**Basics of Web Communication and HTTP**

**1) Webserver**

A web server is a software (and sometimes hardware) that stores, processes, and delivers web pages to users when requested through a browser.

Examples: Apache, Nginx, Microsoft IIS.

It listens for HTTP requests from clients (browsers) and responds with the requested resources (HTML, CSS, images, or data).

**2) Browser**

A browser is a client application (like Chrome, Firefox, Edge) that sends requests to web servers and displays the response to users.

It converts the HTML, CSS, and JavaScript received from the server into a visual web page.

It also manages cookies, caching, security, and rendering.

**3) TCP Connections / Socket**

TCP (Transmission Control Protocol) is a communication protocol that ensures reliable, ordered, and error-checked data transfer between devices.

A socket is an endpoint for sending/receiving data across a network.

When a browser talks to a server, it first creates a TCP connection using sockets before exchanging data.

Example: Browser opens a TCP connection to port 80 (HTTP) or 443 (HTTPS) of the server.

**4) HTTP (HyperText Transfer Protocol)**

HTTP is the protocol used by browsers and web servers to communicate.

It defines how a client (browser) requests data and how a server responds.

Common HTTP methods:

GET → Request data (e.g., load a webpage)

POST → Send data (e.g., submit a form)

PUT / PATCH → Update data

DELETE → Remove data

Example:

Browser sends: GET /index.html HTTP/1.1

Server replies: HTTP/1.1 200 OK + page content.

**5) Viewing Headers – Browser Developer Mode**

Headers are extra information sent along with HTTP requests and responses.

Examples of headers:

Content-Type: text/html → tells browser what kind of data is sent.

Set-Cookie: sessionid=12345 → server sets a cookie.

Authorization: Bearer <token> → client sends authentication.

In browsers (Chrome/Edge/Firefox):

Right-click → Inspect → open Developer Tools.

Go to Network tab.

Reload the page → click on a request → check Headers section.

👉 Together, the flow is:

Browser connects to Web Server using a TCP connection/socket.

It sends a request using the HTTP protocol.

Server responds with headers + content.

You can view all details in Developer Tools → Network → Headers.



**Very simple web client:**

****

This is a manual implementation of a web client (like a mini-browser).

It connects to a server, sends an HTTP request, and prints the response.

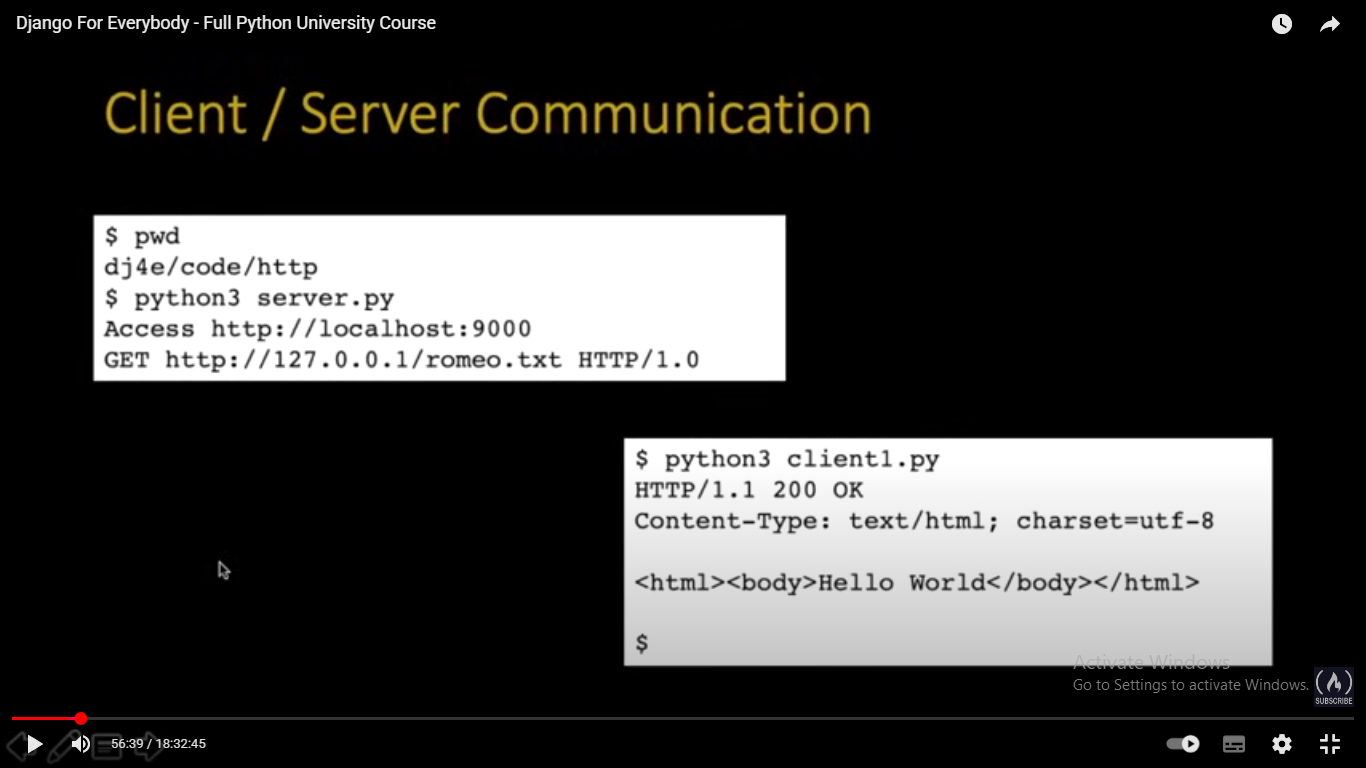
Normally, you’d use Python’s requests library to do this in one line:

**import requests**

**print(requests.get("**[**http://127.0.0.1:9000/romeo.txt").text**](http://127.0.0.1:9000/romeo.txt%22).text)**)**

**—-------------------------------------------------------------------------------------------------------------**

**Client/Server communication**

****

Server (server.py) runs on port 9000 and waits for requests.

Client (client1.py) sends a GET request for romeo.txt.

Server responds with headers + body (HTML page).

Client prints the server’s response.

### **1. Starting the Server**

**$ pwd**

**dj4e/code/http**

**$ python3 server.py**

**Access http://localhost:9000**

* **pwd → prints current working directory (dj4e/code/http).**
* **python3 server.py → runs the Python server program.**
* **The server starts listening on port 9000 at localhost (127.0.0.1).**
* **So now your server is ready to accept HTTP requests.**

### **🔹 2. Client Request**

**GET http://127.0.0.1/romeo.txt HTTP/1.0**

* **This is the request your client (previous client1.py) sends.**
* **It asks for the file romeo.txt from the server using the HTTP GET method.**

### **🔹 3. Running the Client**

**$ python3 client1.py**

* **This executes the Python client program (we saw earlier).**
* **The client connects to the server running on localhost:9000.**
* **It sends the HTTP GET request.**

### **🔹 4. Server Response**

**HTTP/1.1 200 OK**

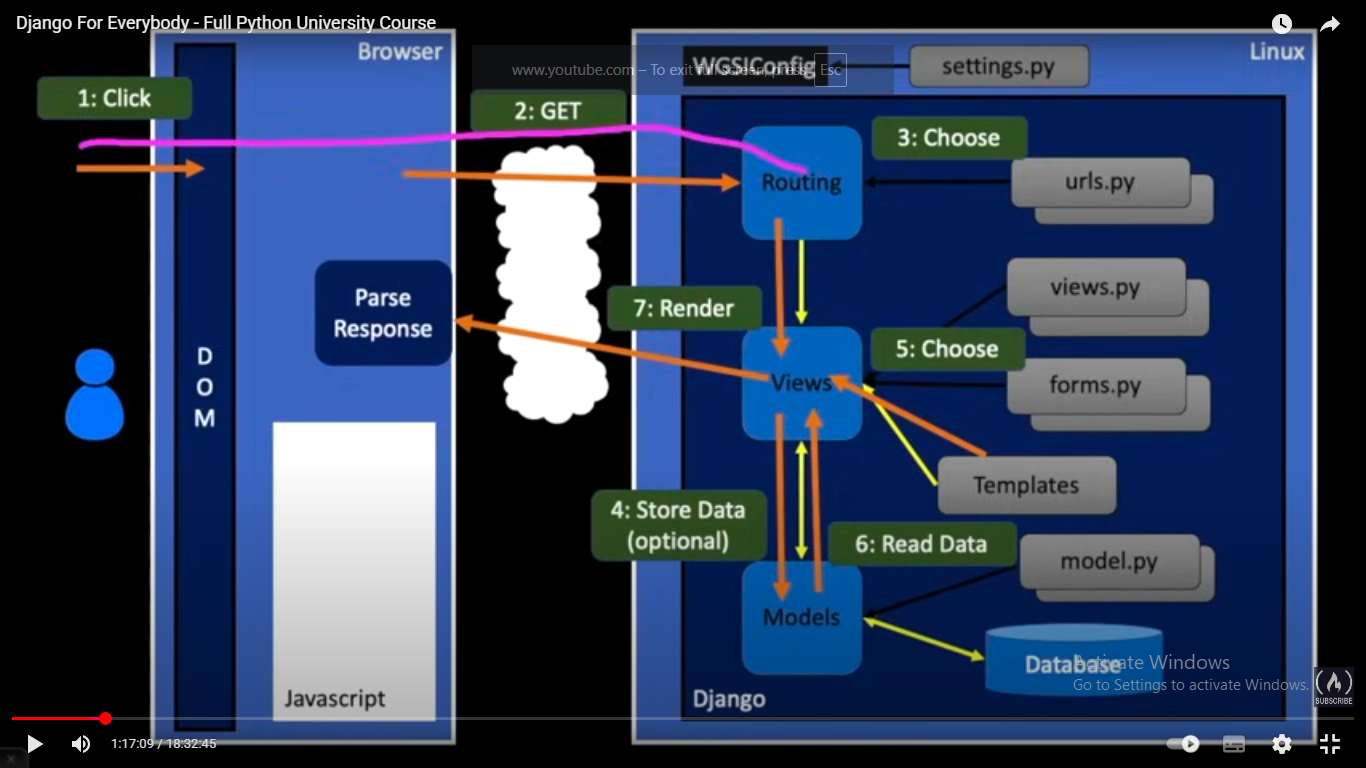
**Content-Type: text/html; charset=utf-8**

**<html><body>Hello World</body></html>**

* **HTTP/1.1 200 OK → Status line**
  + **HTTP/1.1 = protocol version.**
  + **200 OK = success status code (request worked).**
* **Headers**
  + **Content-Type: text/html; charset=utf-8 tells the client that the content is HTML text encoded in UTF-8.**
* **Body**
  + **<html><body>Hello World</body></html> → actual webpage content returned by the server.**

**—----------------------------------------------------------------------------------------------------**

**Django Request Response Cycle:**

****

**—---------------------------------------------------------------------------------------**

**HTML**

[**https://www.wa4e.com/code/css/newtags.htm**](https://www.wa4e.com/code/css/newtags.htm)

[**https://www.wa4e.com/lessons**](https://www.wa4e.com/lessons)

[**https://www.wa4e.com/code/css/index.htm**](https://www.wa4e.com/code/css/index.htm)

**CSS**

[**https://www.wa4e.com/code/css/classid.htm**](https://www.wa4e.com/code/css/classid.htm)

[**https://www.wa4e.com/code/css/navbar.htm**](https://www.wa4e.com/code/css/navbar.htm)

**Use of** [**https://chrispederick.com/**](https://chrispederick.com/) **extension**

**SQL**