**Module 1 – SE -Overview of IT Industry:**

**Question 1:** **What is a Program?**

**Ans:**

A **program** is a set of **instructions** written in a programming language that a computer can understand and execute to perform a specific task or solve a problem.

**Key Points:**

* A program **tells the computer what to do** and **how to do it**.
* It is written using programming languages like **Python, C, Java, C++, JavaScript**, etc.
* Programs are created by **programmers** or **developers**.
* Programs can be **simple**, like a calculator, or **complex**, like a video game or operating system.

**Question 2: Explain in your own words what a program is and how it functions.**

**Ans:**

A **program** is a written set of instructions that tells a computer exactly what to do, step by step. Just like you follow steps in a recipe to cook a dish, a computer follows the steps in a program to complete a task.

**How It Works:**

1. **Writing the Program:**  
   A person (called a programmer) writes instructions using a special language the computer understands — like Python, C, or Java.
2. **Running the Program:**  
   The computer reads these instructions **line by line**, and performs each action exactly as written.
3. **Output:**  
   Based on the instructions, the computer shows the result — like printing a message, saving a file, or playing a sound.

**Question 3: What is Programming?**

**Ans:**

**Programming** is the process of **creating instructions** that a computer can follow to perform specific tasks.

**In Simple Words:**

Programming means **telling the computer what to do**, step by step, using a language it understands (called a **programming language**).

**What Happens in Programming?**

1. **Problem Solving:**  
   First, you think about **what you want the computer to do** — like making a calculator, a game, or a website.
2. **Writing Code:**  
   Then, you **write the steps** (called **code**) using a programming language like **Python, Java, C++, JavaScript**, etc.
3. **Testing and Debugging:**  
   You **run** the code to see if it works. If there are mistakes (called **bugs**), you fix them.
4. **Final Result:**  
   Once everything works, your program is ready to use!

**Question 4:** **What are the key steps involved in the programming process?**

**Ans:**

The **key steps in the programming process** are a series of logical stages that help create an efficient and working program. Here's a breakdown in simple terms:

**1. Understanding the Problem**

* **What it means:** Clearly define what you want the program to do.
* **Example:** You want to create a calculator that adds two numbers.

**2. Planning the Solution**

* **What it means:** Decide how the program will work, step by step.
* **Tools used:** Flowcharts, pseudocode, or simple outlines.
* **Example:**
  1. Ask the user for two numbers
  2. Add them
  3. Show the result

**3. Writing the Code**

* **What it means:** Translate your plan into a programming language (like Python, C, or Java).

**4. Testing the Program**

* **What it means:** Run the program with different inputs to make sure it works correctly.
* **Goal:** Find and fix errors (bugs) in logic or syntax.

**5. Debugging**

* **What it means:** Identify and correct mistakes in the code.
* **Example:** Fixing a typo, wrong formula, or crash.

**6. Finalizing the Program**

* **What it means:** Clean up the code, add comments, and make it user-friendly and efficient.

**7. Deploying and Maintaining**

* **Deploying:** Making the program available to users.
* **Maintaining:** Updating the program if problems come up or if new features are needed.

**Question 5: Types of Programming Languages.**

**Ans:**

Programming languages can be classified in several ways depending on their **level**, **purpose**, and **paradigm**. Below are the most common types:

1. **Based on Level of Abstraction:**

**High-Level Languages:**

* **Easy to understand** and close to human language.
* Used for writing most software today.
* Example: Python, Java, C++, JavaScript

**Low-Level Languages:**

* **Closer to machine code**, harder to read.
* Used for system-level programming (like operating systems).
* Example: Assembly language, Machine code

1. **Based on Programming Paradigm:**

**Procedural Programming Languages:**

* Code is written as a **sequence of steps (procedures/functions)**.
* Example: C, Pascal, BASIC

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

* Based on **objects and classes**; helps manage complex programs.
* Example: Java, Python, C++, C#

**Functional Programming Languages:**

* Focuses on using **functions** and avoiding changing states.
* Example: Haskell, Lisp, Scala

**Scripting Languages:**

* Used for writing **scripts** to automate tasks.
* Example: JavaScript, Python, Bash, PHP

**3. Based on Usage / Purpose**

| **Type** | **Description** | **Examples** |
| --- | --- | --- |
| **General-purpose** | Used for a wide range of applications | Python, Java, C++ |
| **Web Development** | Mainly for building websites | HTML, CSS, JavaScript, PHP |
| **System Programming** | For OS, device drivers, etc. | C, C++, Rust |
| **Scientific/Math** | For calculations, simulations | MATLAB, R, Python |
| **Database Languages** | For managing databases | SQL |

**Question 6:** **What are the main differences between high-level and low-level programming languages?**

**Ans:**

**Main Differences Between High-Level and Low-Level Programming Languages:**

| **Feature** | **High-Level Language** | **Low-Level Language** |
| --- | --- | --- |
| **Abstraction Level** | High – closer to human language | Low – closer to machine language |
| **Ease of Use** | Easy to write, read, and understand | Difficult to write and understand |
| **Examples** | Python, Java, C++, JavaScript | Assembly, Machine Code |
| **Portability** | Highly portable across platforms | Hardware-dependent (not portable) |
| **Execution Speed** | Slower (requires compiler/interpreter) | Faster (direct hardware interaction) |
| **Control Over Hardware** | Less control over memory and CPU | Full control over hardware and system resources |
| **Debugging and Maintenance** | Easier to debug and maintain | Harder to debug and error-prone |
| **Compilation/Translation** | Needs a **compiler** or **interpreter** | Often written for a specific **processor architecture** |

**Question 7:** **World Wide Web & How Internet Works.**

**Ans:**

The **World Wide Web (WWW)** is a **collection of websites and web pages** that you can access through the **Internet** using a browser (like Chrome, Firefox, etc.).

**Key Points:**

* It was invented by **Tim Berners-Lee** in **1989**.
* It uses **web technologies** like **HTML, CSS, JavaScript**, and **URLs**.
* It works on a system of **hyperlinks** that connect web pages.
* It runs on the **Internet**, but **the Web is not the same as the Internet**.

**What is the Internet?**

The **Internet** is the **global network** of **connected computers, servers, and devices** that allows data to travel from one place to another.

**Key Features:**

* It includes **emails, websites, online games, video calls**, etc.
* Uses technologies like **IP (Internet Protocol)** and **TCP (Transmission Control Protocol)**.
* It’s the **infrastructure** that carries all digital communication — including the Web.

**How Does the Internet Work?**

**1. You enter a URL in your browser**

* Example: www.youtube.com

**2. DNS (Domain Name System) translates it**

* Converts the website name into an **IP address** (like 142.250.183.78)

**3. Your computer sends a request**

* This request travels through your **Internet Service Provider (ISP)** to the correct **server**.

**4. Server responds**

* The server where the website is stored **sends data back** to your browser.

**5. Your browser displays the webpage**

* You now see text, images, and videos — the content of the site.

**Question 8:** **Describe the roles of the client and server in web communication.**

**Ans:**

**Roles of Client and Server in Web Communication**

In web communication, the **client** and **server** work together to allow users to access websites, apps, or online services. Here's how each one plays its role:

**Client:**

**What is it?**

The **client** is the device (like your computer, phone, or tablet) and software (usually a **web browser**) that **requests** data from a server.

**Role:**

* Sends **requests** to the server (e.g., asking for a web page).
* Displays the **response** (like a website) received from the server.
* Interacts with users (you!) through interfaces like Chrome, Firefox, or Edge.

**Example:**

* You type www.example.com into your browser → your browser sends a request to get the homepage → waits for server reply.

**Server:**

**What is it?**

The server is a powerful computer or program that stores websites, data, or services, and responds to client requests.

**Role:**

* **Receives requests** from the client.
* **Processes** the request (e.g., fetch data, run code).
* **Sends back** the correct data (HTML, images, videos, etc.).

**Example:**

* When your request reaches the server hosting example.com, it finds the homepage file and sends it back to your browser.

**How They Communicate (Step-by-Step)**

1. **Client** (browser) → sends a **request** (HTTP/HTTPS)
2. **Server** → receives the request, processes it
3. **Server** → sends a **response** (website content)
4. **Client** → displays the content to the user

**Question 9:** **Network Layers on Client and Server.**

**Ans:**

**Network Layers on Client and Server:**

The **client** and **server** communicate over the Internet using a layered structure called the **TCP/IP model** or **OSI model**. Each layer handles a specific part of data communication, ensuring the message goes smoothly from client to server and back.

**TCP/IP Model (Used in Real Internet Communication):**

**Layers on Both Client and Server:**

| Layer (Top to Bottom) | Function |
| --- | --- |
| 4. Application Layer | Interface for applications (e.g., browser, web server). Uses protocols like HTTP, HTTPS, FTP |
| 3. Transport Layer | Breaks data into segments; ensures reliable delivery (uses TCP/UDP) |
| 2. Internet Layer | Adds IP addresses and handles routing the data over the Internet |
| 1. Network Access Layer | Converts data to electrical/optical signals, handles physical delivery over cables, Wi-Fi, etc. |

**Example: What Happens When You Open a Website:**

**Client Side (Browser):**

1. **Application Layer** – Browser sends an HTTP request:  
   GET /index.html HTTP/1.1
2. **Transport Layer** – Breaks request into packets using **TCP**, adds port number.
3. **Internet Layer** – Adds IP address of the server.
4. **Network Access Layer** – Sends the request as signals over Wi-Fi/Ethernet.

**Server Side:**

1. **Network Access Layer** – Receives data signal.
2. **Internet Layer** – Reads the client's IP address to reply correctly.
3. **Transport Layer** – Reassembles TCP segments into a complete request.
4. **Application Layer** – Web server reads the HTTP request, fetches the file, and sends it back.

**Question 10:** **Explain the function of the TCP/IP model and its layers.**

**Ans:**

The **TCP/IP model** (Transmission Control Protocol / Internet Protocol) is a set of rules that govern how data travels across the Internet. It helps devices **communicate** with each other reliably and accurately.

**4 Layers of the TCP/IP Model:**

Each layer has a specific job and works together with the others:

**1. Application Layer (Top Layer):**

**What it does:**

* It’s where the user interacts — your **web browser, email app, or file transfer tool** lives here.
* Defines **how applications use the network**.

**Protocols:**

* HTTP (websites), HTTPS (secure websites), FTP (file transfer), SMTP (email), DNS (domain name system)

**Example:** You open a browser and go to <www.google.com>

**2. Transport Layer:**

**What it does:**

* Breaks large data into **smaller pieces (segments)**.
* Ensures **reliable delivery** (using **TCP**) or faster but unreliable delivery (using **UDP**).
* Adds **port numbers** to identify which app to deliver the data to.

**Protocols:**

* TCP (Transmission Control Protocol): Reliable
* UDP (User Datagram Protocol): Fast but no error checking

**Example:** Makes sure all pieces of a webpage arrive correctly and in order.

**3. Internet Layer:**

**What it does:**

* **Routes the data** between networks using **IP addresses**.
* Adds **source and destination IP** info to each packet.

**Protocols:**

* IP (Internet Protocol), ICMP (for error messages), ARP (address resolution)

**Example:** Decides how to get your message from your city to a server across the world.

**4. Network Access Layer (Also called Link Layer):**

**What it does:**

* Moves data **physically** through wires, Wi-Fi, etc.
* Works with **MAC addresses** and actual hardware (network cards, switches).

**Technologies:**

* Ethernet, Wi-Fi, DSL, Fiber optics

**Example:** Your laptop sends signals through a Wi-Fi router to reach the internet.

**Question 11:** **Types of Internet Connections.**

**Ans:**

There are several ways to connect to the Internet, each with different speeds, costs, and technologies. Here are the common types:

1. **Dial-Up Connection.**

Dial-up is one of the earliest types of Internet connections that uses a **standard telephone line** to connect your computer to the Internet.

2. **Broadband Connection.**

Broadband is a **high-speed internet connection** that is **always on** — meaning you don’t have to dial up each time to connect. It offers much faster speeds than dial-up and supports activities like streaming, gaming, video calls, and large downloads.

3. **Satellite Internet.**

Satellite Internet is a type of internet connection that uses **communication satellites orbiting the Earth** to provide internet access, especially in areas where traditional wired connections (like DSL or cable) are unavailable.

4. **Wireless Internet.**

Wireless Internet refers to connecting to the internet **without using physical cables**. Instead, it uses **radio waves** or other wireless technologies to send and receive data.

**5. Leased Line.**

A **leased line** is a **dedicated, private, and fixed-bandwidth** internet or data connection between two locations. It is rented from an Internet Service Provider (ISP) or telecom company, offering a permanent, exclusive link that is **not shared** with others.

**6. Broadband over Power Lines (BPL).**

Broadband over Power Lines (BPL) is a technology that delivers **internet data over existing electrical power lines**. It uses the electrical grid infrastructure to provide broadband internet access without needing new cables.

**Question 12: : How does broadband differ from fiber-optic internet?**

**Ans:**

**Broadband vs Fiber-Optic Internet:**

| **Aspect** | **Broadband (General)** | **Fiber-Optic Internet** |
| --- | --- | --- |
| **Definition** | A broad term for high-speed internet connections like DSL, cable, satellite, etc. | A type of broadband that uses fiber-optic cables to transmit data as light signals. |
| **Medium Used** | Can use copper wires (DSL), coaxial cables (cable), satellites, or wireless. | Uses thin strands of glass or plastic fibers that carry data using light. |
| **Speed** | Varies widely depending on type; generally up to hundreds of Mbps. | Extremely high speeds, often from 100 Mbps up to multiple Gbps. |
| **Reliability** | Can be affected by distance, weather, or congestion (especially DSL, cable, satellite). | Highly reliable and stable, less affected by distance or interference. |
| **Latency** | Moderate to high latency depending on technology (satellite is highest). | Very low latency — ideal for gaming, video calls, and real-time apps. |
| **Availability** | Widely available, especially DSL and cable. | Limited to areas with fiber infrastructure but rapidly expanding. |
| **Cost** | Usually more affordable; varies by provider and type. | Often more expensive initially but provides better long-term value. |
| **Symmetry** | Usually asymmetric (download speed higher than upload speed). | Often symmetric (equal upload and download speeds). |

**Question 13: What is** **Protocols.**

**Ans:**

A **protocol** is a set of **rules and standards** that define how computers and devices communicate with each other over a network.

**Why Protocols are Important:**

* They ensure that data sent from one device is understood by another.
* They define how data is formatted, transmitted, received, and processed.
* Without protocols, devices wouldn’t know how to “talk” to each other.

**Example:**

* **HTTP (HyperText Transfer Protocol):** Rules for transferring web pages on the internet.
* **TCP/IP (Transmission Control Protocol/Internet Protocol):** Core protocols that govern how data is sent across the internet.
* **FTP (File Transfer Protocol):** Rules for transferring files between computers.

**Question 14:** **What are the differences between HTTP and HTTPS protocols?**

**Ans:**

**Difference Between HTTP and HTTPS:**

| **Feature** | **HTTP** | **HTTPS** |
| --- | --- | --- |
| **Full Form** | HyperText Transfer Protocol | HyperText Transfer Protocol Secure |
| **Security** | ❌ **Not secure** – data is sent as plain text | ✅ **Secure** – data is encrypted using SSL/TLS |
| **Encryption** | No encryption | Encrypts data to protect it from hackers |
| **Port Number** | Uses **port 80** | Uses **port 443** |
| **URL Format** | Starts with http:// | Starts with https:// |
| **SSL Certificate** | Not required | **Required** – verifies the identity of the website |
| **Data Protection** | Vulnerable to eavesdropping and attacks | Protects data from tampering or interception |
| **Browser Indicator** | No padlock icon in browser | Shows 🔒 padlock icon in browser address bar |

**Question 15: What is Application Security:**

**Ans:**