* Module no:1

Q1.Write a simple hello world program into different language programming of your choice compare the structure and syntax.

1. Python

Code:

Python

Print(hello ,world”);

Structure and syntax

* 1.Simplicity and very

minimal.

2.Case sensitivity.

3.no semicolon is required for a simple script.

* 2.C
* Code
* #include<stdio.>h
* Int main();
* Printf(“Hello world\n”);
* Return =0;
* }.

Structure and syntax

The code is enclosed in main function,which is the standard entry point .

Curly braces {}define the body of the function.

Q.2Reserch and create a diagram of how data is transmitted from a client to a server over the internet?

Ans 1. The Request

When you type a web address url into your browser the client first looks up the address of the server using a dns server .

1. The connection

The TCP establishes a connection with sever and breaks your request into smaller.

3.The addressing

The ip address act as a user address use to find the destination.

4 The Journey

These packets are sent across the internet hopping through varuios routers.

5.The Deleivery and Response

The server receives the packets reassemble them into original request.

Q3.Describe the role of Client and server in web communication . In web communication, the client and server operate in a request-response model, with the client initiating a request and the server fulfilling it. The two components often run on separate computers and communicate over the internet using standardized protocols like HTTP.

**The client**

The client is the end-user device or software that requests data or services from a server. A user's web browser is the most common example of a web client.

Key functions of the client include:

* **Initiating communication:** The client initiates the communication by sending a request to the server, often triggered by a user action like typing a URL or clicking a link.
* **Handling the user interface:** The client runs the software that users interact with directly, such as a web browser. It presents the requested information and user controls in a readable, visual format.
* **Requesting resources:** The client sends an HTTP request specifying the web page, image, file, or other resource it needs from the server.
* **Receiving and rendering data:** The client receives the server's HTTP response, which contains the requested data. It then processes and renders this data to display the web page for the user.
* **Executing client-side logic:** The client executes client-side code, such as JavaScript, to provide interactive features and dynamic content on the webpage.

**The server**

The server is a powerful computer system or software program that provides services and resources to clients. A web server's job is to listen for incoming requests and respond by delivering web content.

Key functions of the server include:

* **Storing data:** The server acts as a centralized repository for web content, databases, and other resources that clients may need to access.
* **Listening for requests:** The server waits for incoming requests from clients and is constantly running to ensure availability.
* **Processing requests:** Upon receiving a request, the server processes it to determine what the client is asking for. It retrieves the necessary information from its storage system.
* **Generating dynamic content:** For complex applications, the server can execute server-side scripting (e.g., PHP, Python) to generate dynamic content in real-time, tailoring the response to the client's request.
* **Sending the response:** The server sends an HTTP response back to the client, delivering the requested content or an error message if the request fails.
* **Managing network traffic and security:** Servers handle the traffic flow for many concurrent client requests and incorporate security measures like firewalls and authentication to protect data.

**A web communication example**

When you browse the web, a typical client-server interaction occurs as follows:

1. **Initiation:** A user types a URL (e.g., www.example.com) into their browser (the client) and presses Enter.
2. **DNS lookup:** The browser requests the IP address for www.example.com from a Domain Name System (DNS) server.
3. **Connection:** The browser establishes a connection with the server at the resolved IP address using the TCP/IP protocol.
4. **Request:** The browser sends an HTTP request to the server, asking for the content of the home page.
5. **Processing:** The web server receives the request, finds the HTML, CSS, JavaScript, and image files for the home page, and prepares an HTTP response.
6. **Response:** The server sends the requested files back to the browser.
7. **Rendering:** The browser receives the files and renders the content to display the website on the user's screen.

Q4. Research diffrernt types of internet connections eg,broadband,fiber,satellite,and list their pros and cons.

Ans  transmission technologies, which can be broadly classified into the followin...

Tata Tele Business Services

Types of Internet Connections: Exploring Speed, Reliability & More ...

Comprehensive Guide to Internet Connection Types and Their Benefits. Discover the pros and cons of different internet connection t...

Bajaj Finserv

**Cons**

* **Peak-time slowdowns:** Cable internet is a shared network, so speeds can slow down during peak usage hours when many neighbors are online simultaneously.
* **Distance-based speed reduction:** DSL speeds degrade the farther you are from the provider's central office, making it a poor choice for rural homes.
* **Asymmetrical speeds:** Many cable and DSL plans offer slower upload speeds compared to download speeds, which can be a drawback for those who frequently upload large files or use video conferencing.

**Fiber-optic**

Often considered the "gold standard" for internet connectivity, fiber transmits data as light signals through thin glass or plastic strands.

**Pros**

* **Superior speed:** Fiber offers the fastest speeds, with symmetrical download and upload rates that can be up to 100 times faster than traditional broadband.
* **High reliability:** Fiber is more durable than copper cable and is less susceptible to weather conditions, electromagnetic interference, and power outages.
* **Low latency:** The low signal lag is crucial for activities that demand real-time responsiveness, such as online gaming and video conferencing.
* **High capacity:** The infrastructure is built for high data volumes, so performance remains consistent even with multiple connected devices.

**Cons**

* **Limited availability:** While expanding, fiber infrastructure is not yet universally available, especially in rural or hard-to-reach locations.
* **High installation cost:** Laying new fiber-optic cable is expensive and requires specialized labor, making it a costly investment for providers.
* **Fragility:** Fiber cables are more fragile than copper, making them vulnerable to physical damage during installation and construction.

**Satellite**

Satellite internet provides access via satellites orbiting Earth, communicating with a dish on your property.

**Pros**

* **Universal availability:** Satellite internet can reach almost any location, making it a lifeline for remote and rural areas with no access to other options.
* **Independent infrastructure:** The service is not dependent on terrestrial wires, so it can provide connectivity even after a disaster affects local infrastructure.

**Cons**

* **High latency:** Due to the long distance signals must travel to space and back, satellite connections have a noticeable lag. This makes them unsuitable for fast-paced online gaming and real-time applications.
* **Weather dependency:** Severe weather, such as heavy rain or snow, can interfere with the signal and cause service disruptions.
* **Lower speeds and data caps:** Satellite speeds are generally slower than cable or fiber. Most plans also include data caps, which limit your usage before speeds are throttled.
* **Higher cost:** For the speeds offered, satellite plans are typically more expensive than wired options.

Q5 Design a simple HTTP client server communication in any language.

An HTTP client-server communication can be demonstrated with a simple Python script. The server will wait for a client's request and respond, while the client will connect, send its request, and display the server's response.

This example uses Python's built-in http.server module for the server and the requests library for the client, which is a powerful and easy-to-use library for making HTTP requests.

**1. The HTTP server**

This script creates a simple web server that listens on a specified port. When it receives a GET request, it responds with "Hello, World!".

**How to run the server**

1. Save the code above in a file named .
2. Open your terminal or command prompt.
3. Navigate to the directory where you saved the file.
4. Run the server with the command:
5. The server is now running and waiting for requests.

**2. The HTTP client**

This script acts as the client. It sends a simple HTTP GET request to the server you started and prints the response.

**How to run the client**

1. First, ensure your server is running (from the previous step).
2. Install the requests library if you haven't already: pip install requests.
3. Save the code above in a file named client.py.
4. Open a **new** terminal or command prompt.
5. Navigate to the directory where you saved the file.
6. Run the client with the command: python client.py

**3. What happens during communication**

1. **Client request**: The requests.get(url) function in client.py sends an HTTP GET request to http://localhost:8000, the address of your running server.
2. **Server receives request**: The MyHandler.do\_GET() method in server.py is invoked when it receives the GET request.
3. **Server response**: The server then prepares an HTTP response.
   * self.send\_response(200) sets the status code to 200, indicating success.
   * self.send\_header(...) adds headers, in this case specifying the content type as HTML.
   * self.wfile.write(...) writes the message body, "Hello, World!", which is sent back to the client.
4. **Client receives response**: The requests library in client.py captures the full response from the server.
5. **Client displays results**: The client script then prints the status code and the body of the response for the user to see.

*Q6.Simulate HTTP and FTP Requests using command line tools eg curls*

**curl** is a powerful command-line tool for transferring data with URL syntax, supporting a wide range of protocols, including HTTP and FTP. You can use it to simulate requests and interact with web and file servers directly from your terminal.

**HTTP requests**

**Basic GET request**

This is the most fundamental request, used to retrieve data from a web server. By default, curl performs a GET request when you provide a URL.

**Command:**

sh

curl https://example.com

Use code with caution.

**Output:**  
This command prints the HTML content of the homepage for example.com directly to your terminal.

**GET request with verbose output**

To see what is happening behind the scenes, including the full request headers sent and response headers received, use the -v (verbose) option.

**Command:**

sh

curl -v https://example.com

Use code with caution.

**Output:**  
This displays a detailed log of the communication, from the initial connection to the final data transfer.

**POST request**

A POST request is used to send data to a server, such as when submitting a web form. You can specify the data using the -d or --data option.

**Command:**

sh

curl -X POST -H "Content-Type: application/json" -d '{"name":"John Doe", "job":"Developer"}' https://api.example.com/users

Use code with caution.

**Options:**

* -X POST: Explicitly specifies the HTTP method as POST.
* -H: Adds a custom header. Here, it sets the Content-Type to application/json to inform the server about the data format.
* -d: Includes the data you are sending in the body of the request.

**FTP requests**

**Download a file**

To download a file from an FTP server, use the ftp:// protocol followed by the server address and file path. You can use the -O flag to save the file with the same name as the remote file.

**Command:**

sh

curl -u username:password -O ftp://ftp.example.com/path/to/remote\_file.zip

Use code with caution.

**Options:**

* -u username:password: Provides the authentication credentials for the FTP server. curl will prompt you for a password if you only provide the username.
* -O: Saves the downloaded file in your current directory, using the filename from the remote server (remote\_file.zip).

**Upload a file**

To upload a file to an FTP server, use the -T option, which specifies the local file to upload.

**Command:**

sh

curl -u username:password -T local\_file.txt ftp://ftp.example.com/path/to/upload/

Use code with caution.

**Options:**

* -T: Specifies the local file (local\_file.txt) to transfer.
* ftp://...: The remote directory where the file will be uploaded. The file will keep its original name.

**List files in an FTP directory**

You can list the contents of a directory on an FTP server by pointing curl at the directory URL.

**Command:**

sh

curl -u username:password ftp://ftp.example.com/path/to/directory/

Use code with caution.

**Output:**  
This prints a listing of the files and subdirectories, similar to the output of an ls command.

**Delete a file**

You can send custom FTP commands using the -Q option, which allows you to delete a file.

**Command:**

sh

curl -u username:password ftp://ftp.example.com/path/to/file\_to\_delete.txt -Q "DELE file\_to\_delete.txt"

Use code with caution.

**Option:**

* -Q "DELE file.txt": Sends the custom FTP command DELE to delete the specified file.

AI responses may include mistakes. [Learn more](https://support.google.com/websearch?p=aimode)

38 sites

* HTTP request on the command-line interface via curl - Tolustar

28 Mar 2021 — The CLI allows users to perform operations quickly provided the users know the right commands. There are various CLI...

Tolustar

* Downloading Files With cURL | cURL FTP - Pair Networks

20 May 2025 — Downloading files with curl \* The Basics. \* Setting the output file. \* Viewing the complete request and response. \* ...

Pair Networks

* Popular curl Examples - KeyCDN Support

13 Jun 2022 — The following commands can all be entered directly into your terminal to retrieve a response. \* 1. HTTP GET req

*Q.7 identify and explain three common applications security vulnerabilities.Suggest possible solutions*

*Ans* Three common application security vulnerabilities are SQL injection, cross-site scripting (XSS), and broken authentication. All three can lead to unauthorized access, data theft, and other significant security compromises.

**1. SQL injection (SQLi)**

**Explanation**

SQLi is an injection attack that lets an attacker execute malicious SQL code on a web application's backend database. This happens when an application includes user-supplied input in a database query without proper validation or sanitization.

* **Example:** A login form asks for a username and password. An attacker enters a malicious string like ' OR '1'='1 into the username field, which alters the SQL query to bypass the login process and gain unauthorized access.

**Solution**

* **Use prepared statements:** Parameterized queries ensure that the database treats user input as data, not as an executable SQL command. This is the most effective defense against SQL injection.
* **Employ whitelisting:** Use input validation to accept only a limited set of known safe characters and patterns.
* **Enforce least privilege:** Configure database accounts with only the permissions they absolutely need. For example, a web application should not have administrator-level access to the database.

**2. Cross-site scripting (XSS)**

**Explanation**

XSS is a client-side code injection attack where an attacker injects malicious scripts into a legitimate, trusted website. When a victim's browser loads the page, the malicious script executes and can hijack the user's session, steal their cookies, or redirect them to malicious sites.

* **Example:** An attacker posts a comment on a blog containing malicious JavaScript. When another user views the comment, the script runs in their browser and steals their session token.

**Solution**

* **Encode output:** Always encode untrusted user-controllable data before rendering it in an HTML response. This converts special characters into a safe format so the browser renders it as text, not code.
* **Sanitize input:** If you must allow HTML, use a robust sanitization library to filter and remove any malicious tags and attributes from user input.
* **Use a Content Security Policy (CSP):** A CSP is an HTTP response header that restricts which scripts a webpage can load, acting as a final line of defense.

**3. Broken authentication**

**Explanation**

This vulnerability occurs when an application's authentication and session management functions are poorly implemented. An attacker can exploit these weaknesses to compromise user credentials, impersonate users, or gain unauthorized access to accounts.

* **Example:** An application fails to properly invalidate a session token when a user logs out. An attacker who steals the session cookie can then use it to access the user's account.

**Solution**

* **Implement multi-factor authentication (MFA):** Adding an extra layer of security beyond a password, such as a one-time code or biometric scan, makes it significantly harder for attackers to compromise accounts.
* **Enforce strong password policies:** Use strong hashing algorithms like bcrypt for password storage and enforce policies that prevent weak or commonly used passwords.
* **Use secure session management:** Rotate session IDs after a successful login and ensure they are invalidated after logout or a period of inactivity.

*Q8Identify and classify 5 applications you use daily as either system software or application software?*

ree common application security vulnerabilities are SQL injection, cross-site scripting (XSS), and broken authentication. All three can lead to unauthorized access, data theft, and other significant security compromises.

**1. SQL injection (SQLi)**

**Explanation**

SQLi is an injection attack that lets an attacker execute malicious SQL code on a web application's backend database. This happens when an application includes user-supplied input in a database query without proper validation or sanitization.

* **Example:** A login form asks for a username and password. An attacker enters a malicious string like ' OR '1'='1 into the username field, which alters the SQL query to bypass the login process and gain unauthorized access.

**Solution**

* **Use prepared statements:** Parameterized queries ensure that the database treats user input as data, not as an executable SQL command. This is the most effective defense against SQL injection.
* **Employ whitelisting:** Use input validation to accept only a limited set of known safe characters and patterns.
* **Enforce least privilege:** Configure database accounts with only the permissions they absolutely need. For example, a web application should not have administrator-level access to the database.

**2. Cross-site scripting (XSS)**

**Explanation**

XSS is a client-side code injection attack where an attacker injects malicious scripts into a legitimate, trusted website. When a victim's browser loads the page, the malicious script executes and can hijack the user's session, steal their cookies, or redirect them to malicious sites.

* **Example:** An attacker posts a comment on a blog containing malicious JavaScript. When another user views the comment, the script runs in their browser and steals their session token.

**Solution**

* **Encode output:** Always encode untrusted user-controllable data before rendering it in an HTML response. This converts special characters into a safe format so the browser renders it as text, not code.
* **Sanitize input:** If you must allow HTML, use a robust sanitization library to filter and remove any malicious tags and attributes from user input.
* **Use a Content Security Policy (CSP):** A CSP is an HTTP response header that restricts which scripts a webpage can load, acting as a final line of defense.

**3. Broken authentication**

**Explanation**

This vulnerability occurs when an application's authentication and session management functions are poorly implemented. An attacker can exploit these weaknesses to compromise user credentials, impersonate users, or gain unauthorized access to accounts.

* **Example:** An application fails to properly invalidate a session token when a user logs out. An attacker who steals the session cookie can then use it to access the user's account.

**Solution**

* **Implement multi-factor authentication (MFA):** Adding an extra layer of security beyond a password, such as a one-time code or biometric scan, makes it significantly harder for attackers to compromise accounts.
* **Enforce strong password policies:** Use strong hashing algorithms like bcrypt for password storage and enforce policies that prevent weak or commonly used passwords.
* **Use secure session management:** Rotate session IDs after a successful login and ensure they are invalidated after logout
* What is Cross-Site Scripting (XSS)? How to Prevent it? | Fortinet

Cross-site Scripting (XSS) Meaning. ... In an XSS attack, an attacker uses web-pages or web applications to send malicious code an...

Fortinet

* What is cross-site scripting (XSS) and how to prevent it?

\* Attack surface visibility Improve security posture, prioritize manual testing, free up time. \* CI-driven scanning More proactive...

PortSwigger

* What is Cross Site Scripting (XSS) ? - GeeksforGeeks

22 Sept 2025 — What is Cross Site Scripting (XSS) ? ... Cross-Site Scripting (XSS) is a vulnerability in a web application that al...

GeeksforGeeks

Show all

Identify and classify 5 application you use daily as either system software or application software

**System software**

 manages and controls computer hardware and provides a platform for application software to run. Without system software, a computer would not function. In contrast, **application software** (or "apps") is designed for end-users to perform specific tasks, such as word processing, browsing the internet, or streaming music.

Here are 5 common applications classified by their software type:

**System software**

1. **Android OS:** An operating system based on the Linux kernel that manages and controls all the hardware and software resources on Android mobile devices. It is essential for the phone to function and provides the environment for all other apps to run.

**Application software**

1. **WhatsApp:** An instant messaging application that is designed for a specific purpose: to allow users to communicate through text, voice, and video calls. It runs on top of a mobile operating system like Android or iOS.
2. **Google Chrome:** A web browser application used to access and navigate the internet. It is a program that a user intentionally opens to perform the specific task of browsing.
3. **Microsoft Word:** A word processing application used for a specific task: creating, editing, and formatting text documents. It is part of the Microsoft Office suite and runs on a system's operating system.
4. **Spotify:** A music streaming application that provides a service for users to listen to music. It is designed for the end-user and depends on a system's operating system to function.

Q9Design a basic three tier software architecture diagram for a web application.

**Basic three-tier software architecture for a web application**

A three-tier architecture is a software design pattern that separates an application into three logical and physical layers. This separation of concerns improves scalability, maintainability, and security.

**Diagram**

The following diagram illustrates the flow of a user request through the three tiers of a web application:

* **Request Flow:** The user initiates an action in the Presentation Tier (1). The request is sent to the Application Tier (2), which processes the request using business logic and interacts with the Data Tier (3) if needed. The Application Tier then sends a response back to the Presentation Tier (4), which updates the user interface (5).
* **Response Flow:** The user sees the result of their request displayed in their web browser (5).

mermaid

graph TD

A[User] -->|1. Request (e.g., login, submit form)| B(Presentation Tier);

B -->|2. HTTP Request| C(Application Tier);

C -->|3. Data Access Request (e.g., query, update)| D(Data Tier);

D -->|4. Data Response| C;

C -->|5. HTTP Response| B;

B -->|6. Render UI| A;

subgraph Presentation Tier (Web Browser)

B

end

subgraph Application Tier (Web Server)

C

end

subgraph Data Tier (Database Server)

D

end

Use code with caution.

**Components of each tier**

**1. Presentation Tier (Client)**

The topmost tier, which the end-user interacts with directly.

* **Purpose:** To display information and collect user input.
* **Example Technologies:**
  + **Front-end:** HTML, CSS, JavaScript.
  + **Frameworks:** React, Angular, or Vue.js.
  + **Infrastructure:** A user's web browser.
* **How it works:** The user's browser sends an HTTP request to the Application Tier and displays the resulting HTML and data it receives. This tier does not directly interact with the database.

**2. Application Tier (Server/Business Logic)**

This middle tier contains the application's core logic and functionality.

* **Purpose:** To process user requests, run business logic, and act as an intermediary between the Presentation and Data tiers.
* **Example Technologies:**
  + **Programming Languages:** Python, Java, Ruby, or Node.js.
  + **Web Frameworks:** Django, Spring, Ruby on Rails, or Express.js.
  + **Infrastructure:** An application server or web server.
* **How it works:** The application server receives requests from the client, processes them according to the business rules, and communicates with the Data Tier to retrieve or store information.

**3. Data Tier (Database)**

The back-end tier, which stores and manages all the application's data.

* **Purpose:** To store data persistently, handle data access, and ensure data integrity.
* **Example Technologies:**
  + **Relational Databases:** MySQL, PostgreSQL, or Oracle.
  + **NoSQL Databases:** MongoDB or Cassandra.
  + **Infrastructure:** A dedicated database server.
* **How it works:** This tier only communicates with the Application Tier, never directly with the client. It responds to data requests and manages the storage and retrieval of all application information

Q10.create a case study on the functionality of the presentation ,business ,businesslogic,and data access of a given software system.

Ans

**Case study: Online Retail System (Three-tier architecture)**

This case study uses a fictional online retail system to demonstrate the functionality and interactions between the Presentation, Business Logic, and Data Access layers. The system allows customers to browse products, add items to a shopping cart, and place orders.

**The user action: Adding a product to the cart**

We will analyze how the three layers work together to process a user adding a product to their shopping cart.

**1. Presentation layer: The user interface**

* **Functionality:** This layer is the part of the software the user directly interacts with. Its primary job is to display information from the other layers in an understandable format and to capture user input. In our online retail system, this is the web browser displaying product pages.
* **Scenario:** A customer is browsing the website and sees a "10% off" banner for a specific product. They click the "Add to Cart" button for that item.
* **In action:**
  + The browser renders the webpage with product details, images, and the "Add to Cart" button, based on data provided by the Business Logic layer.
  + When the user clicks the button, the Presentation layer captures this input and sends a request to the Business Logic layer via an API call. This request contains the product ID and the quantity.

**2. Business Logic layer: The application's "brain"**

* **Functionality:** This layer contains the core logic that defines the application's processes, rules, and workflows. It acts as the intermediary, receiving requests from the Presentation layer, processing them, and interacting with the Data Access layer as needed.
* **Scenario:** The user's request to add an item to the cart is received and processed. The system needs to apply the 10% discount and update the cart.
* **In action:**
  + **Receives request:** The Business Logic layer receives the user's request for product XYZ with a quantity of 1.
  + **Validates input:** It first checks if the product ID is valid and if the quantity is a positive number.
  + **Applies business rules:** The system checks for any active promotions. It finds the 10% discount and calculates the new discounted price for product XYZ. It also checks the inventory to ensure the product is in stock.
  + **Coordinates with Data Access layer:** It determines that a database update is required. It sends a request to the Data Access layer to update the user's shopping cart, including the discounted price.
  + **Returns a response:** After the database update is complete, the Business Logic layer generates a response and sends it back to the Presentation layer, indicating that the item was successfully added to the cart.

**3. Data Access layer: The persistent storage**

* **Functionality:** This layer handles all communication with the application's database. It manages all the low-level tasks of storing, retrieving, and manipulating data, shielding the Business Logic layer from the complexities of the specific database technology.
* **Scenario:** The Business Logic layer's request to update the user's cart is fulfilled by this layer.
* **In action:**
  + **Receives request:** The Data Access layer receives a command from the Business Logic layer to update the cart for user ABC with product XYZ, quantity 1, and the discounted price.
  + **Executes database operations:**
    - It translates the command into a specific SQL query (e.g., INSERT or UPDATE).
    - It securely executes the query against the database.
    - It handles any potential database errors, such as a connection failure.
  + **Returns a result:** It returns the status of the operation back to the Business Logic layer (e.g., "cart updated successfully"). The Data Access layer never communicates directly with the user.

**Conclusion**

This three-tier architecture provides several benefits for the online retail system:

* **Separation of Concerns:** Each layer has a single, well-defined responsibility, making the system easier to design, develop, and manage.
* **Maintainability:** An update to the user interface (Presentation layer) does not require changes to the Business Logic or Data Access layers. For example, the website's design could be completely overhauled without touching the business rules.
* **Scalability:** Each layer can be scaled independently. If the business sees a spike in traffic, the Business Logic layer can be scaled to handle more requests without needing to upgrade the database.
* **Security:** The Data Access layer is protected because the Presentation layer never communicates with it directly. All data requests are filtered through the Business Logic layer, which can perform validation and enforce access controls.

Q10explore different types of software environment developments,testing,production,set up a basic environment in a viertual machine. Different software environments serve specific purposes within the software development lifecycle, ensuring that code is written, tested, and deployed to production in a controlled and reliable manner. Virtual machines (VMs) are a common way to set up these isolated environments, preventing conflicts and simulating real-world conditions.

**Different types of software environments**

**1. Development (Dev) environment**

This is the personal sandbox where developers write, test, and debug code locally.

* **Purpose:** To allow developers to iterate quickly on new features or bug fixes without affecting the work of others.
* **Characteristics:**
  + **Isolated and personal:** Typically resides on an individual developer's workstation.
  + **High rate of change:** This is a dynamic space with frequent new code, testing, and debugging.
  + **Tooling:** An Integrated Development Environment (IDE), a code editor, version control like Git, and other necessary SDKs.

**2. Testing (Test or QA) environment**

Here, quality assurance (QA) engineers and developers test code in a controlled setting that mirrors the production environment.

* **Purpose:** To perform rigorous quality checks, including functional, integration, and regression testing, to catch bugs before a release.
* **Characteristics:**
  + **Controlled data:** Often uses a sanitized or controlled dataset that is similar to production data but without sensitive information.
  + **Automation:** Relies on automated testing frameworks to run test suites and ensure consistency.
  + **Environment parity:** Should be as close as possible to the production environment to ensure accurate and reliable results.

**3. Production (Prod) environment**

This is the final, live environment where the software is used by end-users.

* **Purpose:** To provide a stable, secure, and highly available platform for users to access the application.
* **Characteristics:**
  + **Highest standards:** Requires the highest level of reliability, security, and performance.
  + **Real data and traffic:** Handles actual user data and live traffic, so any issues have a direct impact on users.
  + **Strict access control:** Access is typically limited to authorized operations personnel to minimize risk.

**4. Staging (Staging) environment**

A pre-production environment that is a near-identical replica of the production environment.

* **Purpose:** To serve as a final testing ground before release, allowing stakeholders to perform user acceptance testing (UAT) and catch any last-minute issues in a production-like setting.
* **Characteristics:**
  + **Release candidate testing:** The code deployed here is considered release-ready.
  + **Final validation:** Helps ensure a smooth deployment by testing the full application stack in real-world conditions.

**How to set up basic environments in a virtual machine**

Using VMs for setting up environments offers several advantages, including isolation, easy replication, and the ability to run multiple operating systems on a single physical machine.

**Basic steps for setting up environments with VirtualBox**

1. **Install VirtualBox:** Download and install Oracle VM VirtualBox, a free and powerful virtualization tool, on your host operating system (Windows, macOS, or Linux).
2. **Download OS image:** Acquire the .iso file for the operating system you want for your VM. For a lean setup, a Linux server distribution like Ubuntu Server is a common choice.
3. **Create a new VM:**
   * Open VirtualBox and click "New" to start the creation wizard.
   * Give your VM a descriptive name, like "Dev-Environment" or "Test-Server".
   * Select the type and version of the OS you are installing.
   * Allocate RAM and a virtual hard disk. The amount depends on your project's needs and host machine's capacity.
4. **Install the OS:**
   * Start the new VM and select your downloaded .iso file as the startup disk.
   * Follow the on-screen instructions to install the operating system inside the VM.
5. **Configure network access:**
   * To access your server from your host machine (e.g., via SSH or a web browser), you will need to configure port forwarding.
   * In the VM's **Settings > Network** section, set the adapter to "NAT" and configure a port forwarding rule. For example, forward Host Port: 2222 to Guest Port: 22 for SSH access.
6. **Set up the software stack:**
   * Once the OS is installed and running, you can set up your specific software stack (e.g., install a web server like Apache, a database like MySQL, and a language runtime like Node.js).
7. **Create environment templates:**
   * After configuring one VM, use VirtualBox's **Clone** feature to create identical copies for your test and staging environments.
   * By cloning, you ensure that your development and testing environments are consistent with one another, preventing issues caused by environmental differences.

Q11write and upload first source code file to github

Ans:s **tep 1: Write Code**

* Create a local folder (e.g., my-project).
* Inside, create a file (e.g., hello.py) with your source code.

**Step 2: Initialize Git**

* Open your terminal/command prompt.
* Navigate to your project folder: cd my-project
* Initialize a Git repository: git init

**Step 3: Stage and Commit**

* Add your file to staging: git add hello.py (or git add . for all files).
* Commit your changes: git commit -m "Initial commit"

**Step 4: Create GitHub Repo**

* Go to github.com and create a new repository.
* Copy the repository's URL.

**Step 5: Link Repositories**

* Link your local repository to the GitHub one: git remote add origin [your-repo-url]

**Step 6: Push to GitHub**

* Push your code to the main branch: git push -u origin main

Q12 create a github repository and document how to commit and push code changes.

**tep 1: Write Code**

* Create a local folder (e.g., my-project).
* Inside, create a file (e.g., hello.py) with your source code.

**Step 2: Initialize Git**

* Open your terminal/command prompt.
* Navigate to your project folder: cd my-project
* Initialize a Git repository: git init

**Step 3: Stage and Commit**

* Add your file to staging: git add hello.py (or git add . for all files).
* Commit your changes: git commit -m "Initial commit"

**Step 4: Create GitHub Repo**

* Go to github.com and create a new repository.
* Copy the repository's URL.

**Step 5: Link Repositories**

* Link your local repository to the GitHub one: git remote add origin [your-repo-url]

**Step 6: Push to GitHub**

* Push your code to the main branch: git push -u origin mai

Q13create a student account on github and collaborate on a small project with a classmate

Ans: **ating a Student Account on**[GitHub](https://git-scm.com/book/sv/v2/GitHub-Contributing-to-a-Project):

1. **Sign up:** Go to github.com/join to create a free personal account.
2. **Verify student status:** Apply for the **GitHub Student Developer Pack** by visiting education.github.com/pack and clicking "Sign up for Student Developer Pack." You will need to provide proof of enrollment, such as a school-issued email address or a photo of your student ID.
3. **Get verified:** Wait for an email from GitHub confirming your status. This process can take a few days.

**Collaborating on a Project with a Classmate**:

1. **Create the repository:** One student creates a new project repository on GitHub and initializes it with a README file.
2. **Add collaborator:** Go to the repository's **Settings** tab, click **Collaborators** under Access, and add your classmate's GitHub username.
3. **Clone the project:** Both students clone the repository to their local computers using the git clone command.
4. **Work and commit:** Each student works on their part of the code locally, saving changes with git commit.
5. **Push and pull:** To share updates, use git push to upload your changes. To get your classmate's latest updates, use git pull.
6. **Resolve conflicts:** If both students change the same part of a file, Git will create a merge conflict that needs to be fixed manually before committing again.

Q14 create a list of software you use regularly and classify them into following categories system,application,and utilty software

Ans: **System software**

* **Definition:** The foundation that allows a computer to run, managing hardware and providing a platform for other software.
* **Examples:**
  + **Operating Systems:** Windows, macOS, Linux, Android, iOS.
  + **Device Drivers:** Enable the operating system to communicate with specific hardware, like printers or graphics cards.
  + **Firmware:** Low-level software providing control for specific hardware, like a computer's BIOS or UEFI.

**Application software**

* **Definition:** Designed to help users perform specific tasks, ranging from productivity to entertainment.
* **Examples:**
  + **Web Browsers:** Google Chrome, Mozilla Firefox, Safari.
  + **Office Suites:** Microsoft Word (word processing), Microsoft Excel (spreadsheets), Microsoft PowerPoint (presentations).
  + **Creative Tools:** Adobe Photoshop (image editing).
  + **Communication Apps:** Zoom, Slack, Microsoft Teams (video conferencing and messaging).
  + **Media Players:** VLC Media Player (video and audio playback).

**Utility software**

* **Definition:** Tools that help manage, optimize, and maintain the computer system, often running in the background.
* **Examples:**
  + **Antivirus Software:** Norton Antivirus, Avast, Windows Defender (protects against malware).
  + **File Compression Tools:** WinRAR, 7-Zip (reduces file sizes).
  + **Disk Cleanup Tools:** Windows Disk Cleanup (removes unnecessary files to free up storage space).
  + **Backup Software:** Acronis True Image (creates copies of data for recovery).
  + **Disk Defragmenters:** Windows Drive Optimizer (reorganizes fragmented data to improve disk performance).

FOL

Q15 WRITE A report on various types of application software and how they improve productivity

Ans. in want this ans in short

Application software improves productivity primarily by

**automating manual tasks**, which frees up time and reduces human error. It also **enhances collaboration** through tools that allow real-time communication and shared work, no matter the location.

Key ways application software boosts productivity:

* **Automation:** Handles repetitive processes like data entry, invoicing, and report generation, saving significant time.
* **Streamlined workflows:** Manages and tracks project tasks, resources, and deadlines in a single platform, making processes more efficient.
* **Improved communication:** Tools like messaging apps and video conferencing centralize communication and enable faster, more flexible interactions among team members.
* **Better data management:** Enables effective storage, organization, and analysis of large volumes of data for more informed, data-driven decisions.
* **Enhanced accuracy:** Reduces human errors in calculations and record-keeping, ensuring data consistency and reliability.

Q16follow agit tutorial to pracise cloning branching and merging repositories

Ans **1. Clone a repository**

**Purpose:** Create a local copy of a remote repository (e.g., on [GitHub](https://git-scm.com/book/en/v2/Git-Basics-Getting-a-Git-Repository)).

**Instructions:**

1. Navigate to the repository on your Git hosting service (e.g., GitHub, GitLab).
2. Click the "<> Code" button and copy the URL.
3. Open your terminal or command prompt.
4. Navigate to the directory where you want to place the cloned folder using cd <directory\_path>.
5. Use the git clone command followed by the copied URL.

sh

git clone https://github.com/your-user/your-repository.git

Use code with caution.

1. cd into the newly created repository folder.

**2. Branching**

**Purpose:** Create a separate line of development to work on a new feature or fix without affecting the main codebase.

**Instructions:**

1. Create a new branch and switch to it in a single command using git checkout -b.

sh

git checkout -b new-feature-branch

Use code with caution.

1. Make your changes to the project files.
2. Stage your changes for the next commit.

sh

git add .

Use code with caution.

1. Commit your changes with a descriptive message.

sh

git commit -m "Add a new feature"

Use code with caution.

3. Merging

**Purpose:** Combine the changes from your feature branch back into the main branch.

**Instructions:**

1. Switch back to the main branch.

sh

git checkout main

Use code with caution.

1. Pull the latest changes from the remote main branch to ensure your local version is up-to-date.

sh

git pull origin main

Use code with caution.

1. Merge your feature branch into the main branch.

sh

git merge new-feature-branch

Use code with caution.

1. Push the final, merged version of the main branch to the remote repository.

sh

git push origin main

Use code with caution.

1. Optionally, delete your feature branch since it is no longer needed.

sh

git branch -d new-feature-branch

Use code with caution.