Modul-1 : Overview of IT Industry

What is a Program? Explain in your own words what a program is and how it functions. What is Programming?

→A **program** is a set of **instructions** written in a programming language that a computer can execute to perform a specific task. It functions by telling the computer what to do and how to do it, step by step. A program can be simple like printing text or complex like running a video game or a web browser.

How Does a Program Function?

1. **Written in Code**: A human writes the instructions in a programming language (like Python, C, Java, etc.).
2. **Translated for the Computer**:
   - If it's a **compiled language** (like C), the code is turned into machine code **before** it runs.
   - If it's an **interpreted language** (like Python), the code is translated **as it runs**.
3. **Executed by the Computer**: The CPU follows the instructions, one by one, and performs the tasks (like showing text, doing math, or storing information).

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

→**Programming** is the process of designing and creating software by writing code in a programming language. It involves problem-solving, logic, and creativity to translate ideas into functioning applications.

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

Feature	High-Level Language	Low-Level Language
Abstraction	**High (closer to human language)**	**Low (closer to machine hardware)**

Feature	High-Level Language	Low-Level Language
Readability	Easy for humans to read and write	Hard to read — uses symbols or binary codes
Speed/Performance	Slower (due to abstraction and interpretation)	Faster (direct control over hardware)
Hardware Access	Limited (managed by language and OS)	Direct (can interact with memory and CPU)
Portability	Portable across systems (write once, run anywhere)	Not portable — specific to hardware architecture
Examples	Python, Java, C++, JavaScript	Assembly, Machine Code
Ease of Development	Easier — fewer lines of code, built-in libraries	Difficult — more lines, manual memory management
Used By	Most developers (software, apps, web, etc.)	Systems programmers (OS, drivers, embedded)

World Wide Web & How Internet Works

What is the World Wide Web (WWW)?

The World Wide Web (WWW) is a system of linked documents and resources accessed via the Internet. It's what you use when you open a web browser like Chrome, Firefox, or Safari and go to a website (e.g., www.google.com).

Key Points:

- Invented by Tim Berners-Lee in 1989.
- Made up of web pages, which are written in HTML.

- These pages are linked by hyperlinks.
- Accessed through web browsers.

The Web is not the same as the Internet — it runs *on* the Internet.

What is the Internet?

The Internet is a global network of computers that are all connected and can communicate with each other.

Think of it as:

- The infrastructure (like roads, power lines).
- The physical and logical network that carries data (like emails, websites, videos).

How the Internet Works – Step by Step

1. Your Device Connects

- You connect your phone or computer to the Internet via Wi-Fi, ethernet, or mobile data.

2. You Request a Website

- You type a URL like www.example.com into your browser.
- Your browser sends a request to find that site.

3. DNS Resolves the Domain

- The Domain Name System (DNS) translates the domain name (e.g., www.google.com) into an IP address (like 142.250.190.68), which computers use to identify each other.

4. The Request Travels Across the Internet

- The request goes through routers and servers to reach the web server hosting the site.

5. The Web Server Responds

- The server sends back the requested web page (HTML, images, etc.).

6. Your Browser Displays the Page

- The browser takes the code it received and renders it into the web page you see.

Network Layers on Client and Server

TCP/IP Model Layers

Layer	Function
Application	Handles high-level protocols (HTTP, FTP, etc.)
Transport	Reliable data delivery (TCP/UDP)
Internet	IP addressing and routing (IP)
Network Access	Physical transmission of data (Ethernet, Wi-Fi)

Protocols

→

HTTP vs HTTPS

Feature	HTTP	HTTPS
Security	Unencrypted	Encrypted using SSL/TLS
Port	80	443
Data Integrity	Vulnerable	Secure

Feature	HTTP	HTTPS
Use Case	Internal/testing	Public websites, secure transactions

Application Security

Role of Encryption in Securing Applications

Encryption protects sensitive data by converting it into a coded format that is unreadable without a secret key. Its role in applications includes:

- **Data Confidentiality** – Ensures only authorized users can access the data.
- **Data Integrity** – Prevents data tampering during transmission.
- **Secure Communication** – Encrypts data sent over networks (e.g., HTTPS).
- **User Authentication** – Used in password storage and verification.

Software Applications and Its Types.

❖ System vs Application Software

Feature	System Software	Application Software
Purpose	Manages hardware and system resources	Performs user-specific tasks
Examples	Operating systems, drivers	Word processors, browsers
Runs	In the background	User-initiated
Dependency	Essential for system operation	Depends on system software to run

❖ Software Architecture

Importance of Modularity

- **Reusability** – Modules can be reused across projects.
- **Maintainability** – Easier to debug and update.
- **Scalability** – Supports growth without breaking existing code.
- **Collaboration** – Enables teams to work on different modules independently.

❖ Layers in Software Architecture

Importance of Layers

- **Separation of Concerns** – Each layer has a specific responsibility.
- **Reusability** – Components can be reused independently.
- **Maintainability** – Changes in one layer don't affect others.
- **Testing** – Easier to test components in isolation.

Software Environments

❖ Explore and Set Up Software Environments

Importance of Development Environment

→

- **Isolated Testing** – Prevents bugs in development from affecting users.
- **Debugging Tools** – Access to debuggers and logs.
- **Versioning and Experimentation** – Try new features safely.
- **Dependency Management** – Ensures consistent environment for developers.

❖ Source Code vs Machine Code

Aspect	Source Code	Machine Code
Written In	Human-readable languages (Python, Java)	Binary (0s and 1s)
Editable	Yes	No
Purpose	For developers to write and read	For computers to execute
Example	print("Hi")	10110000 01100001

❖ Why is Version Control Important?

- **Tracks Changes** – Keeps history of every change.

- **Team Collaboration** – Multiple developers can work simultaneously.
- **Backups** – Prevents data loss.
- **Rollbacks** – Revert to previous stable versions when needed.

Application Software

Report on Types and Productivity Benefits

Type	Example	Productivity Benefit
Word Processing	MS Word	Document creation
Spreadsheet	Excel	Data analysis and accounting
Database	MS Access	Efficient data storage and retrieval
Presentation	PowerPoint	Visual communication
Communication	Zoom, Slack	Team collaboration and meetings

❖ Role of Application Software in Businesses

- Streamlines tasks (accounting, project management)
- Manages data efficiently (CRM, ERP)
- Enhances communication (email, chat apps)
- Increases productivity through automation and tools
- Supports decision-making with analytics tools

Software Development Process

Main Stages

1. **Requirement Analysis**
2. **System Design**
3. **Implementation (Coding)**
4. **Testing**
5. **Deployment**
6. **Maintenance**

Software Requirement

❖ Importance of Requirement Analysis

- Clarifies what the system must do
- Prevents scope creep and miscommunication
- Guides the development process
- Acts as the foundation for design and testing

Software Analysis

❖ Role of Software Analysis

→

Identifies system requirements and functionalities

Ensures alignment with business goals

Helps break down the system into manageable components

Provides input for system design

System Design

❖ Key Elements of System Design

- Functional Requirements
- Database Design
- Architecture (Monolithic, Microservices)
- Security Considerations
- Scalability & Performance

Software Testing

❖ Importance of Testing

- Ensures software works as expected
- Identifies bugs early

- Verifies security and performance
- Builds user confidence
- Reduces cost of post-release issues

Maintenance

❖ Types of Software Maintenance

1. **Corrective** – Fixing bugs.
2. **Adaptive** – Updating software for new environments.
3. **Perfective** – Enhancing performance or UI.
4. **Preventive** – Making the system more robust

Development

❖ Web vs Desktop Applications

Feature	Web App	Desktop App
Installation	Not required	Required
Platform	Cross-platform	OS-specific
Updates	Centralized	Manual per user
Internet	Required	May work offline
Speed	Depends on internet	Faster performance

Web Applications

❖ Advantages over Desktop Apps

- Access from anywhere with internet
- No installation required
- Instant updates for all users
- Platform-independent

- Easy integration with other online services

Designing

❖ Importance of Software/UI Design

- Enhances user experience
- Reduces learning curve
- Improves efficiency and productivity
- Ensures accessibility and usability

UI/UX Design

❖ What Role Does UI/UX Design Play in Application Development?

UI (User Interface) design focuses on how an application looks, while **UX (User Experience)** design focuses on how it works and feels. Together, they ensure:

- **Ease of navigation** – Users can interact with the app intuitively.
- **Goal alignment** – Helps users complete tasks efficiently.
- **Clarity & usability** – Minimizes confusion and learning curve.
- **Attractive interfaces** – Encourages engagement.
- **Accessibility** – Ensures everyone can use the application.

Mobile Application

❖ Native vs Hybrid Mobile Apps

Feature	Native Apps	Hybrid Apps
Language	Platform-specific (Swift for iOS, Kotlin for Android)	Web technologies (HTML, CSS, JS)
Performance	High	Moderate

Feature	Native Apps	Hybrid Apps
Access to device features	Full	Limited (depends on plugins)
Maintenance	Separate codebases for each OS	One codebase for all platforms
Development time	Longer	Faster
Example	Instagram (native), WhatsApp	Instagram (partially hybrid), Uber (hybrid elements)

DFD (Data Flow Diagram)

❖ Significance of DFDs in System Analysis

- **Visualize data movement** between processes, users, and systems
- **Clarify system functionality** and interactions
- **Helps in identifying redundant or missing processes**
- **Improves communication** among stakeholders

Desktop Application

❖ Pros and Cons of Desktop Apps vs Web Apps

Feature	Desktop Apps	Web Apps
Performance	Faster (uses local resources)	Slower (depends on internet)
Installation	Required	No installation
Portability	OS-dependent	Accessible from any device
Updates	Manual	Automatic
Offline Use	Fully available	Limited or none
Security	Local data storage	Data stored in cloud, may be more vulnerable

Flow Chart

❖ Importance of Flowcharts

- Clarifies logic before coding
- Identifies errors early in planning
- Helps communicate system flow with others
- Improves algorithm design
- Breaks complex problems into smaller steps