REQUEST IN PYTHON

MOHAMMAD HASAN KHODDAMI

introduction

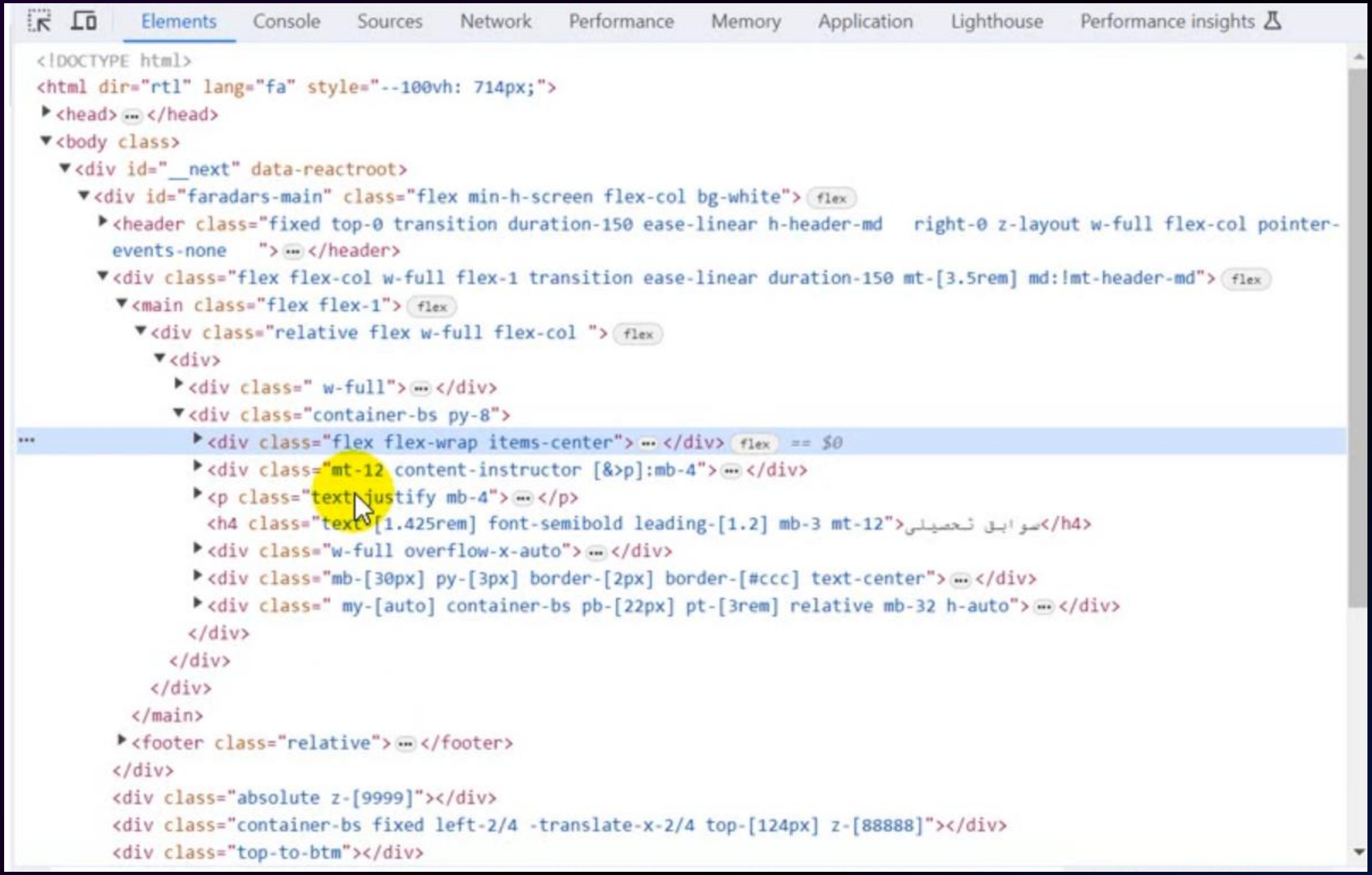
In this presentation we will learn about request library and important concept of API and how to use it.

HTML

- What is HTML ? HTML is abbreviation of Hyper Text Markup language
- Now what is the hyper text ? Hyper text include long html codes
- Note : HTML is not a language programing cause it doesn`t have Loop or function or etc.
- Html uses for create template of websites
- How we can see the html codes of a website ?
  - First open the website
    - Then write click and press on inspect
      - Congrats! Now you can see the html codes of web site

EXAMPLE

As you can see there is no loop no function and no etc.
That’s why we call html markup language
Which means take part of templates together



```
<!DOCTYPE html>
<html dir="rtl" lang="fa" style="--100vh: 714px;">
 ▶<head> … </head>
 ▼<body class>
   ▼<div id="__next" data-reactroot>
     ▼<div id="faradars-main" class="flex min-h-screen flex-col bg-white"> flex
       ▶<header class="fixed top-0 transition duration-150 ease-linear h-header-md   right-0 z-layout w-full flex-col pointer-
        events-none   "> … </header>
       ▼<div class="flex flex-col w-full flex-1 transition ease-linear duration-150 mt-[3.5rem] md:!mt-header-md"> flex
         ▼<main class="flex flex-1"> flex
           ▼<div class="relative flex w-full flex-col "> flex
             ▼<div>
               ▶<div class=" w-full"> … </div>
               ▼<div class="container-bs py-8">
…                ▶<div class="flex flex-wrap items-center"> … </div> flex == $0
                 ▶<div class="mt-12 content-instructor [&>p]:mb-4"> … </div>
                 ▶<p class="text-justify mb-4"> … </p>
                  <h4 class="text-[1.425rem] font-semibold leading-[1.2] mb-3 mt-12">سوابق تحصیلی</h4>
                 ▶<div class="w-full overflow-x-auto"> … </div>
                 ▶<div class="mb-[30px] py-[3px] border-[2px] border-[#ccc] text-center"> … </div>
                 ▶<div class=" my-[auto] container-bs pb-[22px] pt-[3rem] relative mb-32 h-auto"> … </div>
               </div>
             </div>
           </div>
         </main>
       ▶<footer class="relative"> … </footer>
       </div>
       <div class="absolute z-[9999]"></div>
       <div class="container-bs fixed left-2/4 -translate-x-2/4 top-[124px] z-[88888]"></div>
       <div class="top-to-btm"></div>
```

HTTP

- HTTP is abbreviation of Hyper Text Transfer Protocol
- Reminder → Hyper text is HTML and HTTP is a protocol (a process for transferring the HTML) it`s a gate between client and server.
- Now the question is how to improve the security of this protocol?

SSL

- SSL is abbreviation of Secure Socket Layer
- SSL make the information get Hashed and don`t let hackers to attack the information.
- There is different types of algorithms which use in SSL.

HTTPS

- HTTPS = HTTP + SSL
- HTTPS is abbreviation of Hyper Text Transfer Protocol Secure
- It`s a safe protocol for transferring the hyper text(HTML)

$$\begin{tikzcd}
\text{Server} \arrow[r, shift left, "\text{Hashed information}"] & \text{Client} \arrow[l, shift left]
\end{tikzcd}$$

The information is observable just from client and server

CLIENT AND SERVER

CLIENT

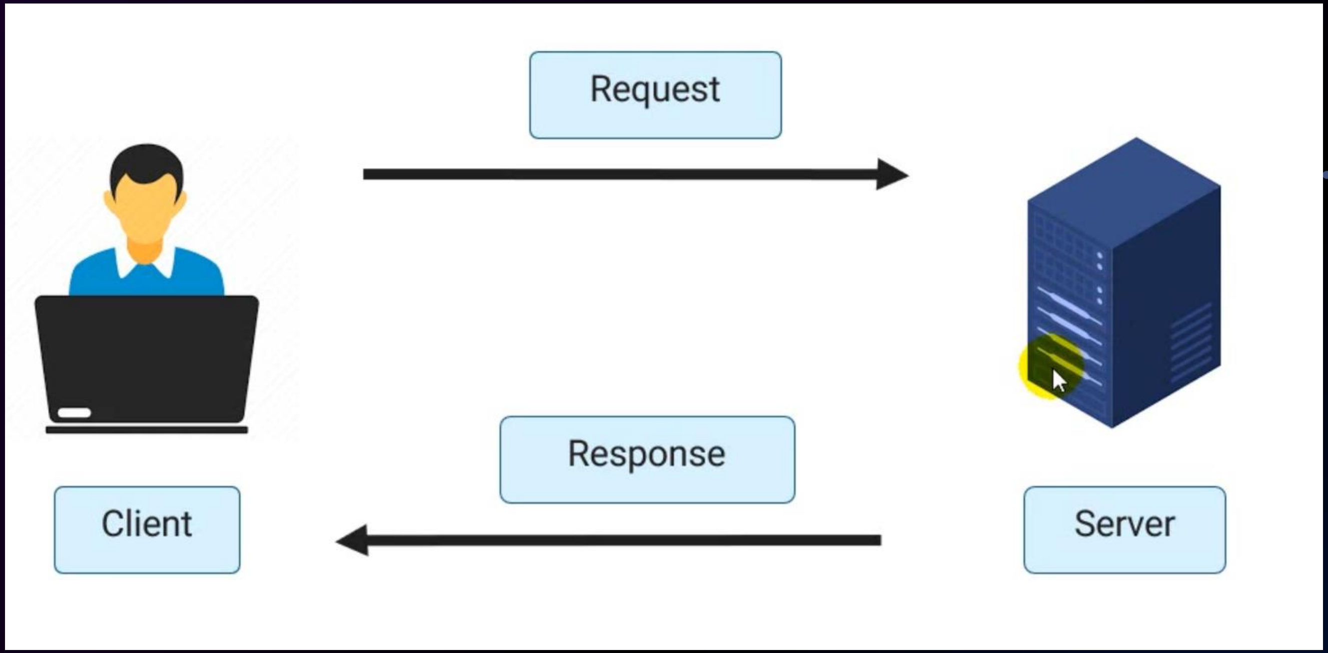
- Client is a user → our phone our computer or everything which use a server , service is a client

SERVER

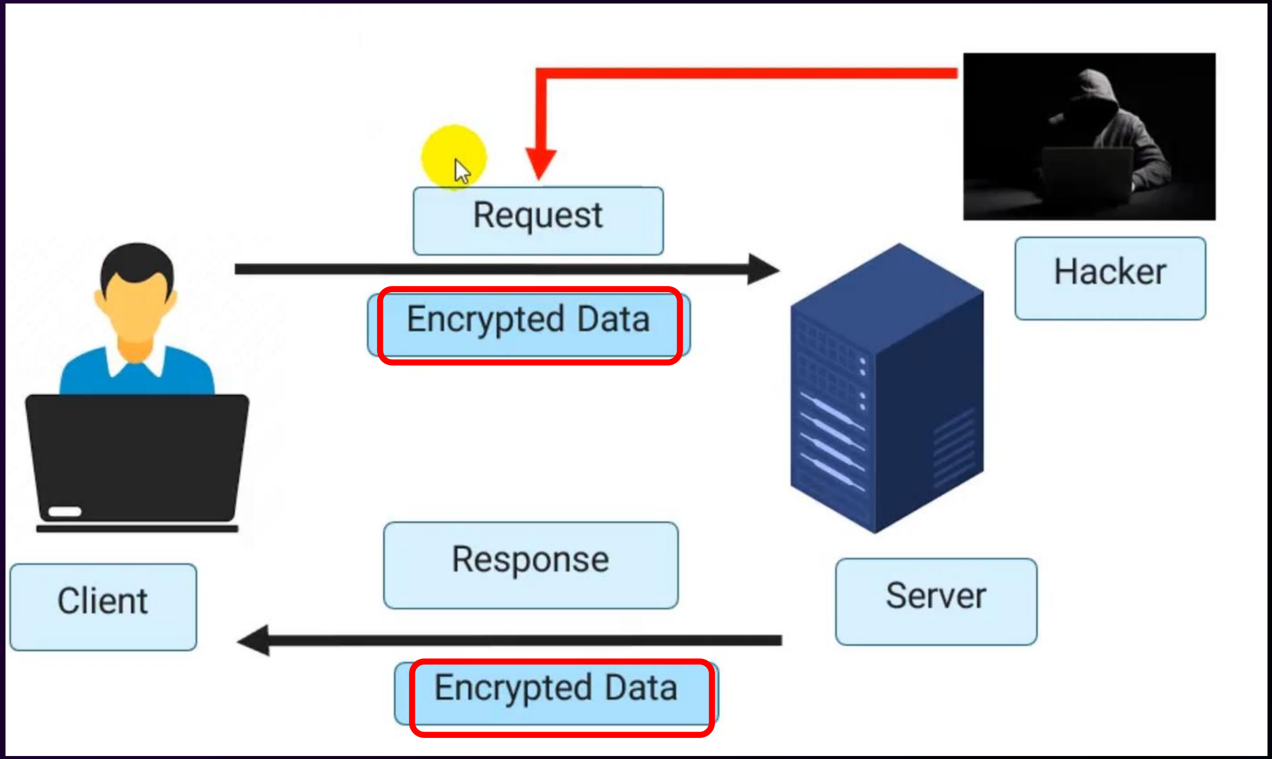
- Every websites or services(like telegram , Instagram and etc.) which we use it everyday is a server.
- Sites : like github , google and etc.

CONNECTING CLIENT AND SERVER BY HTTP

Client sends a request
Server receive the request
And server Response it (sends a response)



CONNECTING CLIENT AND SERVER BY HTTPS



I think everything is clear !

WHAT IS REQUEST AND RESPONSE

- Client sends a request to server by https or http and the purpose is observing some data
- Server sends response to client by http or https
- And our browser make it understandable for us

REQUEST & RESPONSE CONSTRUCTION

- Request and Response contains 2 parts body and header
- Header contains a general data like the type of request , version of browser , login information and etc.
- Every person who signs in a website a token will be created in their browser
- In header part this token requests will send to server
- Token is represented of a authenticated client
- This token rapidly send to server (when you login in website you will remember on that website until you turn off you pc or log out the account)

HEADER VS BODY

HEADER

- Header contains data likes :
  - Authentication , cookies , user-agent , accept-language , Host

BODY

- Body contains other data like forms , address , phone number and etc.
- Every text that we write down in a website is part of body

REQUEST HEADER

HOST

- Shows the Host of Server

ACCEPT-LANGUAGE

- Is the language which websites shows to client

REQUEST HEADER

USER-AGENT

- Shows device and browser data
  - Example : HP LAPTOP , Google chrome…

COOKIE

- Cookies are the personal data of client like username and password
  - When client login in a website he/she username or password are cookie and cookie is the token! If client remove cookie manually and refresh the website he/she has to login again
  - We can conclude that cookie is the token

REQUEST HEADER

AUTHENTICATE

- Sent encoded username and password to server

RESPONSE

HEADER

- STATUS CODE
- SERVER
- SET-COOKIE
- CONTENT-LENTGH
- WWW-AUTHENTICATE

BODY

- HTML FOR SHOWING CLEINT

RESPONSE HEADER

STATUS CODE

- It`s https status code like http status code 200 which shows successful request
- For knowing all status code please read my Article in virgol
- https://virgool.io/@Niklaus/http-status-codes-fo0o0tvji674

SERVER

- Sending some data about server to client

RESPONSE HEADER

SET-COOKIE

- Installing requirements cookie on client browser , like token
- As we said username and password is a cookie

CONTENT-LENTGH

- Is the length of body

RESPONSE HEADER

WWW-AUTHENTICATE

- Shows the algorithm of authentication

GENERAL HEADERS

- General headers are headers which use in request and response

1. Connection : Reports Connection between server and client (Close or Keep alive)
   1. If you client disconnected the connection type is close if connected , connection type is keep alive
2. Date : date of sending request and getting response
3. Cache control : is Content is cache able or not
4. Content-type : shows the content type !

TYPE OF REQUEST

- We have 4 main request methods :
  - 1- GET
  - 2- POST
  - 3- PUT
  - 4- DELETE
- And we have some sub method
  - 1- PATCH
  - 2-OPTION
  - 3- HEAD

GET METHOD

- When client want to receive data from sever have to use get method
- Usually in this method server data`s won`t change (this method use just for read)
- Server has to gives client permission to use get method (read the target url)

POST METHOD

- In post method client want to send data to server like : username and password
- This data sends in body part of request not header
- In this method client adds some information to server like inserting comment or even username and password
- Server has to give client permission to use POST method

PUT METHOD

- PUT method is used for editing data in server like editing password , phone number or etc.
- As the other method server has to give client permission to use PUT method.

DELETE METHOD

- DELETE method is used for delete some data from server
- Note that : as a client we can`t delete whole server just the data`s that belong to us
- Server has to give client permission for DELETE method.

PATCH METHOD

- PATCH method is like put method but we want to upgrade a few information.

OPTION METHOD

- In OPTION method clients can ask server which method are allowed to use here !
  - Example make it more clear
    - For example I open the youtbue.com and use OPTION method on that
    - Youtube will answer which method that I can use on website
      - if I don`t have an account I can just watch videos so I can use GET (read only)
      - If I have an account I can watch videos and share my videos so I can use GET and POST also I can edit my captions or delete my videos which means I can use PUT and DELETE too.
- IN HEADER WE CAN SEE THE RESPONSE

HEAD METHOD

- HEAD method is like the GET method but we just receive the header data which means don`t use body part . Just header! (server sends a response of header)

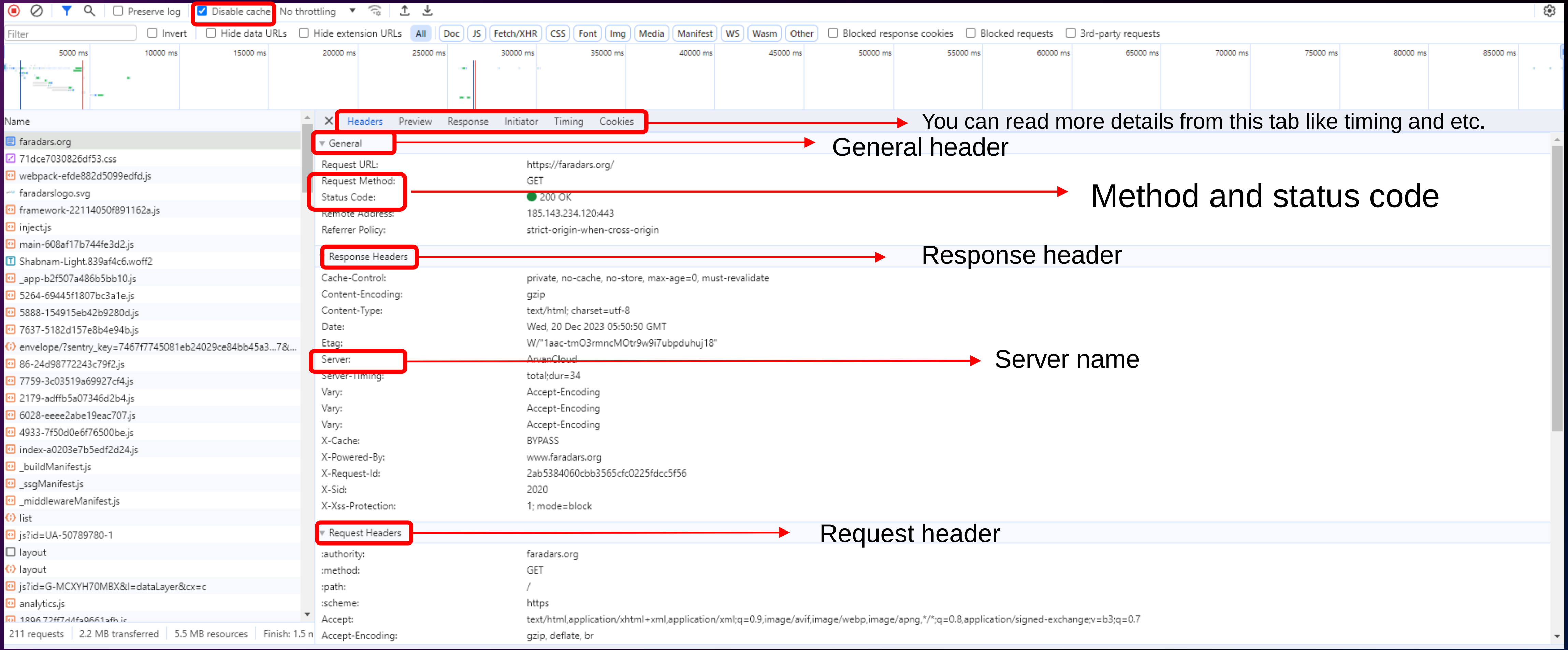
STATUS 200

- If server gives clients permission for one named methods http status code is 200

REALESTIC EXAMPLE

- Now we want to inspect a website codes and explain the request from that
- First we open a website then click on inspect then click on network (this is for understanding better)

REALESTIC EXAMPLE



You can read more details from this tab like timing and etc.

General header

Method and status code

Response header

Server name

Request header

```
faradars.org
71dce7030826df53.css
webpack-efde882d5099edfd.js
faradarslogo.svg
framework-22114050f891162a.js
inject.js
main-608af17b744fe3d2.js
Shabnam-Light.839af4c6.woff2
_app-b2f507a486b5bb10.js
5264-69445f1807bc3a1e.js
5888-154915eb42b9280d.js
7637-5182d157e8b4e94b.js
envelope/?sentry_key=7467f7745081eb24029ce84bb45a3...7&...
86-24d98772243c79f2.js
7759-3c03519a69927cf4.js
2179-adffb5a07346d2b4.js
6028-eeee2abe19eac707.js
4933-7f50d0e6f76500be.js
index-a0203e7b5edf2d24.js
_buildManifest.js
_ssgManifest.js
_middlewareManifest.js
list
js?id=UA-50789780-1
layout
layout
js?id=G-MCXYH70MBX&l=dataLayer&cx=c
analytics.js
1806 72ff7d4fa0661afb.js

211 requests | 2.2 MB transferred | 5.5 MB resources | Finish: 1.5 n
```

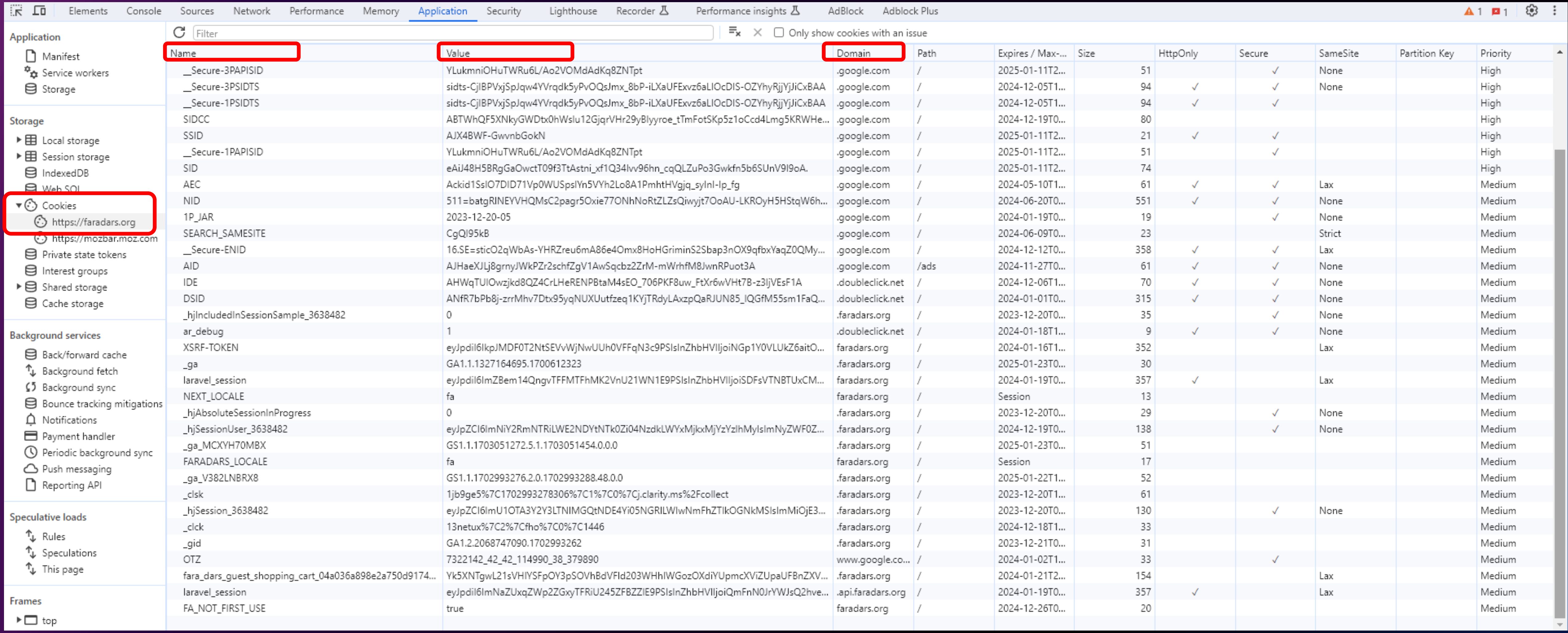
```
▼ General
Request URL:         https://faradars.org/
Request Method:      GET
Status Code:         ● 200 OK
Remote Address:      185.143.234.120:443
Referrer Policy:     strict-origin-when-cross-origin

Response Headers
Cache-Control:       private, no-cache, no-store, max-age=0, must-revalidate
Content-Encoding:    gzip
Content-Type:        text/html; charset=utf-8
Date:                Wed, 20 Dec 2023 05:50:50 GMT
Etag:                W/"1aac-tmO3rmncMOtr9w9i7ubpduhuj18"
Server:              ArvanCloud
Server-Timing:       total;dur=34
Vary:                Accept-Encoding
Vary:                Accept-Encoding
Vary:                Accept-Encoding
X-Cache:             BYPASS
X-Powered-By:        www.faradars.org
X-Request-Id:        2ab5384060cbb3565cfc0225fdcc5f56
X-Sid:               2020
X-Xss-Protection:    1; mode=block

▼ Request Headers
:authority:          faradars.org
:method:             GET
:path:               /
:scheme:             https
Accept:              text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding:     gzip, deflate, br
```

Practice

- Open a website inspect and read the network part and compare it with request header and response header.

INSPECT- APPLICATION

- For seeing cookies in better way we open the application in inspect



Name	Value	Domain	Path	Expires / Max-...	Size	HttpOnly	Secure	SameSite	Partition Key	Priority
__Secure-3PAPISID	YLukmniOHuTWRu6L/Ao2VOMdAdKq8ZNTpt	.google.com	/	2025-01-11T2...	51		✓	None		High
__Secure-3PSIDTS	sidts-CjIBPVxjSpJqw4YVrqdk5yPvOQsJmx_8bP-iLXaUFExvz6aLIOcDIS-OZYhyRjjYjJiCxBAA	.google.com	/	2024-12-05T1...	94	✓	✓	None		High
__Secure-1PSIDTS	sidts-CjIBPVxjSpJqw4YVrqdk5yPvOQsJmx_8bP-iLXaUFExvz6aLIOcDIS-OZYhyRjjYjJiCxBAA	.google.com	/	2024-12-05T1...	94	✓	✓			High
SIDCC	ABTWhQF5XNkyGWDtx0hWslu12GjqrVHr29yBlyyroe_tTmFotSKp5z1oCcd4Lmg5KRWHe...	.google.com	/	2024-12-19T0...	80					High
SSID	AJX4BWF-GwvnbGokN	.google.com	/	2025-01-11T2...	21	✓	✓			High
__Secure-1PAPISID	YLukmniOHuTWRu6L/Ao2VOMdAdKq8ZNTpt	.google.com	/	2025-01-11T2...	51		✓			High
SID	eAiJ48H5BRgGaOwctT09f3TtAstni_xf1Q34lvv96hn_cqQLZuPo3Gwkfn5b6SUnV9I9oA.	.google.com	/	2025-01-11T2...	74					High
AEC	Ackid1SsIO7DID71Vp0WUSpslYn5VYh2Lo8A1PmhtHVgjq_syInl-Ip_fg	.google.com	/	2024-05-10T1...	61	✓	✓	Lax		Medium
NID	511=batgRINEYVHQMsC2pagr5Oxie77ONhNoRtZLZsQiwyjt7OoAU-LKROyH5HStqW6h...	.google.com	/	2024-06-20T0...	551	✓	✓	None		Medium
1P_JAR	2023-12-20-05	.google.com	/	2024-01-19T0...	19		✓	None		Medium
SEARCH_SAMESITE	CgQI95kB	.google.com	/	2024-06-09T0...	23			Strict		Medium
__Secure-ENID	16.SE=sticO2qWbAs-YHRZreu6mA86e4Omx8HoHGriminS2Sbap3nOX9qfbxYaqZ0QMy...	.google.com	/	2024-12-12T0...	358	✓	✓	Lax		Medium
AID	AJHaeXJLj8grnyJWkPZr2schfZgV1AwSqcbz2ZrM-mWrhfM8JwnRPuot3A	.google.com	/ads	2024-11-27T0...	61	✓	✓	None		Medium
IDE	AHWqTUIOwzjkd8QZ4CrLHeRENPBtaM4sEO_706PKF8uw_FtXr6wVHt7B-z3ljVEsF1A	.doubleclick.net	/	2024-12-06T1...	70	✓	✓	None		Medium
DSID	ANfR7bPb8j-zrrMhv7Dtx95yqNUXUutfzeq1KYjTRdyLAxzpQaRJUN85_IQGfM55sm1FaQ...	.doubleclick.net	/	2024-01-01T0...	315	✓	✓	None		Medium
_hjIncludedInSessionSample_3638482	0	.faradars.org	/	2023-12-20T0...	35		✓	None		Medium
ar_debug	1	.doubleclick.net	/	2024-01-18T1...	9	✓	✓	None		Medium
XSRF-TOKEN	eyJpdiI6IkpJMDF0T2NtSEVvWjNwUUh0VFFqN3c9PSIsInZhbHVlIjoiNGp1Y0VLUkZ6aitO...	faradars.org	/	2024-01-16T1...	352			Lax		Medium
_ga	GA1.1.1327164695.1700612323	.faradars.org	/	2025-01-23T0...	30					Medium
laravel_session	eyJpdiI6ImZBem14QngvTFFMTFhMK2VnU21WN1E9PSIsInZhbHVlIjoiSDFsVTNBTUxCM...	faradars.org	/	2024-01-19T0...	357	✓		Lax		Medium
NEXT_LOCALE	fa	faradars.org	/	Session	13					Medium
_hjAbsoluteSessionInProgress	0	.faradars.org	/	2023-12-20T0...	29		✓	None		Medium
_hjSessionUser_3638482	eyJpZCI6ImNiY2RmNTRiLWE2NDYtNTk0Zi04NzdkLWYxMjkxMjYzYzlhMyIsImNyZWF0Z...	.faradars.org	/	2024-12-19T0...	138		✓	None		Medium
_ga_MCXYH70MBX	GS1.1.1703051272.5.1.1703051454.0.0.0	.faradars.org	/	2025-01-23T0...	51					Medium
FARADARS_LOCALE	fa	faradars.org	/	Session	17					Medium
_ga_V382LNBRX8	GS1.1.1702993276.2.0.1702993288.48.0.0	.faradars.org	/	2025-01-22T1...	52					Medium
_clsk	1jb9ge5%7C1702993278306%7C1%7C0%7Cj.clarity.ms%2Fcollect	.faradars.org	/	2023-12-20T1...	61					Medium
_hjSession_3638482	eyJpZCI6ImU1OTA3Y2Y3LTNIMGQtNDE4Yi05NGRILWIwNmFhZTIkOGNkMSIsImMiOjE3...	.faradars.org	/	2023-12-20T0...	130		✓	None		Medium
_clck	13netux%7C2%7Cfho%7C0%7C1446	.faradars.org	/	2024-12-18T1...	33					Medium
_gid	GA1.2.2068747090.1702993262	.faradars.org	/	2023-12-21T0...	31					Medium
OTZ	7322142_42_42_114990_38_379890	www.google.co...	/	2024-01-02T1...	33		✓			Medium
fara_dars_guest_shopping_cart_04a036a898e2a750d9174...	Yk5XNTgwL21sVHlYSFpOY3pSOVhBdVFld203WHhlWGozOXdiYUpmcXViZUpaUFBnZXV...	.faradars.org	/	2024-01-21T2...	154			Lax		Medium
laravel_session	eyJpdiI6ImNaZUxqZWp2ZGxyTFRiU245ZFBZZIE9PSIsInZhbHVlIjoiQmFnN0JrYWJsQ2hve...	.api.faradars.org	/	2024-01-19T0...	357	✓		Lax		Medium
FA_NOT_FIRST_USE	true	faradars.org	/	2024-12-26T0...	20					Medium

EXPLANATION OF LAST SLIDE

- As you know cookies are the tokens. (actually cookies are storing tokens but for understanding better we say cookies are tokens !)
- Cookies include a name and value from given domain
- Now if we login in faradars there will be a new cookie which is our username and password
- As a practice open a website check the cookies then login on that website then check it again.

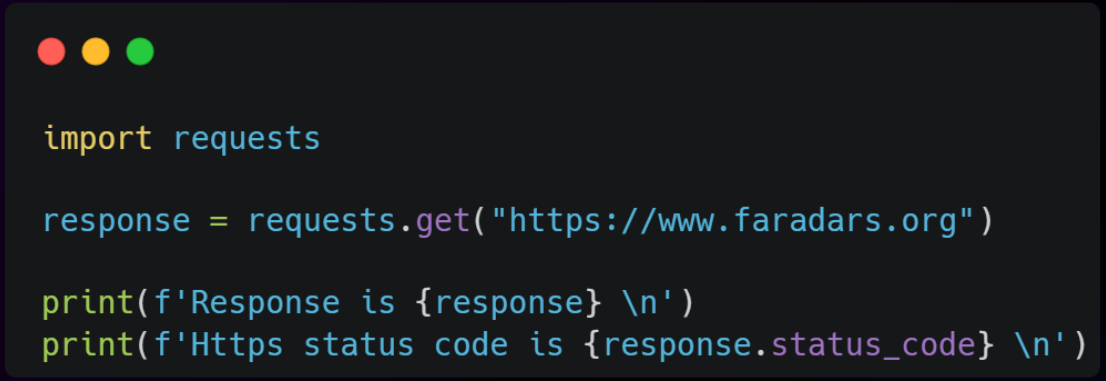
SENDING REQUEST BY GET

- Requirements :
  - 0- INTERNET !
  - 1- VS Code
  - 2- Python
  - 3- Requests library(module)
    - For installing Requests module use following command on cmd:
      - pip install requests

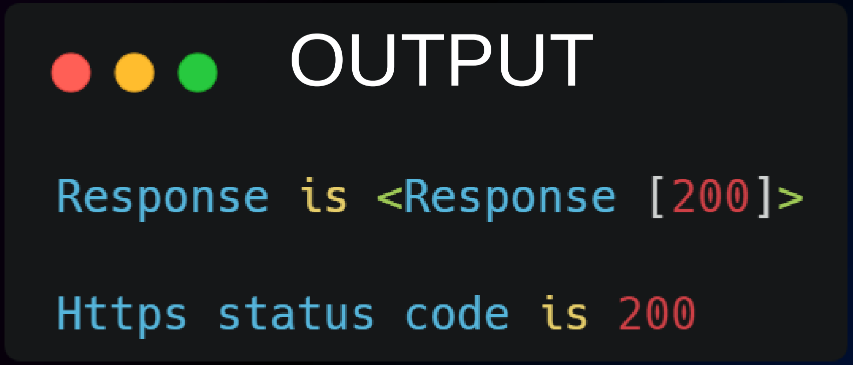
SENDING REQUEST BY GET

First we have to import requests

Then we using get built in function the requirement element is URL of the website as string
Then we print response and status code
It`s 200 !



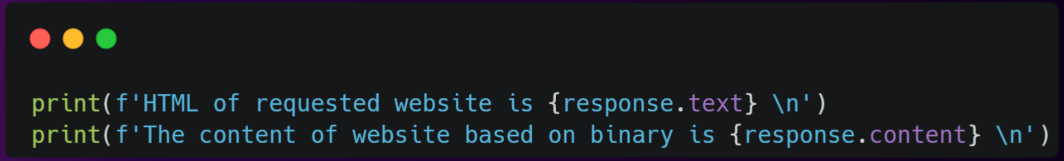
```python
import requests

response = requests.get("https://www.faradars.org")

print(f'Response is {response} \n')
print(f'Https status code is {response.status_code} \n')
```



OUTPUT

```
Response is <Response [200]>

Https status code is 200
```

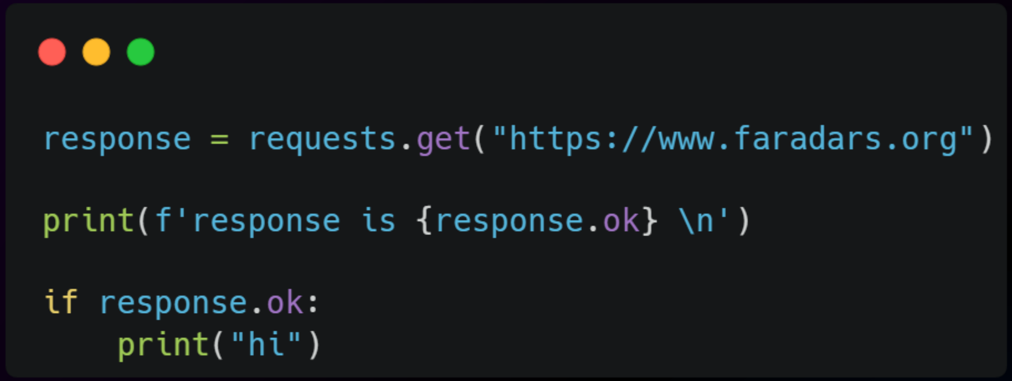
NOTE: also we can get the HTML of the website and content based on binary but the output is huge!
So you have to try it by yourself



```python
print(f'HTML of requested website is {response.text} \n')
print(f'The content of website based on binary is {response.content} \n')
```

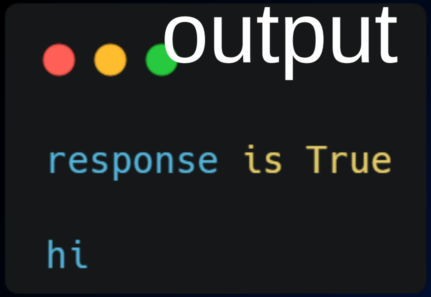
RESPONSE.OK

Response.ok menas if response is 200 return me TRUE
if it`s not Return me False.



```
response = requests.get("https://www.faradars.org")

print(f'response is {response.ok} \n')

if response.ok:
    print("hi")
```

output



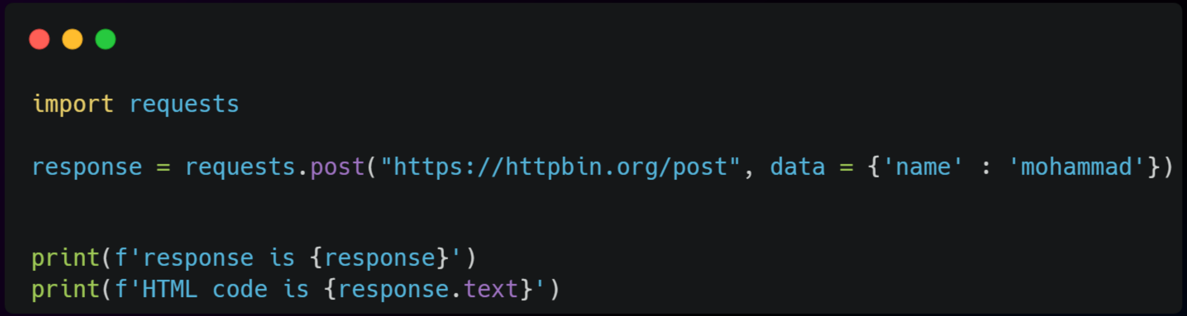
```
response is True

hi
```

SENDING REQUEST BY POST, PUT , PATCH

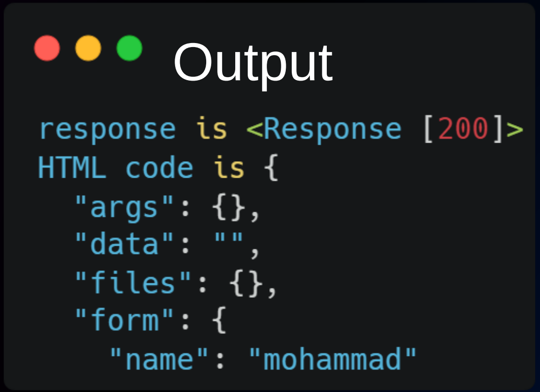
- As we said if we want to send some data`s to server using POST method
- In post method we have to send data and usually our data is dictionary (or Json)
  - Know as key and value
- Many websites don`t let us to use POST method so we can use httpbin.org
  - It`s free source website for education which let us to use POST method

SENDING REQUEST BY POST

- In request.post
  - First we write down or url
  - Second we send the data as context manager
  - Finite !



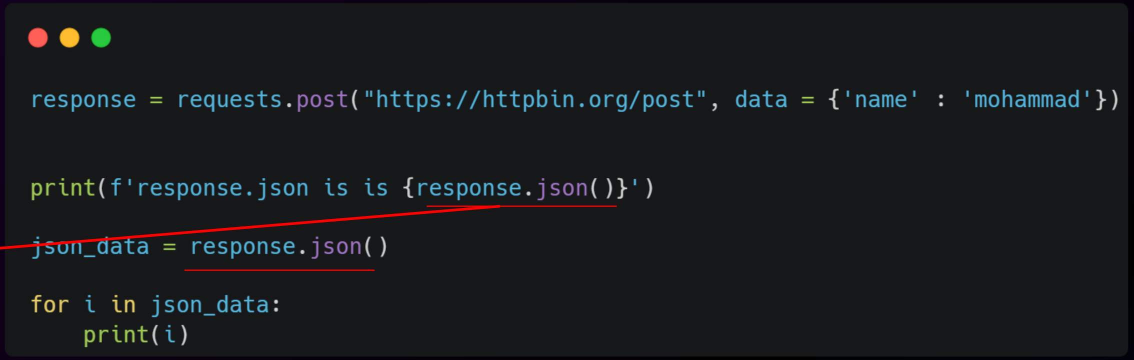
```python
import requests

response = requests.post("https://httpbin.org/post", data = {'name' : 'mohammad'})


print(f'response is {response}')
print(f'HTML code is {response.text}')
```



Output

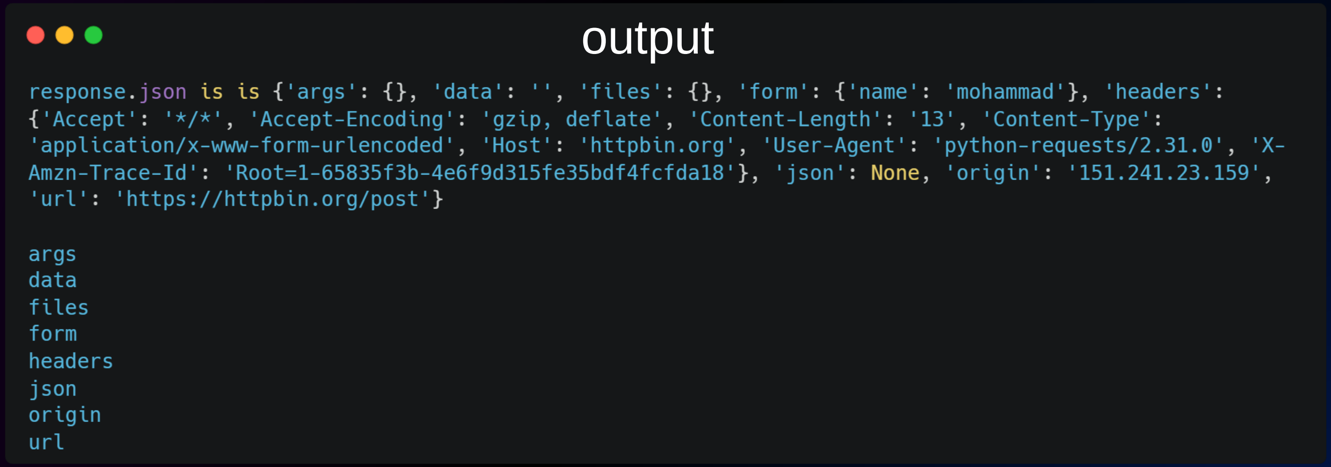
```
response is <Response [200]>
HTML code is {
  "args": {},
  "data": "",
  "files": {},
  "form": {
    "name": "mohammad"
```

RESPONSE.JSON



```
response = requests.post("https://httpbin.org/post", data = {'name' : 'mohammad'})


print(f'response.json is is {response.json()}')

json_data = response.json()

for i in json_data:
    print(i)
```

.json is built in function



output

```
response.json is is {'args': {}, 'data': '', 'files': {}, 'form': {'name': 'mohammad'}, 'headers':
{'Accept': '*/*', 'Accept-Encoding': 'gzip, deflate', 'Content-Length': '13', 'Content-Type':
'application/x-www-form-urlencoded', 'Host': 'httpbin.org', 'User-Agent': 'python-requests/2.31.0', 'X-
Amzn-Trace-Id': 'Root=1-65835f3b-4e6f9d315fe35bdf4fcfda18'}, 'json': None, 'origin': '151.241.23.159',
'url': 'https://httpbin.org/post'}

args
data
files
form
headers
json
origin
url
```

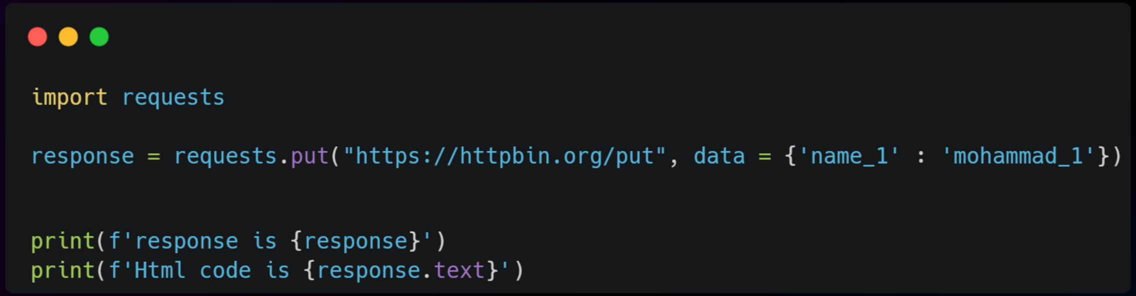
Why do we use Json ?

- Cause data are to important or complicated
- Some Websites just support json

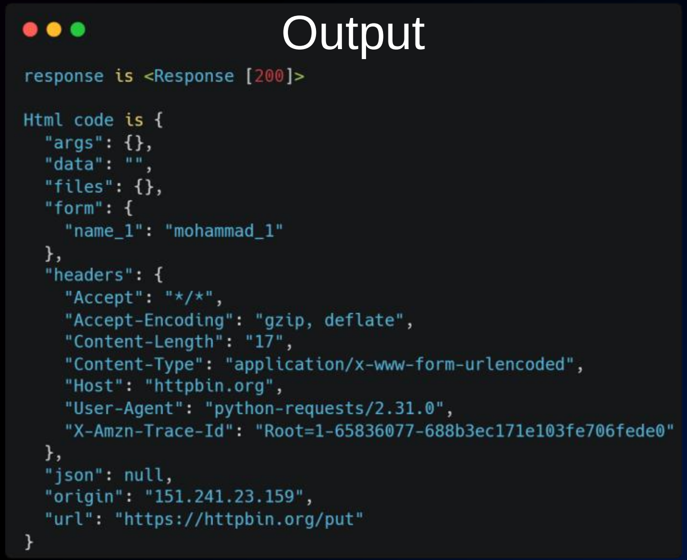
SENDING REQUEST BY PUT

- As we said we use PUT when we want to edit or upgrade data

Everything likes last slides!
And as u can see our data`s upgraded in output



```python
import requests

response = requests.put("https://httpbin.org/put", data = {'name_1' : 'mohammad_1'})


print(f'response is {response}')
print(f'Html code is {response.text}')
```

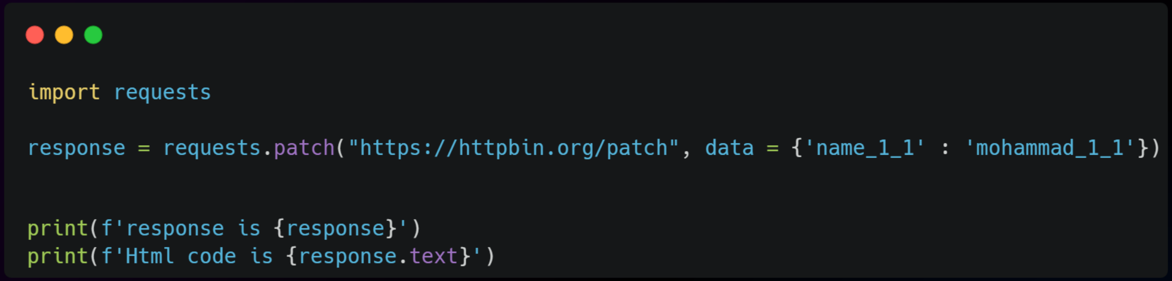


Output

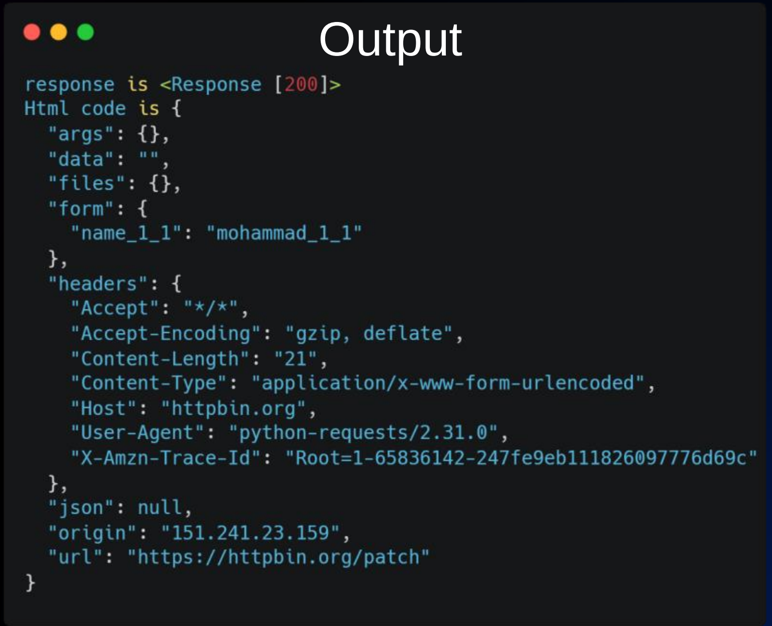
```
response is <Response [200]>

Html code is {
  "args": {},
  "data": "",
  "files": {},
  "form": {
    "name_1": "mohammad_1"
  },
  "headers": {
    "Accept": "*/*",
    "Accept-Encoding": "gzip, deflate",
    "Content-Length": "17",
    "Content-Type": "application/x-www-form-urlencoded",
    "Host": "httpbin.org",
    "User-Agent": "python-requests/2.31.0",
    "X-Amzn-Trace-Id": "Root=1-65836077-688b3ec171e103fe706fede0"
  },
  "json": null,
  "origin": "151.241.23.159",
  "url": "https://httpbin.org/put"
}
```

SENDING REQUEST BY PATCH

- PATCH method is used for few edits like changing a url of website in data



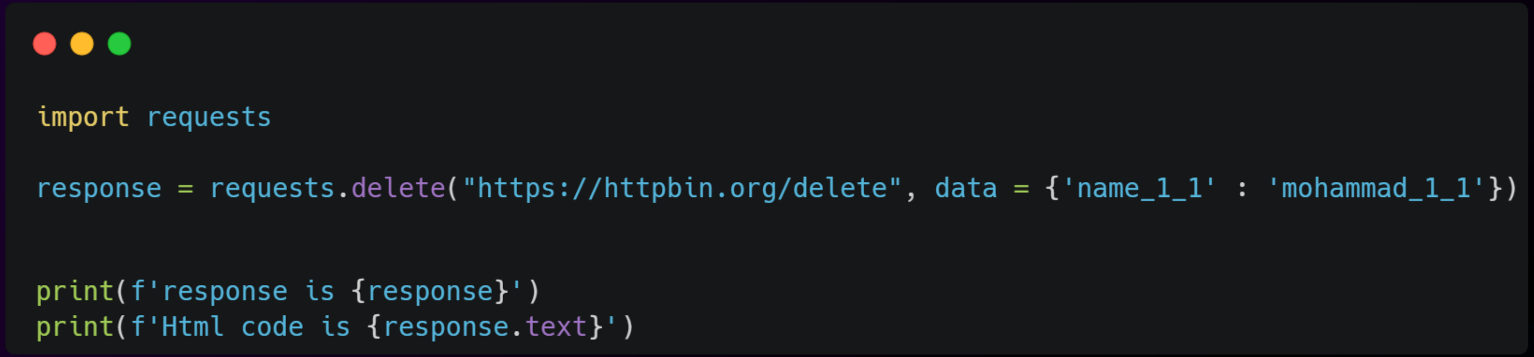
```python
import requests

response = requests.patch("https://httpbin.org/patch", data = {'name_1_1' : 'mohammad_1_1'})


print(f'response is {response}')
print(f'Html code is {response.text}')
```



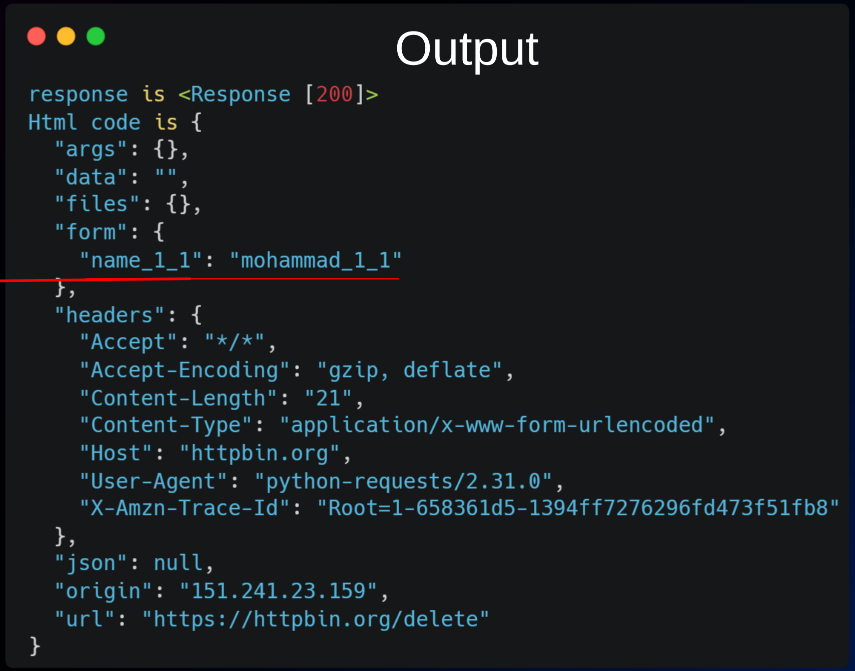
Output

```
response is <Response [200]>
Html code is {
  "args": {},
  "data": "",
  "files": {},
  "form": {
    "name_1_1": "mohammad_1_1"
  },
  "headers": {
    "Accept": "*/*",
    "Accept-Encoding": "gzip, deflate",
    "Content-Length": "21",
    "Content-Type": "application/x-www-form-urlencoded",
    "Host": "httpbin.org",
    "User-Agent": "python-requests/2.31.0",
    "X-Amzn-Trace-Id": "Root=1-65836142-247fe9eb111826097776d69c"
  },
  "json": null,
  "origin": "151.241.23.159",
  "url": "https://httpbin.org/patch"
}
```

SENDING REQUEST BY DELETE



```python
import requests

response = requests.delete("https://httpbin.org/delete", data = {'name_1_1' : 'mohammad_1_1'})


print(f'response is {response}')
print(f'Html code is {response.text}')
```



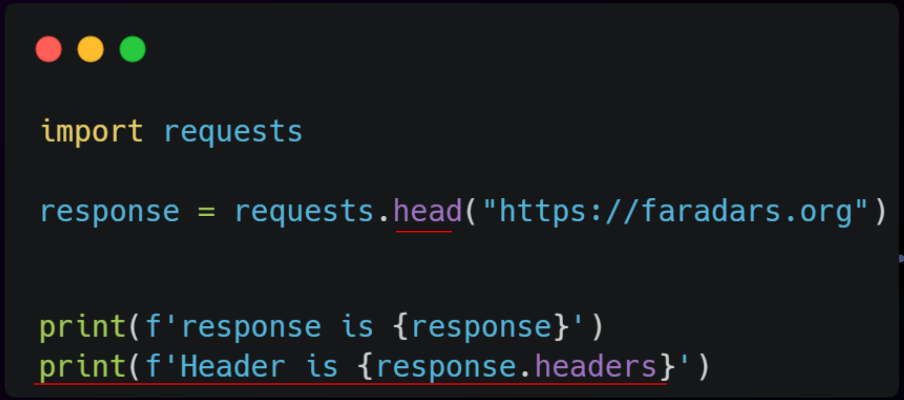
Output

```
response is <Response [200]>
Html code is {
  "args": {},
  "data": "",
  "files": {},
  "form": {
    "name_1_1": "mohammad_1_1"
  },
  "headers": {
    "Accept": "*/*",
    "Accept-Encoding": "gzip, deflate",
    "Content-Length": "21",
    "Content-Type": "application/x-www-form-urlencoded",
    "Host": "httpbin.org",
    "User-Agent": "python-requests/2.31.0",
    "X-Amzn-Trace-Id": "Root=1-658361d5-1394ff7276296fd473f51fb8"
  },
  "json": null,
  "origin": "151.241.23.159",
  "url": "https://httpbin.org/delete"
}
```

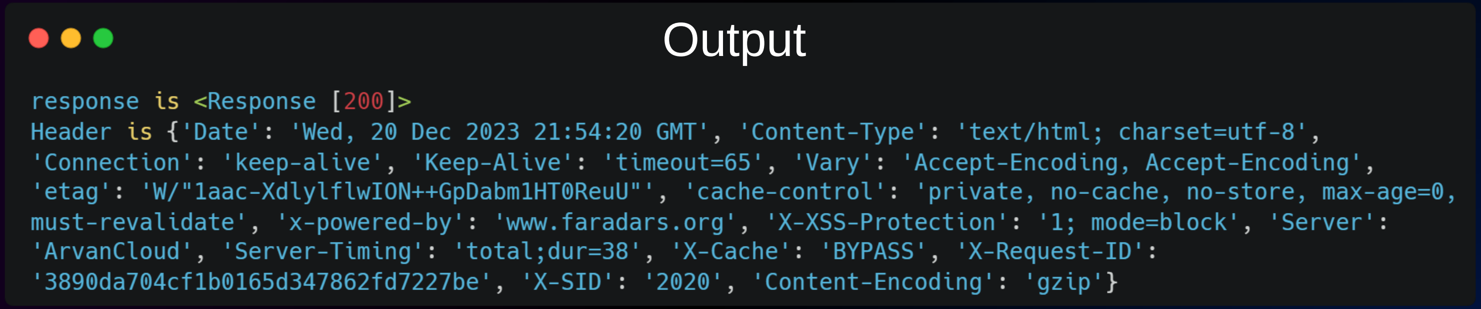
This Is a context that we sent it on website

SENDING REQUEST BY HEAD METHOD

By head method we can read the header of website.
And I think everything is clear
The output is a json format that we can read header data from that



```python
import requests

response = requests.head("https://faradars.org")


print(f'response is {response}')
print(f'Header is {response.headers}')
```



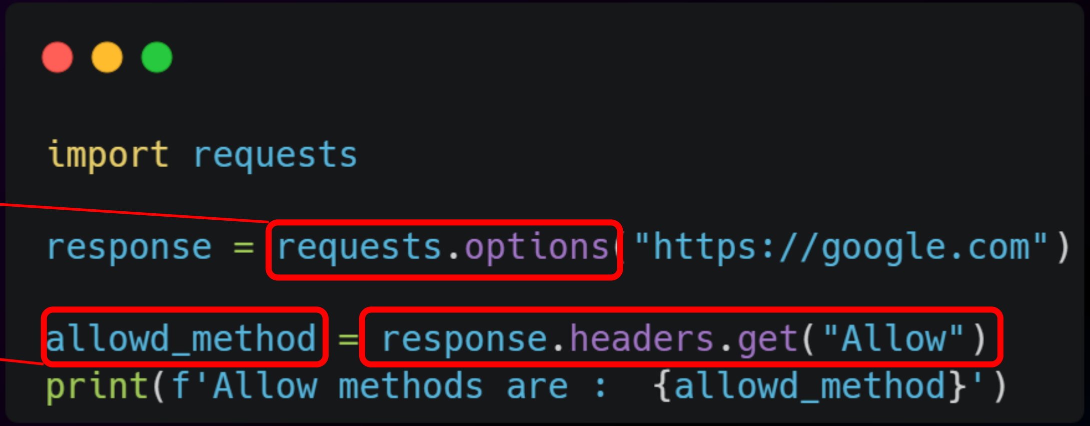
Output

```
response is <Response [200]>
Header is {'Date': 'Wed, 20 Dec 2023 21:54:20 GMT', 'Content-Type': 'text/html; charset=utf-8',
'Connection': 'keep-alive', 'Keep-Alive': 'timeout=65', 'Vary': 'Accept-Encoding, Accept-Encoding',
'etag': 'W/"1aac-XdlylflwION++GpDabm1HT0ReuU"', 'cache-control': 'private, no-cache, no-store, max-age=0,
must-revalidate', 'x-powered-by': 'www.faradars.org', 'X-XSS-Protection': '1; mode=block', 'Server':
'ArvanCloud', 'Server-Timing': 'total;dur=38', 'X-Cache': 'BYPASS', 'X-Request-ID':
'3890da704cf1b0165d347862fd7227be', 'X-SID': '2020', 'Content-Encoding': 'gzip'}
```

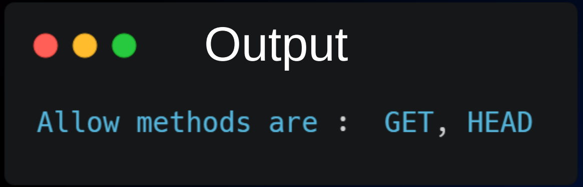
SENDING REQUEST BY OPTION

Here we asking for allowed methods of website

Now we creating variable and asking to get method to find the methods that are allowed.(from the header of website)



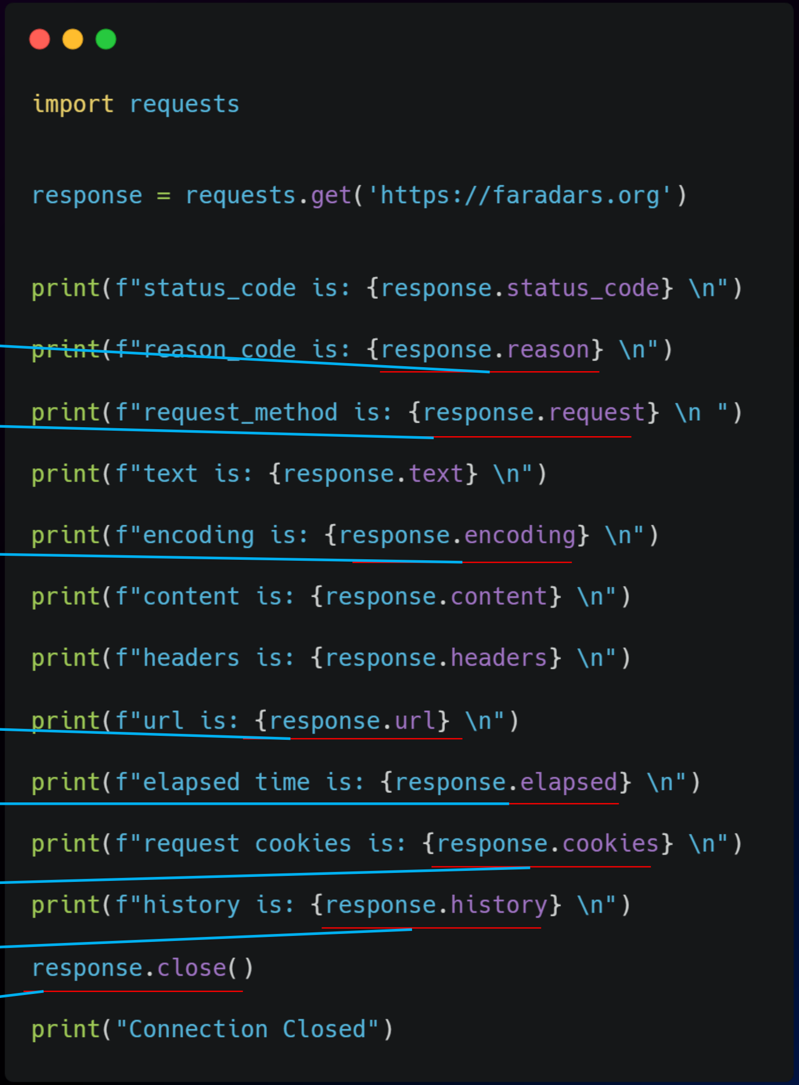
```
import requests

response = requests.options("https://google.com")

allowd_method = response.headers.get("Allow")
print(f'Allow methods are :  {allowd_method}')
```



Output

```
Allow methods are :  GET, HEAD
```

RESPONSE METHODS



```python
import requests

response = requests.get('https://faradars.org')

print(f"status_code is: {response.status_code} \n")
print(f"reason_code is: {response.reason} \n")
print(f"request_method is: {response.request} \n ")
print(f"text is: {response.text} \n")
print(f"encoding is: {response.encoding} \n")
print(f"content is: {response.content} \n")
print(f"headers is: {response.headers} \n")
print(f"url is: {response.url} \n")
print(f"elapsed time is: {response.elapsed} \n")
print(f"request cookies is: {response.cookies} \n")
print(f"history is: {response.history} \n")
response.close()
print("Connection Closed")
```

The content of status code

The type of request that we sent(we used GET)

Encoding! Like utf-8

The target URL

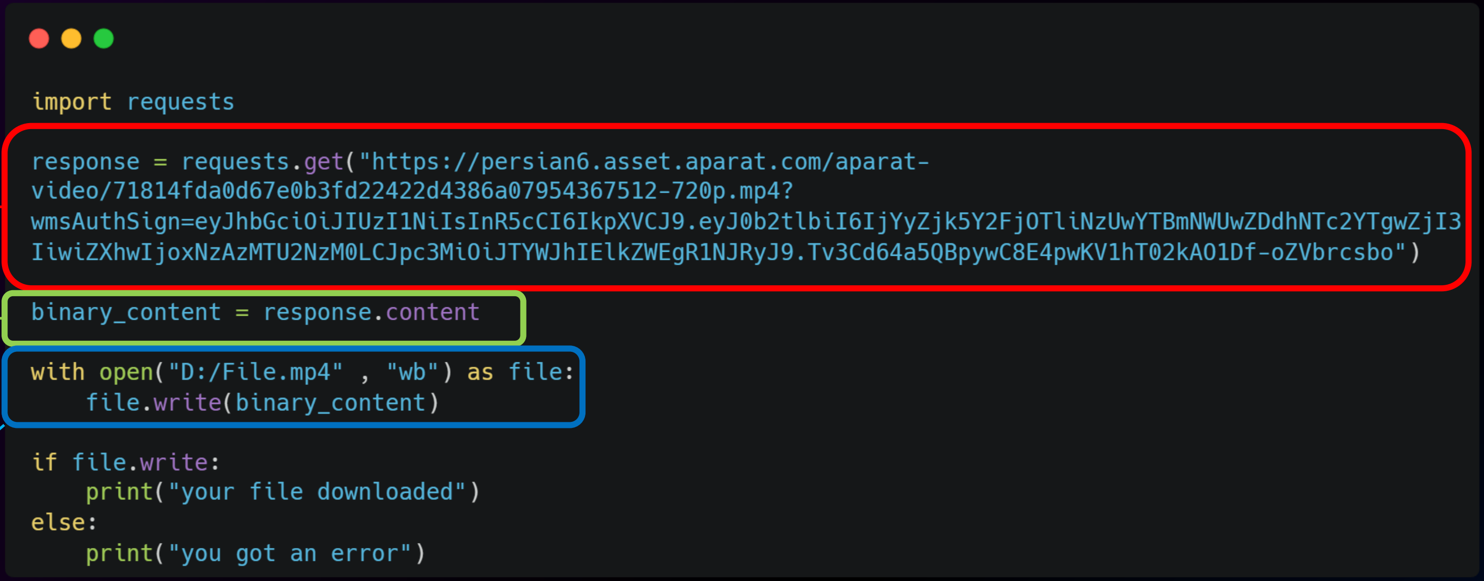
The time that took for response

Installed cookies

The history of request

Important in security , make the connection lost with the server

DOWNLOADING WEB CONTENT BY CONTENT METHOD



```
import requests

response = requests.get("https://persian6.asset.aparat.com/aparat-
video/71814fda0d67e0b3fd22422d4386a07954367512-720p.mp4?
wmsAuthSign=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0b2tlbiI6IjYyZjk5Y2FjOTliNzUwYTBmNWUwZDdhNTc2YTgwZjI3
IiwiZXhwIjoxNzAzMTU2NzM0LCJpc3MiOiJTYWJhIElkZWEgR1NJRyJ9.Tv3Cd64a5QBpywC8E4pwKV1hT02kAO1Df-oZVbrcsbo")
binary_content = response.content
with open("D:/File.mp4" , "wb") as file:
    file.write(binary_content)

if file.write:
    print("your file downloaded")
else:
    print("you got an error")
```

First I use get request and assign it into variable

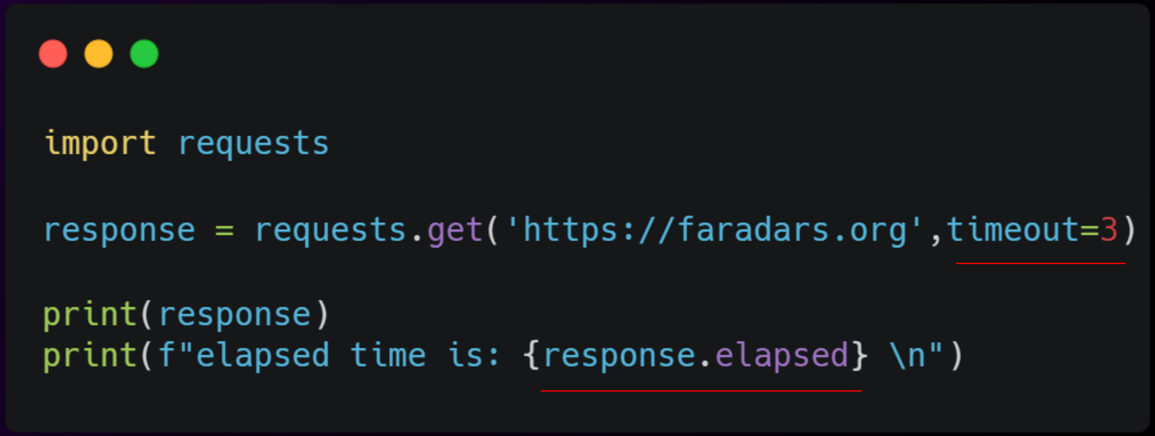
Response.content was our binary code (you can see the binary code by using print(response.content) and I assign it into variable

Then I use with context manager to write binary file into an mp4 file , (wb → write binary). And I ask python to write my binary_content as a file

Note : you can`t change the file format !

ADDING TIME TO REQUEST

- In professional programing we can add a timeout to our request (but usually don`t use in basic)
- You can add timeout and if timeout limit passed… give the client an error message

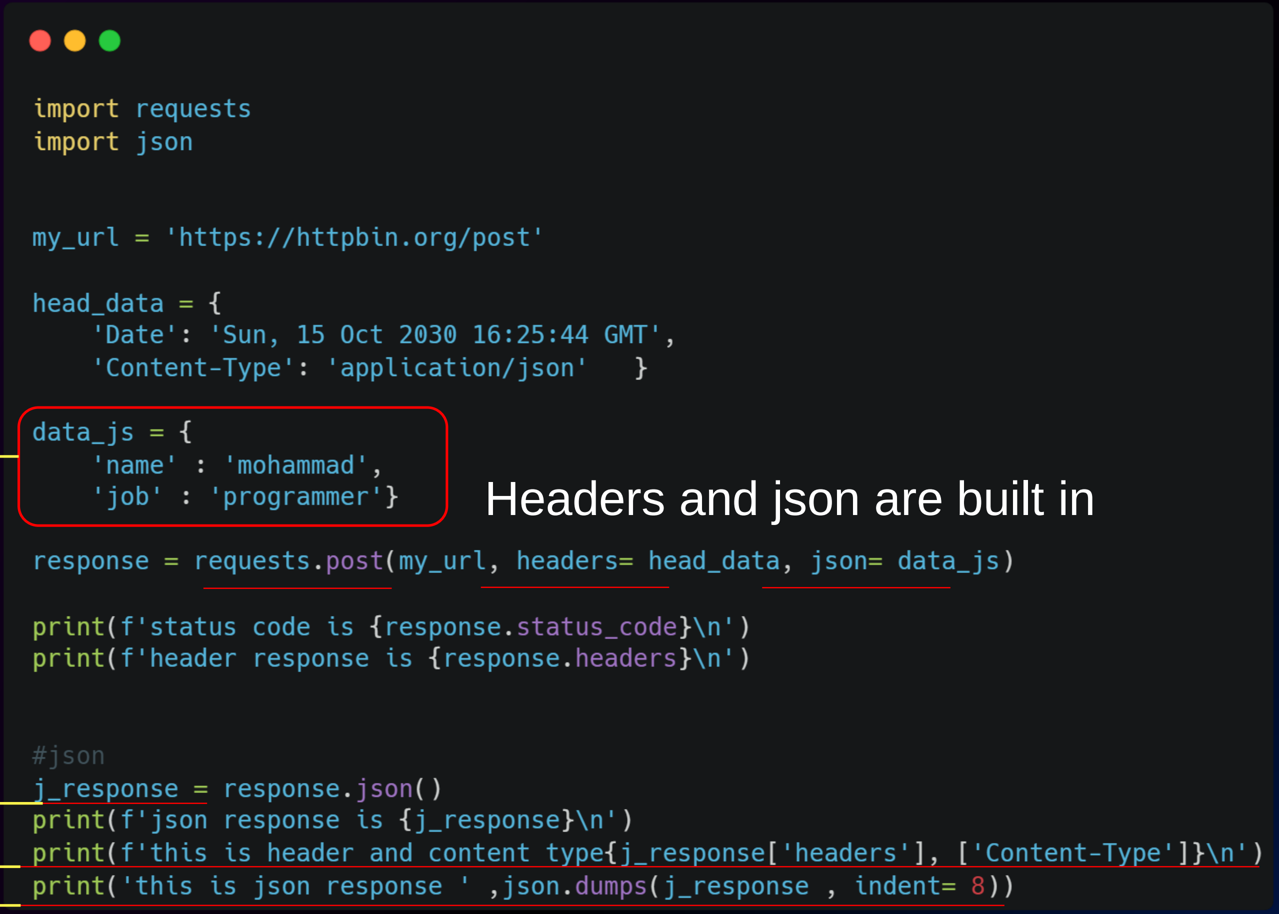


```python
import requests

response = requests.get('https://faradars.org',timeout=3)

print(response)
print(f"elapsed time is: {response.elapsed} \n")
```

USING HEADER AND JSON IN SENDING POST REQUEST

- I think everything is clear in the title we have to just know the syntax
- But before that there is some notes that you have pay attention to it
- 1- Know the json type , what is json and etc.
- 2- how we can iterate in json data
- 3- next slide i`ll show you the code and explanation

RESULT



```
import requests
import json


my_url = 'https://httpbin.org/post'

head_data = {
    'Date': 'Sun, 15 Oct 2030 16:25:44 GMT',
    'Content-Type': 'application/json'    }

data_js = {
    'name' : 'mohammad',
    'job' : 'programmer'}

response = requests.post(my_url, headers= head_data, json= data_js)

print(f'status code is {response.status_code}\n')
print(f'header response is {response.headers}\n')


#json
j_response = response.json()
print(f'json response is {j_response}\n')
print(f'this is header and content type{j_response['headers'], ['Content-Type']}\n')
print('this is json response ' ,json.dumps(j_response , indent= 8))
```

Sending parameters as json, remember json is like a dictionary

Headers and json are built in

Using json.dumps for reading data`s better

Convert data's to json

Iterate in json file

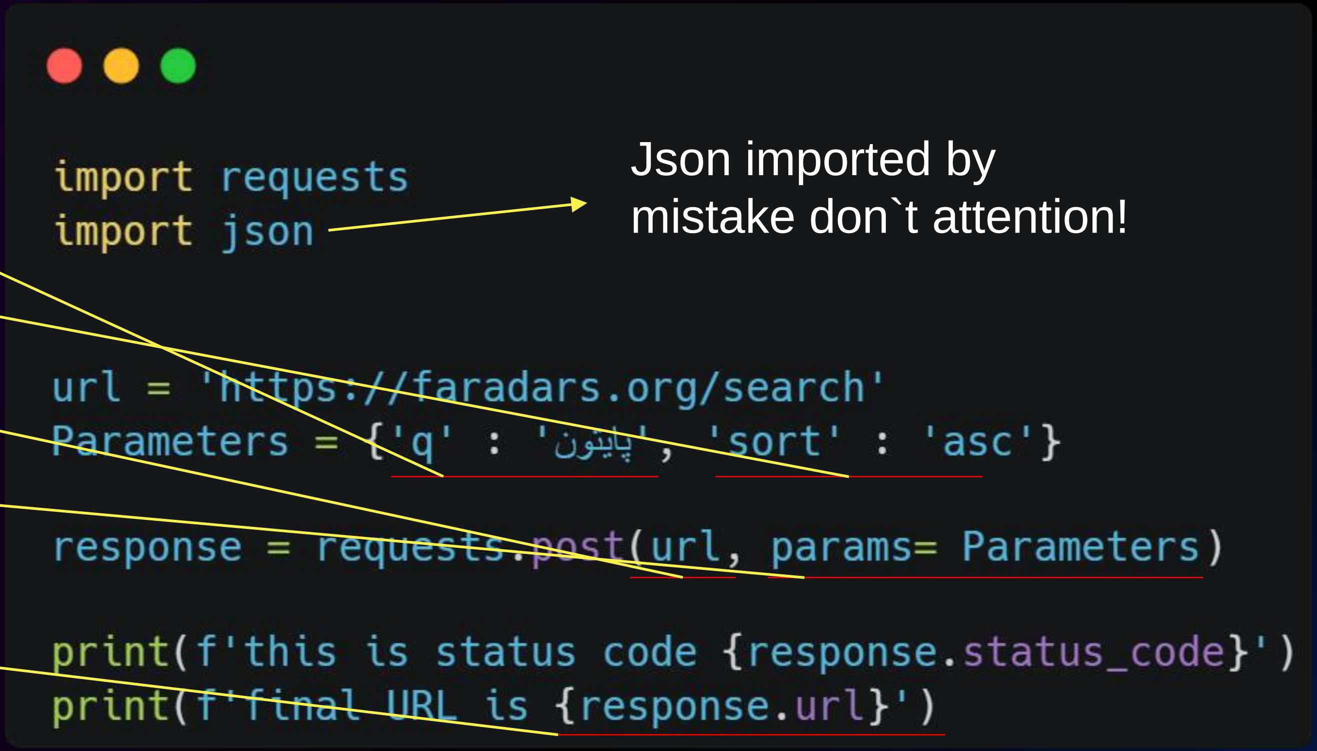
USING PARAMS FOR SENDNING QUERIES

- Params means : parameters and we can send it by requests.post
  - We can use it for searching , api and etc.
- Now what is query string ?
  - Some time we see the question mark (?) at sites url`s which means the sites is categories the data's for example:
  - You want to buy a new laptop you made some filter like brand , color , cpu or etc. before each filter you can see the question mark (?) then you see your filter then you see →(&) like:
    - url?filter1&filter2&filter3&…… real life example:
    - https://www.digikala.com/search/category-notebook-netbook-ultrabook/asus/?attributes%5B2285%5D%5B0%5D=19736&attributes%5B2292%5D%5B0%5D=19828&color_palettes%5B0%5D=8&types%5B0%5D=4

These are filters

SEARCH BY PARAMS

- For searching in website in params we use q as we said in last slide
  - It is good to knows about sorting methods (cause have a sorting parameter too)



```python
import requests
import json

url = 'https://faradars.org/search'
Parameters = {'q' : 'پایتون', 'sort' : 'asc'}

response = requests.post(url, params= Parameters)

print(f'this is status code {response.status_code}')
print(f'final URL is {response.url}')
```

The context that we want to search

Sorting

Sending url to request

Sending parameters into url

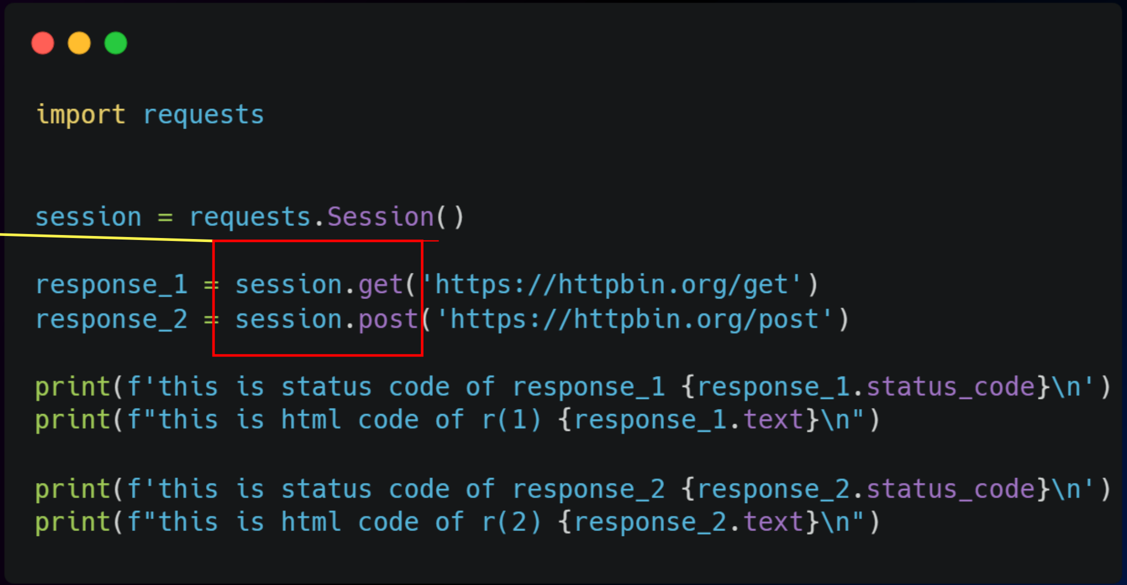
Getting new url from the request

Json imported by mistake don`t attention!

SENDING SEVERAL REQUEST BY SESSION METHOD

- Why do we use session ? Be cause in session method our connection with server will be keep alive
- For example when I login into website , website gives me token(for logging in) session will save that token till close the connection

Attention session is a class



```python
import requests


session = requests.Session()

response_1 = session.get('https://httpbin.org/get')
response_2 = session.post('https://httpbin.org/post')

print(f'this is status code of response_1 {response_1.status_code}\n')
print(f"this is html code of r(1) {response_1.text}\n")

print(f'this is status code of response_2 {response_2.status_code}\n')
print(f"this is html code of r(2) {response_2.text}\n")
```

USING COOKIE PARMS FOR SENDING COOKIE

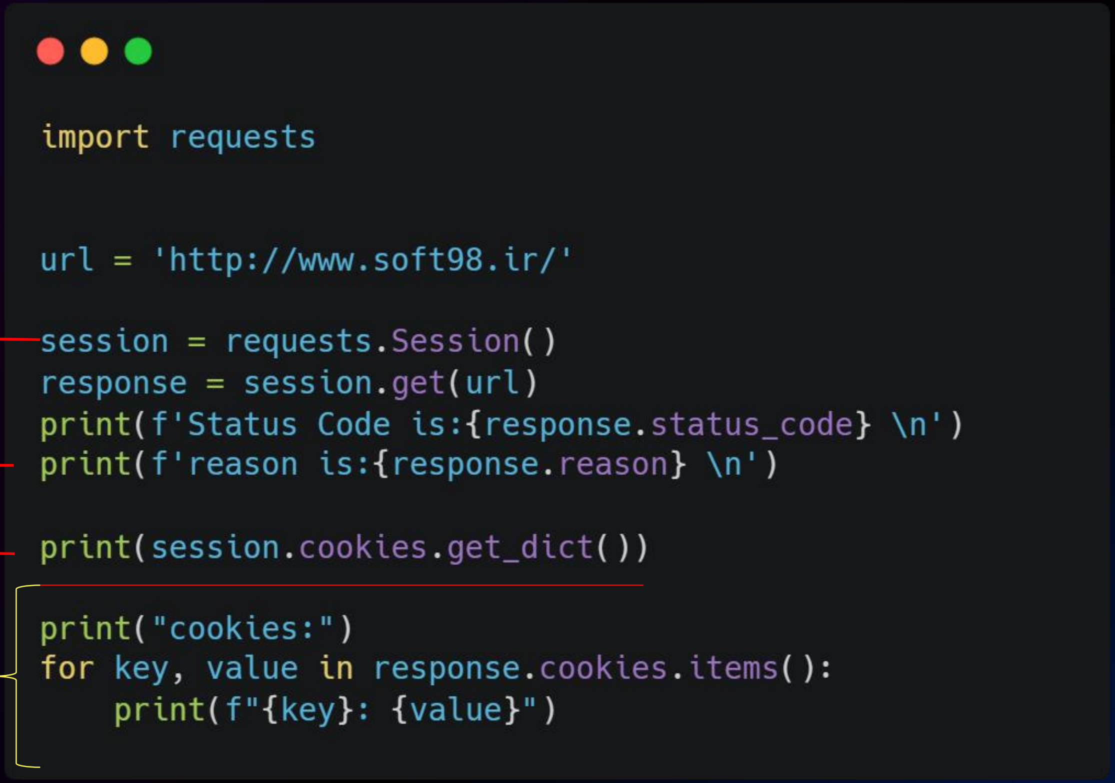
- By this method we can seeing the cookies of website

Opening a session

If we got an error what is the reason?

For showing cookies to us

For loop to seeing cookies better



```python
import requests


url = 'http://www.soft98.ir/'

session = requests.Session()
response = session.get(url)
print(f'Status Code is:{response.status_code} \n')
print(f'reason is:{response.reason} \n')

print(session.cookies.get_dict())

print("cookies:")
for key, value in response.cookies.items():
    print(f"{key}: {value}")
```

AUTHENTICATION BY HTTPBASICAUTH AND HTTPDIGESTAUTH

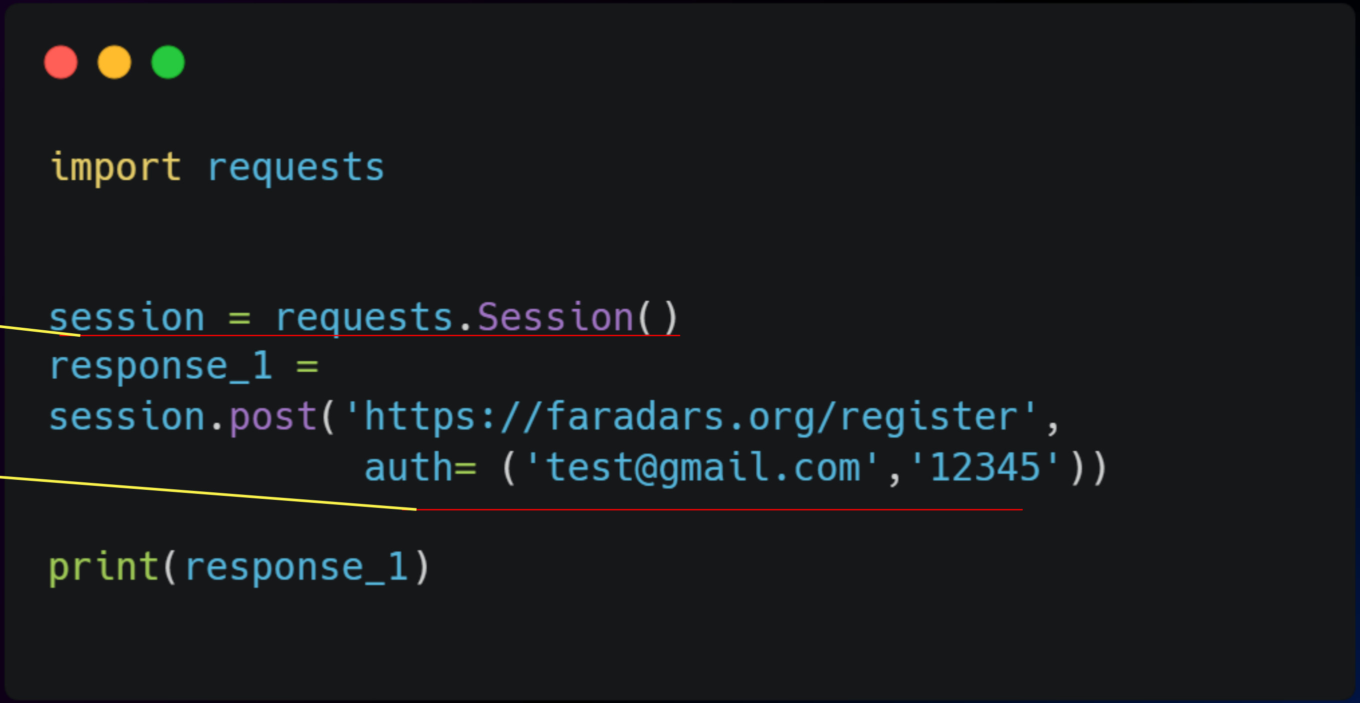
- First : what is different between basic and digest method ?
  - In basic method data`s security is to low and hackers can have access to our information easily but launching basicauth is easier than digestauth
    - By default python uses basicauth
  - In digesauth our user and password are encrypted by hash
- For authentication we can use POST and GET method there is no difference but as security way there is a big difference between these two
  - Because GET method sending our user and password as query string to server which means shows are username and password in url !
  - Then we have to use POST method !

METHOD 1

- NOTE : in all 3 methods is better to use Session because after authentication we want to send some request so… use session !

Opening session

Use auth built in and sends email and password



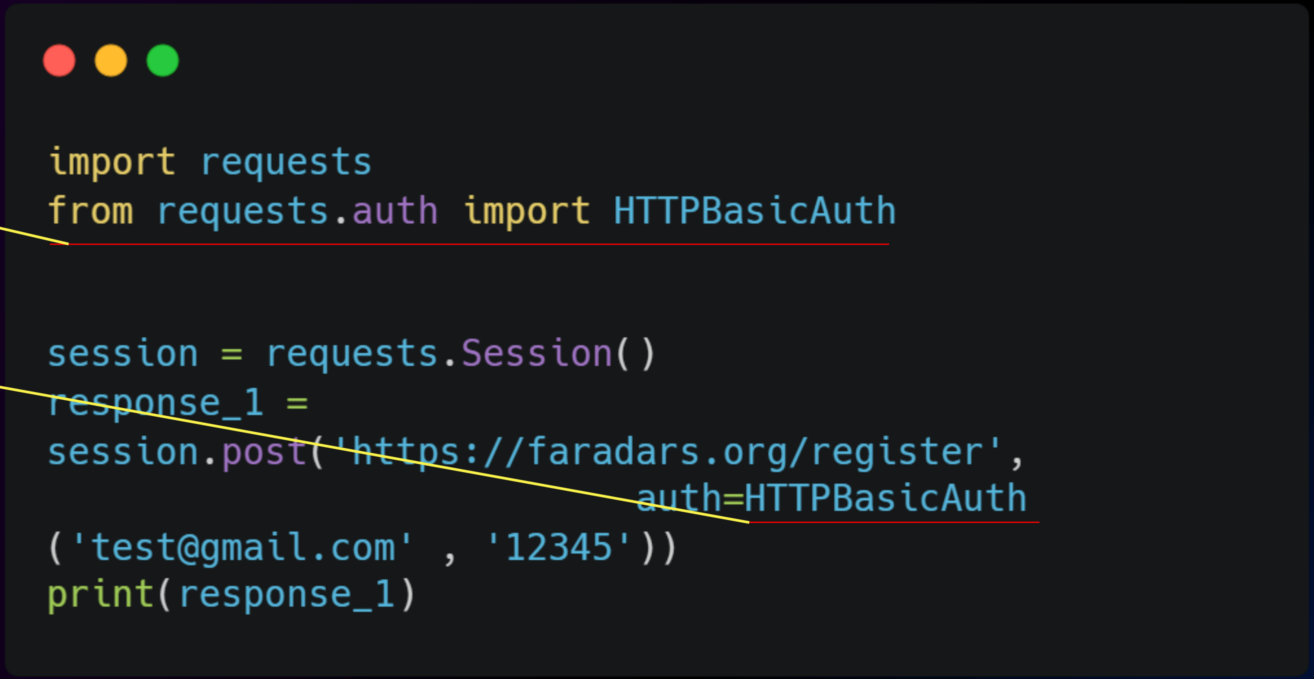
```python
import requests


session = requests.Session()
response_1 =
session.post('https://faradars.org/register',
                auth= ('test@gmail.com','12345'))

print(response_1)
```

METHOD 2

- Using httpbasicauth is like using auth built in and not gonna hash the username and password

Import httpsbasicauth

Use httpbasicauth in auth built in

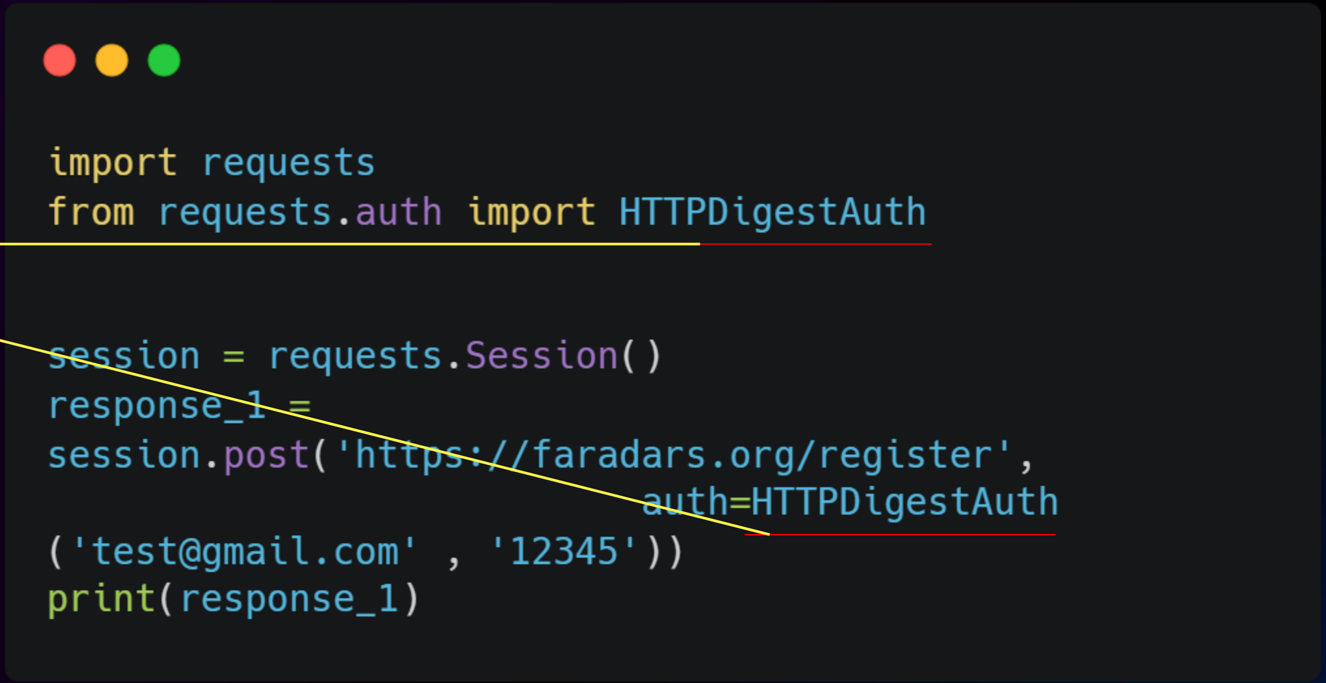


```python
import requests
from requests.auth import HTTPBasicAuth


session = requests.Session()
response_1 =
session.post('https://faradars.org/register',
                                   auth=HTTPBasicAuth
('test@gmail.com' , '12345'))
print(response_1)
```

METHOD 3

Now we use httpdigest auth which will hash our data`s

Import httpdigestauth and use it in post request in auth built in

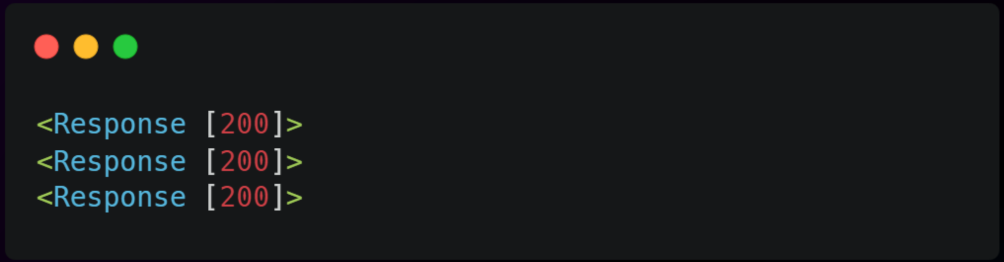


```
import requests
from requests.auth import HTTPDigestAuth


session = requests.Session()
response_1 =
session.post('https://faradars.org/register',
                                          auth=HTTPDigestAuth
('test@gmail.com' , '12345'))
print(response_1)
```

RESULT AND WHY ?!

On those 3 methods we that we used we got three success response ! But why ? We used test@gmail.com !

Because response[200] doesn`t mean that we could login
Means we got an answer from server
And websites with high security use this trick to don`t let clinets get hacked .
If hacker tests username and passwords he/she will get 200status code and his/her not gonna find out what is the username and password



```
<Response [200]>
<Response [200]>
<Response [200]>
```

THANKS FOR YOUR ATTENTION

MOHAMMADH.KHODDAMI@GMAIL.COM