Module 1

OVERVIEW OF IT INDUSTRY

1. What is a Program?

- A program is a set of instructions written in a programming language that a computer can understand and follow to perform specific tasks. It helps automate processes, solve problems, or manage data efficiently.

2. What is Programming?

- Programming is the process of designing and writing these instructions (called code) using programming languages. It involves creating logic, solving problems, and instructing the computer on how to handle tasks step by step.

3. Key Steps in Programming Process:

- Problem Understanding: Understand what needs to be done.
- Planning: Create a step-by-step plan or flowchart.
- Coding: Write the actual code using a programming language.
- Testing: Run the program and find any errors or bugs.
- Debugging: Fix the errors found during testing.
- Deployment: Make the program available for use.
- Maintenance: Update and fix the program as needed over time.

4. Types of Programming Languages:

- **High-level Languages:** Easy to read, understand, and use (e.g., Python, Java, C++). They are close to human language.
- **Low-level Languages:** Difficult to read but give more control over hardware (e.g., Assembly language, Machine code). They are close to machine language.

5. World Wide Web & How Internet Works:

- **The World Wide Web is a system of internet servers that support documents formatted in HTML. These documents are linked using hyperlinks and accessed using the internet. A client (like a browser) sends a request to the server (which hosts the website), and the server sends back the requested information or webpage.**

6. TCP/IP Model and Its Layers:

- TCP/IP (Transmission Control Protocol/Internet Protocol) is a set of rules for data communication over the internet. It has four layers:
- **Application Layer:** Interacts with software applications (e.g., browsers).
- **Transport Layer:** Ensures reliable data transfer (e.g., TCP).
- **Internet Layer:** Handles the addressing and routing of data (e.g., IP).
- **Network Access Layer:** Manages the physical transmission of data.

7. Client and Server Communication:

- **In this model, the client sends requests (like opening a webpage), and the server responds by sending the required data. This process is the basis of web browsing and online services.**

8. Broadband vs Fiber-Optic Internet:

- **Broadband:** High-speed internet over cable, DSL, or wireless networks. Common and widely available.
- **Fiber-Optic:** Uses light to transmit data through glass fibers. It is much faster, more reliable, and supports higher bandwidth.

9. HTTP vs HTTPS:

- HTTP (HyperText Transfer Protocol): Used for transferring web pages but not secure.
- HTTPS (HTTP Secure): A secure version of HTTP. It uses encryption (SSL/TLS) to protect data during transmission.

10. Role of Encryption in Applications:

- Encryption secures data by converting it into a coded format. Only authorized users can decode it. It is used in secure websites, apps, banking systems, and messaging apps to prevent unauthorized access.

11. System Software vs Application Software:

- **System Software:** Controls and manages computer hardware (e.g., Operating System, Drivers).
- **Application Software:** Helps users do tasks (e.g., MS Word, Photoshop, Browsers).

12. Modularity in Software Architecture:

- **Modularity means dividing software into smaller parts (modules). Each module performs a specific function. This helps in easier testing, debugging, and future upgrades.**

13. Layers in Software Architecture:

- **Software is often divided into layers such as presentation (UI), business logic, and data layers. This separation makes it easier to develop, test, and manage large applications.**

14. Software Environments:

- A software development environment includes tools like code editors (VS Code), compilers, and debuggers. It provides everything needed to build and test software efficiently.

15. Source Code vs Machine Code:

- **Source Code:** Human-readable instructions written in a programming language.
- **Machine Code:** Binary code that the computer understands directly. Source code is converted to machine code using a compiler or interpreter.

16. Version Control:

- Version control systems like Git help track changes in code, revert to previous versions, and collaborate with team members efficiently.

17. GitHub for Students:

- GitHub allows students to store projects, track progress, and collaborate with others. It also helps in creating an online portfolio for showcasing skills

18. Open Source vs Proprietary Software:

- **Open Source:** Code is publicly available. Users can modify and distribute it (e.g., Linux).
- **Proprietary:** Owned by a company, and users need permission or license to use (e.g., Microsoft Office).

19. Git for Collaboration:

- **Git allows multiple developers to work on the same project, merge their work, and manage versions without overwriting each other’s code.**

20. Application Software in Businesses:

- **Application software helps businesses in activities like accounting, inventory management, communication, and data analysis (e.g., Tally, Excel).**

21. Software Development Process:

- Requirement Gathering
- System Design
- Development
- Testing
- Deployment
- Maintenance

22. Requirement Analysis:

- In this phase, developers understand what the users need. It helps avoid errors and ensures the software does what the user expects.

23. Software Analysis:

- Analyzing software helps in designing better solutions by understanding the system's behavior, limitations, and requirements.

24. System Design:

- Key elements:
- User Interface Design
- Data Design
- Process Design
- System Architecture

25. Software Testing:

- Testing ensures the software is working as expected. It finds bugs, checks performance, and improves software quality.

26. Types of Software Maintenance:

- **Corrective:** Fixing bugs
- **Adaptive:** Updating to new systems
- **Perfective:** Improving performance
- **Preventive:** Avoiding future problems

27. Web vs Desktop Applications:

- **Web Apps:** Run in browsers, accessible anywhere, require internet.
- **Desktop Apps:** Installed on computers, work offline, may be faster.

28. Advantages of Web Applications:

- Accessible from any device
- Easy to update
- No installation required

29. UI/UX Design in Development:

- UI (User Interface) and UX (User Experience) focus on how an application looks and feels. Good design makes the app user-friendly and attractive.

30. Native vs Hybrid Mobile Apps:

- **Native Apps:** Made for a specific platform (e.g., Android or iOS). Better performance.
- **Hybrid Apps:** Work on multiple platforms with one codebase. Easier and cheaper to develop.

31. Importance of DFD (Data Flow Diagram):

- **DFDs show how data moves through a system. They help understand, analyze, and design the system clearly.**

32. Desktop Application Pros and Cons:

- **Pros:**
- Works offline
- Better performance
  Cons:
- Requires installation
- Difficult to update

33. Flowcharts in Programming:

- Flowcharts help in planning the logic of a program using diagrams. They make it easier to understand and debug programs.