Python Full stack

- By Bhavna Vaishnav

Python stack

4 layers :

How the web works

Frontend

Backend (Python)

Databases, Deployment and real world engg

Chapters :

1. Web Fundamentals
2. Frontend Basics (Refer another notebook)
3. Python Core (Refer another notebook)
4. Backend Frameworks
   - Flask
   - Django - FastAPI
5. Databases
   - SQL - others
   - ORM

   } Refer another notebook
6. Rest APIs
7. Authentication & security
8. Version Control
9. Deployment and DevOps Basics
10. Testing & Best Practices.

Chapter 1.1

How browsers Talk to servers

1. The Big Picture (First Principles)

The web is just communication between two programs.

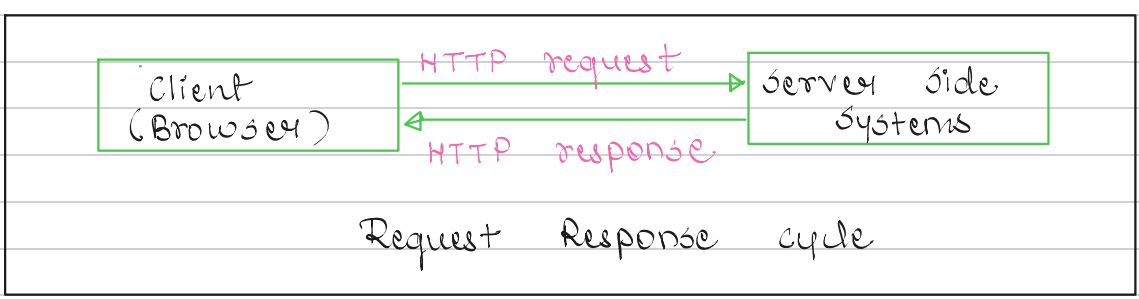
1. Browser - a program running on your laptop / phone
2. Server - program running on another computer somewhere on the internet.

They do not share memory.
They do not know each other
They only talk using messages.

Those messages follow strict rules called HTTP.

2. Who starts the conversation?

Always the browser
A server never talks randomly to a browser.



Request Response cycle

Example :

- You type google.com
- You press Enter
- Browser sends a message
- Server replies
- Conversation ends

This is called a request - response model.

What Exactly is a Request?

A request is a structured text message.

Example:

```
Get /search?q=python HTTP/1.1
Host: google.com
User-Agent: Chrome
```

This says:

- GET → I want data
- /search?q=python → this specific resource
- Host → which server
- Headers → extra information about me.

What does the server do?

1. Reads the request
2. Decides what logic to run
3. Generates a response
4. Sends it back.

Each request is treated like:

"Who are you? What do you want?"

Server does not remember you by default.

That's why HTTP is stateless.

What is a Response?

A response is also a structured message.

Example:

```
HTTP/1.1 200 OK
Content-Type: text/html
<html> </html>
```

It has:
- Status code → result of the request
- Headers → metadata
- Body → actual content.

Status Codes (Critical Concept)

Status codes tell the browser what happened.

Code	Meaning
200	Success
301	Redirect
400	Bad request
401	Unauthorized
403	Forbidden
404	Not found
500	Server error

If you understand status code, you can debug 50% of web issues instantly.

What happens when you open a website?

1. Browser asks DNS: "where is google.com"
2. Browser sends HTTP request to that IP
3. Server responds with HTML
4. Browser parses HTML
5. Browser requests CSS, JS, images (more req)
6. Server responds to each.
7. Browser renders the page.

A single page load can trigger 50-200 HTTP req.

Web development = controlling how server responds to req

What logic goes where ?

Client Responsibility

- UI rendering
- Form input
- Button clicks
- Basic validation
- Sending requests

Server responsibility

- Authentication
- Database access
- Business rules
- Security checks
- Data processing

Chapter 1.2

Http & Https

What is HTTP ?

HTTP = HyperText Transfer Protocol

It is a protocol. A protocol is just a set of rules
- How to send data
- in what format
- in what order
- How to understand errors.

What problems does HTTP solve ?

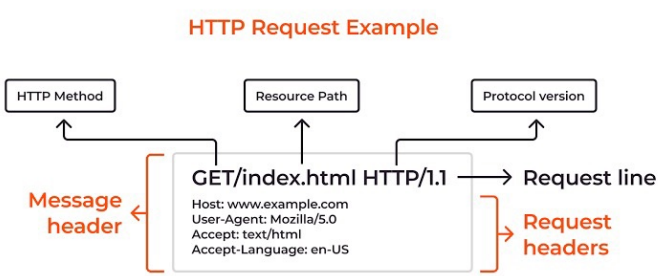
HTTP answers these questions :
- How does the browser ask for a page ?
- How does the server reply ?
- How do both know when communication is complete ?
- How do they report success or failure ?

HTTP is the language browsers and servers speak.

HTTP is stateless

stateless = server does not remember previous requests

Structure of an HTTP request :



```
POST /?id=1 HTTP/1.1
```
Request line

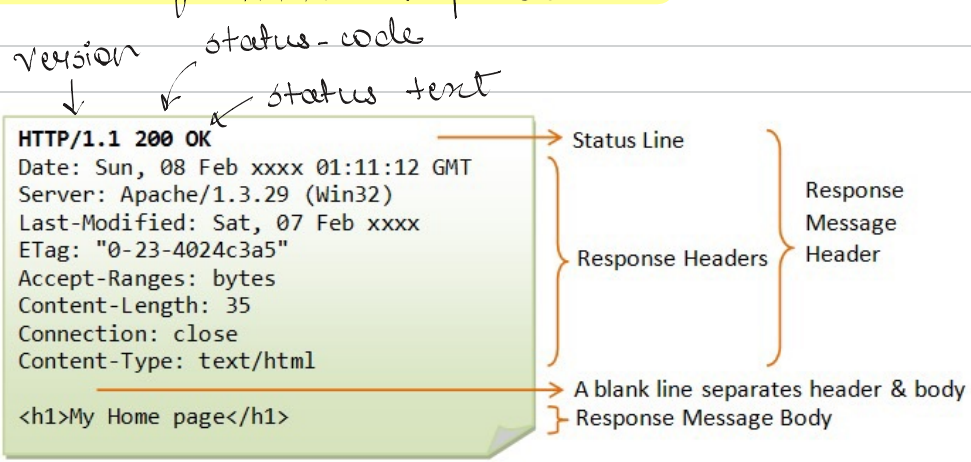
```
Host: www.swingvy.com
Content-Type: application/json; charset=utf-8
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:53.0)
Gecko/20100101 Firefox/53.0
Connection: close
Content-Length: 136
```
Header

```
{
  "status": "ok",
  "extended": true,
  "results": [
    {"value": 0, "type": "int64"},
    {"value": 1.0e+3, "type": "decimal"}
  ]
}
```
Body message

HTTP methods :

Method	Meaning
GET	Ask for data
POST	Send data
PUT	Update data
DELETE	Remove data
PATCH	Partial update

structure of HTTP response :



The Browser :
- Reads status code
- interprets headers
- Renders body

Big Problem with HTTP

It is a plain text which means anyone on the internet can read it, Passwords can be stolen, data can be modified.

What is HTTPS ?

HTTPS = HTTP + encryption
The extra S stands for Secure.
HTTPS uses : SSL / TLS
Public - key cryptography
Encryption + integrity + authenticity
Basically, same HTTP rules but wrapped in encryption.

Simplified flow :
1. Browser connects to server
2. Server sends certificate
3. Browser verifies certificate
4. Encryption keys are exchanged
5. Secure tunnel is created
6. HTTP data flows inside this tunnel.

After this :
- No one can read the data
- No one can modify it
- You know you're talking to the real server.

Feature	HTTP	HTTPS
Encryption	No	Yes
Security	Unsafe	Secure
Speed	Slightly faster	Negligible difference
SEO	Penalized	Preferred
Modern Usage	Deprecated	Standard

Request / Response Cycle:

It is the complete journey of a browser's request to a server and the server's reply back to browser

> One request → One response

- No response without request
- No second response for the same request

Step by step (for one real request):

Step 1: User Action

You:
- Type a URL
- Click a link
- Submit a form
- Click a button

Browser decides:
"I need to send a request"

Step 2: DNS Resolution

Browser asks:
"Where is this domain located?"

DNS replies with IP add.

No IP → no request

Step 3: TCP Connection

Browser:
- Opens a conn to server
- TLS handshake if HTTPS

This is transport layer

Step 4: HTTP Request is sent

Browser is waiting
Nothing is rendered yet

Step 5: Server Processing

On the server:
1. Request is received
2. URL is matched
3. Authentication checked
4. Business logic executed
5. Database queried
6. Response prepared

This is where:
Flask, Django, FastAPI do their job.

Step 6: HTTP Response is sent

Server sends response and forgets the request.

Step 7: Browser Handles Response

Browser:
- Reads status code
- Parses header
- Renders HTML
- Executes JS
- Requests additional resources

Each CSS/JS/img triggers new request/response cycles.

Browser → Request → Server
Browser ← Response ← Server

Repeat this hundres of times per page.

What Breaks the cycle?

Issue	Result
DNS failure	Request never sent
Network error	No response
Server crash	500 error
Timeout	Browser gives up

When you refresh :
- Browser sends a new request
- Server sends a new response

What is a Status Code ?

A status code is a number sent by the server that answers one question :
"What happened to request ?"
Browser always looks at the status code first, even before reading the content

Status Code Families

Range	Meaning
1xx	Information
2xx	Success
3xx	Redirection
4xx	Client error
5xx	Server error

404 - Not found (Client Mistake)
Server is working, Code is running, Requested resource does not exist.

Real World Analogy

200 → Package delivered
404 → Address doesn't exist
500 → Warehouse problem

URL - The address of the web :

A URL tells the browser where to send the request and what to ask for.

Cookies - Memory for stateless HTTP

HTTP is stateless. Cookies add memory.

A cookie is :

- small piece of data
- stored in browser
- sent with every request to that domain.

URL = where
Query params = filters
Headers = metadata
Cookies = memory