Module 1 – Overview of IT Industry

Question: What is a Program?

Answer: A **program** is a set of instructions written in a programming language that tells a computer what to do. These instructions are executed by the computer to perform specific tasks, such as calculations, data processing, or interacting with users.

**THEORY EXERCISE**: Explain in your own words what a program is and how it functions.

Answer: A **program** is a set of instructions that tells a computer what to do. These instructions are written in a programming language and are designed to perform specific tasks. A program can be as simple as displaying text on a screen or as complex as running a search engine.

**How a Program Functions**

1. **Writing the Code** – A programmer writes instructions using a programming language (e.g., Python, Java, C).
2. **Translation to Machine Code** – The written code is converted into machine language that the computer understands. This happens through **compilation** (in compiled languages like C) or **interpretation** (in interpreted languages like Python).
3. **Execution** – The computer follows the instructions step by step to perform the task.
4. **Output/Results** – Based on the instructions, the program provides an output (e.g., displaying text, processing data, or controlling hardware).

Question: What is Programming?

Answer: **Programming** is the process of creating programs by writing code that instructs a computer to perform tasks. It involves logic, problem-solving, and creativity to develop software applications.

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

Answer: **Key Steps in the Programming Process**

The programming process involves several systematic steps to design, write, and test a program. Below are the main stages:

**1. Problem Definition**

* Understand the problem you want to solve.
* Define clear objectives and requirements for the program.
* Identify inputs, processes, and expected outputs.

*Example:* A program to calculate the area of a rectangle needs length and width as inputs and should return the area as output.

**2. Planning and Algorithm Design**

* Develop a step-by-step plan (algorithm) to solve the problem.
* Use flowcharts or pseudocode to outline logic before writing actual code.

*Example (Pseudocode for calculating area):*

START

Read length

Read width

Compute area = length × width

Display area

END

**3. Writing the Code (Implementation)**

* Choose an appropriate programming language (Python, C, Java, etc.).
* Write the code following the algorithm and syntax rules of the chosen language.

*Example (Python code for area calculation):*

length = float(input("Enter length: "))

width = float(input("Enter width: "))

area = length \* width

print("Area of rectangle:", area)

**4. Compilation/Interpretation**

* If using a **compiled language** (C, Java), the source code is compiled into machine code before execution.
* If using an **interpreted language** (Python, JavaScript), the code is executed line by line.

**5. Testing and Debugging**

* Run the program with different inputs to check if it works correctly.
* Identify and fix errors (bugs) in logic, syntax, or runtime execution.
* Ensure the program meets all requirements.

**6. Execution and Deployment**

* Once tested, the program is executed in a real-world environment.
* The software can be packaged and deployed for users.

**7. Maintenance and Updates**

* Regularly update the program to fix bugs, improve efficiency, and add new features.
* Monitor performance and collect user feedback for improvements.

Question: Types of Programming Languages

**1. Low-Level Languages**

These languages are close to machine code and provide direct hardware control.

**a) Machine Language (Binary Code)**

· Written in **0s and 1s** (binary).

· Directly understood by computers.

· Difficult for humans to write and debug.

✅ **Example:** 10110100 11001101 (Binary representation of an instruction)

**b) Assembly Language**

· Uses symbolic **mnemonics** instead of binary (e.g., MOV A, B instead of 1010 0101).

· Requires an **assembler** to convert into machine code.

· Faster and more efficient but harder to write than high-level languages.

✅ **Example (Assembly Code):**

MOV AL, 61h

ADD AL, 5h

**2. High-Level Languages**

These are human-friendly languages that use English-like syntax.

**a) Procedural Programming Languages**

· Based on **step-by-step instructions** (procedures or functions).

· Code is structured and executed sequentially.

✅ **Examples:**

· **C**

· **FORTRAN**, **Pascal**

**b) Object-Oriented Programming (OOP) Languages**

· Uses **objects** and **classes** to organize code.

· Supports principles like **encapsulation, inheritance, and polymorphism**.

✅ **Examples:**

· **Java**

· **C++**, **Python (supports OOP)**

**c) Functional Programming Languages**

· Based on **mathematical functions**.

· Uses **immutable data** and avoids changing state.

✅ **Examples:**

· **Haskell**, **Lisp**, **Scala**

**d) Scripting Languages**

· Used for **automation, web development, and task execution**.

· Often **interpreted** instead of compiled.

✅ **Examples:**

· **Python**

· **JavaScript**, **Perl**, **PHP**

**3. System Programming Languages**

* Designed for **low-level system tasks** like OS development.
* Provides **direct memory access and hardware control**.

✅ **Examples:**

· **C**, **Rust**

**4. Database Query Languages**

* Used for interacting with **databases**.
* Retrieves, updates, and manages data.

✅ **Example (SQL – Structured Query Language)**

**5. Markup and Style Sheet Languages**

* Not traditional programming languages but used for structuring and styling content.

✅ **Examples:**

* **HTML (HyperText Markup Language)**
* CSS (Cascading Style Sheets)

·

**THEORY EXERCISE**: What are the main differences between high-level and low-level programming languages?

Answer:

1**. Abstraction Level:**

* **High-level**: Provides more abstraction from the hardware, making it easier to write and read. They focus on human-readable syntax and are closer to natural language (e.g., Python, Java, C++).
* **Low-level**: Closer to machine code and provides less abstraction. They are more difficult to read and write but offer more control over hardware (e.g., Assembly, Machine code).

**2. Ease of Use:**

* **High-level**: Easier to learn and use because they include more user-friendly syntax and are designed to be closer to human thought processes.
* **Low-level**: More complex to use, as programmers must manage memory, registers, and direct hardware manipulation.

**3. Portability:**

* **High-level**: More portable across different platforms, as they are abstracted from the underlying hardware (e.g., Java can run on any system with a Java Virtual Machine).
* **Low-level**: Less portable, as they are tailored for a specific type of hardware architecture (e.g., Assembly code is often written for specific processors).

**4. Performance:**

* **High-level**: Generally slower, as the high-level abstractions add overhead. Code needs to be interpreted or compiled before it runs.
* **Low-level**: Typically faster, as it interacts directly with the hardware, with less abstraction overhead.

**5. Control Over Hardware:**

* **High-level**: Provides less control over hardware, as many low-level operations are abstracted away.
* **Low-level**: Offers greater control over hardware, which is essential for tasks requiring fine-grained resource management (e.g., operating system development, embedded systems).

**6. Development Speed:**

* **High-level**: Faster development since you can focus on problem-solving without dealing with the intricacies of the hardware.
* **Low-level**: Slower development, as it involves manual management of hardware and resources, which can be error-prone and time-consuming.

**7. Error Handling:**

* **High-level**: Includes built-in error handling mechanisms, like exceptions, making it easier to debug.
* **Low-level**: Fewer built-in mechanisms for error handling, making debugging harder and requiring more manual checks.

Question: World Wide Web & How Internet Works

Answer: The **World Wide Web (WWW)** is a system of interlinked web pages and resources that can be accessed through the internet using a web browser. It was invented by **Tim Berners-Lee** in 1989 and is based on the following components:

1. **Web Pages**: Documents containing text, images, videos, and links, usually written in **HTML (HyperText Markup Language)**.
2. **URLs (Uniform Resource Locators)**: Addresses used to locate web pages (e.g., https://www.google.com).
3. **HTTP/HTTPS (HyperText Transfer Protocol)**: A protocol that allows communication between web browsers and web servers.
4. **Web Browsers**: Software applications like **Google Chrome, Mozilla Firefox, or Safari** that retrieve and display web pages.

The **WWW is a service that runs on the Internet**, enabling users to browse and interact with websites.

**How the Internet Works**

The **Internet** is a global network of interconnected computers and devices that communicate using standardized protocols.

**Key Components of the Internet:**

1. **IP Addresses (Internet Protocol Address)**
   * Every device on the internet has a unique IP address (e.g., 192.168.1.1 or 2001:db8::ff00:42:8329 for IPv6).
   * IP addresses help identify and locate devices on the network.
2. **DNS (Domain Name System)**
   * Converts human-readable domain names (e.g., www.google.com) into IP addresses.
   * Works like a phonebook for the internet.
3. **ISP (Internet Service Provider)**
   * Companies like **AT&T, Comcast, Jio, or Airtel** provide internet access to users.
   * ISPs connect homes, businesses, and data centers to the internet.
4. **Routers & Switches**
   * **Routers** direct internet traffic between networks.
   * **Switches** manage communication within local networks.
5. **Data Transmission & Protocols**
   * The **TCP/IP (Transmission Control Protocol/Internet Protocol)** ensures data is sent and received accurately.
   * **Packets**: Data is broken into small chunks called packets, which travel across networks and are reassembled at the destination.
6. **Servers & Clients**
   * **Servers** host websites, applications, and data.
   * **Clients** (such as smartphones or computers) request information from servers.
7. **How a Website Loads:**
   * You type a URL into a browser.
   * The browser contacts a **DNS server** to find the IP address of the website.
   * The request is sent to the **web server** hosting the website.
   * The web server processes the request and sends back the web page.
   * The browser displays the page for the user.

**THEORY EXERCISE**: Describe the roles of the client and server in web communication.

Answer: In web communication, **clients** and **servers** play crucial roles in transmitting and processing data over the internet. They follow a **request-response model**, where the client sends requests, and the server processes and responds to them.

**1. Client Role**

A **client** is any device (computer, smartphone, tablet) that initiates a request to access a web service or website. The client typically runs a **web browser** (e.g., Chrome, Firefox, Edge) or an application that interacts with web servers.

**Functions of the Client:**

* **Initiates a Request**: The client sends an HTTP/HTTPS request to a web server when a user enters a URL or clicks a link.
* **Receives Data**: The client receives the response from the server, usually in the form of HTML, CSS, JavaScript, or JSON data.
* **Processes and Renders Content**: The web browser processes the received data and displays the website to the user.
* **Stores Cookies and Cache**: Some data is stored locally for faster access and improved user experience.

**Example of Client Activity:**

* A user types www.example.com in a browser.
* The browser contacts a **DNS server** to get the website’s IP address.
* It then sends an **HTTP GET request** to the web server.
* The server responds with the requested webpage, which the browser displays.

**2. Server Role**

A **server** is a remote computer or system that hosts websites, applications, and databases. It listens for incoming requests from clients and responds with the required data.

**Functions of the Server:**

* **Processes Client Requests**: The server receives HTTP/HTTPS requests and determines how to handle them.
* **Retrieves Data from a Database**: If the request requires dynamic data, the server fetches it from a database (e.g., MySQL, PostgreSQL).
* **Executes Business Logic**: In dynamic applications, the server may process user input and perform tasks like authentication or data updates.
* **Sends a Response**: The server sends the requested data (HTML pages, JSON API responses, images, etc.) back to the client.

**Example of Server Activity:**

* A user requests a login page.
* The server processes the request and fetches the login form.
* If the user submits credentials, the server validates them against a database.
* It sends back a response indicating whether the login was successful.

Question: Network Layers on Client and Server

Answer: In web communication, both the **client** and **server** communicate using the **OSI (Open Systems Interconnection) Model** or the **TCP/IP Model**. These models define how data is transmitted over a network in a structured manner.

1. OSI Model and Its Layers

| **Layer No.** | **Layer Name** | **Client Role** | **Server Role** |
| --- | --- | --- | --- |
| **7** | **Application Layer** | Sends HTTP/HTTPS requests via a web browser | Responds to requests with web pages or data |
| **6** | **Presentation Layer** | Encrypts/decrypts data (e.g., SSL/TLS) | Encrypts/decrypts data before transmission |
| **5** | **Session Layer** | Manages session state (e.g., cookies, authentication) | Maintains session for multiple client requests |
| **4** | **Transport Layer** | Uses TCP/UDP to ensure reliable or fast data transfer | Uses TCP/UDP to manage data segmentation and delivery |
| **3** | **Network Layer** | Assigns an IP address and routes packets to the destination | Routes packets back to the client using IP addresses |
| **2** | **Data Link Layer** | Converts data into frames and sends it over a physical network | Receives frames, processes them, and forwards them |
| **1** | **Physical Layer** | Sends data as electrical signals or wireless signals | Receives signals and converts them into digital data |

2. TCP/IP Model and Its Layers

| **TCP/IP Layer** | **Client Role** | **Server Role** |
| --- | --- | --- |
| **Application Layer** | Sends HTTP/HTTPS requests using a browser or app | Responds with website content or API data |
| **Transport Layer** | Uses TCP for reliable transmission or UDP for speed | Ensures proper data delivery using TCP or UDP |
| **Internet Layer** | Assigns an IP address and routes packets | Routes response packets back to the client |
| **Network Access Layer** | Converts data into signals for transmission | Converts received signals into data |

**How Data Travels Through Layers (Example: Opening a Website)**

1. **Client (Browser) Sends a Request**
   * The request starts at the **Application Layer** (e.g., HTTP request for www.example.com).
   * It passes through **Transport, Network, and Data Link Layers**, and is transmitted over the **Physical Layer**.
2. **Server Receives and Processes the Request**
   * The request moves from the **Physical Layer** up to the **Application Layer**.
   * The web server processes the request and sends a response.
3. **Client Receives the Response**
   * The response travels through the **same layers in reverse order**.
   * The **browser displays the web page** for the user.

**Application Software**

**Q: Theory Exercise: What is the role of application software in businesses?**

Application software helps businesses perform specific tasks efficiently. It automates processes, manages data, and improves productivity. Examples include:

**· MS Office** (for document creation and data analysis)

**· CRM software** (for customer management)

**· Accounting software** (for financial management)

**Software Development Process**

**Q: What are the main stages of the software development process?**

**A:** The main stages are:

**1. Requirement Analysis** – Identifying user needs.

**2. Planning** – Defining scope, cost, and schedule.

**3. Design** – Creating system architecture.

**4. Implementation (Coding)** – Writing the actual software.

**5. Testing** – Checking for bugs and errors.

**6. Deployment** – Releasing the software.

**7. Maintenance** – Updating and fixing issues.

**Software Requirement**

**Q: Why is the requirement analysis phase critical in software development?**

**A:** It ensures that the software meets user needs, prevents miscommunication, reduces development costs, and minimizes future changes by defining features and constraints clearly.

**Software Analysis**

**Q: What is the role of software analysis in the development process?**

**A:** It helps developers understand system requirements, identify potential issues, and create functional specifications. This ensures the software meets business and user needs efficiently.

**System Design**

**Q: What are the key elements of system design?**

**A:** The key elements include:

**· Architecture Design** – Defining the system structure.

**· User Interface (UI) Design** – Ensuring usability and accessibility.

**· Database Design** – Organizing data efficiently.

**· Security Design** – Implementing safety measures.

**· Integration Design** – Ensuring compatibility with other systems.

**Software Testing**

**Q: Why is software testing important?**

**A:** Software testing ensures that the application is free of bugs, meets requirements, and functions correctly. It improves security, performance, and user experience.

**Maintenance**

**Q: What types of software maintenance are there?**

**A: Types of software maintenance are :**

**1. Corrective Maintenance** – Fixing bugs and errors.

**2. Adaptive Maintenance** – Modifying software to work in a new environment.

**3. Perfective Maintenance** – Improving performance and adding new features.

**4. Preventive Maintenance** – Making updates to prevent future issues.

**Development**

**Q: What are the key differences between web and desktop applications?**

**A: The key differences between web and desktop applications are :**

| **Feature** | **Web Application** | **Desktop Application** |
| --- | --- | --- |
| Access | Runs on a browser | Installed on a computer |
| Updates | Updated automatically | Requires manual updates |
| Internet | Requires an internet connection | Works offline |
| Performance | Depends on the internet speed | Faster on powerful hardware |

**Web Application**

**Q: What are the advantages of using web applications over desktop applications?**

**A:**

**· Accessibility** – Can be used from any device with a browser.

**· No Installation Required** – Runs online without installation.

**· Automatic Updates** – Updates are managed by the provider.

**· Cross-Platform Compatibility** – Works on different operating systems.

**Designing**

**Q: What role does UI/UX design play in application development?**

**A:** UI/UX design improves user experience by making the application visually appealing, easy to navigate, and efficient. Good design increases user engagement and satisfaction.

**Mobile Application**

**Q: What are the differences between native and hybrid mobile apps?**

**A: The differences between native and hybrid mobile apps are :**

| **Feature** | **Native App** | **Hybrid App** |
| --- | --- | --- |
| Development | Built for a specific platform (Android/iOS) | Uses web technologies to run on multiple platforms |
| Performance | Faster and more efficient | Slightly slower due to web view |
| User Experience | Better UI/UX with platform-specific design | Consistent but less optimized |
| Development Cost | Higher | Lower |

**Data Flow Diagram (DFD)**

**Q: What is the significance of DFDs in system analysis?**

**A:** DFDs help in understanding how data moves within a system. They visualize data input, processing, and output, making it easier to identify inefficiencies, redundancies, and potential improvements in system design.

**Desktop Application**

**Q: What are the pros and cons of desktop applications compared to web applications?**

**A:**

| **Aspect** | **Desktop Application** | **Web Application** |
| --- | --- | --- |
| **Performance** | Faster, uses system resources directly | Slower, depends on internet speed |
| **Internet Dependency** | Works offline | Requires internet access |
| **Installation** | Needs to be installed on a device | Runs on a browser, no installation needed |
| **Updates** | Manual updates required | Automatic updates from the server |
| **Security** | More secure if offline | Prone to online threats |
| **Accessibility** | Limited to installed devices | Accessible from any device with a browser |