



**Faculty of Engineering & Technology
Electrical & Computer Engineering Department**

Computer Network

ENCS3320

Project 1 Report

Prepared by:

Nirmeen Al-Sheikh - 1200200	Section: 1
Mariam Hamad - 1200837	Section: 3
Leena Affouri - 1200335	Section: 1

Instructor: Dr. Abdalkarim Awad

Date: 30/5/2023

Abstract:

The goal of this project is to learn about and understand the principles of networking by running some commands and making connections between one server and many clients. We also made HTML webpages using socket programming.

Contents:

Contents

Abstract:	2
-----------------	---

The goal of this project is to learn about and understand the principles of networking by running some commands and making connections between one server and many clients. We also made HTML webpages using socket programming.....2

Contents:.....	3
----------------	---

Solution of the project:	5
--------------------------------	---

Part 1:	5
---------------	---

1-What are Ping, Tracert, Nslookup, and Telnet?	5
--	---

2-run the following commands:	5
--	---

A: Ping a device in the same network.....	5
---	---

B: Ping www.harvard.edu	6
-------------------------------	---

C- Tracert www.harvard.edu.....	7
---------------------------------	---

D: Nslookup www.harvard.edu	7
-----------------------------------	---

.....	7
-------	---

Part 2:	8
---------------	---

The server code:.....	8
-----------------------	---

The output of server:	8
-----------------------------	---

First client code:.....	9
-------------------------	---

The output of the first client:	9
---------------------------------------	---

Second client code:	10
---------------------------	----

10
----	-------

The output of the second client:.....	10
---------------------------------------	----

Explain of part 2:	11
--------------------------	----

Part 3:	12
---------------	----

1- Check if the request is / or /index.html or /main_en.html or /en (for example localhost: 9977/ or localhost: 9977/en) then the server should send main_en.html file with Content-Type: text/html.	12
---	----

.....	12
-------	----

The output:	14
-------------------	----

localhost: 9977/	14
-------------------------------	----

2- If the request is /ar then the server response with main_ar.html which is an Arabic version of main_en.html	17
3- If the request is an .html file then the server should send the requested html file with Content-Type: text/html. You can use any html file.	18
4- If the request is a .css file then the server should send the requested css file with Content-Type: text/css. You can use any CSS file.....	19
5- If the request is a .png then the server should send the png image with Content-Type: image/png. You can use any image.....	21
6- If the request is a .jpg then the server should send the jpg image with Content-Type: image/jpeg. You can use any image.....	21
7- Use the status code 307 Temporary Redirect to redirect the following	22
a- If the request is /yt then redirect to YouTube website.....	22
b- If the request is /so then redirect to stackoverflow.com website.....	23
c- If the request is /rt then redirect to ritaj website.....	23
8- If the request is wrong or the file doesn't exist the server should return a simple HTML webpage that contains (Content-Type: text/html)	24
9- The program should print the HTTP requests on the terminal window (command line window).....	26

Solution of the project:

Part 1:

1-What are Ping, Tracert, Nslookup, and Telnet?

Ping: is a network tool that sends a packet of data to a specified IP address or domain name to analyze the response time, TTL (time to live), RTT (round trip time), and overall connection between the sender and the destination.

Tracert: is a command-line utility that traces the path of network packets from the source to the destination.

Nslookup: is a command-line tool used to query DNS to retrieve information about a domain, including its IP address, IPV4 address, and alias name.

Telnet: is a network protocol used to establish a remote connection to a specific device or server.

2-run the following commands:

A: Ping a device in the same network

```
C:\Users\soFTech>ping 172.19.0.1

Pinging 172.19.0.1 with 32 bytes of data:
Reply from 172.19.0.1: bytes=32 time=12ms TTL=255
Reply from 172.19.0.1: bytes=32 time=4ms TTL=255
Reply from 172.19.0.1: bytes=32 time=5ms TTL=255
Reply from 172.19.0.1: bytes=32 time=5ms TTL=255

Ping statistics for 172.19.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 4ms, Maximum = 12ms, Average = 6ms
```



In the previous screenshot, we made Pinging 172.19.0.1 with 32 bytes of data: the ping utility is sending packets of size 32 bytes to the specified IP address.

Reply from 172.19.0.1: bytes = 32, time = 12ms TTL=255: These lines represent the replies received from the destination IP address (a smartphone). Each line indicates a successful reply, displaying the response time in milliseconds (time = 12 ms), the size of the packet in bytes (bytes = 32), and the Time to Live (TTL) value, which represents the number of hops the packet can traverse before being discarded (TTL = 255).

The "time=12ms" indicates that the round trip time for the ICMP (Internet Control Message Protocol) echo request and reply packets to travel from the sender (laptop) to the destination (172.19.0.1) and back to the sender is 12 milliseconds.

Ping statistics for 172.19.0.1, this line provides a summary of the ping statistics for the destination IP address (a smartphone).

packets: Sent = 4, Received = 4, Lost = 0 (0% loss): This line indicates the number of packets sent, received, and lost during the ping operation. In our situation, all packets were received successfully, resulting in 0% packet loss.

Approximate round trip times in milliseconds: This section provides additional information about the response times for the sent packets. Minimum = 4ms, Maximum = 12ms, Average = 6ms: These values represent the minimum, maximum, and average response times, respectively, measured in milliseconds.

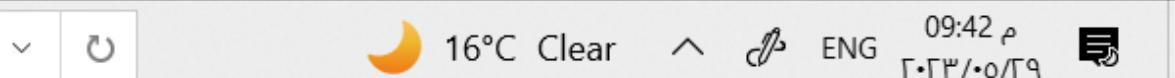
B: Ping www.harvard.edu

```
C:\Users\ASUS STRIX G15>ping www.harvard.edu

Pinging pantheon-systems.map.fastly.net [146.75.122.133] with 32 bytes of data:
Reply from 146.75.122.133: bytes=32 time=180ms TTL=52
Reply from 146.75.122.133: bytes=32 time=178ms TTL=52
Reply from 146.75.122.133: bytes=32 time=195ms TTL=52
Reply from 146.75.122.133: bytes=32 time=204ms TTL=52

Ping statistics for 146.75.122.133:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 178ms, Maximum = 204ms, Average = 189ms

C:\Users\ASUS STRIX G15>
```



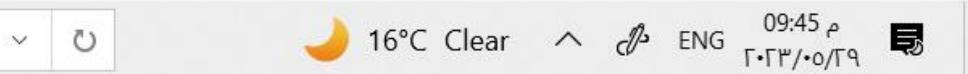
C- Tracert www.harvard.edu

```
C:\Users\ASUS STRIX G15>tracert www.harvard.edu

Tracing route to pantheon-systems.map.fastly.net [199.232.82.133]
over a maximum of 30 hops:

 1   7 ms    2 ms    2 ms  172.19.0.1
 2   8 ms    7 ms   28 ms  10.10.10.1
 3   4 ms    4 ms   3 ms  46.43.64.201
 4   69 ms   60 ms   67 ms  te0-7-0-7-1.agr21.mrs02.atlas.cogentco.com [149.6.155.145]
 5   61 ms   115 ms   58 ms  be2753.ccr32.mrs02.atlas.cogentco.com [154.54.39.13]
 6   65 ms   61 ms   56 ms  149.6.154.6
 7   87 ms   69 ms   71 ms  199.232.82.133

Trace complete.
```

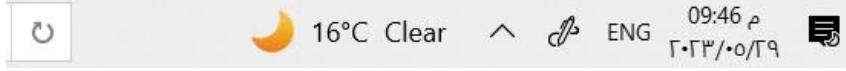


The output shows that the packets pass through seven network devices before reaching the destination, with round-trip times ranging from 4 to 115 milliseconds. The first two hops are on the local network, followed by several hops on the Cogent Communications network, before finally reaching the destination at IP address 199.232.82.133. This server appears to be hosting the website for Harvard University.

D: Nslookup www.harvard.edu

```
C:\Users\ASUS STRIX G15>nslookup www.harvard.edu
Server: UnKnown
Address: 172.21.21.153
```

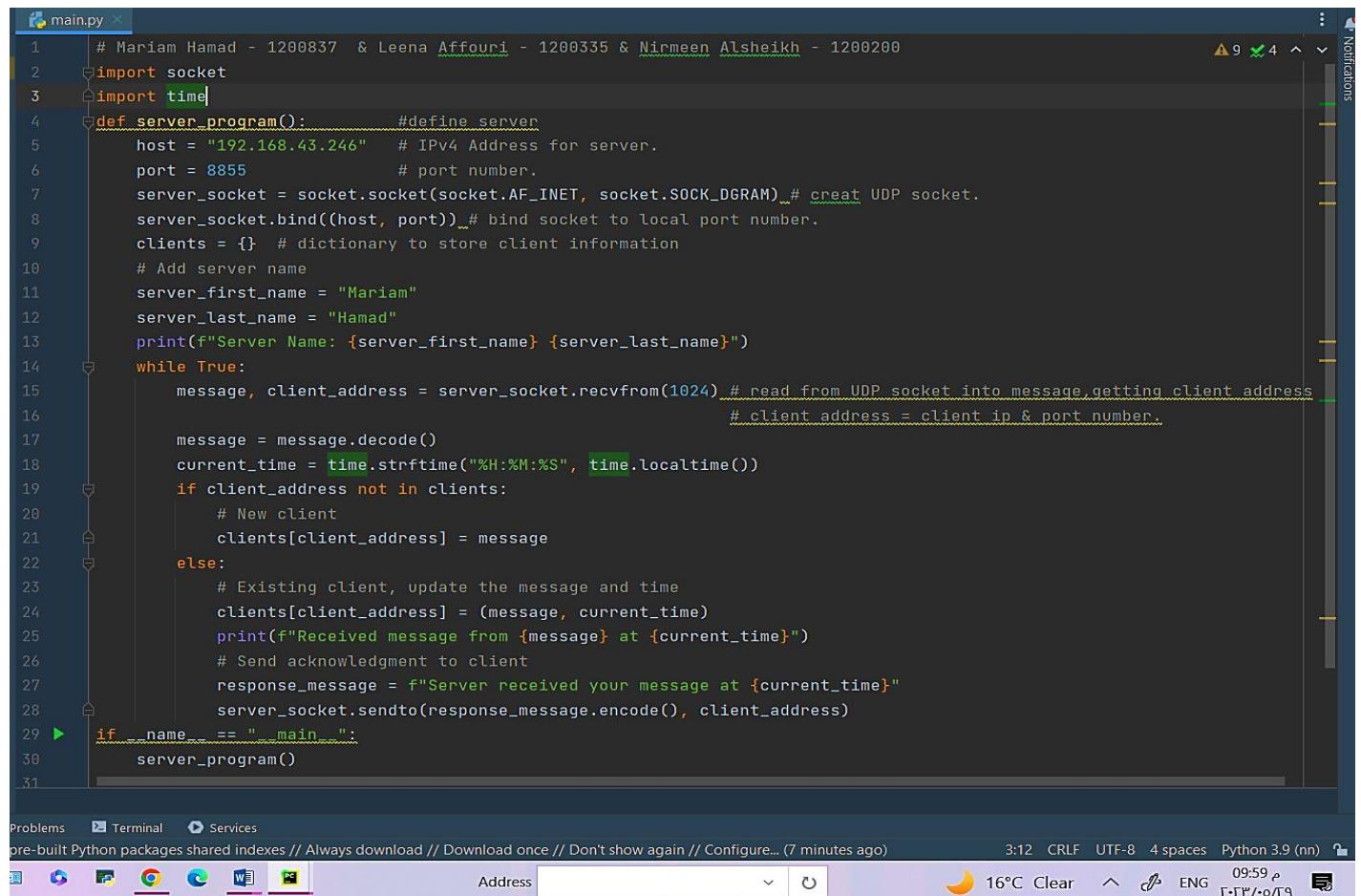
```
Non-authoritative answer:
Name:  pantheon-systems.map.fastly.net
Addresses: 2a04:4e42:54::645
          199.232.82.133
Aliases:  www.harvard.edu
```



The output above shows the alias name that represents two names for the same IP address. In our case, it shows the actual name for the website, www.harvard.edu, and we discovered it is the alias name, and as we know in the DNS records, the real name takes the type CNAME and the alias name takes the type A.

Part 2:

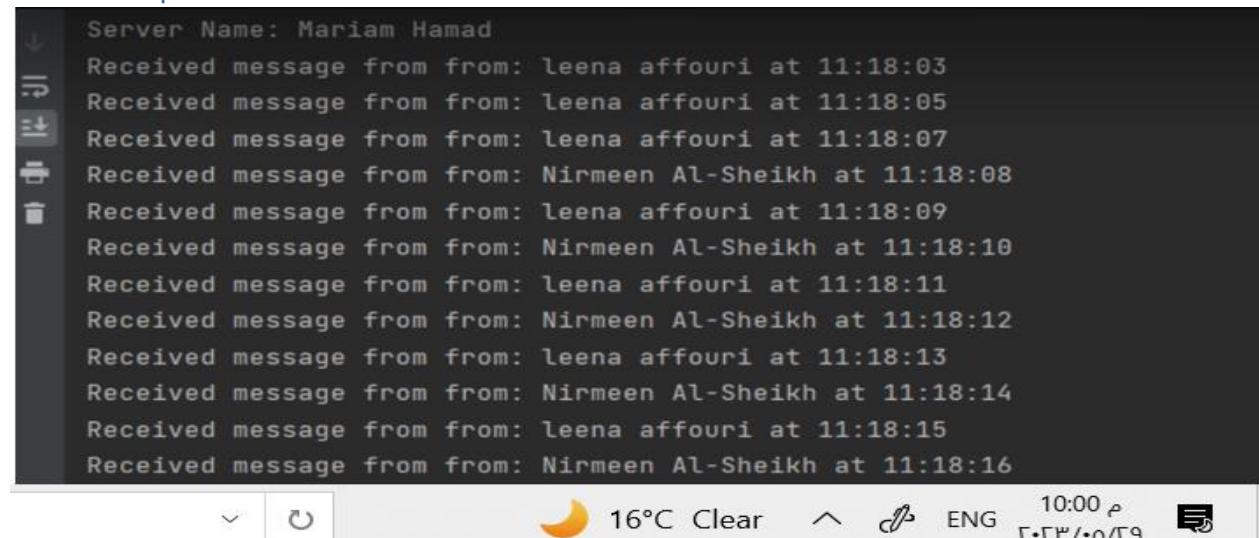
The server code:



A screenshot of a code editor showing a Python script named `main.py`. The code defines a `server_program()` function that creates a UDP socket, binds it to a specific host and port, and then enters a loop to receive messages from clients. It decodes the messages, prints them along with the current time, and sends an acknowledgment back to the client. The code also handles new clients by adding them to a dictionary of client addresses and their messages. The code editor interface includes tabs for `main.py`, `Terminal`, and `Services`. The terminal below shows the output of the server program.

```
1 # Mariam Hamad - 1200837 & Leena Affouri - 1200335 & Nirmeen Alsheikh - 1200200
2 import socket
3 import time
4 def server_program():      #define server
5     host = "192.168.43.246" # IPv4 Address for server.
6     port = 8855             # port number.
7     server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM) # creat UDP socket.
8     server_socket.bind((host, port)) # bind socket to local port number.
9     clients = {} # dictionary to store client information
10    # Add server name
11    server_first_name = "Mariam"
12    server_last_name = "Hamad"
13    print(f"Server Name: {server_first_name} {server_last_name}")
14    while True:
15        message, client_address = server_socket.recvfrom(1024) # read from UDP socket into message,getting client address
16                                # client address = client ip & port number.
17        message = message.decode()
18        current_time = time.strftime("%H:%M:%S", time.localtime())
19        if client_address not in clients:
20            # New client
21            clients[client_address] = message
22        else:
23            # Existing client, update the message and time
24            clients[client_address] = (message, current_time)
25            print(f"Received message from {message} at {current_time}")
26            # Send acknowledgment to client
27            response_message = f"Server received your message at {current_time}"
28            server_socket.sendto(response_message.encode(), client_address)
29    if __name__ == "__main__":
30        server_program()
31
```

The output of server:

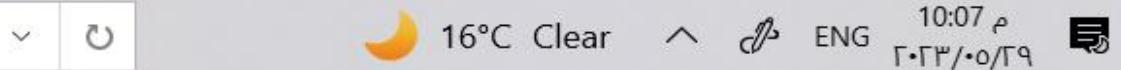


A screenshot of a terminal window showing the output of the `main.py` script. The server prints its name and then repeatedly receives messages from two clients, `leena affouri` and `Nirmeen Al-Sheikh`, and prints them along with the current time. The terminal interface includes icons for file operations and a status bar at the bottom.

```
Server Name: Mariam Hamad
Received message from from: leena affouri at 11:18:03
Received message from from: leena affouri at 11:18:05
Received message from from: leena affouri at 11:18:07
Received message from from: Nirmeen Al-Sheikh at 11:18:08
Received message from from: leena affouri at 11:18:09
Received message from from: Nirmeen Al-Sheikh at 11:18:10
Received message from from: leena affouri at 11:18:11
Received message from from: Nirmeen Al-Sheikh at 11:18:12
Received message from from: leena affouri at 11:18:13
Received message from from: Nirmeen Al-Sheikh at 11:18:14
Received message from from: leena affouri at 11:18:15
Received message from from: Nirmeen Al-Sheikh at 11:18:16
```

First client code:

```
nan.py > client_program
 1 # Mariam Hamad - 1200837
 2 # Leena Affouri - 1200335
 3 # Nirmeen Alsheikh - 1200200
 4 import socket
 5 import time
 6
 7 def client_program():
 8     host = "192.168.43.255" # Broadcast address
 9     port = 8855 # Server port number
10     client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM) # create UDP socket for server
11     client_socket.setsockopt(socket.SOL_SOCKET, socket.SO_BROADCAST, 1) # Enable broadcasting
12     client_socket.settimeout(2) # Set timeout for receiving response
13     student_name = "Nirmeen Al-Sheikh" # sender's first and last names
14     num = 0
15     while True:
16         message = f"from: {student_name}" # Message containing student's name
17         client_socket.sendto(message.encode(), (host, port)) # Send message to the broadcast address
18
19         try:
20             data, address = client_socket.recvfrom(1024) # Receive response from server
21             print(f" {data.decode()}")
22
23         except socket.timeout:
24             print("No response from server")
25
26         num += 1
27         time.sleep(2) # Wait for 2 seconds before sending the next message
28
29     client_socket.close() # Close the connection
30
31 if __name__ == '__main__':
32     client_program()
```



The output of the first client:

A screenshot of a terminal window titled 'TERMINAL'. The window has tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', and 'TERMINAL', with 'TERMINAL' being the active tab. The terminal displays the following text:

```
Server received your message at 11:29:27
Server received your message at 11:29:29
Server received your message at 11:29:31
Server received your message at 11:29:33
Server received your message at 11:29:35
Server received your message at 11:29:37
Server received your message at 11:29:40
```

Below the terminal window is another instance of the Windows taskbar with identical icons.

Second client code:

The screenshot shows the PyCharm IDE interface with a Python project named 'pythonProject4'. The main window displays the 'main.py' file containing the following code:

```
# Mariam Hamad - 1200837 & Leena Affouri - 1200335 & Nirmeen Alsheikh - 1200200
import socket
import time

def client_program():
    host = "192.168.43.255" # Broadcast address
    port = 8855 # Server port number
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM) # create UDP socket for server
    client_socket.setsockopt(socket.SOL_SOCKET, socket.SO_BROADCAST, 1) # Enable broadcasting
    client_socket.settimeout(2) # Set timeout for receiving response

    student_name = "leena affouri" # sender's first and last names

    num = 0
    while True:
        message = f"from: {student_name}" # Message containing student's name
        client_socket.sendto(message.encode(), (host, port)) # Send message to the broadcast address

        try:
            data, address = client_socket.recvfrom(1024) # Receive response from server
            print(f" {data.decode()}")
        except socket.timeout:
            print("No response from server")

        num += 1
        time.sleep(2) # Wait for 2 seconds before sending the next message
    client_socket.close() # Close the connection

if __name__ == '__main__':
    client_program()
```

The status bar at the bottom shows the terminal output: "Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built Python packages shared indexes // Always download // Download once // Don't show again // Configure... (2 minutes)" and "26.76 CRLF UTF-8 4 spaces Python 3.9 (pythonProject4)". The system tray shows the date and time as "5/29/2023 10:12 PM".

The output of the second client:

The terminal window shows the following repeated output:

```
Server received your message at 11:29:27
Server received your message at 11:29:29
Server received your message at 11:29:31
Server received your message at 11:29:33
Server received your message at 11:29:35
Server received your message at 11:29:37
Server received your message at 11:29:40
```

The status bar at the bottom shows the terminal output: "Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built Python packages shared indexes // Always download // Download once // Don't show again // Configure... (2 minutes)" and "26.76 CRLF UTF-8 4 spaces Python 3.9 (pythonProject4)". The system tray shows the date and time as "5/29/2023 10:17 PM".

Explain of part 2:

In this part, we have one server and two clients. The clients send messages to the server every two seconds. The server receives a message from each client and displays a message on its terminal window containing the first and last name of the client, along with the time that the message reached the server. At the same time, the client receives a message from the server, including the time that the message reached the server, after we guarantee that the clients and the server are connected on the same network and the client uses the server IP address.

Part 3:

1- Check if the request is **/ or /index.html or /main_en.html or /en** (for example **localhost: 9977/ or localhost: 9977/en**) then the server should send **main_en.html** file with Content-Type: **text/html**.

```
from socket import *
import urllib.request

serverPort = 9977
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind(("127.0.0.1", serverPort))
serverSocket.listen(1)
print("The web server is ready to receive")

while True:
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024).decode()
    print("Received sentence:", sentence)
    ip = addr[0]
    port = addr[1]
    sentence_parts = sentence.split()
    if len(sentence_parts) >= 2:
        object = sentence_parts[1]
        print("Requested object:", object)

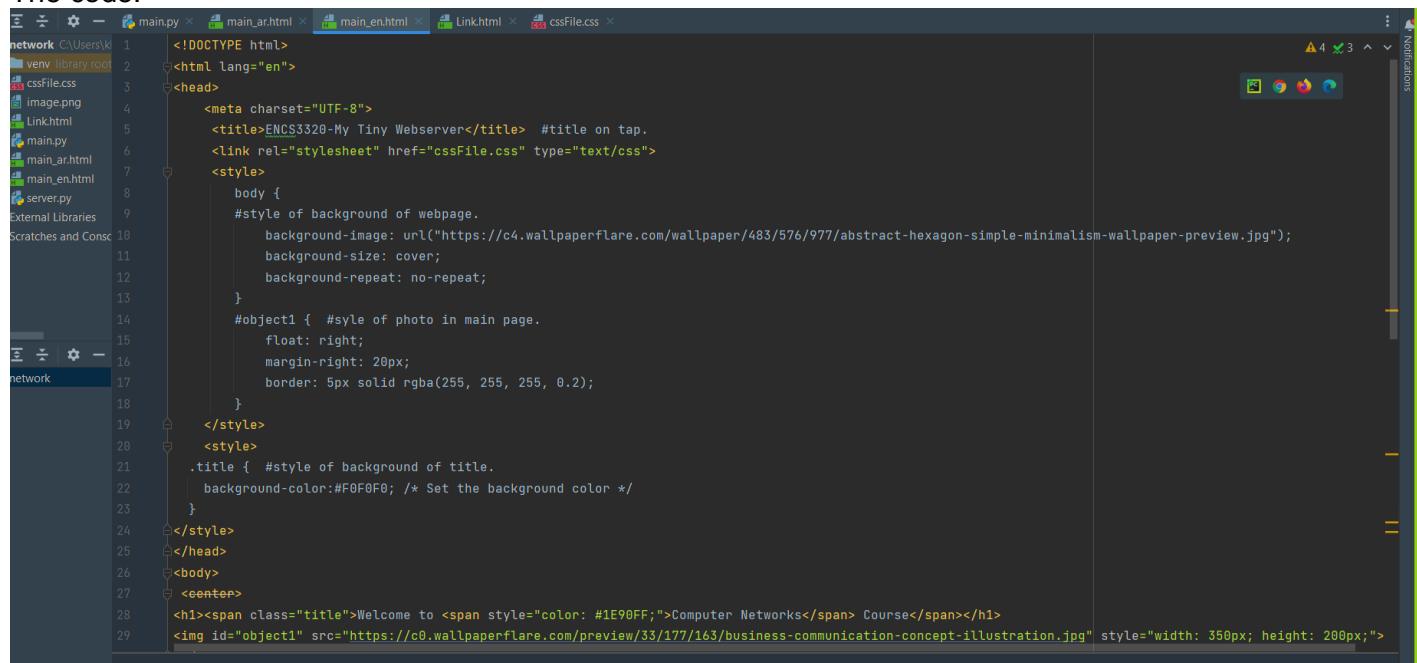
        if object == '/' or object == '/index.html' or object == '/main_en.html' or object == '/en':
            connectionSocket.send("HTTP/1.1 200 OK \r\n".encode())
            connectionSocket.send("Content-Type: text/html \r\n".encode())
            connectionSocket.send("\r\n".encode())
            file1 = open("main_en.html", "rb")
            connectionSocket.send(file1.read())

        elif object == '/ar':
            connectionSocket.send("HTTP/1.1 200 OK \r\n".encode())
```

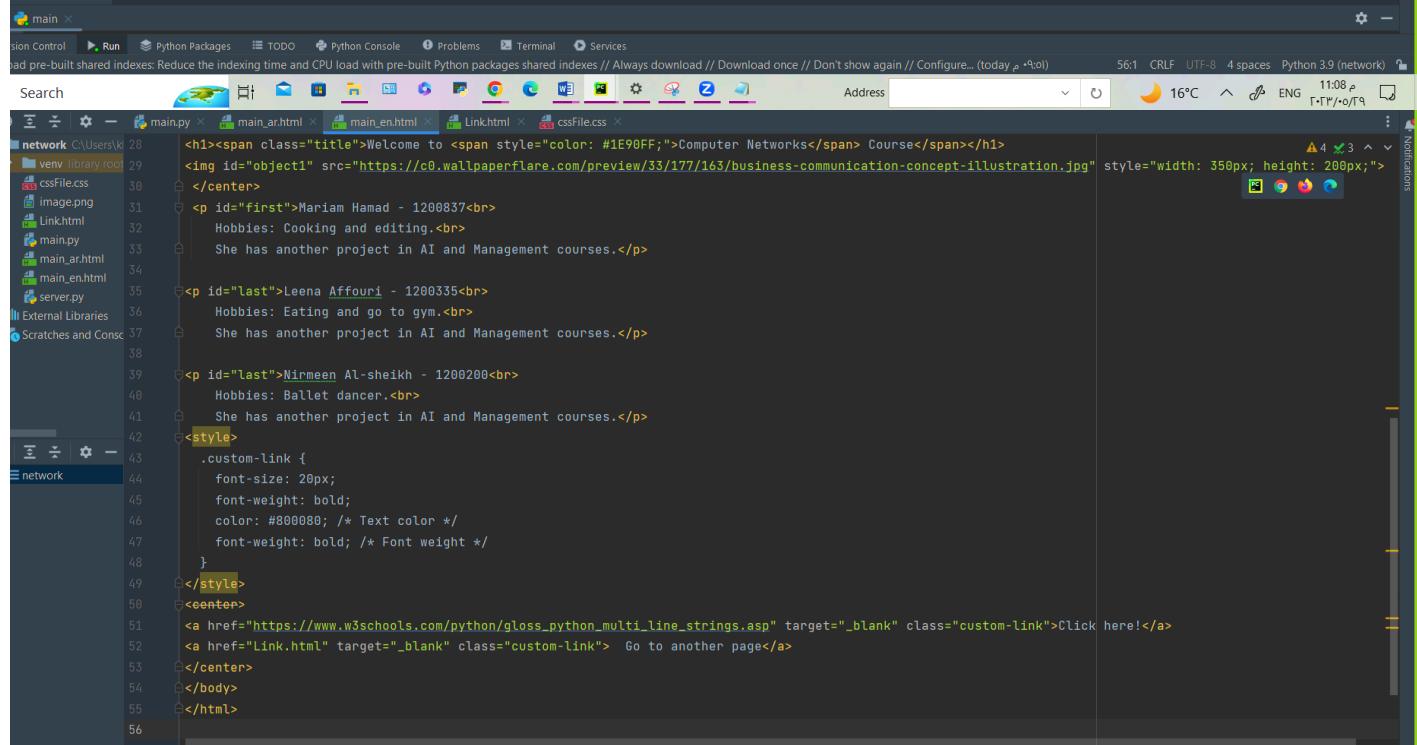
In this part, we use socket programming to implement a simple but complete web server in Python on port 9977. First, the program checks the request in the browser. There are four types of requests; if it's one of them, then the server should send the main_en.html file.

The `main_en.html` file contain HTML webpage that contains:

The code:

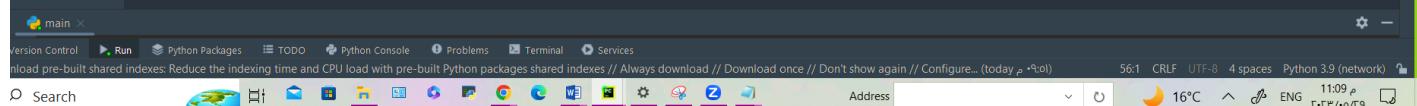


```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>ENC33320-My Tiny Webserver</title> #title on tap.
    <link rel="stylesheet" href="cssFile.css" type="text/css">
    <style>
      body {
        #style of background of webpage.
        background-image: url("https://c4.wallpaperflare.com/wallpaper/483/576/977/abstract-hexagon-simple-minimalism-wallpaper-preview.jpg");
        background-size: cover;
        background-repeat: no-repeat;
      }
      #object1 { #style of photo in main page.
        float: right;
        margin-right: 20px;
        border: 5px solid rgba(255, 255, 255, 0.2);
      }
    </style>
    <style>
      .title { #style of background of title.
        background-color:#F0F0F0; /* Set the background color */
      }
    </style>
  </head>
  <body>
    <center>
      <h1><span class="title">Welcome to <span style="color: #1E90FF;">Computer Networks</span> Course</span></h1>
      
    </center>
  </body>
</html>
```



```
<h1><span class="title">Welcome to <span style="color: #1E90FF;">Computer Networks</span> Course</span></h1>

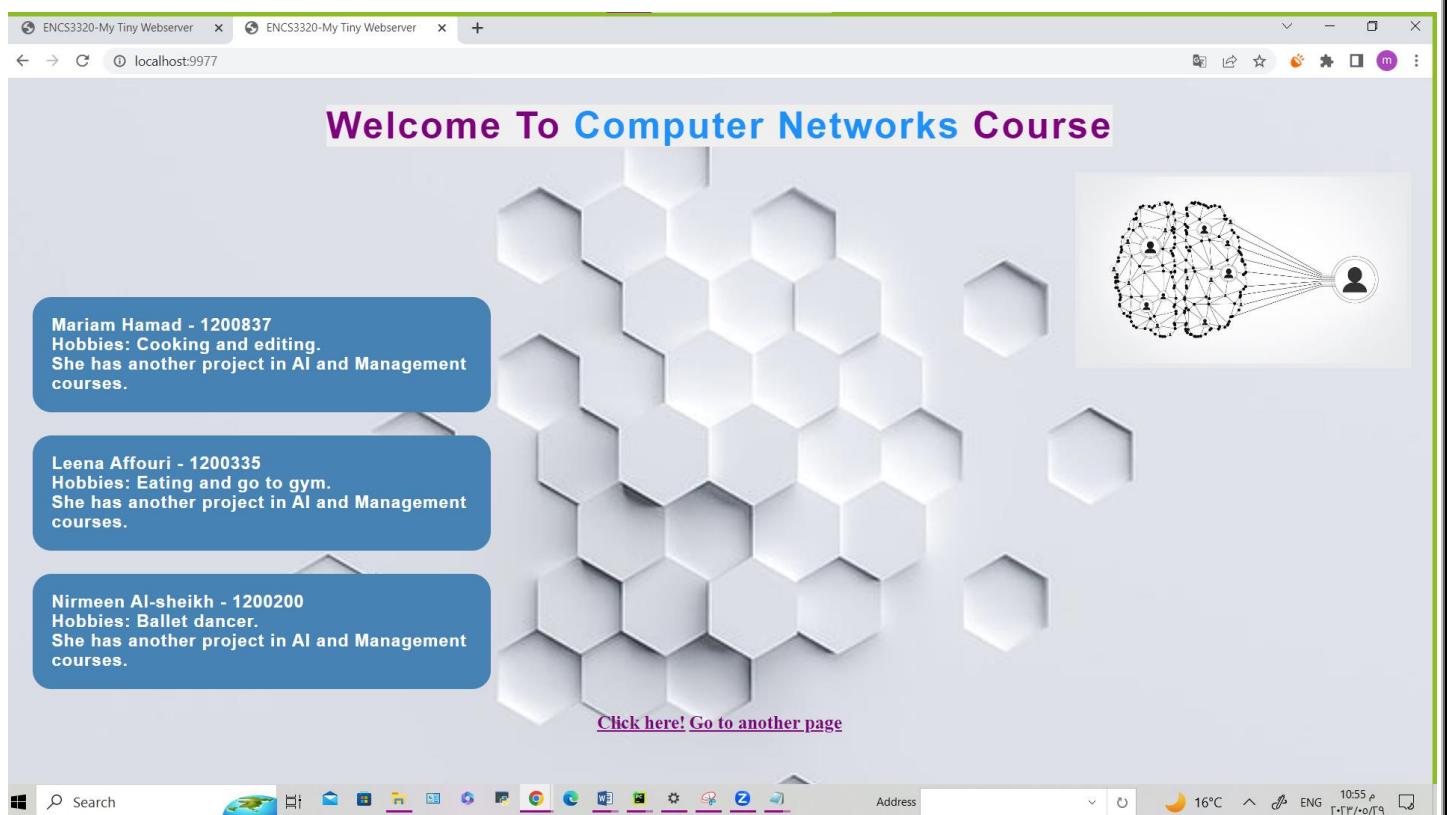
</center>
<p id="first">Mariam Hamad - 1200837<br>
  Hobbies: Cooking and editing.<br>
  She has another project in AI and Management courses.</p>
<p id="last">Leena Affouri - 1200335<br>
  Hobbies: Eating and go to gym.<br>
  She has another project in AI and Management courses.</p>
<p id="last">Nirmeen Al-sheikh - 1200200<br>
  Hobbies: Ballet dancer.<br>
  She has another project in AI and Management courses.</p>
<style>
  .custom-link {
    font-size: 20px;
    font-weight: bold;
    color: #800080; /* Text color */
    font-weight: bold; /* Font weight */
  }
</style>
<center>
  <a href="https://www.w3schools.com/python/gloss_python_multi_line_strings.asp" target="_blank" class="custom-link">Click here!</a>
  <a href="Link.html" target="_blank" class="custom-link"> Go to another page</a>
</center>
</body>
</html>
```



The file that was sent by the server contains an HTML webpage that has in the title our course code ENCS3320-My Tiny Webserver. Then in the webpage we have in the header Welcome to our course Computer Networks," then in the body we have our names and ID's and some information about us, including projects that we have done during different courses, some skills, and hobbies. Also, we add a background, another image, and finally a pattern to open a W3Schools page. and we use CSS to make the page look nice. It is the background text box and the color, size, and type of the texts. The width and length of the text box and the position of boxes.

The output:

localhost: 9977/



localhost: 9977/en

Mariam Hamad - 1200837
Hobbies: Cooking and editing.
She has another project in AI and Management courses.

Leena Affouri - 1200335
Hobbies: Eating and go to gym.
She has another project in AI and Management courses.

Nirmeen Al-sheikh - 1200200
Hobbies: Ballet dancer.
She has another project in AI and Management courses.

Welcome To Computer Networks Course

Click here! [Go to another page](#)

When click in click here.

ENCS3320-My Tiny Webserver | ENCS3320-My Tiny Webserver | ENCS3320-My First Webserver | Python Multiline Strings | موقع الويب المصنفو... | ENCS3320

Tutorials ▾ References ▾ Exercises ▾ Bootcamps Videos Upgrade Get Certified Create Website Sign Up Log in

HTML CSS JAVASCRIPT SQL PYTHON JAVA PHP BOOTSTRAP HOW TO W3.CSS C C++ C# REACT R

Python Tutorial

- Python HOME
- Python Intro
- Python Get Started
- Python Syntax
- Python Comments
- Python Variables
- Python Data Types
- Python Numbers
- Python Casting
- Python Strings
- Python Booleans
- Python Operators
- Python Lists
- Python Tuples
- Python Sets
- Python Dictionaries
- Python If...Else
- Python While Loops
- Python For Loops
- Python Functions
- Python Lambda
- Python Arrays
- Python Classes/Objects

French Arabic English

احصل على فرصة ممولة منخفضة ونحو 24/5 دون التأهيل محفوظ بالمخاطر وقد يؤدي إلى حضارة راس المال طالب مكافأة إيداع لغاية \$10000 Windsor Brokers

Build your career. Get Full Access.

Lifelong access to all W3Schools courses and certifications!

SAVE 770\$ Start today

Python Multiline Strings

◀ Python Glossary

Multiline Strings

You can assign a multiline string to a variable by using three quotes:

Example

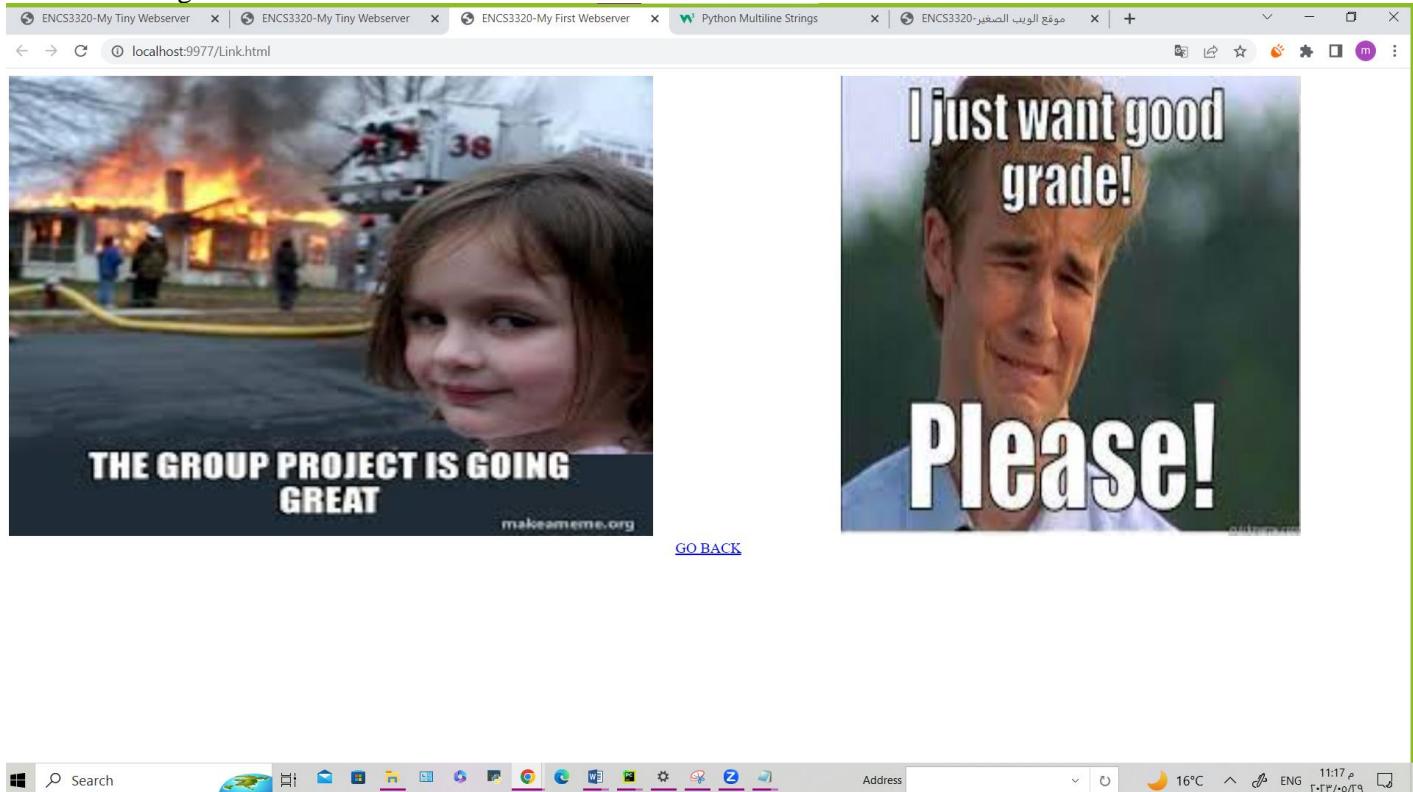
You can use three double quotes:

```
a = """Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua."""
print(a)
```

Get your own Python Server

Windows Search Address 16°C ENG 10:54 AM

When click in go to another link.

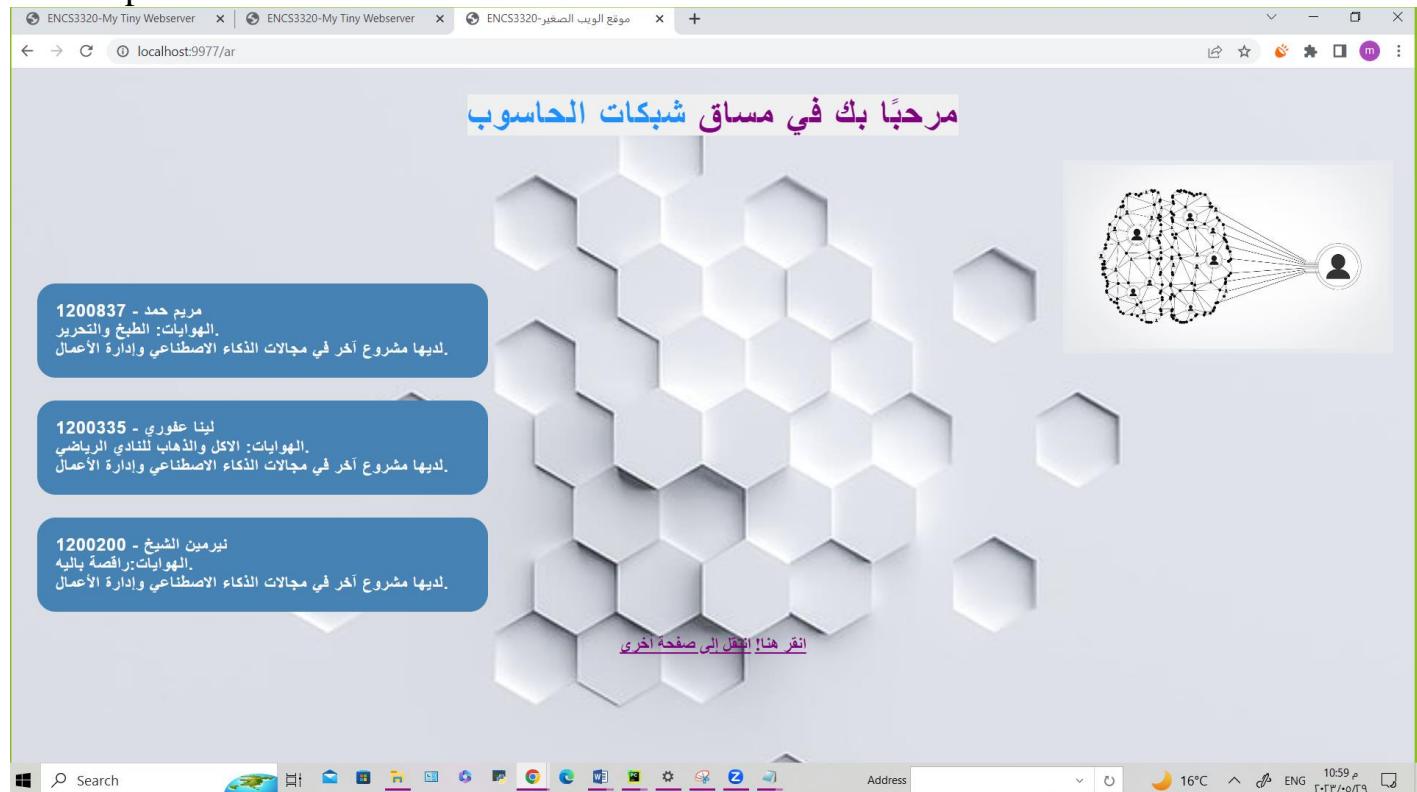


2- If the request is /ar then the server response with `main_ar.html` which is an Arabic version of `main_en.html`

The code:

```
1  <!DOCTYPE html>
2  <html lang="ar">
3  <head>
4      <meta charset="UTF-8">
5      <title>ENCS3320 - موقعي التوب المفتوح</title>
6      <link rel="stylesheet" href="cssFile.css" type="text/css">
7      <style>
8          body {
9              background-image: url("https://c4.wallpaperflare.com/wallpaper/483/576/977/abstract-hexagon-simple-minimalism-wallpaper.jpg");
10             background-size: cover;
11             background-repeat: no-repeat;
12         }
13
14         #object1 {
15             float: right;
16             margin-right: 20px;
17             border: 5px solid rgba(255, 255, 255, 0.2);
18         }
19     </style>
20     <style>
21         .title {
22             background-color: #F0F0F0; /* Set the background color */
23         }
24     </style>
25 </head>
26 <body>
27 <center>
28     <h1><span class="title">شيكات الحاسوب</span><span style="color: #1E90FF;">من هنا بك في مساق</span></h1>
29     
30 </center>
31 <p id="first">1200837 - محمد حمد -  
الهوايات: الطبخ والتحرير.  
نديها مشروع آخر في مجالات التكاء، الاصطدام وإدارة الأعمال.
32 <p id="last">1200335 - لينا عذوري  
الهوايات: الرحلات والتأديب الرياضي.  
نديها مشروع آخر في مجالات التكاء، الاصطدام وإدارة الأعمال.
33 <p id="last">1200200 - سيدرين الشيخ -  
الهوايات: رقصة بالرقص.  
نديها مشروع آخر في مجالات التكاء، الاصطدام وإدارة الأعمال.
34     <style>
35         .custom-link {
36             /* Add your desired styles here */
37             font-size: 20px;
38             font-weight: bold;
39             color: #800080; /* Text color */
40             font-weight: bold; /* Font weight */
41         }
42     </style>
43 <center>
44     <a href="https://www.w3schools.com/python/gloss_python_multi_line_strings.asp" target="_blank" class="custom-link">انقر هنا!</a>
45     <a href="Link.html" target="_blank" class="custom-link">انقل إلى صفحة أخرى</a>
46 </center>
47 </body>
48 </html>
49
50 elif object == '/ar':
51     connectionSocket.send("HTTP/1.1 200 OK \r\n".encode())
52     connectionSocket.send("Content-Type: text/html \r\n".encode())
53     connectionSocket.send("\r\n".encode())
54     file2 = open("main_ar.html", "rb")
55     connectionSocket.send(file2.read())
```

The output:



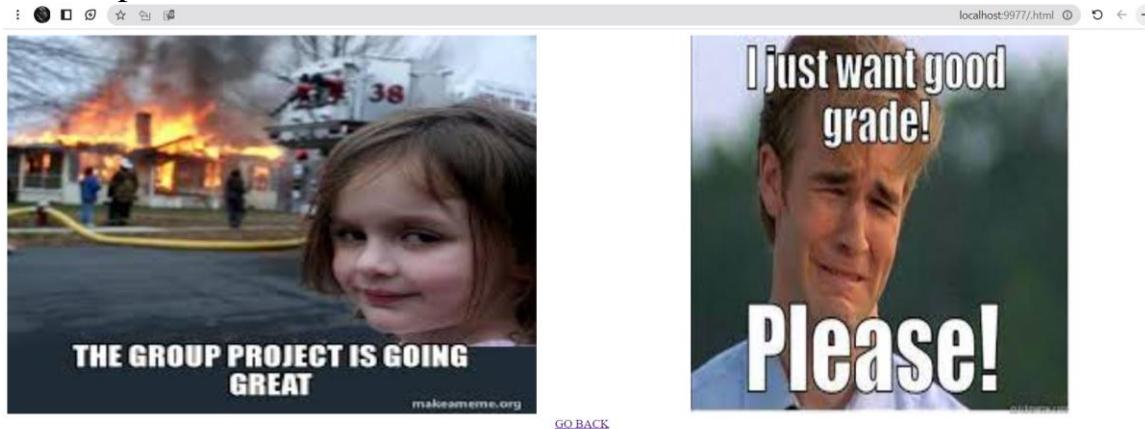
This is a translated copy of the webpage in Arabic.

3- If the request is an **.html file** then the server should send the requested html file with Content-Type: text/html. You can use any html file.

The code:

```
elif object.endswith('.html'):
    connectionSocket.send("HTTP/1.1 200 OK \r\n".encode())
    connectionSocket.send("Content-Type: text/html \r\n".encode())
    connectionSocket.send("\r\n".encode())
    file3 = open("Link.html", "rb")
    connectionSocket.send(file3.read())
```

The output:



4- If the request is a **.css** file then the server should send the requested css file with Content-Type: text/css. You can use any CSS file

The code:

```
1  h1 {  
2      font-family: "Roboto", Arial, sans-serif;  
3      font-size: 40px;  
4      font-weight: bold;  
5      text-align: center;  
6      color: #800080;  
7      letter-spacing: 2px;  
8      text-transform: capitalize;  
9  
10 }  
11  
12 p {  
13     background-color: #4682B4;  
14     border-radius: 20px;  
15     color: white;  
16     font-weight: bold;  
17     text-align: center;  
18     font-family: "Roboto", Arial, sans-serif;  
19     font-size: 18px;  
20     padding: 20px;  
21     margin: 25px;  
22     width: 450px;  
23     letter-spacing: 1px;  
24  
25 }  
26 #first {  
27     text-align: left;  
28     margin-right: 100px; /* Add some right margin */  
29     margin-top: 160px;  
30 }
```

```
26     #first {
27         text-align: left;
28         margin-right: 100px; /* Add some right margin */
29         margin-top: 160px;
30
31     }
32
33     #last {
34         text-align: left;
35         margin-right: 100px; /* Add some left margin */
36         margin-top: 10px;
37
38     }
39
40     #object1 {
41         width: 200px;
42         height: 200px;
43         object-fit: cover;
44         display: block;
45         margin: 0 auto;
46     }
```

The output:

ENC3320-My Tiny Webserver | ENC3320-My Tiny Webserver | ENC3320-My First Webserver | Python Multiline Strings | ENC3320-الويب الصغير | localhost:9977/css

```
h1 {
    font-family: "Roboto", Arial, sans-serif;
    font-size: 40px;
    font-weight: bold;
    text-align: center;
    color: #800080;
    letter-spacing: 2px;
    text-transform: capitalize;
}
p {
    background-color: #4682B4;
    border-radius: 20px;
    color: white;
    font-weight: bold;
    text-align: center;
    font-family: "Roboto", Arial, sans-serif;
    font-size: 18px;
    padding: 20px;
    margin: 25px;
    width: 450px;
    letter-spacing: 1px;
}
#first {
    text-align: left;
    margin-right: 100px; /* Add some right margin */
    margin-top: 10px;
}
#last {
    text-align: left;
    margin-right: 100px; /* Add some left margin */
    margin-top: 10px;
}
#object1 {
    width: 200px;
    height: 200px;
    object-fit: cover;
    display: block;
    margin: 0 auto;
}
```

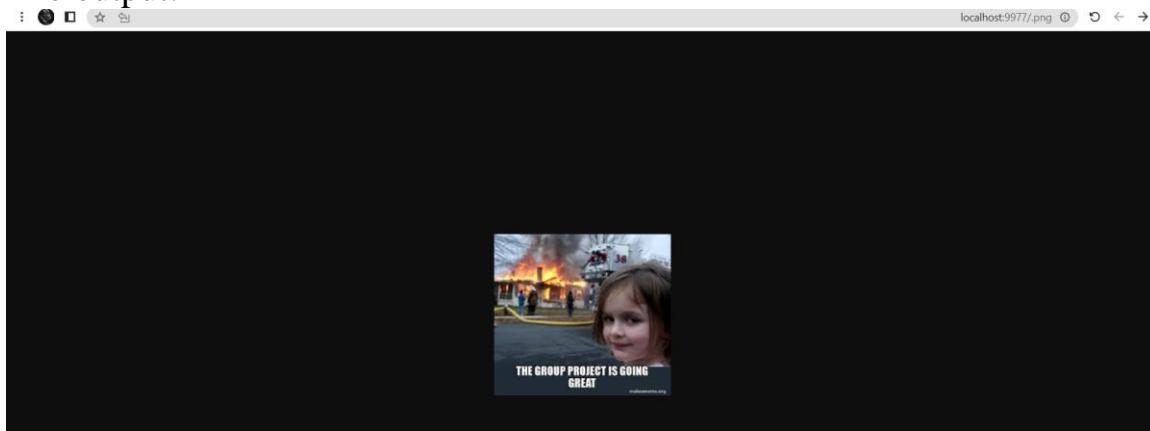


5- If the request is a .png then the server should send the png image with Content-Type: image/png. You can use any image.

The code:

```
elif object.endswith('.png'):
    url = "https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQ2ZE3og8SzGwmH6wIEIDFiMtTI9zTLxxLHXA&usqp=CAU.png"
    connectionSocket.send("HTTP/1.1 200 OK \r\n".encode())
    connectionSocket.send("Content-Type: image/png \r\n".encode())
    connectionSocket.send("\r\n".encode())
    # Download the image from the URL and save it locally
    urllib.request.urlretrieve(url, "image.png")
    # Open the local image file and send its content
    file5 = open("image.png", "rb")
    connectionSocket.send(file5.read())
```

The output:

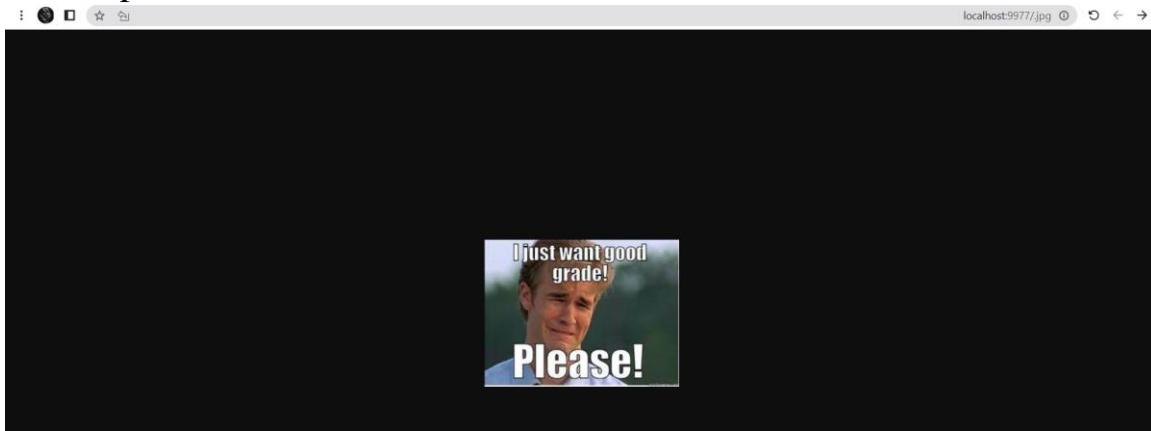


6- If the request is a .jpg then the server should send the jpg image with Content-Type: image/jpeg. You can use any image.

The code:

```
elif object.endswith(".jpg"):
    url = "https://i.imgur.com/fvrP7rn.jpg"
    connectionSocket.send("HTTP/1.1 200 OK\r\n".encode())
    connectionSocket.send("Content-Type: image/jpg \r\n".encode())
    connectionSocket.send("\r\n".encode())
    # Download the image from the URL and save it locally
    urllib.request.urlretrieve(url, "image.jpg")
    # Open the local image file and send its content
    file5 = open("image.jpg", "rb")
    connectionSocket.send(file5.read())
```

The output:



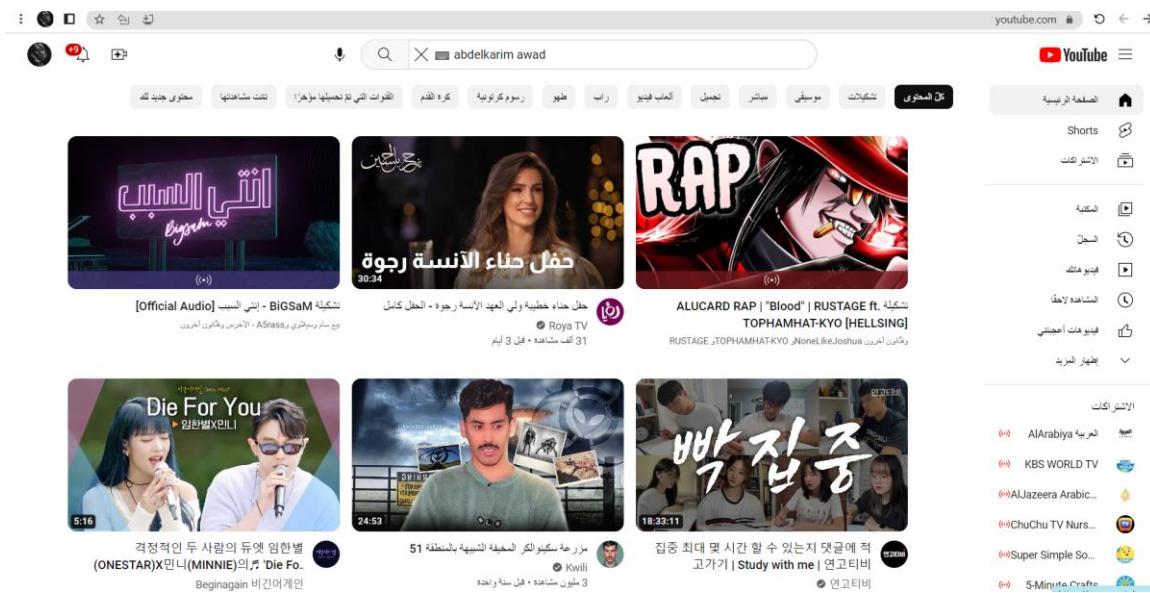
7- Use the status code 307 Temporary Redirect to redirect the following

a- If the request is /yt then redirect to YouTube website

The code:

```
elif object == '/yt':  
    connectionSocket.send("HTTP/1.1 307 Temporary Redirect \r\n".encode())  
    connectionSocket.send("Content-Type: text/html \r\n".encode())  
    connectionSocket.send("Location: https://www.youtube.com/ \r\n".encode())  
    connectionSocket.send("\r\n".encode())
```

The output:

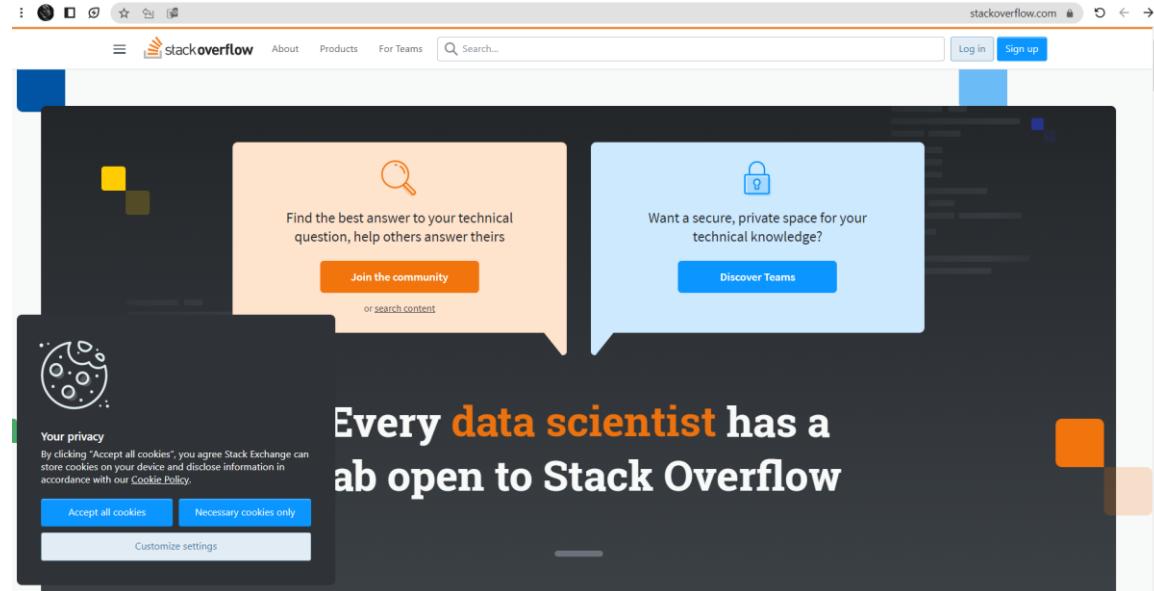


b- If the request is /so then redirect to stackoverflow.com website

The code:

```
elif object == '/so':
    connectionSocket.send("HTTP/1.1 307 Temporary Redirect \r\n".encode())
    connectionSocket.send("Content-Type: text/html \r\n".encode())
    connectionSocket.send("Location: https://stackoverflow.com \r\n".encode())
    connectionSocket.send("\r\n".encode())
```

The output:

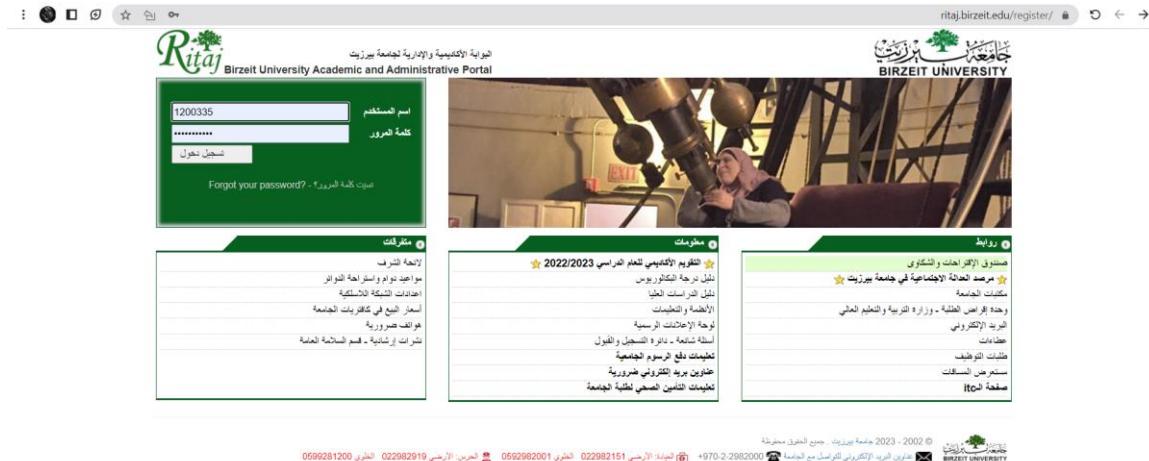


c- If the request is /rt then redirect to ritaj website

The code:

```
elif object == '/rt':
    connectionSocket.send("HTTP/1.1 307 Temporary Redirect \r\n".encode())
    connectionSocket.send("Content-Type: text/html \r\n".encode())
    connectionSocket.send("Location: https://ritaj.birzeit.edu// \r\n".encode())
    connectionSocket.send("\r\n".encode())
```

The output:



8- If the request is wrong or the file doesn't exist the server should return a simple HTML webpage that contains (Content-Type: text/html)

The code:

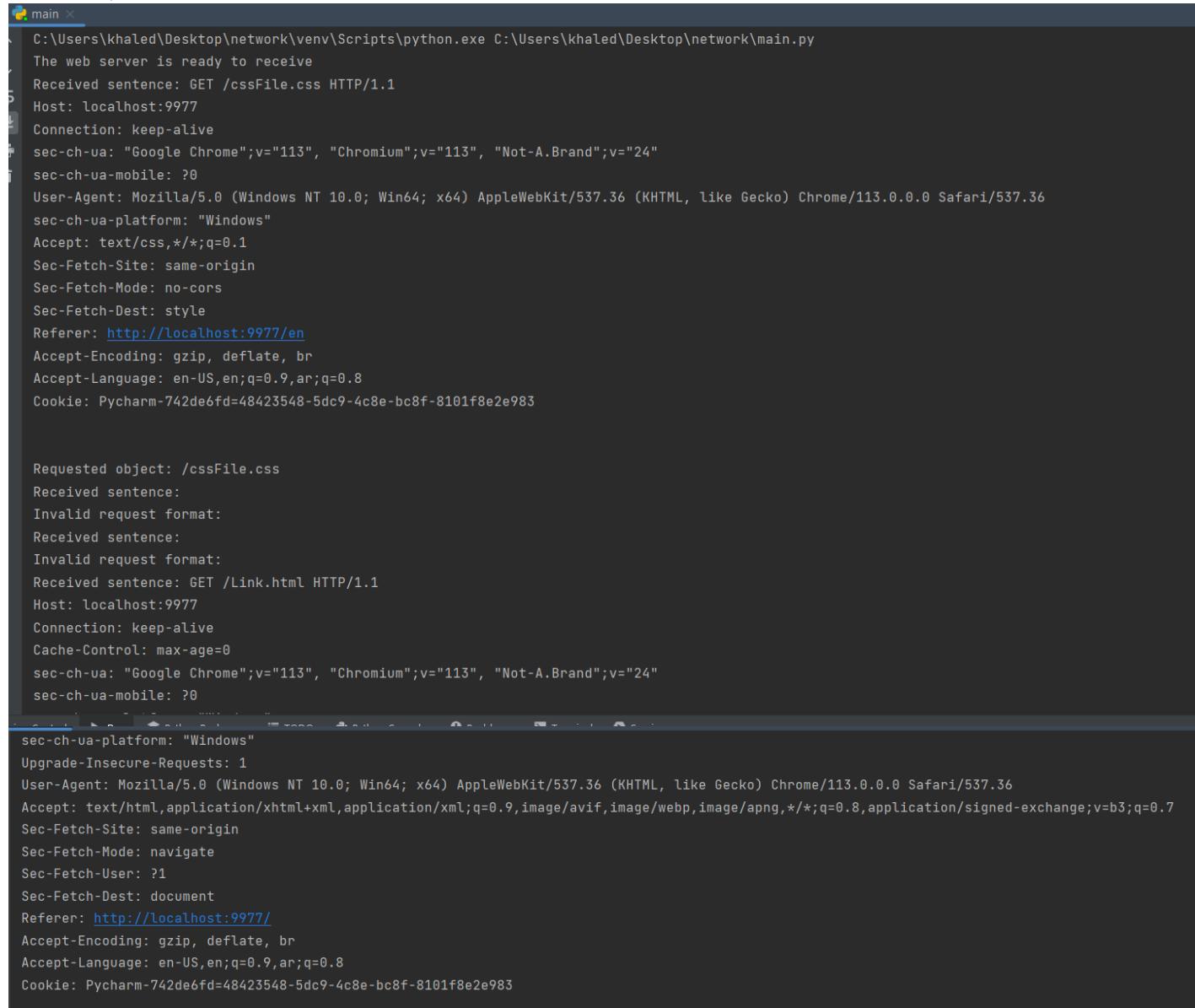
```
else:  
    error_message = f'''HTTP/1.1 404 Not Found <html> <head> <title>Error 404</title> <style>  
        body {{ background-image: url("back.jpg");  
            background-repeat: no-repeat;  
            background-attachment: fixed;  
            background-size: 100% 100%; }}  
        h1 {{ margin-top: 80px; color: red; font-weight: bold; text-align: center; }}  
        h2 {{background-color: #4682B4;  
            border-radius: 20px;  
            color: white;  
            font-weight: bold;  
            text-align: center;  
            font-family: "Roboto", Arial, sans-serif;  
            font-size: 18px;  
            padding: 20px;  
            margin: 25px auto;  
            max-width: 450px;  
            letter-spacing: 1px;}} </style> </head> <body> <h1>The file is not found</h1>  
    <h2>Mariam Hamad - 1200837</h2>  
    <h2>Leena Affouri - 1200335</h2>  
    <h2>Nirmeen Al-sheikh - 1200200</h2>  
    <h2><b>Client IP: <ip></b></p>  
    <h2><b>Client Port: <port></b></p> </body> </html> '''.encode('utf-8')  
connectionSocket.send("HTTP/1.1 404 Not Found\r\n".encode())  
connectionSocket.send("Content-Type: text/html\r\n".encode())  
connectionSocket.send("\r\n".encode())  
connectionSocket.send(error_message)  
connectionSocket.close()  
  
else:  
    print("Invalid request format:", sentence)
```

The output:



If the request is anything else, it shows a page with an error message that the file was not found and our names and IDs, as well as the client IP and client port.

9- The program should print the **HTTP requests** on the **terminal window** (command line window).



The screenshot shows a terminal window titled "main" with the following content:

```
C:\Users\khaled\Desktop\network\venv\Scripts\python.exe C:\Users\khaled\Desktop\network\main.py
The web server is ready to receive
Received sentence: GET /cssFile.css HTTP/1.1
Host: localhost:9977
Connection: keep-alive
sec-ch-ua: "Google Chrome";v="113", "Chromium";v="113", "Not-A.Brand";v="24"
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.0.0 Safari/537.36
sec-ch-ua-platform: "Windows"
Accept: text/css,*/*;q=0.1
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: no-cors
Sec-Fetch-Dest: style
Referer: http://localhost:9977/en
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9,ar;q=0.8
Cookie: Pycharm-742de6fd=48423548-5dc9-4c8e-bc8f-8101f8e2e983

Requested object: /cssFile.css
Received sentence:
Invalid request format:
Received sentence:
Invalid request format:
Received sentence: GET /Link.html HTTP/1.1
Host: localhost:9977
Connection: keep-alive
Cache-Control: max-age=0
sec-ch-ua: "Google Chrome";v="113", "Chromium";v="113", "Not-A.Brand";v="24"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.0.0 Safari/537.36
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
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://localhost:9977/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9,ar;q=0.8
Cookie: Pycharm-742de6fd=48423548-5dc9-4c8e-bc8f-8101f8e2e983
```

```
main x

Requested object: /Link.html
Received sentence: GET /cssFile.css HTTP/1.1
Host: localhost:9977
Connection: keep-alive
sec-ch-ua: "Google Chrome";v="113", "Chromium";v="113", "Not-A.Brand";v="24"
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.0.0 Safari/537.36
sec-ch-ua-platform: "Windows"
Accept: text/css,*/*;q=0.1
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: no-cors
Sec-Fetch-Dest: style
Referer: http://localhost:9977/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9,ar;q=0.8
Cookie: Pycharm-742de6fd=48423548-5dc9-4c8e-bc8f-8101f8e2e983

Requested object: /cssFile.css
Received sentence: GET /ar HTTP/1.1
Host: localhost:9977
Connection: keep-alive
Cache-Control: max-age=0
sec-ch-ua: "Google Chrome";v="113", "Chromium";v="113", "Not-A.Brand";v="24"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.0.0 Safari/537.36
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
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9,ar;q=0.8
Cookie: Pycharm-742de6fd=48423548-5dc9-4c8e-bc8f-8101f8e2e983

Requested object: /ar
Received sentence:
Invalid request format:
Received sentence: GET /.css HTTP/1.1
Host: localhost:9977
Connection: keep-alive
Cache-Control: max-age=0
sec-ch-ua: "Google Chrome";v="113", "Chromium";v="113", "Not-A.Brand";v="24"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.0.0 Safari/537.36
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
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9,ar;q=0.8
Cookie: Pycharm-742de6fd=48423548-5dc9-4c8e-bc8f-8101f8e2e983
```

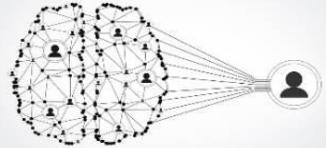
Screenshot from another device 192.168.1.109:

7:11 47% 47%

192.168.1.109:9977/ar

موقع الويب الـ...-ENCS3320
192.168.1.109:9977

مرحبا بك في مساق شبكات الحاسوب



1200837 مريم حمد -
الهوايات: الطبخ والتحرير.
لديها مشروع آخر في مجالات الذكاء الاصطناعي
وإدارة الأعمال.

1200335 لينا عفوري -
الهوايات: الاكل والذهاب للنادي الرياضي.
لديها مشروع آخر في مجالات الذكاء الاصطناعي
وإدارة الأعمال.

1200200 نيرمين الشيخ -
الهوايات: راقصة باليه.
لديها مشروع آخر في مجالات الذكاء الاصطناعي
وإدارة الأعمال.

[انقر هنا! انتقل إلى صفحة أخرى](#)

7:11

47%



192.168.1.109:9977/en|

X

7:11

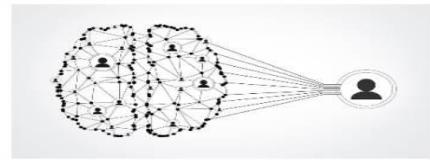
47%

X ENCS3320-My Tiny We...
192.168.1.109:9977



#title on tap.

Welcome To Computer Networks Course



Mariam Hamad - 1200837
Hobbies: Cooking and editing.
She has another project in AI and Management courses.

Leena Affouri - 1200335
Hobbies: Eating and go to gym.
She has another project in AI and Management courses.

Nirmeen Al-sheikh - 1200200
Hobbies: Ballet dancer.
She has another project in AI and Management courses.

[**Click here! Go to another page**](#)

7:11 ... 47%



192.168.1.109:9977/mml



7:11 ... 47%

Error 404
192.168.1.109:9977



HTTP/1.1 404 Not Found

The file is not found

Mariam Hamad - 1200837

Leena Affouri - 1200335

Nirmeen Al-sheikh - 1200200

Client IP: 192.168.1.107

Client Port: 45054



Menu ▾

Sign Up

Log in



HTML



Python Multiline Strings

[◀ Python Glossary](#)

Multiline Strings

You can assign a multiline string to a variable by using three quotes:

Example

[Get your own Python Server](#)

You can use three double quotes:

```
a = """Lorem ipsum dolor sit amet,  
consectetur adipiscing elit,  
sed do eiusmod tempor incididunt  
ut labore et dolore magna  
aliqua."""  
print(a)
```

X ⚠ ENCS3320-My First W...
192.168.1.109:9977



[GO BACK](#)

THE END

