**What is a Program?**

* A program is a set of instructions that a computer follows to do a specific task.

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

* A program is a set of instructions written by a person to tell the computer what to do.
* These instructions are written using a programming language like Python, Java, or C.
* The computer reads these instructions step-by-step and performs actions like adding numbers, showing messages, or opening files.
* The main goal of a program is to solve a problem or complete a task – for example, a calculator app, a game, or a website.

**What is Programming?**

* Programming is the process of writing instructions that a computer can understand and follow to perform specific tasks. These instructions are written using programming languages like Python, Java, or C++.

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

1. **Understand the Problem**  
   → Know what needs to be done.
2. **Plan the Solution**  
   → Write steps or use a flowchart.
3. **Write the Code**  
   → Use a programming language to type your solution.
4. **Test the Code**  
   → Run it and fix any bugs.
5. **Debug if Needed**  
   → Correct errors or unexpected behavior.
6. **Maintain the Code**  
   → Update or improve it over time.

**Types of Programming Languages**

1. Procedural Programming Languages
2. Object-Oriented Programming Languages
3. Logical Programming Languages
4. Functional Programming Languages

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

**1. Abstraction**

* **High-Level**: More abstract, closer to human language (e.g., Python, Java).
* **Low-Level**: Closer to machine language (e.g., Assembly, C).

**2. Ease of Use**

* **High-Level**: Easier to read, write, and understand.
* **Low-Level**: Harder to read and write, requires understanding of hardware.

**3. Control over Hardware**

* **High-Level**: Less control over hardware.
* **Low-Level**: More control over hardware and memory.

**4. Portability**

* **High-Level**: Code is more portable across different machines.
* **Low-Level**: Code is more machine-specific.

**5. Execution Speed**

* **High-Level**: Generally slower due to abstraction.
* **Low-Level**: Faster execution since it’s closer to the machine.

**World Wide Web & How Internet Works**

* The World Wide Web (WWW) is a vast system of interconnected documents and other resources that are linked by hyperlinks and URLs (Uniform Resource Locators). It is a platform that allows access to various content like text, images, videos, and interactive features over the internet.

**How the Internet Works**

The internet is a global network of interconnected computers and servers. It enables data exchange and communication across the world. Here's a simplified breakdown of how it works:

1. **Client-Server Model**: The internet relies on a client-server model, where the client (your device) requests data from the server (where the data is stored). The server processes the request and sends back the necessary data to the client.
2. **Data Transmission**: When you send a request (like opening a webpage), your device communicates with a server using the Internet Protocol (IP). Your device and the server are connected via physical networks (such as cables, fiber optics, etc.), and the data is transmitted through these pathways.
3. **Routers and DNS**: Data travels through routers, which direct the data to the correct location. When you enter a URL, the Domain Name System (DNS) translates the human-readable address into an IP address, helping the data reach the right server.
4. **Protocols**: Different protocols help manage how data is transferred. For example, HTTP/HTTPS for web pages, FTP for file transfer, and TCP/IP to ensure reliable data transmission.
5. **Browser Request-Response Cycle**: When you type a URL in your browser, it sends a request to the server. The server then processes the request and sends the data back, displaying the webpage to you.

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

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

**Client**

* The **client** is usually **your device** (like your phone or computer).
* It **asks for information** or services.
* Example: When you open a website, your browser (Chrome, Firefox) is the client that **sends a request** to the server.

**Server**

* The **server** is a **powerful computer** that **stores websites** and data.
* It **receives requests** from the client and **sends back** the needed information.
* Example: It sends the website content (like text, images, videos) back to your browser.

**Simple Analogy:**

* **Client** = You ordering food at a restaurant.
* **Server** = The kitchen that prepares and serves your food.

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

The **TCP/IP model** is like a **set of rules** that helps computers **communicate over the internet**. It breaks the communication process into **layers**, and each layer has a specific job.

1. **Application Layer** – The top layer where user applications access network services.
2. **Transport Layer** – Handles end-to-end communication and data delivery between devices.
3. **Internet Layer** – Responsible for addressing and routing the data across networks.
4. **Network Access Layer** – Deals with the physical transmission of data over the network.

**Explain Client Server Communication**

**What is a Client?**

* A **client** is the **device or app** that **asks for something**.
* Example: Your phone, browser, or app like Chrome, WhatsApp, etc.

**What is a Server?**

* A **server** is a **powerful computer** that **gives what was asked**.
* It stores websites, files, videos, or data and sends them when requested.

**How Client-Server Communication Works:**

1. **Client sends a request** → Like: “Hey server, give me Google homepage.”
2. **Server gets the request** → Checks what’s being asked.
3. **Server sends back a response** → Like: “Here’s the Google homepage.”
4. **Client shows the result** → You see the page on your screen.

**How does broadband differ from fiber-optic internet?**

**Broadband vs Fiber-Optic Internet**

| **Feature** | **Broadband** | **Fiber-Optic Internet** |
| --- | --- | --- |
| **Type of Connection** | General term for high-speed internet (includes DSL, cable, etc.) | Uses **fiber-optic cables** (glass wires) to send data as **light** |
| **Speed** | Fast, but can slow down during peak hours | **Much faster**, even with many users |
| **Technology** | Uses **electrical signals** via copper cables | Uses **light signals** through glass fibers |
| **Reliability** | Can be affected by distance and weather | More **reliable** and stable |
| **Availability** | Available in most areas | Limited to certain cities or areas (but growing) |
| **Cost** | Usually **cheaper** | Usually **more expensive**, but worth it for speed |

**In Simple Terms:**

* **Broadband** = umbrella term for regular fast internet (like cable or DSL).
* **Fiber-Optic** = super-fast, modern internet that uses light to transfer data, making it faster and more reliable.

**Example:**

* Watching a video on broadband might buffer sometimes.
* On fiber-optic, it loads almost instantly—even in 4K.

**What are the differences between HTTP and HTTPS protocols?**

**HTTP vs HTTPS**

| **Feature** | **HTTP** | **HTTPS** |
| --- | --- | --- |
| **Full Form** | HyperText Transfer Protocol | HyperText Transfer Protocol Secure |
| **Security** | ❌ Not secure | ✅ Secure (uses SSL/TLS encryption) |
| **Data Protection** | Data is sent in **plain text** | Data is **encrypted** |
| **URL Format** | Starts with http:// | Starts with https:// |
| **Use Case** | Used for normal websites | Used for **secure** websites (banking, login, etc.) |
| **Padlock Icon** | ❌ No padlock in browser | ✅ Shows a padlock 🔒 in browser |

**In Short:**

* **HTTPS = HTTP + Security**
* Always prefer **HTTPS** for safety!

What is the role of encryption in securing applications Software Applications and Its Types

**Encryption** is a method of protecting data by converting it into a code that can only be read by someone who has the correct **decryption key**. It plays a very important role in keeping **software applications secure**.

**Main Roles of Encryption:**

1. Protects Sensitive Data
2. Ensures Data Privacy
3. Provides Secure Communication
4. Helps in Authentication
5. Builds User Trust

**Software Applications and Its Types**

1. Applications Software
2. System Software
3. Middleware Software
4. Driver Software
5. Programming Software

**What is the difference between system software and application software?**

**Difference Between System Software and Application Software**

| **Feature** | **System Software** | **Application Software** |
| --- | --- | --- |
| **Definition** | System software is a collection of programs that manage and control computer hardware and provide a platform for running application software. | Application software is designed to perform specific tasks for the user, such as creating documents, managing data, or playing media. |
| **Purpose** | Its primary purpose is to operate the computer hardware and provide a platform for running application software. | Its primary purpose is to perform specific user-oriented tasks or solve specific problems. |
| **Examples** | Operating System (OS), Device Drivers, Utility Programs (e.g., Antivirus, Disk Management) | Word processors (MS Word), Spreadsheet software (Excel), Browsers (Chrome), Games |
| **Interaction with User** | System software usually works in the background and may not interact directly with the user (unless there is an issue or need for configuration). | Application software directly interacts with the user to fulfill specific tasks. |
| **Installation** | Typically pre-installed on a computer or device and cannot be easily replaced or removed by the user. | Installed by the user or purchased from a store, and can be easily added or removed. |
| **Examples of Functions** | Manages hardware resources, controls memory, and provides an interface for application software. | Helps users accomplish tasks like creating documents, sending emails, or editing photos. |

**In Simple Terms:**

* **System Software** is like the "foundation" or "engine" that runs the computer.
* **Application Software** is like the "tools" that allow users to do tasks on the computer.

For example:

* **System Software**: Windows OS, macOS
* **Application Software**: Microsoft Word, Google Chrome

**What is the significance of modularity in software architecture?**

**Modularity** means breaking a big system into smaller, easier-to-manage pieces or parts. Each part does a specific job and can be worked on separately

1. **Improved Maintainability**: Easier to update or fix individual parts of the system without affecting the whole software.
2. **Reusability**: Modules can be reused across different projects, saving time and resources.
3. **Scalability**: New features or modules can be added easily without major changes to the system.
4. **Separation of Concerns**: Different parts of the system are managed by separate modules, making the code more organized and understandable.
5. **Parallel Development**: Different teams can work on separate modules at the same time, speeding up development.
6. **Better Testing and Debugging**: Issues can be isolated to specific modules, making testing and debugging faster and easier.
7. **Flexibility and Customization**: Modules can be swapped or replaced as needed to meet specific requirements.

**Why are layers important in software architecture?**

1. **Improved Maintainability:** Layers allow easier updates or fixes without affecting the entire system.
2. **Reusability:** Each layer can be reused in different applications, saving time and effort.
3. **Scalability:** Layers let the system grow without major changes, adding new features or functions smoothly.
4. **Separation of Concerns:** Each layer handles a specific part of the system, making it simpler to manage.
5. **Parallel Development:** Different teams can work on separate layers at the same time without interfering with each other.
6. **Better Testing and Debugging:** Layers make it easier to test and fix parts of the system individually.
7. **Flexibility and Customization:** Layers allow easy swapping or modifying of parts based on needs or requirements.

**Explain the importance of a development environment in software production.**

1. **Code Writing & Testing:** It provides the tools (like editors, compilers, debuggers) needed to write and test code efficiently.
2. **Error Detection:** It helps catch errors early with features like syntax highlighting, auto-completion, and debugging tools.
3. **Consistency:** Ensures the development process is the same for everyone in the team, avoiding environment-related issues.
4. **Faster Development:** Speeds up the development process with built-in tools, libraries, and automation features.
5. **Safe Experimentation:** Developers can safely test new features or changes without affecting the actual (live) software.

**What is the difference between source code and machine code?**

1. **Source Code:**
   * It is the human-readable code written by a programmer in a programming language like Java, Python, or C++.
   * Example: print("Hello World");
   * It needs to be translated before the computer can understand it.
2. **Machine Code:**
   * It is the computer-readable code made up of 0s and 1s (binary).
   * The CPU can directly execute machine code.
   * It is generated by compiling or interpreting the source code.

**In short:**

*Source code* is for **humans to read and write**,  
 *Machine code* is for **computers to execute**.

**Why is version control important in software development?**

1. **Tracks all code changes** made over time.
2. **Helps teams work together** without conflicts.
3. **Allows going back** to previous working versions.
4. **Makes bug fixing easier** by comparing code changes.
5. **Supports testing new features** safely in separate branches.
6. **Keeps development organized** and avoids confusion.

**What are the benefits of using Github for students?**

Here are the benefits of using GitHub for students in **simple one-line points**:

1. **Stores and manages code** online for easy access.
2. **Helps learn real-world version control** using Git.
3. **Makes it easy to share projects** with teachers or classmates.
4. **Allows collaboration** on group projects.
5. **Shows coding skills** to future employers.
6. **Free student benefits** like tools, hosting, and training.
7. **Tracks progress** and changes over time in your code.

**What are the differences between open-source and proprietary software?**

| **Point** | **Open-Source Software** | **Proprietary Software** |
| --- | --- | --- |
| 1. **Source Code** | Visible and can be modified by anyone | Hidden and only modified by the company/developer |
| 2. **Cost** | Usually free to use | Often paid/licensed |
| 3. **Customization** | Easily customizable by users | Limited customization options |
| 4. **Support** | Community-based support | Official vendor or company support |
| 5. **Usage Rights** | Freely usable, sharable, and distributable | Restricted use as per license terms |
| 6. **Examples** | Linux, LibreOffice, Firefox | Windows, MS Office, Adobe Photoshop |
| 7. **Security Fixes** | Anyone can contribute to fixing bugs | Fixes come only from the official company |

**How does GIT improve collaboration in a software development team?**

1. **Tracks Changes**  
   Git records every change made to the code, so team members can see who did what and when.
2. **Branching**  
   Developers can work on their own branches without affecting the main project, which avoids conflicts.
3. **Merging**  
   Git allows team members to combine their code changes easily when they’re ready.
4. **Version Control**  
   Everyone can go back to a previous version of the code if something breaks.
5. **Conflict Resolution**  
   Git highlights conflicts when two people change the same part of the code, making it easy to fix.
6. **Remote Collaboration**  
   Teams can work together from anywhere by syncing their code through platforms like GitHub.
7. **Accountability**  
   Since every change is logged with the author's name, it brings transparency and responsibility.

**What is the role of application software in businesses?**

1. **Improves Efficiency**  
   It helps automate tasks like billing, accounting, or scheduling, saving time and effort.
2. **Supports Decision Making**  
   Software like spreadsheets and data analysis tools help businesses make smart decisions.
3. **Enhances Communication**  
   Email, messaging apps, and video conferencing tools help employees and teams stay connected.
4. **Manages Data**  
   Application software helps store, organize, and retrieve important business data safely.
5. **Customer Management**  
   CRM software helps businesses track customer interactions, improving service and sales.
6. **Increases Productivity**  
   Tools like word processors, presentation software, and project managers help get more work done.

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

The main stages of the **software development process** are as follows:

1. Planning
2. Analysis
3. Design
4. Implementation
5. Testing
6. Maintenance

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

The **requirement analysis phase** is critical in software development for several reasons:

1. **Clear Understanding of Needs**: Helps understand what users truly need from the software.
2. **Prevents Scope Creep**: Stops unnecessary changes during development.
3. **Accurate Project Planning**: Assists in estimating time, cost, and resources.
4. **Reduces Risk of Failure**: Identifies and addresses potential problems early.
5. **Improves Communication**: Ensures everyone involved understands the project.
6. **Basis for Testing**: Provides a foundation to create tests and verify the software works.

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

Software analysis helps to clearly define the system’s requirements, identify issues, and ensure the final product meets user needs and expectations.

**Role of Software Analysis in Development**

**1. Understanding Requirements**

* Software analysis helps to **understand** what the user needs and what the system should do.
* This is where developers figure out **how to solve the problem**.

**2. Identifying Key Features**

* It helps identify the **main features** that need to be included, like what the software must do, how it should behave, and what functions it must have.

**3. Setting the Foundation**

* It creates a **blueprint** or **plan** for developers to follow, ensuring everyone is on the same page before coding starts.

**4. Avoiding Mistakes Early**

* By analyzing the requirements in detail, **potential problems** or **misunderstandings** can be found early, saving time and costs later.

**5. Communication Tool**

* It helps communicate the **project's goals** to all team members, ensuring everyone understands what needs to be built.

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

1. **Requirements Analysis**: Understand what the system needs to do to solve the right problem.
2. **Architecture Design**: Plan the system’s structure and how its components will interact.
3. **Component Design**: Break the system into smaller parts and define how they work together.
4. **Data Design**: Plan how data will be stored, processed, and accessed securely.
5. **Interface Design**: Design how users or other systems will interact with the system.
6. **Security Design**: Plan how to protect the system from threats and vulnerabilities.
7. **Scalability & Performance**: Ensure the system can grow and perform well under stress.
8. **Testing & Validation**: Plan how to test the system to make sure it works as expected.

**Why is software testing important?**

1. **Find Bugs Early**: Testing helps detect issues before the software is used by others.
2. **Improve Quality**: Ensures the software works as intended and meets user needs.
3. **Save Costs**: Fixing bugs early is cheaper than fixing them after release.
4. **Ensure Security**: Testing identifies potential security vulnerabilities.
5. **User Experience**: Ensures the software is user-friendly and performs well.
6. **Compliance**: Verifies the software meets industry standards and regulations.

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

There are four main types of software maintenance:

1. **Corrective Maintenance**: Fixes bugs or issues found in the software after it has been deployed.
2. **Adaptive Maintenance**: Updates the software to keep it compatible with changes in the environment, such as new operating systems or hardware.
3. **Perfective Maintenance**: Improves the software by adding new features, enhancing performance, or making the software more user-friendly.
4. **Preventive Maintenance**: Involves making changes to the software to prevent future problems or issues, ensuring its longevity.

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

**Web Applications vs Desktop Applications**

| **Feature** | **Web Applications** | **Desktop Applications** |
| --- | --- | --- |
| **Installation** | No installation needed, run in a browser. | Must be installed on your computer. |
| **Access** | Accessible from anywhere with an internet connection. | Only accessible on the computer where it’s installed. |
| **Updates** | Automatically updated online. | Manual updates required by the user. |
| **Platform Dependence** | Works on any device with a browser (cross-platform). | Platform-specific (Windows, macOS, etc.). |
| **Performance** | Depends on internet speed and server. | Faster performance as it runs directly on the device. |
| **Internet Requirement** | Requires an internet connection. | Can work offline after installation. |
| **Security** | Security depends on the server and network. | More secure as it runs locally on the system. |
| **Examples** | Google Docs, Gmail, Facebook. | Microsoft Word, Photoshop, VLC Media Player. |

**In Short:**

* **Web apps**: Accessible anywhere, no installation needed, need the internet.
* **Desktop apps**: Installed on your computer, work offline, and usually faster.

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

**Advantages of Web Applications**

**1. No Installation Required**

* Web apps don’t need to be installed on your device. Just open them in a browser!

**2. Access Anywhere**

* As long as you have an internet connection, you can use the app from any device (phone, tablet, laptop).

**3. Automatic Updates**

* No need to worry about manually updating. Web apps update automatically on the server.

**4. Cross-Platform Compatibility**

* Works on any operating system (Windows, macOS, Linux) as long as you have a browser.

**5. Easier Collaboration**

* Many web apps allow **real-time collaboration**, like Google Docs, where multiple users can work together.

**6. Centralized Data Storage**

* Data is stored on the server, so you don’t need to worry about losing it if something happens to your device.

**7. Cost-Effective**

* You don’t need to buy or install software for every device. Just use it through a browser!

**8. No Dependency on Hardware**

* Web apps run on **server-side** computing power, so your device doesn’t need to be super powerful.

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

**Role of UI/UX Design in Application Development**

**1. UI (User Interface) Design – Visuals & Interaction**

* **What it means**: UI design focuses on how the app looks and how users interact with it (buttons, menus, icons, etc.).
* **Why it matters**: A clean and visually appealing interface makes the app **easy to use** and **attractive**, keeping users engaged.

**2. UX (User Experience) Design – Feel & Usability**

* **What it means**: UX design focuses on the **overall experience** of the user, like how easy it is to navigate, find things, and get tasks done.
* **Why it matters**: Good UX ensures the app is **intuitive** and **efficient**, making the user feel **satisfied** and reducing frustration.

**Why UI/UX Design is Important:**

**- Improves User Satisfaction**

* A well-designed UI/UX leads to happy users who enjoy using the app and want to return.

**- Boosts Usability**

* Users can **navigate easily**, find what they need faster, and **complete tasks smoothly**.

**- Increases Retention**

* An app with great UI/UX design keeps users coming back and improves the app’s **success**.

**- Enhances Brand Image**

* A sleek and functional UI/UX gives your app a professional, trustworthy vibe.

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

**Native Apps vs Hybrid Apps**

| **Feature** | **Native Apps** | **Hybrid Apps** |
| --- | --- | --- |
| **Development** | Built for a specific platform (iOS or Android). | Built once and run on multiple platforms (iOS, Android). |
| **Performance** | Fast and responsive since they use native code. | Slower than native, as they use web technologies inside a native wrapper. |
| **User Experience** | Better UX, as it follows the platform's guidelines and feels natural. | Not as smooth, may not look/feel like native apps. |
| **Access to Device Features** | Full access to device features (camera, GPS, etc.). | Limited access to device features unless additional plugins are used. |
| **Development Cost** | Higher, since you need to build separate apps for iOS and Android. | Lower, as you develop one app for all platforms. |
| **Maintenance** | Requires updating separate apps for each platform. | Easier to maintain, since you update one app for all platforms. |
| **Examples** | Instagram, WhatsApp, Facebook (separate apps for iOS and Android). | Facebook (older versions), Twitter, Uber (hybrid approach). |

**In Short:**

* **Native apps**: Built for a specific platform, better performance and UX, but higher cost and maintenance.
* **Hybrid apps**: Built once for multiple platforms, easier to maintain, but slightly slower and less smooth.

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

**1. Shows How Data Moves**

* DFDs help visualize how **data flows** in a system — from **input to output**.

**2. Simple and Clear**

* It gives a **clear picture** of the system without writing code — using symbols like arrows, circles, and boxes.

**3. Helps in Understanding**

* Makes it easy for developers, clients, and non-technical people to **understand the system**.

**4. Identifies Problems Early**

* Helps spot **gaps**, **errors**, or **confusion** in the process **before** development begins.

**5. Better Communication**

* Acts like a **map** that all team members can follow, so everyone knows how the system works.

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

**Pros**

* **Speed and Performance**: Usually faster because they run directly on the user's device without needing an internet connection.
* **Offline Access**: Can work without the internet once installed.
* **Better Hardware Access**: Can interact more deeply with hardware (e.g., printers, scanners, graphics cards).
* **Enhanced Security**: Data is stored locally, reducing risk of server-side breaches (if local device is secure).

**Cons**

* **Installation Required**: Must be downloaded and installed on each device, which can be time-consuming.
* **Updates Are Manual**: Users have to update the app themselves unless automatic updates are built in.
* **Platform Dependent**: A Windows app won’t run on macOS or Linux without specific versions or emulators.
* **Less Accessible Remotely**: Can’t be accessed from anywhere like web apps unless remote access is configured.

**How do flowcharts help in programming and system design?**

Flowcharts help in programming and system design in the following ways:

1. **Visual Clarity**
   * They show the steps of a process in a clear and easy-to-understand diagram.
2. **Better Planning**
   * Flowcharts help plan the logic of a program or system before actual coding starts.
3. **Problem Solving**
   * They help break down complex problems into smaller, manageable steps.
4. **Improves Communication**
   * Flowcharts allow team members and stakeholders to understand the flow of a system easily.
5. **Debugging and Maintenance**
   * They help identify errors and make changes in the logic without confusion.