Fuat Yiğit Koçyiğit – 16429085948

CMPE472 Section 1

26.10.2022

TED UNIVERSITY

CMPE 472

Programming Assignment I: Web Server

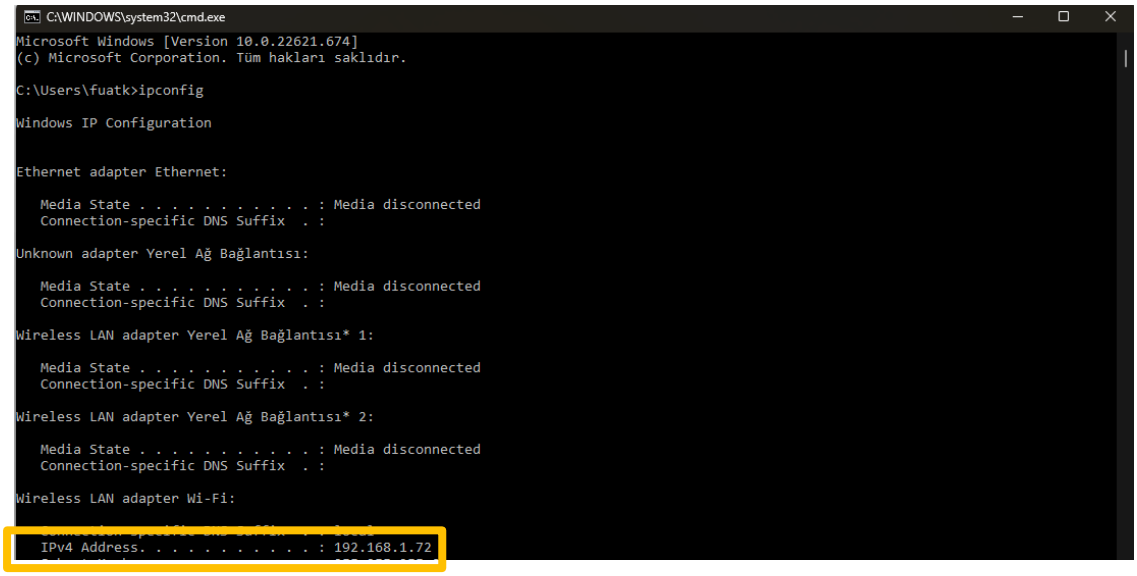
Assignment Summary

In this assignment, I learned the basics of socket programming for TCP connections in Python. I learned creating a socket, binding it to a specific address and port, as well as sending and receiving an HTTP packet.

The web server I created with python is able to accept and parse the HTTP requests, return the requested file from the server (my HTML page in this assignment), create an HTTP response message according to the situation (HTTP 404 Not Found if the requested file does not exist and HTTP 200 OK if the requested file is found and returned) and send that response to the client.

Assignment Steps

- In order to create a server program, I firstly needed my **IP number**. To learn my IP number, I opened **cmd.exe** program. Then, I typed "**ipconfig**" command and got these results:

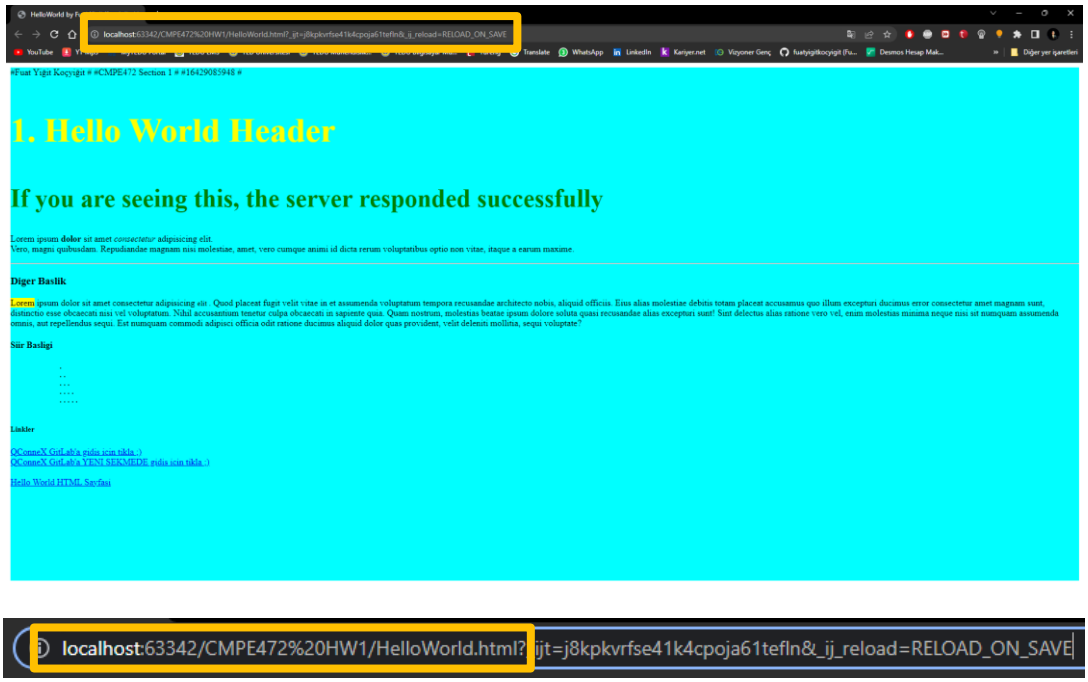


Cmd.exe ipconfig results

As it could be seen in the yellow box, I learned that my IP address is "**192.168.1.72**".

- After that, I **created a python project** which includes "**main.py**" and "**HelloWorld.html**" files. In the main.py file, I firstly **pasted the sceleton code**.

- Then, I opened up the HelloWorld.html file and **created a random html page**. I opened it in Google Chrome browser and got these results:



HelloWorld.html file results

In this page, I see that the used port is **63342**. So, I will continue using it as my port for the server.

1. After getting all the required information and created the html file, I opened the main.py file. In the **first fill space (# Prepare a sever socket)**, I wrote these codes:

With the ***serverSocket.bind*** function, I created my socket with giving the IP address and Port number **(('192.168.1.72', 63342))**.

After that, with ***serverSocket.listen(1)***, I started the server to listen incoming requests from that socket.

```
# Prepare a sever socket
# Fill in start
# serverSocket.bind((IP Address, Port))
serverSocket.bind(('192.168.1.72', 63342))
serverSocket.listen(1)
# Fill in end
```

2. In the **second fill space (# Establish the connection)**, I wrote these codes:

With the **serverSocket.accept()** method, server accepts the incoming requests. After return, socket connection is established.

```
# Fill in start
connectionSocket, addr = serverSocket.accept()
# Fill in end
```

3. In the **third fill space**, I wrote these codes:

With the **connectionSocket.recv(1024).decode()** method, server receives bytes from the connected socket and decode to read.

```
# Fill in start
message = connectionSocket.recv(1024).decode()
# Fill in end
```

4. In the **fourth fill space**, I wrote these codes:

With the **f.read()** method, we created the outputdata that we will use as output.

```
# Fill in start
outputdata = f.read()
# Fill in end
```

5. In the **fifth fill space (# Send one HTTP header line into socket)**, I wrote these codes:

With the ***connectionSocket.send()*** function, server transmits the HTTP header line to the socket with the **HTTP/1.1 200 OK** message.

```
# Send one HTTP header line into socket
# Fill in start
connectionSocket.send("\nHTTP/1.1 200 OK\nContent-Type: text/html\n\n".encode())
# Fill in end
```

6. In the **sixth fill space (# Send response message for file not found)**, I wrote these codes:

With the ***connectionSocket.send()*** function again, I send the failed response message to the socket with **HTTP/1.1 404 Not Found** message.

```
# Send response message for file not found
# Fill in start
connectionSocket.send("\nHTTP/1.1 404 Not Found\n\n".encode())
# Fill in end
```

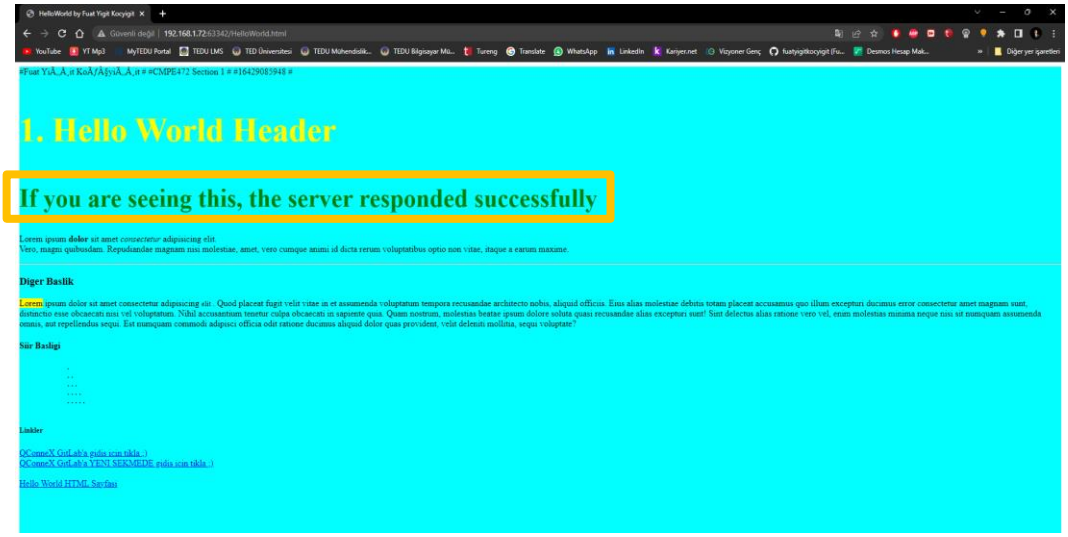
7. In the **seventh fill space (# Close client socket)**, I wrote these codes:

```
# Close client socket
# Fill in start
connectionSocket.close()
# Fill in end
```

With the ***connectionSocket.close()*** function, I closed the client socket successfully.

Screenshots

1. By entering the ***"http://192.168.1.72:63342/HelloWorld.html"*** link, we can see that client browser is verifying that it actually receives the contents of the HTML file from the server.



2. By entering the ***"http://192.168.1.72:63342/ByeWorld.html"*** link, we can see the result we get while trying to get a file that is not present at the server. It got the 404 message as expected.