Module 3 – Fundamentals of IT

THEORY EXERCISE:

📄 What is a Program? (In Simple Words)

A **program** is a set of **instructions** written by a person (a programmer) to make a **computer do a specific task**.

Just like a recipe tells a chef what to do step-by-step, a program tells a computer what actions to perform, like:

- Showing a message on screen
- Adding two numbers
- Opening a website
- Saving a file

⚙ How Does a Program Function?

1. **Written in a Language Computers Understand**
   Programs are written using **programming languages** like Python, C++, or Java.
2. **Stored as Code**
   The code is saved as a file (like `.py`, `.cpp`, etc.).
3. **Run by a Processor**
   When you run the program:
   - The **CPU (central processing unit)** reads each instruction **one by one**.
   - It performs the task: math, showing text, saving data, etc.
4. **Input → Process → Output**
   A program usually follows this basic flow:
   - **Input**: Data from user or file
   - **Process**: Computer thinks/calculates
   - **Output**: Shows result on screen or stores it

✅ Example:

Imagine a calculator app:

- You enter: 5 + 3
- The program reads your input
- It processes the addition
- It shows: **8**

THEORY EXERCISE:

: **What are the key steps involved in the programming process?**

✅ **1.** Understanding the Problem

- First, you must clearly understand **what the program should do**.
- Example: "Create a program to calculate the average of three numbers."

✅ **2.** Planning the Solution

- Think and plan **how the program will solve the problem**.
- You may use:
  - **Flowcharts** (diagrams)
  - **Pseudocode** (simple English instructions)

✅ **3.** Writing the Code

- Use a **programming language** (like Python, C++, Java) to write the instructions.
- This is called **coding or development**.

✅ **4.** Compiling or Interpreting

- The code needs to be **converted** into a form the computer understands (machine code).
- Some languages use a **compiler** (like C++).
- Others use an **interpreter** (like Python).

✅ **5.** Testing the Program

- Run the program with different inputs to **check for errors (bugs)**.
- Make sure it gives the correct output.

✅ **6.** Debugging

- If there are any mistakes or bugs, **find and fix them**.

✅ **7.** Final Execution

- Once it works correctly, the program is ready to be **used by others** or **deployed** to a real system.

✅ **8.** Maintenance and Updates

- After launch, you may need to:
  - Add new features
  - Fix problems
  - Improve performance

THEORY EXERCISE:

What are the main differences between high-level and low-level programminglanguages?

🏭 1. Level of Abstraction

High-Level Language	Low-Level Language
Closer to **human language**	Closer to **machine language**
Easy to read and write	Harder to read and understand
Example: Python → `print("Hello")`	Example: Assembly → `MOV AX, 4C00h`

🛠️ 2. Ease of Use

High-Level	Low-Level
Easier to learn for beginners	Requires technical knowledge of hardware
Automatic memory management (in most cases)	Manual memory and CPU management

---

⚡ 3. Speed and Performance

High-Level	Low-Level
Slower execution (due to abstraction)	Very fast and efficient
Compiled or interpreted into machine code	Already close to machine code

---

 4. Hardware Control

High-Level	Low-Level
Limited control over hardware	Full control over memory and hardware
Used for apps, websites, games	Used for systems, firmware, drivers

---

 5. Examples

- **High-Level Languages**: Python, Java, C++, JavaScript, PHP
- **Low-Level Languages**: Assembly language, Machine code (binary)

THEORY EXERCISE:

Describe the roles of the client and server in web communication

Client (User Side):

- **Role**: The client is the device or software (usually a **web browser**) that **requests information** from the server.
- **Examples**: Google Chrome, Mozilla Firefox, Microsoft Edge.
- **Function**:
    1. The user enters a web address (URL).
    2. The browser sends an **HTTP/HTTPS request** to the server.
    3. It waits for a response and then **displays the website** or data received.

Server (Website Side):

- **Role**: The server is a **powerful computer** that **stores websites, applications, databases, and resources**.
- **Function**:
    1. Receives the client's request.
    2. Processes the request (fetches data, runs code, etc.).
    3. Sends back a **response**, usually in the form of an HTML page, image, or data.

Example of Communication:

1. You type `www.example.com` in a browser (client).
2. The client sends a request to the web server.
3. The server finds the page and sends it back.
4. The browser displays the page to you.

THEORY EXERCISE:

Explain the function of the TCP/IP model and its layers

✅ Function of the TCP/IP Model:

- **Organizes how data travels** from one device to another.
- **Ensures reliable communication** between devices.
- **Breaks down complex networking tasks** into manageable layers, each with a specific role.

📶 Layers of the TCP/IP Model:

The TCP/IP model has **4 layers**:

---

1. Application Layer (Top Layer)

- **Purpose**: Provides services directly to the user or application.
- **Protocols**: HTTP (web), FTP (file transfer), SMTP (email), DNS (domain lookup)
- **Function**: Manages how applications access network services.

🔻 Example: When you visit a website, your browser uses **HTTP** to communicate with the server.

---

2. Transport Layer

- **Purpose**: Ensures reliable data delivery between devices.
- **Protocols**:
    - **TCP**: Reliable, connection-oriented (e.g., for websites).
    - **UDP**: Fast, connectionless (e.g., for video streaming).

- **Function**: Breaks data into segments, checks for errors, ensures data is delivered in order.

🔻 Example: TCP checks if your message reaches the other side without errors.

---

3. Internet Layer

- **Purpose**: Determines **routing** of data between networks.
- **Protocol**: **IP** (Internet Protocol)
- **Function**: Adds IP addresses, decides the best path for data to travel.

🔻 Example: Like a GPS, it decides the route your data takes to reach the destination.

---

4. Network Access Layer (Link Layer)

- **Purpose**: Handles **physical transmission** of data.
- **Function**: Manages device addressing (MAC), network hardware (like switches), and media (like cables or Wi-Fi).

🔻 Example: Sends data over Ethernet or Wi-Fi from your computer to a router.

THEORY EXERCISE:

Explain Client Server Communication

✅ Client-Server Communication Explained

Client-server communication is the way two devices (the **client** and the **server**) interact over a network to exchange data.

🔄 How it Works (Step-by-Step):

1. **Client Sends a Request**:
    - A **client** (like a web browser or mobile app) starts communication by sending a **request**.
    - Example: You type `www.google.com` in your browser. The browser sends an HTTP request to Google's server.
2. **Server Processes the Request**:

    - The **server** (a remote computer hosting the website or service) receives the request.
    - It processes it (e.g., fetching data from a database or running a program).
3. **Server Sends a Response**:
    - After processing, the server sends a **response** back to the client.
    - Example: An HTML page, a file, or some data.
4. **Client Displays the Data**:
    - The client receives the response and **displays the content** to the user.

🌐 Common Example:

You open your browser and go to a website:

- **Client**: Your browser (e.g., Chrome)
- **Request**: "Get me the home page of www.example.com"
- **Server**: The computer hosting that site responds with the website files
- **Response**: The site loads in your browser

□ Key Concepts:

Term	Description
Client	Sends request, receives data (e.g., browsers, apps)
Server	Receives request, processes it, sends data back
Request	Message from client to server (e.g., "Give me this page")
Response	Message from server to client (e.g., "Here's the page")
Protocol	Rules used for communication (e.g., HTTP, FTP)

🔐 Bonus – Protocols Used:

- **HTTP/HTTPS** – for websites
- **FTP** – for file transfer
- **SMTP** – for sending emails
- **WebSockets** – for real-time communication