals.
- Digital medical records can help doctors check patient history.

For example: In Pakistan, remote patient monitoring can help people living in rural areas where hospitals are far away.



Figure 5.9: Healthcare Applications

iv. **Education (Smart Learning Environment):** IoT enhances the learning experience by making classrooms interactive and efficient. The following applications are used in the education sector.

- **Smart Classrooms:** Digital boards, projectors, and connected devices improve teaching methods.
- **Attendance Systems:** Biometric systems automatically record students' and teachers' attendance.

- **Resource Management:** IoT helps manage lighting, temperature, and equipment in classrooms.
- **Remote Learning:** Students can access lessons through connected devices.

For example: In government colleges and schools of Sindh, biometric attendance and digital classrooms can help improve management and learning.



Smart Classroom



Attendance Systems



Resource Management



Remote Learning

Figure 5.10: Education Applications

v. **Home Automation (Smart Homes):** IoT is widely used in homes to automate daily tasks and improve comfort. Commonly useful applications are:

- **Smart Lighting:** Lights can be controlled remotely or automatically.
- **Smart Security Systems:** Cameras and sensors provide real-time monitoring and alerts.
- **Smart Appliances:** Devices like refrigerators, ACs, and washing machines can be controlled via mobile apps.
- **Voice Control:**

Systems such as voice assistants enable users to control devices with voice commands.

For example: A person can use a mobile app to turn off lights or check a security camera while away from home.



Smart Lighting



Smart Security Systems



Smart Appliances



Voice-Controlled Devices

Figure 5.11: Home Automation Applications

vi. Smart Farming: IoT is revolutionizing agriculture by enhancing crop productivity and improving resource efficiency. Smart farming is increasingly adopting IoT-based applications to improve agricultural efficiency and productivity. Agriculture is very important in Pakistan, especially in Sindh. IoT can help farmers save water, improve crop production, and monitor weather conditions. It helps in different farming methods such as:

- Soil moisture sensors can check water needs.
- Weather sensors can provide weather information.
- Automated irrigation can control water supply.
- Drones and cameras can monitor crops.



Soil Moisture Sensors



Weather Sensors



Automated Irrigation Systems



Drones and Cameras



Livestock Monitoring Systems

Figure 5.12: Smart Farming Applications

For example: In rural areas of Sindh, farmers grow crops such as wheat, rice, cotton, sugarcane, and vegetables. Smart irrigation systems can help farmers use water carefully, especially in areas where water shortage is a serious problem.

5.2.1 Scope of Artificial Intelligence (AI) in Education in Pakistan:

Artificial Intelligence is gradually becoming an important part of Pakistan's education system. With the increasing use of digital technologies and online learning platforms, AI is helping improve teaching, learning, and administrative processes.

i. Practical Uses of AI in Pakistan:

- Implementation of **Learning Management Systems (LMS)**
- Growth of **online and virtual classrooms**, especially after COVID-19
- Use of **automated grading systems** and digital assessment
- Development of **career counselling systems** based on student performance
- Government initiatives promoting **digital and AI-based education**

ii. Benefits of AI in Education:

- Help students learn at their own speed
- Support teachers in checking work
- Identify students who need extra help
- Improve online learning
- Support students with disabilities
- Help institutions make better decisions using data

iii. Major Challenges:

- Limited internet access in many rural areas
- Lack of computers and digital devices
- Need for teacher training
- High cost of AI-based systems
- Data privacy and security issues
- Risk of overdependence on technology

5.2.2 Career and Skill Connection in IoT and AI:

IoT and AI are creating new career opportunities in Pakistan. Students who learn computing skills can work in software houses, industries, telecom companies, banks, hospitals, schools, agriculture, and government departments. Government and private organizations such as the **Sindh IT Department, SITC**, and telecom companies like **Jazz and Telenor Pakistan** are actively working on IoT-based solutions, digital infrastructure, and smart systems.

Future Career:

- **IoT Engineer** who designs and develops smart systems and connected devices.
- **Embedded Systems Engineer**, works with sensors, microcontrollers, and hardware.
- **Network Engineer**, manages communication between devices.
- **Data Analyst**: analyzes data generated by IoT systems.
- **System Administrator**, maintains IoT infrastructure and services.



IoT Engineer



Embedded Systems Engineer



Network Engineer



Data Analyst



AI Developer



System Administrator

Figure 5.13: Career Opportunities

Skill required:

- **Programming:** Basic coding in Python, Java, C++, etc.
- **Electronics:** Understanding sensors, circuits and smart devices.
- **Networking:** Wi-Fi, Bluetooth, and internet communication protocols.
- **Data handling:** Data collection, cleaning, and analysis.
- **Problem-Solving:** Identifying problems and finding solution to real-life problems.
- **Communication:** Explaining ideas clearly and building a strong understanding for users.
- **Ethics:** Using technology safely and responsibly.



Students can prepare for future careers by learning basic programming languages, improving their English communication skills, practising problem-solving techniques, and understanding how technology is used in real life.

5.3 Information Sources:

Information is useful data that helps people understand facts, make decisions, and solve problems using computers. Today, the internet provides a lot of information, but not all of it is accurate or reliable.

Information sources are ways to find knowledge and data. Information is important for learning, research, and making decisions. These sources are grouped into primary, secondary, and tertiary sources, and each plays a different role in academic work.

- i. **Primary Sources:** Primary sources give original, firsthand information that comes straight from people, places, events, experiments, or official records. **For example,** interview, speeches, diaries, letters, photographs, NADRA records, official records, etc.

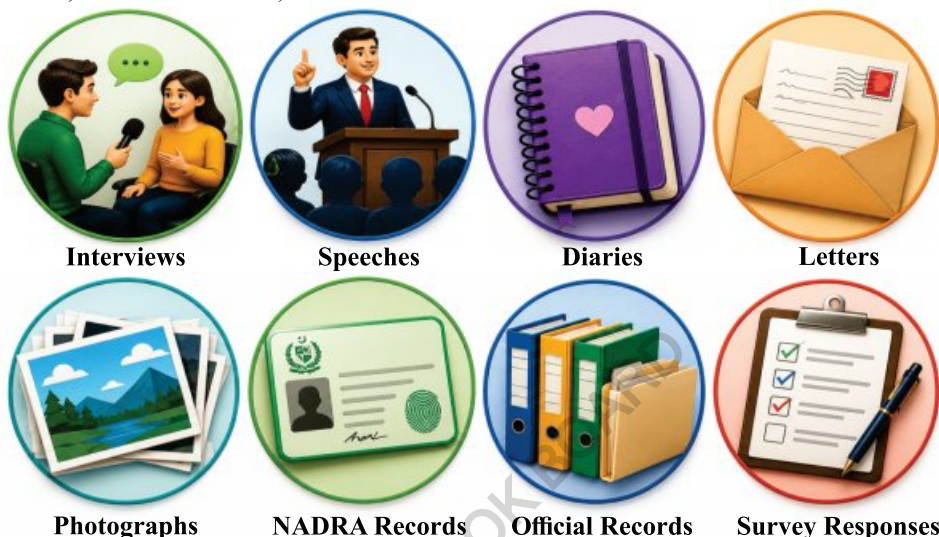


Figure 5.14: Primary Source of Information

The main features of primary sources are as follows:

- Provide original and direct information.
- Accurate and authentic data
- Support research and investigations

For example, NADRA is a primary source of information because it is collected directly from individuals and officially verified. For example, when a citizen applies for a CNIC, personal details like name, date of birth, and fingerprints are recorded and authenticated. Therefore, NADRA records constitute original, reliable first-hand information.

When Primary Sources Become Unreliable:

Primary sources may become unreliable if:

- Information is incomplete.
- Data is recorded incorrectly.
- False information is provided.

- ii. **Secondary Sources:** A **secondary source** is information that has already been collected and explained by someone else. It does not give original or first-hand data. Instead, it helps us understand, summarize, or analyze information from primary sources more simply.

Secondary Sources:

- Textbooks and reference books
- Research papers and review articles
- News reports and documentaries
- Biographies and historical analysis

These sources are helpful for learning because they organize information and make it easier to follow.



Figure 5.15: Secondary Source of Information

For example, reports published by the Pakistan Bureau of Statistics use data collected from censuses and surveys to present summaries, graphs, and analysis. These reports are considered secondary sources because they are based on already collected primary information.

When Secondary Sources Become Unreliable:

Secondary sources may become unreliable if:

- **Outdated information**
- **Lack of proper references**
- **No expert verification**

- iii. **Tertiary Sources:** Tertiary sources provide summarized and organized information derived from both primary and secondary sources, such as encyclopedias, dictionaries, handbooks, Glossaries, Indexes, etc. These sources are mainly used for quick reference and basic understanding. A few common features:

- Provide summarized and organized information.
- Based on both primary and secondary sources
- Easy to access and understand
- Useful for quick revision and general knowledge



Figure 5.16: Tertiary Source of Information

For example, Encyclopedias and online databases like Wikipedia articles about Pakistan, which summarize information from books, research papers, and reports, are considered tertiary sources.

When Tertiary Sources Become Unreliable:

Tertiary sources may become unreliable if:

- Unknown or unverified websites
- Oversimplified or incomplete information

5.3.1 Career and Skill Development through Information Sources:

In Pakistan, different types of information sources help students and professionals build careers and develop important skills.

Future Career:

- **News hosts and journalists** collect news, verify facts, and present accurate and reliable information to the public.
- **Researchers** use trustworthy sources of information to conduct studies and share their findings.
- **Teachers** use accurate information to prepare lessons and help students understand different topics.

- **Content writers** create articles and learning materials based on well-checked and reliable information.
- **Data analysts** study data from reliable sources to identify trends and support better decision-making.



Figure 5.17: Career Opportunities from Information Source

Skill required:

- Reading and understanding information
- Checking reliable sources
- Writing clearly
- Organizing data
- Avoiding fake news
- Giving references
- Presenting information honestly

5.4 Impacts of Computing In Various Fields of Life:

Computing has both positive and negative impacts on society. It has made life easier, faster, and more connected. At the same time, it has created some problems, such as overuse of technology, privacy risks, and job changes. Students should understand both sides of computing so that they can use technology responsibly.

Positive and Negative Impacts of Computing:

Field	Positive Impact	Negative Impact
Education	Online classes, e-books, videos, and learning apps make studying easier and more flexible.	Excessive screen time can reduce attention and affect health.
Communication	Emails, messaging apps, and video calls allow fast and easy communication.	People may interact less in person, reducing face-to-face communication.
Business	Online shopping, digital banking, and automation improve speed and efficiency.	Some traditional jobs may be replaced by machines and technology.
Healthcare	Digital patient records, telemedicine, and smart devices improve diagnosis and treatment.	Patient data can be misused if not properly secured.
Agriculture	Weather forecasting and smart irrigation systems help increase crop production.	Limited internet access in rural areas can reduce the use of modern tools.
Daily Life	Smart devices and online services save time and make life more convenient.	People may become too dependent on technology.

Smart and Responsible Use of Computing:

Responsible use means using technology safely, ethically, and wisely.

Students should:

- Protect their personal information
- Use strong passwords
- Avoid sharing false information
- Be respectful to others when you are online.
- Use technology to learn new things and for positive activities
- Make sure to balance your screen time with physical activities.

An Analytical Approach to the Impacts of Computing:

- Computing improves efficiency and accessibility, especially in education and healthcare.
- Computing creates new jobs but can also replace some older jobs.
- It helps people around the world communicate, but sometimes we spend less time talking face-to-face.
- Technology makes life more convenient, but we can become too dependent on it.

5.5 Assistive Technologies:

Assistive technologies are devices, software, tools, and systems that help people with disabilities or functional limitations. These technologies support people in education, communication, movement, learning, and daily life. It helps create an inclusive society. They allow more people to participate in school, work, and community life. Assistive technologies can be categorized into different groups based on their functions:

- i. **Mobility aids:** Mobility aids help people move from one place to another. The following aids are included: wheelchairs, walkers, crutches, and stair lifts.

For example, a student who has difficulty walking may use a wheelchair to move around the school or college.

- ii. **Sensory aids:** This technology enhances or supports sensory functions, particularly vision and hearing, thereby enabling improved interaction with the environment. The following aids are included: hearing aids, screen readers, Braille displays, Electronic white canes and magnifying devices for individuals with visual impairments.

For example, screen readers (e.g., NVDA, VoiceOver) convert text on a computer into speech for students with visual difficulties.

- iii. **Communication aids:** This technology assists individuals with speech or expressive difficulties, enabling more effective communication. The following aids are included: speech-generating devices, text-to-speech applications, communication boards, and mobile applications designed to support speech.

For example, a child who cannot speak clearly may use a picture communication board to express needs and ideas.

- iv. **Cognitive and learning aids:** These aids help people with memory, reading, writing, learning, and understanding. The most common aids include educational applications, spell-check tools, text-to-speech software, reminder applications, audiobooks, grammar checking tools, concept mapping tools, and digital planners.

For example, A student with reading difficulty may use an audiobook to understand a lesson.



Mobility Aids



Sensory Aids



Communication Aids



Cognitive Learning Aids

Figure 5.18: Assistive Technologies

5.5.1 Importance of Assistive technologies:

These technologies have a profound impact across domains such as education, communication, healthcare, and personal independence.

i. **Equal Access to Education:**

- They help students with disabilities learn better in the classroom.
- Tools like screen readers and speech-to-text programs make learning materials easier to access.
- Allows students to participate in online and physical learning equally.

ii. **Improved Communication:**

- They support people who have trouble speaking, hearing, or seeing.
- Communication devices and apps make it easier for users to share their thoughts clearly.
- These tools help students, teachers, and others interact more easily.

iii. **Independence in Daily Life:**

- They help people do tasks on their own, without always needing help.
- Mobility aids help people with physical disabilities move around safely.
- These technologies build confidence and help people become more self-reliant in their daily lives.

iv. **Better Healthcare Support:**

- Hospitals use assistive technologies for patient monitoring and rehabilitation.
- They help doctors keep track of patient conditions more accurately.
- Specialized assistive devices can also help people recover more effectively.

v. **Equal Opportunities in Employment:**

- They allow people with disabilities to work efficiently in many different jobs.
- Workplace technologies help people do their jobs independently.
- These tools promote fairness and help make job opportunities more inclusive.

vi. **Social Inclusion:**

- They help reduce barriers between people with and without disabilities.
- These technologies encourage people to take part in social, educational, and work activities.
- They help build a more inclusive and supportive society.

5.5.2 Career Connection in Assistive Technologies:

Assistive technologies are creating new and meaningful career opportunities in Pakistan. Students interested in computing, healthcare, design, and engineering can build successful careers in this growing field.

i. Careers in Assistive Technologies:

- **Assistive Software Developer:** Creates software like screen readers and speech-to-text tools to help people with disabilities use computers easily.
- **AI and Machine Learning Specialist:** Develops smart systems that improve accessibility, such as voice assistants and automated recognition tools.
- **Biomedical Engineer:** Designs and develops assistive medical devices like artificial limbs, hearing aids, and other supportive equipment.
- **UX/UI Designer (Accessibility Expert):** Designs websites and apps that are simple, clear, and easy to use for all users, including people with disabilities.
- **Rehabilitation Technologist:** Helps patients learn how to use assistive devices properly to improve their daily life and independence.

Career Connection in Assistive Technologies in Pakistan



Assistive
Software Developer



AI and Machine
Learning Specialist



Biomedical Engineer



UX/UI Designer



Rehabilitation
Technologist



Special Education
Technology Support Officer

Figure 5.19: Career Connection in Assistive Technologies

ii. Skills Required:

Skill Area	Description
Programming	Developing useful software and apps
Design	Making systems easy for all users
Healthcare Knowledge	Understanding users' physical and learning needs
Problem-Solving	Creating solutions for real-life challenges
Communication	Working with teachers, doctors, students, and families
Empathy	Understanding and respecting the needs of all users



Career Opportunities

This unit explores how computing technologies such as Artificial Intelligence (AI), the Internet of Things (IoT), and data analytics are transforming daily life, from smart homes to hospitals and farms. These emerging fields offer some of the fastest-growing career opportunities in technology today.

Artificial Intelligence Engineer

Designs intelligent systems and algorithms that let machines make decisions, such as facial recognition or recommendation engines.

IoT (Internet of Things) Engineer

Develops connected devices and sensor networks used in smart homes, wearables, and smart cities.

Data Analytics Specialist

Analyzes data gathered from smart devices and sensors to support automated decision-making.

Robotics / Smart Systems Engineer

Combines AI, IoT, and hardware knowledge to design automated and robotic systems.

Machine Learning Engineer

Builds and trains models that allow smart systems to improve their performance using real-world data.

Technology Consultant

Advises businesses, schools, and hospitals on adopting AI and IoT solutions to improve efficiency.



Summary

- ◆ Computing is applied across various sectors, including education, healthcare, banking, agriculture, industry, communication, and daily life.
- ◆ Artificial Intelligence enables machines to learn from data, recognize patterns, and make decisions. It is used in facial recognition, chatbots, video recommendations, and smart systems.
- ◆ AI works alongside IoT and Data Analytics. IoT collects data, Data Analytics examines it, and AI uses the insights to make informed decisions.
- ◆ IoT devices connect to the internet, enabling them to collect, share, and use data automatically.

- ◆ The main components of IoT are sensors, actuators, connectivity, data processing, cloud computing, and user interface.
- ◆ IoT is implemented in smart cities, industry, healthcare, education, smart homes, and agriculture.
- ◆ In Pakistan, IoT can enhance traffic management, healthcare services, classroom administration, industrial production, and water-efficient agriculture.
- ◆ Data Analytics involves collecting, storing, cleaning, analyzing, and presenting data to extract valuable insights.
- ◆ Data Analytics supports schools, banks, hospitals, businesses, and government offices in making informed decisions.
- ◆ Information sources are divided into primary, secondary, and tertiary sources.
- ◆ Students should check information carefully before using it because not all online information is reliable.
- ◆ Computing has positive impacts, such as faster communication, easier learning, better healthcare, online banking, and improved farming.
- ◆ Computing also has negative impacts, including privacy risks, excessive screen time, job displacement, and overreliance on technology.
- ◆ Responsible use of computing means using technology safely, ethically, and wisely.
- ◆ Assistive technologies support individuals with disabilities in mobility, learning, communication, hearing, vision, and daily activities.
- ◆ These technologies promote equal access to education, healthcare, employment, and social participation.
- ◆ This unit helps students understand the connection between computing, daily life, society, career opportunities, and Pakistan's future development.

Exercise

- 1. Encircle the correct answer**
- i) A smartwatch records heart rate and sends it to a mobile phone. Which technology is mainly used here?
 - (a) Word Processing
 - (b) Internet of Things
 - (c) Printing Technology
 - (d) Analog Technology
 - ii) In a smart irrigation system, which device checks whether the soil is dry or wet?
 - (a) Actuator
 - (b) Sensor
 - (c) Monitor
 - (d) Smart hearing aid

- iii) A school removes duplicate student records while preparing examination results. Which step of Data Analytics is this?
 - (a) Data collection
 - (b) Data visualization
 - (c) Data cleaning
 - (d) Data storage
- iv) Which of the following is a primary source of information?
 - (a) Textbook
 - (b) Interview
 - (c) Dictionary
 - (d) Summary website
- v) A student uses an unknown website without an author name or references. This source is likely to be:
 - (a) Reliable
 - (b) Primary
 - (c) Unreliable
 - (d) Official
- vi) Which technology supports people with hearing, vision, learning, or movement difficulties?
 - (a) Gaming technology
 - (b) Manufacturing technology
 - (c) Assistive technology
 - (d) Antivirus technology
- vii) Which of the following is an actuator in a smart water system?
 - (a) Soil moisture sensor
 - (b) Water valve
 - (c) Temperature sensor
 - (d) Camera sensor
- viii) Which type of source usually provides summarized information for quick reference?
 - (a) Primary source
 - (b) Secondary source
 - (c) Tertiary source
 - (d) Unverified source
- ix) Which method is used to present data through charts and graphs?
 - (a) Data collection
 - (b) Data storage
 - (c) Data cleaning
 - (d) Data visualization
- x) Which assistive technology converts on-screen text into speech?
 - (a) Screen reader
 - (b) Smart meter
 - (c) Motion sensor
 - (d) Water pump

2. Fill in the Blanks

- i) A _____ detects changes such as temperature, movement, light, or soil moisture.
- ii) An _____ performs a physical action, such as opening a valve or switching on a fan.
- iii) Correcting duplicate, missing, or incorrect records is called data _____.
- iv) An interview is an example of a _____ source of information.
- v) Screen readers and hearing aids are examples of _____ technologies.

3. Provide short answers to the following questions.

- i) How does Artificial Intelligence help a mobile phone recognize a user's face?
- ii) What is the difference between a sensor and an actuator?
- iii) How does cloud computing support students and teachers in Pakistan?
- iv) What problems can occur if people become overdependent on technology?
- v) Why are communication and problem-solving skills important for future computing careers?

4. Provide detailed answers to the following questions.

- i) Explain how IoT components work together in a smart irrigation system for a farmer in Sindh. Include sensors, actuators, connectivity, data processing, and user interface in your answer.
- ii) A college wants to improve its examination result system. Explain how Data Analytics can help from data collection to data visualization.
- iii) A student finds information from a social media post and an official government website. Explain how the student should decide which source is more reliable.
- iv) Analyze the positive and negative impacts of computing.
- v) Discuss the career opportunities related to AI, IoT, Data Analytics, and Assistive Technologies in Pakistan. What skills should students develop for these careers?



Activities

- 1. Think of one smart device used in your home, school, or community. Does it use AI, IoT, or Data Analytics? Explain your answer in two or three sentences.
- 2. Which smart city application is most needed in your city or town? Write one reason for your answer.
- 3. Think about a farming problem in Pakistan. How can IoT help solve it?
- 4. Choose one career from IoT or AI. Write two reasons why this career interests you and list three skills required for it.
- 5. Do you enjoy reading, writing, checking facts, or organizing information? Choose one career from this section and explain why it may be suitable for you.
- 6. Write one positive and one negative impact of computing that you have observed in your daily life.
- 7. Identify one assistive technology used in schools or hospitals. Write how it helps people and which career is related to it.

Digital Literacy

Student Learning Outcomes:

After the completion of this unit students will be able to:

- ◆ Perform advanced searches to locate information and/or design a data collection approach to gather original data (e.g., qualitative interviews, surveys, prototypes, simulations)
- ◆ Define data collection strategies (e.g. qualitative interviews, surveys, prototypes, simulations)
- ◆ Apply best practices to present primary & secondary data for a research question
- ◆ Design data-collection approach to gather original data
- ◆ Present data using appropriate digital tools (such as graphs or infographics in worksheets, presentations, reports etc.)



Introduction to Unit:

Digital literacy refers to the ability to understand and use digital devices, the internet, and online tools in a safe, effective, and responsible manner. In our increasingly digital world, possessing digital literacy is essential for students, educators, professionals, and everyone else. It enables individuals to find information, communicate, produce digital content, and solve problems through technology.

A digitally literate student can locate information, organize it properly, turn it into something useful, and share it responsibly with others.



Figure 6.1 Digital literacy



Digital literacy is not only using devices; it is using digital information safely and wisely.

6.1 Digital Literacy in the Information Age:

The Information Age is a period where vast amounts of data are produced, shared, and accessed instantly through the internet, mobile devices, and digital platforms. In such an environment, simply having access to technology is not enough. Students and users must develop the skills to understand, manage, and use information wisely.

Digital literacy goes beyond basic computer skills. It involves a combination of technical, cognitive, and social abilities that enable individuals to function successfully in a digital society.

6.2 Types of Data:

The data gathered through a survey, investigation, or classroom project, can be described with qualitatively and quantitatively, while some data gives numbers that can be counted, measured, or compared.

6.2.1 Qualitative Data:

Qualitative data describes what people think, feel, notice, or experience. It is useful when the aim is to understand reasons, attitudes, and behaviour rather than numbers or measures.

Examples include students' opinions about online classes, descriptions of study habits, and observation notes written during a computer lab activity.

6.2.2 Quantitative Data:

Quantitative data deals with numbers. It is useful when students need to count, measure, compare, calculate averages, or display results in charts and graphs. Examples include the number of students using the internet daily, the marks obtained in a Computer Science test, or the time spent online per day in hours.

6.2.3 Comparing Qualitative and Quantitative Data:

Table 6.1 Comparison of qualitative and quantitative data

Type of data	Nature	Examples	Best used for
Qualitative	Descriptive, non-numerical	opinions, reasons, behaviour	understanding why and how something happens
Quantitative	Numerical, countable or measurable	hours, scores, percentages, totals	comparing results and making graphs

6.3 Data-Collection Strategies:

Digital literacy is not limited to finding information that already exists. Students should also learn how to gather original information.

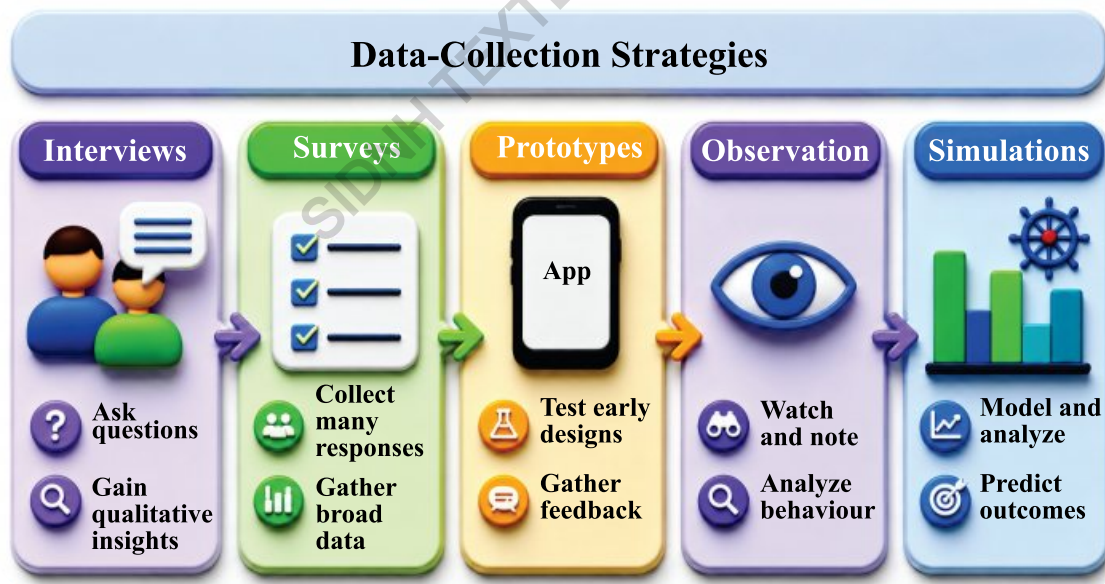


Figure 6.2 Common methods used to collect data

6.3.1 Interviews:

An interview is a planned conversation used to gather information directly from a person.

- It may be one-to-one or in a small group.
- Questions are often planned in advance.
- The answers are usually qualitative, although some may be factual.

6.3.2 Surveys / Questionnaires:

A survey is useful when the same set of questions needs to be asked from many people. It may be prepared on paper or created with digital tools such as Google Forms or Microsoft Forms.

The image displays two screenshots of the Microsoft Forms interface. The left screenshot shows a specific form titled 'Digital Literacy Survey' with the description 'This survey is about how students use digital tools for learning and daily life.' It contains three questions: 1. 'Do you use a computer or smartphone every day?' with radio button options 'Yes' and 'No'; 2. 'How often do you use the internet for study purposes?' with radio button options 'Daily', 'A few times a week', 'Rarely', and 'Never'; 3. 'Which devices do you use most frequently for online learning?' with checkbox options 'Smartphone', 'Laptop', and 'Desktop computer'. The right screenshot shows the Microsoft Forms dashboard with a search bar, 'New Form' and 'New Quiz' buttons, and a list of pinned forms including 'Digital Literacy Survey' (1 response), 'Class Feedback Form' (9 responses), 'Customer Satisfaction Survey' (14 responses), and another 'Digital Literacy Survey' (1 response). A 'Templates' section is also visible at the bottom.

Figure 6.3

Microsoft Forms interface showing how to create and manage a digital literacy survey form.

Unit 6: Digital literacy

The image displays the Google Forms interface. On the left, a 'Questions' view of a 'Digital Literacy Survey' is shown. It includes a title, a description, and three multiple-choice questions. The first question asks about daily computer or smartphone use, the second about internet frequency for study, and the third about devices used for online learning. A 'Submit' button is at the bottom. On the right, the 'Recent forms' and 'Template gallery' views are shown. The 'Recent forms' view lists three forms: 'Digital Literacy Survey', 'Class Feedback Form', and 'Event Registration'. The 'Template gallery' view shows three templates: 'Digital Literacy Survey', 'Class Feedback Form', and 'Customer Satisfaction'.

Figure 6.4 Google Forms interface showing the creation of a digital literacy survey form.

- It can collect quantitative data through ratings and multiple-choice questions.
- It can also collect qualitative data through open-ended responses.



Online survey tools such as Google Forms and Microsoft Forms can automatically collect responses and display basic charts, making data analysis easier for students.

6.3.3 Prototypes:

A prototype is a simple early version of a product, model, or solution. It allows an idea to be tested before time and effort are spent on the final version.

- Users try the prototype and give feedback.
- That feedback becomes useful data for improvement.

6.3.4 Observation:

In observation, the researcher watches what people do or what happens in a situation and records the results carefully.

- No direct questions are asked.
- The data may be qualitative such as notes, or quantitative such as counts.

6.3.5 Simulations:

A simulation is a computer-based model of a real situation.

- Variables can be changed to see what happens.
- Simulations are useful when real experiments are costly, risky or difficult.

6.4 Primary and Secondary Data:

Whenever data is used, it is important to know where it came from. The source of data affects how much control the researcher had over collecting it and how confidently the results can be used. Based on source, the data can be categorized as primary and secondary data.

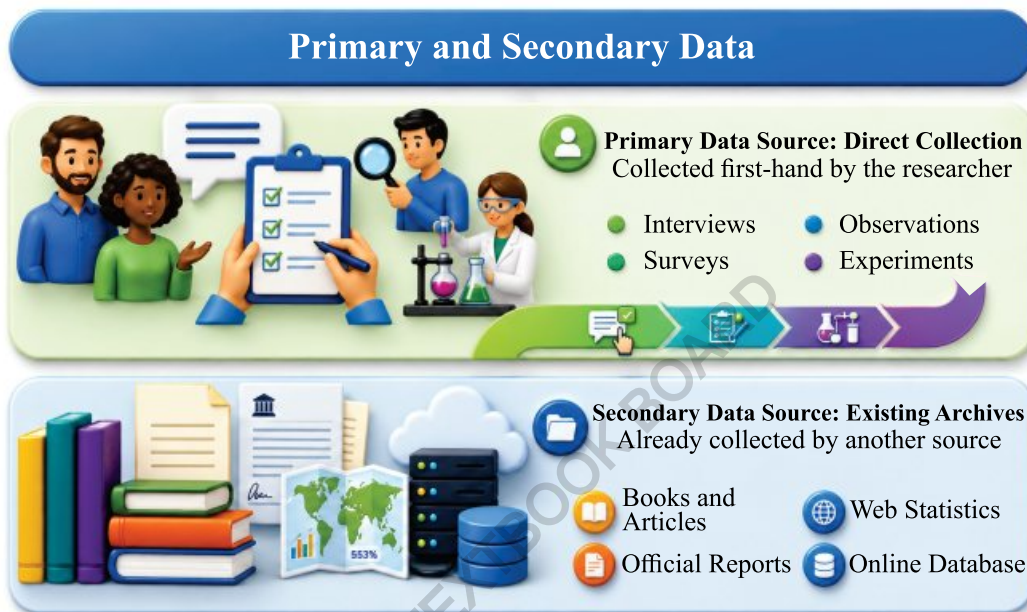


Figure 6.5 Primary and secondary sources of data

6.4.1 Primary Data:

Primary data is collected first-hand by the researcher for a particular purpose. It is gathered directly from people, events, or situations rather than taken from an already published source. The potential source of primary data can be

- interviews
- surveys
- observations
- experiments

6.4.2 Secondary Data:

Secondary data has already been collected by someone else and is being used again for a new purpose. The potential source of secondary data can be

- books and research articles
- official reports
- web statistics
- online databases

6.4.3 Difference Between Primary and Secondary Data:

Table 6.2 Difference between primary and secondary data

Aspect	Primary data	Secondary data
Source	collected directly by the researcher	already collected by others
Examples	survey responses, interviews, observations	reports, articles, websites, databases
Purpose	prepared for a specific study	reused for a new purpose
Control	high control over questions and collection process	limited control because the data already exists



Primary data gives fresh information collected for a specific purpose, while secondary data helps compare new findings with information already published by others.

6.5 Designing a Data-Collection Approach:

Before collecting data, it is essential to design a data collection approach. A data-collection approach is the plan used to gather accurate, relevant, and useful information for a study or project.

The data collection approach usually consists of the following steps.

- Step 1: Define the research question clearly.
- Step 2: Decide the type of data needed: quantitative, qualitative or both.
- Step 3: Select suitable methods such as a survey, interview or observation.
- Step 4: Choose the sample or target group.
- Step 5: Prepare the tools such as forms, questions and observation sheets.
- Step 6: Collect the data carefully and accurately.
- Step 7: Store the data digitally in a spreadsheet and keep a backup copy.
- Step 8: Follow ethical practice and respect privacy.

6.6 Presenting Data Using Digital Tools:

Collecting data is only one part of the work. Students also need to present their findings in a form that other people can understand quickly and easily. Digital tools make this much easier. The data can be presented using spreadsheets, presentations and Infographics.

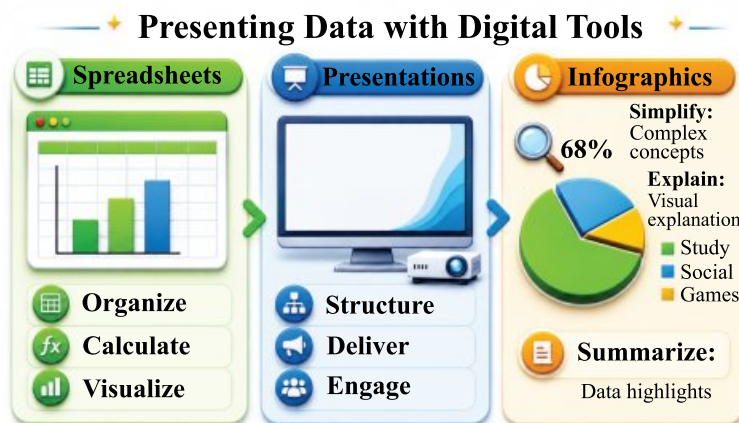


Figure 6.6 Digital tools used to present data clearly

6.6.1 Spreadsheets:

Spreadsheets such as Excel, Google Sheets, and Apple Numbers help students arrange information in rows and columns, carry out simple calculations, and prepare data for charts and analysis.

- enter data systematically.
- calculate totals, averages and percentages.
- sort and filter records.
- create charts and graphs automatically.

6.6.2 Presentations:

Presentation software such as PowerPoint, Google Slides, Canva, Adobe Express, Gamma, and Slides AI helps students explain their findings to an audience in a clear and organized way.

- begin with a clear title and research question.
- use short bullet points instead of long paragraphs.
- include tables, charts and key findings.
- keep the design simple and readable.

6.6.3 Infographics:

An infographic is a visual way of presenting information, facts, or data so that people can understand the main idea quickly and easily. Tools such as Canva, Adobe Express, Power BI, Tableau, and Looker Studio can be used to create infographics and interactive dashboards.

- choose a format that suits the research question and audience.
- highlight the most important findings, results and conclusions.
- use clear visuals, limited text and accurate information.



A good chart or infographic can explain complex data faster than a long paragraph. However, visuals must always be accurate and based on real data.

6.6.4 Reports:

Reports are structured documents used to present collected data, findings, analysis, and conclusions in a clear and organized way. Digital tools such as Microsoft Word, Google Docs, Canva Docs, Notion, and Adobe Express can help students create well-formatted reports. AI-based tools such as ChatGPT, Microsoft Copilot, Google Gemini, Power BI with Copilot, Tableau AI, and Canva Magic Write can also support report writing by helping students summarize data, improve wording, organize sections, and check grammar.

- start with a clear title, introduction, and purpose of the report.
- present the collected data using tables, charts, or graphs where needed.
- explain the main findings in simple and clear language.
- include conclusions and recommendations based on the data.
- use AI tools carefully to summarize, organize, and improve the report, but always review the information for accuracy.

6.7 Case Study: Digital Inquiry Project:

Empowering Students in the Digital Age



Figure 6.7 Digital inquiry workflow from research question to final artefact

6.7.1 Research Question:

How does daily mobile phone use affect the study habits of students?

6.7.2 Case Study Introduction:

Students today use mobile phones for learning, communication, entertainment, and social media. A mobile phone can help students search for information, watch educational videos, attend online classes, and communicate with teachers and classmates. However, excessive use of mobile phones may also distract students from their studies.

In this case study, students investigate how mobile phone use affects the study habits of students. They perform advanced searches, collect original data, organize the data in a spreadsheet, and create a digital artefact such as a report, presentation, infographic, or dashboard.

6.7.3 Learning Outcomes Covered:

- perform advanced searches to locate relevant information;
- collect primary data through surveys, interviews, or observations;
- use secondary data from reliable online sources;
- organize and analyze data using digital tools;
- communicate results and conclusions using charts, reports, presentations, or infographics;
- create a digital artefact that answers a research question.

6.7.4 Scenario:

A group of students notices that many classmates use mobile phones during free periods, at home, and sometimes while studying. Some students say mobile phones help them learn better, while others say mobile phones waste time and reduce concentration.

The Computer Science teacher asks the class to investigate this issue using digital literacy skills. The students must answer whether mobile phone use helps or harms the study habits of students.

6.7.5 Methodology:

Step 1: Perform Advanced Searches

Students should first search for background information using search engines and reliable websites. They should use advanced search operators to narrow down the results and locate more useful information.



Figure 6.8 Advanced search and source evaluation

Suggested Search Queries:

- "mobile phone use" "study habits" students
- "mobile phone use" "academic performance" filetype:pdf
- digital distraction students site:.edu
- student screen time and learning filetype:pdf

Source Evaluation Checklist:

Question	Yes/No
Is the author or organization clearly mentioned?	
Is the information recent?	
Is the website reliable, such as educational, government, or research-based?	
Does the source provide facts, data, or references?	
Is the information balanced and not biased?	

Step 2: Identify Data Types

For this project, students will collect both qualitative and quantitative data.

Type of Data	Examples in this Case Study
Quantitative data	Number of hours students use mobile phones daily; percentage of students who feel distracted; average study time per day.
Qualitative data	Students' opinions about mobile phone use; reasons for distraction; suggestions for responsible use.

Step 3: Select Data-Collection Methods

Students may use one or more of the following methods to collect original data.

- Survey / Questionnaire: useful when the same questions must be asked to many students.
- Interview: useful for understanding opinions and experiences in more detail.
- Observation: useful for recording visible behaviour in a classroom or computer lab without collecting private information.

Step 4: Prepare the Survey



Collecting Primary Data



A class survey can collect both numbers and opinions

QUESTIONNAIRE

✓

1. Daily phone use?



✓

2. Use phone for study?



✓

3. Feel distracted?



✓

4. One useful study app?







Class XI responses

Figure 6.9 Collecting primary data through a class survey

Sample Survey Questions:

No.	Question	Type of Data
1	How many hours do you use your mobile phone daily?	Quantitative
2	Do you use your mobile phone for study purposes?	Quantitative
3	Which activity do you use your phone for the most?	Quantitative
4	Does mobile phone use distract you from studying?	Quantitative
5	How many hours do you study daily without using your phone?	Quantitative
6	How does your mobile phone help you in your studies?	Qualitative
7	What problems do you face due to mobile phone use?	Qualitative
8	Suggest one way students can use mobile phones more responsibly.	Qualitative

Step 5: Choose the Sample

The sample is the group of people from whom data will be collected. For this case study, students may collect responses from 30 to 50 Class XI students, including students from different sections and students with different mobile phone habits.



Remember:

A good sample should represent the group being studied.

Step 6: Collect Primary Data

Students should collect original data through their own survey, interview, or observation. They should ask permission before collecting responses and must not record unnecessary personal information.

Ethical Guidelines:

- ask permission before collecting responses;
- avoid collecting sensitive personal information;
- do not record names unless necessary;
- keep responses private;
- use the data only for educational purposes.



Respecting privacy is an important part of digital literacy. Good researchers collect only the information they need.

Step 7: Use Secondary Data

Students should also include one or two secondary sources, such as research articles, educational websites, official reports, or reliable statistics from online databases. These sources help students compare their class findings with information already published by others.

Example: An online article explains that excessive screen time may affect concentration. Students can compare this idea with their own survey results.

Step 8: Organize Data in a Spreadsheet

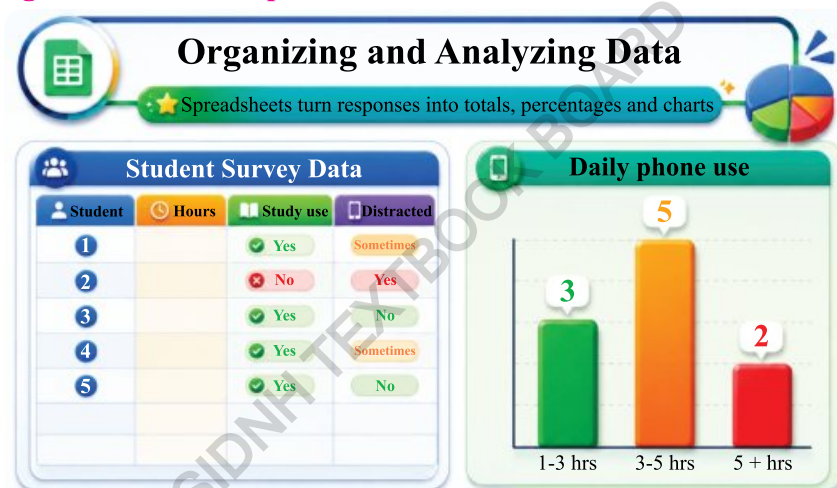


Figure 6.10 Spreadsheet analysis and chart creation

Students should enter survey results into Excel, Google Sheets, or any spreadsheet software. A spreadsheet helps students sort responses, calculate totals, find averages, and create charts.

Student No.	Daily Phone Use	Study Use	Main Activity	Feels Distracted	Daily Study Time
1	3 hours	Yes	YouTube lectures	Sometimes	2 hours
2	5 hours	No	Social media	Yes	1 hour
3	2 hours	Yes	Searching notes	No	3 hours

Students can then calculate the total number of responses, average mobile phone use, average study time, percentage of students who feel distracted, and the most common mobile phone activity.

Step 9: Create Charts and Graphs

Students should create at least two visuals to make their findings easier to understand.

- Bar chart: daily mobile phone usage categories such as less than 1 hour, 1–3 hours, 3–5 hours, and more than 5 hours.
- Pie chart: main purpose of mobile phone use, such as study, social media, gaming, entertainment, or communication.
- Column chart: number of students who feel distracted: yes, no, or sometimes.

Step 10: Analyze the Results

After collecting responses from 40 students, a group may find that most students use mobile phones for 3 to 5 hours daily. About half of the students may report that they use mobile phones for educational purposes, such as watching lectures, searching notes, and communicating with classmates. However, many students may also report that social media notifications disturb their concentration while studying.

The survey may show that students who use mobile phones mainly for study have better time management than students who use phones mostly for entertainment. The results suggest that mobile phones can be useful learning tools if they are used responsibly.

Step 11: Write the Conclusion

The research shows that mobile phones have both positive and negative effects on study habits. They help students access learning materials, search for information, and communicate with teachers and classmates. However, excessive use of social media, gaming, and entertainment apps can reduce study time and concentration. Therefore, mobile phones should not be avoided completely. Instead, students should learn to use them wisely by setting time limits, turning off unnecessary notifications, and using educational apps during study time.

Step 12: Recommendations

- use mobile phones mainly for educational purposes;
- turn off notifications while studying;
- set daily screen-time limits;
- avoid social media during study hours;
- use apps for notes, dictionaries, lectures, and reminders;
- take breaks from screens to protect health and concentration.

6.7.6 Digital Artefact:



Figure 6.11 Final artefact options for communicating research findings

A digital artefact is a final product created using digital technologies. Students must create one final artefact that answers the research question. They may choose a digital report, presentation, infographic, or dashboard.

Artefact Option	Suitable Digital Tools	Main Purpose
Digital Report	Microsoft Word, Google Docs, Canva Docs, Notion	Explain the full study in a structured document.
Presentation	PowerPoint, Google Slides, Canva, Gamma, Slides AI	Share key findings with an audience.
Infographic	Canva, Adobe Express, Power BI, Tableau, Looker Studio	Show the main results visually and quickly.
Dashboard	Power BI, Tableau, Looker Studio	Explore data through interactive charts and filters.

Artefact Design Structure:

1. Title
2. Research Question
3. Introduction
4. Method of Data Collection
5. Primary Data Findings
6. Secondary Data Summary
7. Charts and Graphs
8. Conclusion
9. Recommendations
10. References



A digital artefact is not only a decorated file. It must answer the research question, show evidence, communicate findings, and present a clear conclusion.



Career Opportunities

This unit develops skills in advanced searching, designing data-collection methods such as surveys and interviews, and presenting data effectively using digital tools like graphs and infographics. These research and communication skills are valuable across virtually every modern profession.

Market Research Analyst

Designs surveys and interviews to collect primary data and presents findings to guide business decisions.

Data Journalist

Uses advanced search and data-collection skills to research stories and present them with visuals and infographics.

UX Researcher

Gathers qualitative and quantitative data from users through interviews and surveys to improve product design.

Digital Marketing Specialist

Collects and analyzes audience data, presenting results through dashboards, graphs, and reports.

Research Assistant / Analyst

Applies structured data-collection approaches (surveys, prototypes, simulations) to support academic or scientific research.

Content Creator / Infographic Designer

Presents primary and secondary research findings using engaging digital tools and visual storytelling.



Summary

- ◆ Digital literacy involves searching carefully, checking source reliability, collecting evidence, and communicating findings responsibly.
- ◆ A strong project begins with a focused research question and a suitable plan for using primary and/or secondary data.
- ◆ Interviews, surveys, prototypes, simulations, and observation each produce different kinds of evidence and should be chosen thoughtfully.
- ◆ Sampling, bias, privacy, consent, and citation all affect the quality and ethics of student research.
- ◆ Data should be cleaned and organized before analysis so that tables, charts, and conclusions are accurate.
- ◆ Different visuals serve different purposes: tables show exact values, bar charts compare categories, line graphs show trends, and infographics summarize key findings quickly.
- ◆ A strong digital artefact connects the research question, method, findings, conclusion, and references in a clear structure.
- ◆ Results should be communicated differently for different audiences while remaining accurate, readable, and evidence-based.



Exercise

1. Encircle the correct answer

- i) Which of the following best describes digital literacy?
 - (a) The ability to type quickly using a keyboard
 - (b) The ability to use digital tools to find, evaluate, create, and communicate information responsibly
 - (c) The ability to install software on a computer
 - (d) The ability to use social media for entertainment
- ii) A student asks classmates, “How does mobile phone use affect your concentration during study?” What type of data is mainly collected?
 - (a) Quantitative data because it can be counted easily
 - (b) Secondary data because classmates already know the answer
 - (c) Qualitative data because it collects opinions and experiences
 - (d) Numerical data because mobile phone use is related to time

- iii) Which search query is most suitable for finding a reliable PDF report about internet usage in Pakistan?
 - (a) internet usage Pakistan
 - (b) best internet use in Pakistan social media
 - (c) "internet usage in Pakistan" filetype:pdf
 - (d) Pakistan internet all websites latest
- iv) Which option is the best example of primary data?
 - (a) A government report downloaded from the internet
 - (b) A research article written by another author
 - (c) Survey responses collected directly from Class XI students
 - (d) Statistics copied from an online database
- v) Why are spreadsheets useful when presenting research data?
 - (a) They automatically make the research question correct
 - (b) They help organize data, perform calculations, and create charts
 - (c) They replace the need for collecting original data
 - (d) They ensure all online information is reliable
- vi) A digital artefact created after research should mainly:
 - (a) repeat long paragraphs from websites
 - (b) communicate results and conclusions clearly
 - (c) include decoration only
 - (d) avoid any data visuals
- vii) Data collected directly by the researcher is called:
 - (a) Secondary data
 - (b) Processed data
 - (c) Primary data
 - (d) Digital data
- viii) Which search operator helps search within a specific website or domain?
 - (a) OR
 - (b) AND
 - (c) NOT
 - (d) site:
- ix) Copying someone else's work without giving credit is called:
 - (a) Copyright
 - (b) Referencing
 - (c) Citation
 - (d) Plagiarism
- x) Asking permission before collecting responses is part of research:
 - (a) Analysis
 - (b) Sampling
 - (c) Presentation
 - (d) Ethics

2. Fill in the Blanks

- i) A well-defined _____ helps a student choose suitable data-collection methods and stay focused during research.
- ii) Data collected directly by the researcher through surveys, interviews or observation is called _____ data.

- iii) Data taken from reports, books, websites or databases prepared by others is called _____ data.
- iv) Copying words, images or ideas without giving credit is called _____.
- v) Asking permission before collecting responses is part of research _____.

3. Provide short answers to the following questions.

- i) What is primary data? Give one example.
- ii) What is secondary data? Give one example.
- iii) What is plagiarism and why should it be avoided?
- iv) What is meant by data privacy?
- v) Write any three ways to protect personal information while using digital technologies.

4. Provide detailed answers to the following questions.

- i) Explain the concept of digital literacy. Why is it necessary for students in modern education?
- ii) Describe the difference between qualitative and quantitative data with suitable examples.
- iii) Explain any four data-collection strategies: interviews, surveys, prototypes, observations, and simulations.
- iv) Describe the steps involved in designing a data-collection approach for a research question.
- v) Explain how spreadsheets, presentations, infographics, and reports can be used to communicate research findings.



Activities

1. In groups of 3–4 students, choose a topic such as smartphone use in college, online learning during examinations, or internet habits of students.
2. Form a clear research question, identify your target sample, and design a survey or interview with at least 8 questions. Make sure the wording is fair, clear, and ethical.
3. Collect responses from at least 20 students, enter the data into a spreadsheet, clean the data, and create at least one table and one chart or infographic.
4. Prepare a short digital artefact (3–5 slides, infographic, or brief report) that presents findings, conclusions, the reliability of any secondary sources used, and one recommendation for the intended audience.

GLOSSARY

Terms are listed alphabetically. The unit tag beside each entry shows where the term is first introduced.

A

Abstraction	<i>Unit 2</i>	Hiding unnecessary details and focusing only on the essential features required to solve a problem.
Actuator	<i>Unit 5</i>	A device that performs a physical action (such as movement or switching) in response to a system command, converting a digital signal into a physical action.
Advanced Search	<i>Unit 6</i>	The use of special search operators (e.g. quotation marks for exact phrases, filetype:, site:) to narrow down search results and find more relevant, reliable information.
Agile Model	<i>Unit 1</i>	An iterative software development approach that breaks a project into short cycles called sprints, allowing continuous feedback and flexible changes throughout development.
Algorithm	<i>Unit 2</i>	A step-by-step logical procedure used to solve a specific problem, independent of any programming language.
Alternate Key	<i>Unit 4</i>	A candidate key that is not chosen as the primary key, but which could still uniquely identify records in a table.
Analog Signal	<i>Unit 1</i>	A continuous signal that can take any value within a range and changes smoothly over time, like a wave (e.g. human voice, temperature).
AND Gate	<i>Unit 1</i>	A logic gate whose output is 1 only when all of its inputs are 1.
Artificial Intelligence (AI)	<i>Unit 3</i>	The capability of computational systems to perform tasks typically associated with human intelligence, such as learning, reasoning, problem-solving, perception, and decision-making.
Artificial Intelligence (AI) Applications	<i>Unit 5</i>	Technology that enables machines to think, learn, and make decisions, often working with IoT and Data Analytics as the "brain" of a smart system.
Assistive Technology	<i>Unit 5</i>	Devices, software, tools, and systems that help people with disabilities or functional limitations participate more fully in education, communication, mobility, and daily life.
Attribute	<i>Unit 4</i>	A property or characteristic of an entity in a database (e.g. Name, Class, and Marks are attributes of a Student entity).

B

Binary Digit (Bit)	<i>Unit 1</i>	The smallest unit of digital data, having only two possible values: 0 (OFF/LOW/FALSE) or 1 (ON/HIGH/TRUE).
Binary Search	<i>Unit 2</i>	A fast searching algorithm that repeatedly divides a sorted list in half to locate a target value; it only works on sorted data.
Boolean (bool)	<i>Unit 3</i>	A Python data type that holds one of only two values: True or False.

Glossary

Boolean Algebra	<i>Unit 1</i>	A branch of mathematics, developed by George Boole, that deals with only two values (TRUE/1 and FALSE/0) and forms the foundation of digital electronics and programming logic.
Bubble Sort	<i>Unit 2</i>	A simple sorting algorithm that repeatedly compares adjacent items and swaps them if they are in the wrong order, gradually "bubbling" the largest values to the end of the list.
Built-in Library	<i>Unit 3</i>	A library that comes packaged with Python and does not need to be installed separately before use.

C

Candidate Key	<i>Unit 4</i>	Any field, or set of fields, in a table that could uniquely identify each record; a table may have several candidate keys, only one of which becomes the primary key.
Cascade Update / Cascade Delete	<i>Unit 4</i>	Referential-integrity operations where a change (update) or removal (delete) of a record in a parent table is automatically applied to all related records in child tables.
Clarity	<i>Unit 2</i>	An evaluation criterion for an algorithm concerned with how readable and easy to understand its steps are.
Cloud Computing	<i>Unit 5</i>	A technology that allows users to store data and use services over the internet instead of relying solely on their own computers or mobile phones.
Cognitive / Learning Aid	<i>Unit 5</i>	An assistive technology category that helps with memory, reading, writing, and understanding, such as text-to-speech software, audiobooks, and digital planners.
Communication Aid	<i>Unit 5</i>	An assistive technology category that helps people with speech or expressive difficulties communicate more effectively, such as speech-generating devices or communication boards.
Communication Model	<i>Unit 1</i>	A set of rules and layers that explains how data travels from one device to another through a network, so that sender and receiver follow the same steps.
Correctness	<i>Unit 2</i>	An evaluation criterion for an algorithm concerned with whether it produces the right output for all valid inputs.
Cybersecurity	<i>Unit 3</i>	The practice of protecting computer systems, networks, and programs from digital attacks.

D

Data (Database context)	<i>Unit 4</i>	Unprocessed, raw facts, figures, or observations collected from various sources; it can be quantitative or qualitative and becomes information once organized and interpreted.
Data Analytics	<i>Unit 5</i>	The process of collecting, organizing, cleaning, studying, and presenting data in order to find useful information and support better decision-making.
Data Collection (Digital Literacy)	<i>Unit 6</i>	The process of gathering original information for a study or project, using methods such as interviews, surveys, prototypes, observation, or simulations.
Data Science	<i>Unit 3</i>	The study of data to extract meaningful insights that support business or research decisions.

Glossary

Database	<i>Unit 4</i>	An organized collection of related data, stored in a structured manner, so that it can be easily accessed, managed, and updated.
DBMS (Database Management System)	<i>Unit 4</i>	Software that allows users to create, manage, and interact with a database, acting as a bridge between the user and the stored data.
Debugger	<i>Unit 3</i>	A software tool used to find and fix errors in a program.
Decomposition	<i>Unit 2</i>	The process of breaking a complex task or system into smaller, more manageable parts.
Digital Artefact	<i>Unit 6</i>	The final product created using digital technologies such as a report, presentation, infographic, or dashboard that answers a research question.
Digital Literacy	<i>Unit 6</i>	The ability to understand and use digital devices, the internet, and online tools in a safe, effective, and responsible manner.
Digital Signal	<i>Unit 1</i>	A signal with only two levels, HIGH (1) and LOW (0), that jumps instantly between these values, forming a square-wave pattern.
Discrete Quantity	<i>Unit 1</i>	A separate, countable quantity with specific fixed values (e.g. the number of students in a class), as opposed to a continuous quantity.

E

Efficiency	<i>Unit 2</i>	An evaluation criterion for an algorithm concerned with how well it uses time and memory (space) resources.
Encapsulation / Decapsulation	<i>Unit 1</i>	The process of adding (encapsulation) or removing (decapsulation) header information as data passes down or up through the layers of a communication model such as OSI or TCP/IP.
Entity	<i>Unit 4</i>	Any real-world object or concept about which data is stored in a database (e.g. Student, Teacher, Course).
ER Model (Entity-Relationship Model)	<i>Unit 4</i>	A conceptual design tool used to plan a database before it is built, showing what data needs to be stored and how different pieces of data relate to each other.
Extension (VS Code)	<i>Unit 3</i>	An add-on feature that can be installed in an IDE such as VS Code to enhance its functionality.

F

Field	<i>Unit 4</i>	A column in a database table that represents a specific type of data or attribute (e.g. Name, RollNumber).
Float	<i>Unit 3</i>	A Python data type that stores numbers with a decimal point.
Foreign Key	<i>Unit 4</i>	A field in one table that refers to the primary key of another table, used to create relationships between tables.
Frameworks	<i>Unit 3</i>	Pre-built structures that automate common tasks and give developers a foundation for application development.

H

High-Level Language	<i>Unit 3</i>	A programming language designed to be easy for humans to read and write (e.g. Python), as opposed to low-level machine code.
----------------------------	---------------	--

I

IDE (Integrated Development Environment)	<i>Unit 3</i>	A software tool that combines an editor, compiler/interpreter, and debugger, allowing a programmer to write, run, and manage code in one place.
Indentation	<i>Unit 3</i>	The use of spaces at the beginning of a line in Python to define and separate blocks of code.
Infographic	<i>Unit 6</i>	A visual way of presenting information, facts, or data including interactive reports and dashboards so the main idea can be understood quickly.
Information	<i>Unit 4</i>	Data that has been organized, processed, and interpreted so that it carries meaning and supports decision-making.
Information Age	<i>Unit 6</i>	The current period in which vast amounts of data are produced, shared, and accessed instantly through the internet, mobile devices, and digital platforms.
Integer (int)	<i>Unit 3</i>	A Python data type that stores whole numbers without a decimal point.
Internet of Things (IoT)	<i>Unit 5</i>	A system in which physical devices are connected to the internet and can collect and share data to perform tasks automatically.
Interview (Data Collection)	<i>Unit 6</i>	A planned conversation, one-to-one or in a small group, used to gather information directly from a person; answers are usually qualitative.

J

Junction (Bridge) Table	<i>Unit 4</i>	A table used to break a many-to-many (M:N) relationship into two one-to-many relationships that a relational database can properly support.
--------------------------------	---------------	---

L

Linear Search	<i>Unit 2</i>	A searching algorithm that checks each item in a list one by one, in order, until the target value is found or the list ends.
Logic Gate	<i>Unit 1</i>	An electronic circuit that performs a basic Boolean operation (such as AND, OR, or NOT) on one or more binary inputs to produce a single binary output.
Logical Error	<i>Unit 3</i>	An error in which a program runs without crashing but produces the wrong output, due to a mistake in the program's logic.
Low-Level Language	<i>Unit 3</i>	A programming language that is closer to binary machine code and therefore more difficult for humans to read and write.

M

Mid-Level Language	<i>Unit 3</i>	A programming language that bridges the gap between low-level and high-level languages, offering both hardware access and human readability.
Mobility Aid	<i>Unit 5</i>	An assistive technology category that helps people move from one place to another, such as wheelchairs, walkers, or crutches.

N

NAND Gate	<i>Unit 1</i>	A universal logic gate that gives the opposite output of an AND gate, output is 0 only when all inputs are 1.
------------------	---------------	---

Glossary

NOR Gate	<i>Unit 1</i>	A universal logic gate that gives the opposite output of an OR gate, output is 1 only when all inputs are 0.
NOT Gate (Inverter)	<i>Unit 1</i>	A logic gate that reverses its single input, a 0 becomes 1, and a 1 becomes 0.

O

Observation (Data Collection)	<i>Unit 6</i>	A data-collection method in which the researcher watches and records what people do, without asking direct questions.
OR Gate	<i>Unit 1</i>	A logic gate whose output is 1 when at least one of its inputs is 1.
OSI Model	<i>Unit 1</i>	A 7-layer reference model (Open Systems Interconnection) used to explain and design how data communication happens between devices on a network.

P

Pattern Recognition	<i>Unit 2</i>	Identifying regularities, repetitions, or predictable changes within data to simplify a solution.
Primary Data	<i>Unit 6</i>	Data collected first-hand by the researcher for a particular purpose, gathered directly from people, events, or situations.
Primary Key	<i>Unit 4</i>	A field, or combination of fields, that uniquely identifies each record in a table; a table has exactly one primary key, and it cannot be NULL.
Primary Source	<i>Unit 5</i>	A source that gives original, first-hand information collected directly from people, places, events, or official records (e.g. an interview or an official record such as NADRA).
Privacy/Ethics	<i>Unit 6</i>	The responsible-research practice of asking permission before collecting data, avoiding unnecessary personal information, and using data only for its intended purpose.
Program	<i>Unit 3</i>	A set of instructions given to a computer to perform a specific task.
Programming Language	<i>Unit 3</i>	A special language, with clearly defined rules (syntax), used to write instructions for a computer to follow.
Protocol	<i>Unit 1</i>	An agreed set of rules that governs how data is formatted, transmitted, and received between devices in a network.
Prototype	<i>Unit 6</i>	A simple, early version of a product, model, or solution that allows an idea to be tested and improved through user feedback before the final version is built.
Pseudocode	<i>Unit 2</i>	A human-readable, semi-formal representation of an algorithm, written in plain English with structured keywords (e.g. IF, FOR), used before writing actual code.

Q

Qualitative Data	<i>Unit 6</i>	Descriptive, non-numerical data that describes what people think, feel, notice, or experience.
Quantitative Data	<i>Unit 6</i>	Numerical data that can be counted, measured, or compared.
Query	<i>Unit 4</i>	A request made to a database to retrieve, filter, or summarize specific data from one or more tables.

R

RDBMS (Relational Database Management System)	<i>Unit 4</i>	A DBMS that organizes data into related tables, based on the relational model, and is one of the most widely used database models in modern computing.
Record	<i>Unit 4</i>	A row in a database table representing a single entry or instance of data (e.g. one student's complete details).
Referential Integrity	<i>Unit 4</i>	A rule ensuring that a foreign key in one table always refers to an existing, valid record in another (parent) table, keeping data across related tables consistent.
Relational Schema	<i>Unit 4</i>	A formal representation of a database's structure, defining how data is organized into tables and what fields each table contains.
Relationship (1:1, 1:M, M:N)	<i>Unit 4</i>	The way entities in a database are logically connected: one-to-one (1:1), one-to-many (1:M), or many-to-many (M:N).
Report (Digital Literacy)	<i>Unit 6</i>	A structured document used to present collected data, findings, analysis, and conclusions in a clear and organized way.
Research Question	<i>Unit 6</i>	The clearly defined question that a study or project sets out to answer, guiding the choice of data type and data-collection method.
Runtime Error	<i>Unit 3</i>	An error that occurs while a program is actually running, interrupting its execution.

S

SDLC (Software Development Life Cycle)	<i>Unit 1</i>	A structured process, made up of defined phases (requirement analysis, design, coding, testing, deployment, maintenance), that guides software development from beginning to end.
Searching	<i>Unit 2</i>	The process of locating a specific element's position or existence within a list of data.
Secondary Data	<i>Unit 6</i>	Data that has already been collected by someone else and is reused for a new purpose (e.g. books, official reports, online databases).
Secondary Source	<i>Unit 5</i>	A source that presents information already collected and explained by someone else, used to summarize or analyze primary-source information (e.g. a textbook or a news report).
Selection Sort	<i>Unit 2</i>	A sorting algorithm that repeatedly finds the smallest (or largest) remaining item and moves it into its correct position.
Sensor	<i>Unit 5</i>	A device that collects information from the physical world, such as temperature, light, sound, movement, or pressure.
Sensory Aid	<i>Unit 5</i>	An assistive technology category that supports vision or hearing, such as hearing aids, screen readers, or Braille displays.
Simulation	<i>Unit 6</i>	A computer-based model of a real situation in which variables can be changed to observe outcomes; useful when real experiments are costly, risky, or difficult.
Sorting	<i>Unit 2</i>	Arranging a collection of items into a specific order (ascending, descending, or chronological).

Glossary

Source Evaluation	<i>Unit 6</i>	The process of judging whether an information source is trustworthy, by checking its author, currency, reliability, and evidence.
Spreadsheet	<i>Unit 6</i>	Software used to arrange information in rows and columns, perform calculations, sort/filter records, and create charts and graphs.
String (str)	<i>Unit 3</i>	A Python data type that stores text or a sequence of characters.
Survey / Questionnaire	<i>Unit 6</i>	A data-collection method that asks the same set of questions to many people, which can gather both quantitative and qualitative data.
Syntax Error	<i>Unit 3</i>	An error that occurs when the rules of a programming language are not followed, preventing the program from running at all.

T

Table	<i>Unit 4</i>	The basic structure of a database used to store data in rows (records) and columns (fields), similar to a spreadsheet.
TCP/IP Model	<i>Unit 1</i>	A 4-layer model (Application, Transport, Internet, Network Access) that is actually used to run communication on the Internet today.
Tertiary Source	<i>Unit 5</i>	A source that provides summarized, organized information derived from both primary and secondary sources, such as an encyclopedia or dictionary, mainly used for quick reference.
Third-Party Library	<i>Unit 3</i>	A library that is not included with Python by default and must be installed separately (e.g. via pip) before it can be used.
Traceback	<i>Unit 3</i>	A detailed error message provided by Python that identifies the type, location, and cause of an error in a program.
Truth Table	<i>Unit 1</i>	A table that lists every possible combination of inputs to a logic circuit or Boolean expression, along with the resulting output.

V

VS Code (Visual Studio Code)	<i>Unit 3</i>	A free, lightweight, and powerful code editor used to write and manage code in multiple programming languages.
-------------------------------------	---------------	--

W

Waterfall Model	<i>Unit 1</i>	A sequential software development model in which each phase (requirements, design, coding, testing, deployment, maintenance) must be completed before the next one begins.
------------------------	---------------	--

X

XNOR Gate	<i>Unit 1</i>	A logic gate whose output is 1 when its two inputs are the same (both 0 or both 1).
XOR Gate	<i>Unit 1</i>	A logic gate whose output is 1 only when its two inputs are different from each other.



Complete Teaching Guide

Computer Science

Grade XI

Teacher's Companion

How to use this guide:

For each unit, this guide provides an overview, learning objectives, a topic breakdown with suggested time and teaching approach, recommended activities, assessment guidance, and common misconceptions to address. Teachers may adapt period counts to their own timetable.

UNITS AT A GLANCE

UNIT	TITLE	DURATION	FOCUS
1	Computer Systems	3 weeks (9–10 periods)	Digital logic, SDLC, networking models
2	Computational Thinking and Algorithms	3 weeks (9–10 periods)	CT strategies, sorting & searching
3	Programming Fundamentals	4 weeks (12–14 periods)	Python programming fundamentals
4	Database Management	4 weeks (12–14 periods)	Relational databases & MS Access
5	Applications and Impacts of Computing	3 weeks (8–9 periods)	AI, IoT, data analytics, impacts, assistive tech
6	Digital Literacy	3 weeks (9–10 periods)	Digital literacy & inquiry project

Chapter 1

COMPUTER SYSTEMS

This unit introduces three foundational areas of computing: Digital Logic (how computers represent and process data using binary values and logic gates), the Software Development Life Cycle “SDLC” (how software is planned, built, tested and maintained, comparing Waterfall and Agile models), and Communication Models (how devices exchange data over networks using the OSI and TCP/IP models). (Suggested duration: 3 weeks (9–10 periods)).

Learning Objectives:

- Differentiate analog vs. digital data and discrete vs. continuous quantities.
- Apply Boolean algebra/operators; build and read truth tables and logic expressions.
- Identify logic gates and build circuits in LogiSim.
- Explain SDLC phases and compare Waterfall and Agile models using case studies.
- Describe the OSI 7-layer model and TCP/IP 4-layer model, and the role/protocols of each layer.

Topic Breakdown & Suggested Time:

Topic	Key Concepts	Suggested Teaching Approach
1.1 Digital Logic (4 periods)	Analog vs digital, Boolean algebra, logic gates, truth tables, logic diagrams, LogiSim	Use the stairs-vs-ramp analogy; demonstrate each gate live in LogiSim; have students build truth tables before circuits
1.2 SDLC (3 periods)	SDLC phases, Waterfall model, Agile model, testing types, maintenance	Walk through both case studies; use a group activity comparing Waterfall vs Agile for different project types
1.3 Communication Models (3 periods)	Communication model, OSI 7 layers, encapsulation/decapsulation, TCP/IP 4 layers	Teach the mnemonic "Please Do Not Throw Sausage Pizza Away"; run the ping activity for a hands-on network demo

Resources & Materials Needed:

- Computer lab with internet access; LogiSim Evolution installed (or web-based version)
- Projector/whiteboard for circuit diagrams and OSI/TCP-IP layer charts
- Command prompt/terminal access for the ping activity

Recommended Activities:

- Build and verify a truth table for a compound Boolean expression in LogiSim.
- Design a logic circuit for a burglar alarm (OR-gate application).
- Draft a simple SRS document for a calculator app.
- Run "ping www.google.com" and identify the OSI layer/protocol involved.

Assessment:

- MCQs and fill-in-the-blanks for quick recall of terms (gates, SDLC phases, layers).
- Short-answer questions comparing concepts (analog vs digital; TCP vs UDP; Waterfall vs Agile).
- Detailed/long questions: full gate truth tables, SDLC phase explanation, OSI layer functions with protocols.
- Practical/lab assessment: LogiSim circuit built correctly from a given expression.

Common Misconceptions to Address:

- Students confuse the OSI model (7 layers, conceptual) with TCP/IP (4 layers, practical), stress they describe the same process at different granularity.
- Students assume Agile has no planning/documentation, clarify it is iterative, not undocumented.
- Students think NOR/NAND are just "opposite" gates rather than universal gates that can build any other gate.

**Chapter
2**

COMPUTATIONAL THINKING AND ALGORITHMS

This unit builds computational thinking skills: analyzing real-world problems and solving them using algorithmic strategies decomposition, pattern recognition, and abstraction. Students learn to design, evaluate and compare algorithms, focusing on two sorting algorithms (Bubble Sort, Selection Sort) and two searching algorithms (Linear Search, Binary Search), and how to judge an algorithm's correctness, efficiency, and clarity. (Suggested duration: 3 weeks (9–10 periods))

Learning Objectives:

- Analyze real-world problems and apply computational thinking to solve them.
- Explain the importance and properties of a good algorithm, and write pseudocode.
- Apply the strategies of decomposition, pattern recognition, and abstraction to a problem.
- Trace, compare and implement Bubble Sort and Selection Sort.
- Trace, compare and implement Linear Search and Binary Search.
- Evaluate an algorithm's correctness, efficiency (time/space), and clarity, and choose the right algorithm for a given problem.

Topic Breakdown & Suggested Time

Topic	Key Concepts	Suggested Teaching Approach
2.1–2.2 CT & Algorithms (2 periods)	Computational thinking, what is an algorithm, importance of algorithms, algorithm vs pseudocode	Start with an everyday task (making tea) to introduce step-by-step thinking; contrast algorithm vs pseudocode with a simple example
2.3 Algorithmic Strategies (2 periods)	Decomposition, Pattern Recognition, Abstraction	Use relatable real-life examples for each strategy; have students identify the strategy used in short scenarios
2.4 Sorting & Searching (4 periods)	Bubble Sort, Selection Sort, Linear Search, Binary Search algorithm, pseudocode, worked example	Trace each algorithm step-by-step on the board with a small list of numbers; have students hand-trace before coding/pseudocoding
2.5–2.6 Evaluating & Choosing Algorithms (2 periods)	Correctness, efficiency, clarity; decision guide for choosing an algorithm	Use the simple decision guide (sorted? size? speed vs memory?) with practice scenarios to pick the right algorithm

Resources & Materials Needed:

- Whiteboard/projector for tracing sorting and searching steps
- Attendance register or similar real data set for hands-on sorting/searching activities
- Index cards or number cards for a physical sorting demonstration

Recommended Activities:

- Arrange students in the assembly queue by height, physical demonstration of sorting.
- Sort names in the attendance register in descending alphabetical order by hand.
- Find a classmate's name in the attendance register using the Binary Search technique.
- Trace Bubble Sort and Selection Sort on the same list and compare the number of swaps/comparisons.

Assessment:

- MCQs and fill-in-the-blanks for quick recall of terms (algorithm, sorting, searching, CT strategies).
- Short-answer questions defining pattern recognition, decomposition, correctness, efficiency, and clarity.
- Detailed/long questions: write steps of Selection Sort and Binary Search; justify choice of searching algorithm for large datasets.
- Practical assessment: manually trace a sort/search algorithm on a given list and show intermediate steps.

Common Misconceptions to Address:

- Students confuse an algorithm (language-independent logic) with actual program code, reinforce that pseudocode/algorithm precedes coding.
- Students think Binary Search works on any list, stress it requires sorted data first.
- Students assume a faster algorithm is always "better", clarify that clarity and correctness matter alongside efficiency, especially for small datasets.

Chapter 3

PROGRAMMING FUNDAMENTALS

This unit introduces Python programming: what programming and programming languages are, setting up and using an IDE (VS Code), Python basics (variables, data types, input/output), operators (arithmetic, assignment, relational, logical, membership, bitwise), control structures (sequence, selection, repetition), built-in and third-party libraries, and debugging (syntax, runtime, and logical errors). (Suggested duration: 4 weeks (12–14 periods))

Learning Objectives:

- Understand the concept of programming and identify real-world applications of Python.
- Use the elements of an IDE (VS Code) to write, run, and debug Python code.
- Understand variables, data types, and use input()/print() for I/O handling.
- Apply arithmetic, assignment, relational, logical, membership, and bitwise operators.
- Use selection (if/elif/else) and repetition (for/while) control constructs to solve problems.
- Use built-in and third-party Python libraries, and debug code by identifying error types.

Topic Breakdown & Suggested Time:

Topic	Key Concepts	Suggested Teaching Approach
Programming Basics & IDE (2 periods)	Programming languages (low/mid/high level), Python applications, VS Code interface	Demo real-world Python use cases; do a guided tour of VS Code before coding starts
Variables, Data Types & I/O (2 periods)	Variables, dynamic typing, input()/print(), type conversion	Use the "container/storage box" analogy for variables; live-code input()/print() and show that input() always returns a string
Operators (2 periods)	Arithmetic, assignment, relational, logical, membership, bitwise operators	Build a single running example program, adding one operator category at a time; use truth tables for logical/bitwise operators
Control Structures (4 periods)	Sequence; Selection (if, if-else, if-elif-else, nested if); Repetition (for, while, nested loops, break/continue)	Trace flowcharts alongside code; have students convert everyday decisions/repeated tasks into if/loop structures before typing code
Libraries & Debugging (2–3 periods)	Built-in libraries (math, random, datetime), third-party libraries (pip install), syntax/runtime/logical errors, tracebacks	Demonstrate pip install live; intentionally break a working program to show each error type and how to read a traceback

Resources & Materials Needed:

- Computer lab with Python and VS Code installed (with Python extension)
- Internet access for pip install demonstrations (third-party libraries)
- Projector for live-coding and error/traceback demonstrations

Recommended Activities:

- Write and execute a first Python program in VS Code.
- Build a program combining sequence, selection, and repetition (e.g. grading or number-checking program).
- Generate a student report card using built-in library functions.
- Build an arithmetic calculator, a "guess the number" game, or a simple ATM simulation.

Assessment:

- MCQs and fill-in-the-blanks on syntax basics (comments, variable names, indentation, data types).
- Short-answer questions comparing for vs while loops, break vs continue, and error types.
- Detailed/long questions: explain data types and operators with tables and code, and write control-structure programs with syntax.
- Practical/lab assessment: write, run and debug a short Python program.

Common Misconceptions to Address:

- Students think input() returns a number by default, stress it always returns a string that must be explicitly converted.
- Students confuse '=' (assignment) with '==' (comparison), reinforce with side-by-side examples.
- Students assume any error means the program is completely wrong, clarify syntax vs runtime vs logical errors.

**Chapter
4****DATABASE MANAGEMENT**

This unit introduces relational database concepts and practical database management using MS Access. Students learn database components (tables, keys, constraints), the Entity-Relationship (ER) Model and relationship types (1:1, 1:M, M:N), referential integrity (cascade update/delete), relational schema development through a Library Management System case study, and hands-on creation of tables, forms, queries, and data summaries/visualizations in MS Access. (Suggested duration: 4 weeks (12–14 periods))

Learning Objectives:

- Understand database concepts, elements, and the role of a DBMS.
- Understand relationship types (1:1, 1:M, M:N) and referential integrity (cascade update/delete).
- Develop a relational schema and Entity-Relationship (ER) Model from a real-life scenario.
- Create tables in MS Access using appropriate data types, keys, and constraints.
- Create forms and queries based on defined criteria, and design reports.
- Analyze datasets to produce summary statistics and data visuals using a relational DBMS.

Topic Breakdown & Suggested Time:

Topic	Key Concepts	Suggested Teaching Approach
Database Concepts & Components (2 periods)	Data vs information, DBMS, tables/records/fields, keys and integrity constraints, RDBMS	Use a familiar dataset (class attendance/marks register); identify fields, records, and a primary key together
ER Model & Referential Integrity (3 periods)	Entities, attributes, relationships (1:1, 1:M, M:N), referential integrity, cascade update/delete	Draw simple ER diagrams together on the board; use a junction-table example to resolve an M:N relationship
Relational Schema Case Study (2 periods)	Library Management System: requirement analysis → entities → attributes → cardinality → final ER schema	Walk through the case study step-by-step; have students replicate the same steps for their own mini-scenario
Tables, Forms & Queries in MS Access (4–5 periods)	Creating tables (Datasheet/Design view, data types), Form Wizard, SELECT queries, filtering (AND/OR), grouping, summary statistics, charts	Hands-on lab: build one table, one form, and 2–3 queries progressively in the same MS Access database

Resources & Materials Needed:

- Computer lab with Microsoft Access installed
- Sample dataset (e.g. student records) for hands-on table/form/query practice
- Projector/whiteboard for ER diagrams and schema walkthroughs

Recommended Activities:

- Create a Student table with at least five fields and assign a primary key.
- Design an ER diagram for a school system (Student–Course–Teacher).
- Create a data-entry form using the Form Wizard.
- Build queries with AND/OR conditions and a grouped query; create a bar/pie chart from the summary.

Assessment:

- MCQs and fill-in-the-blanks on database terms (DBMS, ER Model symbols, junction table, cascade rules, candidate key).
- Short-answer questions distinguishing data vs information, field vs record, and explaining referential integrity.
- Detailed/long questions: design a full ER model and relational schema for a given scenario.
- Practical/lab assessment: build a table, form, and at least one filtered/grouped query in MS Access.

Common Misconceptions to Address:

- Students confuse a table's primary key with any unique-looking field, clarify only one key is chosen to uniquely identify each record.
- Students think M:N relationships can be modeled directly, stress the need for a junction/linking table.
- Students treat forms, queries, and reports as interchangeable, clarify their distinct roles.

Chapter 5

APPLICATIONS AND IMPACTS OF COMPUTING

This unit explores how computing is applied across daily life and various sectors (education, healthcare, banking, agriculture, industry). It covers Artificial Intelligence (AI), the Internet of Things (IoT), Data Analytics, and how these work together; evaluating information sources (primary, secondary, tertiary); the positive and negative impacts of computing on society and responsible/ethical use; and assistive technologies that support people with disabilities. (Suggested duration: 3 weeks (8–9 periods))

Learning Objectives:

- Explain how AI, IoT, and Data Analytics work together in smart systems (e.g. smart homes, smart irrigation).
- Describe real-world uses of IoT and AI in different areas, including career and skill connections.
- Distinguish primary, secondary, and tertiary information sources and judge their reliability.
- Analyze the positive and negative impacts of computing across fields of life and identify responsible-use practices.
- Describe categories of assistive technologies and their importance.
- Identify career opportunities and required skills related to AI, IoT, Data Analytics, and Assistive Technologies.

Topic Breakdown & Suggested Time:

Topic	Key Concepts	Suggested Teaching Approach
AI, IoT & Data Analytics (3 periods)	What AI/IoT/Data Analytics are, how they work together, IoT components, uses in different areas	Use the smart-home/smart-irrigation example to trace the flow: sensor (IoT) → data (Analytics) → decision (AI)
Information Sources (1–2 periods)	Primary, secondary, and tertiary sources; judging reliability of information	Bring examples (textbook, interview, social media post) and have students classify and rate reliability
Impacts of Computing (2 periods)	Positive/negative impacts across education, communication, business, healthcare, agriculture; responsible and ethical use	Use the impacts table as a discussion starter; have students debate one positive vs one negative impact per field
Assistive Technologies (2 periods)	Mobility, sensory, communication, cognitive/learning aids; importance for inclusion; related careers	Show real examples/videos of screen readers, speech-to-text, mobility aids; connect to relatable scenarios

Resources & Materials Needed:

- Projector/internet access to show short videos/demos of AI, IoT devices, and assistive technologies
- Sample information sources (textbook excerpt, news article, social media post) for the reliability activity
- Whiteboard for the positive/negative impacts discussion table

Recommended Activities:

- Trace a smart irrigation system end-to-end: sensor → data processing → actuator decision.
- Classify a set of given sources as primary, secondary, or tertiary and justify reliability.
- Debate the positive vs. negative impact of computing in one chosen field.
- Research and present one assistive technology and the career path connected to building it.

Assessment:

- MCQs on recognizing technology use-cases (sensor vs actuator, IoT vs AI, source type).
- Short-answer questions on how AI/IoT/Data Analytics work together and the difference between a sensor and an actuator.
- Detailed/long questions: explain a full IoT system for a real scenario; analyze positive and negative impacts.
- Reflective assessment: a short written response on responsible/ethical technology use.

Common Misconceptions to Address:

- Students think AI, IoT, and Data Analytics are the same thing, clarify their distinct but connected roles (collect → analyze → decide).
- Students assume all online information is equally reliable, reinforce checking authorship, sourcing, and cross-referencing.
- Students view assistive technologies as only for severe disabilities, broaden understanding to everyday accessibility tools.

Chapter
6

DIGITAL LITERACY

This unit develops digital literacy: using advanced search techniques and evaluating information sources; distinguishing qualitative vs quantitative and primary vs secondary data; designing data-collection approaches (interviews, surveys, prototypes, observation, simulations); and presenting findings responsibly using digital tools (spreadsheets, presentations, infographics). It culminates in a Digital Inquiry Project where students research a question and produce a final digital artefact. (Suggested duration: 3 weeks (9–10 periods))

Learning Objectives:

- Perform advanced searches to locate information and evaluate source reliability.
- Differentiate qualitative vs quantitative data, and primary vs secondary data.
- Design a suitable data-collection approach (interview, survey, prototype, observation, or simulation) for a research question.
- Apply best practices to present primary and secondary data for a research question.
- Present data using appropriate digital tools such as spreadsheets, graphs, and infographics.
- Plan and produce a digital artefact that answers a research question.

Topic Breakdown & Suggested Time:

Topic	Key Concepts	Suggested Teaching Approach
Digital Literacy & Data Types (2 periods)	Digital literacy in the Information Age, qualitative vs quantitative data, advanced search operators, source evaluation	Demonstrate a live advanced search (quotation marks, site: operator) and compare results

Data-Collection Strategies (2 periods)	Interviews, surveys/questionnaires, prototypes, observation, simulations; primary vs secondary data	Give each strategy a real scenario and have students match the best method to it
Designing a Data-Collection Approach (1–2 periods)	Choosing a method based on the research question, privacy/consent/ethics	Have students draft a research question and justify their chosen data-collection method
Presenting Data with Digital Tools (2 periods)	Spreadsheets, presentations, infographics, choosing the right visual for the message	Show the same dataset as a table, bar chart, and infographic; discuss when each is most effective
Digital Inquiry Project Case Study (2–3 periods)	Research question → methodology → primary/secondary findings → digital artefact	Walk through the case study as a model, then have students plan their own mini research question and artefact

Resources & Materials Needed:

- Computer lab/internet access for advanced search and data-presentation practice
- Spreadsheet software and a presentation/infographic tool (e.g. Canva, PowerPoint)
- Sample datasets and example survey/interview questions for practice

Recommended Activities:

- Run advanced searches using quotation marks and the site: operator, and compare result quality.
- Draft a short survey/questionnaire for a chosen research question and identify possible bias.
- Turn a small dataset into a spreadsheet chart and an infographic, then compare their impact.
- Plan and outline a mini Digital Inquiry Project: research question, method, and final artefact choice.

Assessment:

- MCQs and fill-in-the-blanks on key terms (primary/secondary data, advanced search operators, digital artefact).
- Short-answer questions comparing qualitative vs quantitative data and primary vs secondary data.
- Detailed/long questions: design a data-collection approach for a given research question.
- Project-based assessment: a complete digital artefact answering a research question.

Common Misconceptions to Address:

- Students think any online information is trustworthy, reinforce checking author, date, evidence, and source reliability.
- Students confuse primary and secondary data, clarify primary is collected directly, secondary comes from existing sources.
- Students assume more visuals automatically make a presentation better, stress choosing the visual that fits the message.

All rights are reserved with the **SINDH TEXTBOOK, BOARD, JAMSHORO.**

Prepared by **SINDH TEXTBOOK, BOARD, JAMSHORO.**

Reviewed by **Directorate of Curriculum Assessment and Research Sindh, Jamshoro**

Prescribed by the Board of Intermediate and Secondary Education Hyderabad, Sukkur, Shaheed Benazirabad, Larkana, Mirpurkhas and Karachi for Secondary School Examination.

Approved by the **School Education and Literacy Department, Government of Sindh.**

Letter No. SELD/HCW/18/2018: Dated 22nd July, 2026 for the province of Sindh.

قومی ترانہ

پاک سرزمین شاد باد کیشور حسین شاد باد

تونشانِ عزمِ عالی شان آرضِ پاکستان

مرکزِ یقین شاد باد

پاک سرزمین کا نظام ثبوتِ انوثِ عوام

قوم، ملک، سلطنت پائندہ تابندہ باد

شاد باد منزلِ مراد

پرچم ستارہ و ہلال رہبرِ ترقی و کمال

ترجمانِ ماضی شانِ حال جانِ استقبال

سایہِ خداے دُعا لجلال

Serial No.		Publisher Code No.	
Month and Year of Publication	Edition	Quantity	Price