CSE-476 Mobile Communications Networks TERM PROJECT REPORT

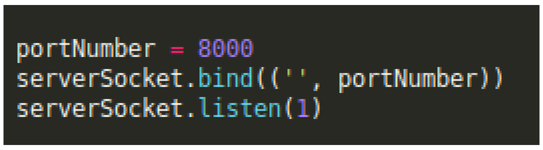
Emirhan UZUN
171044019

Assignment 1 : Web Server

In this assignment, we are asked to develop a simple Web server in Python that is capable of processing only one request. Specially, out Web server will create a connection socket when contacted by a client (browser); receive the HTTP request from this connection; parse the request to determine the specific file being requested; get the requested file from the server's file system; create an HTTP response message consisting of the requested file preceded by header lines; and send the response over the TCP connection to the requesting browser. If a browser request a file that is not present in our server, our server should return a "404 Not Found" error message.
We had a skeleton code for Web server and I filled it with some researches.
Now I had a WebServer.py and I create a http file which is "HelloWorld.html". After completing the skeleton code, I send a http header line into socket. Then I send the content of the requested file to the client. For the files which is not present my server, I return an error message which is "404 Not Found". I kept WebServer.py and HelloWorld.hhtml file into same directory. After run the code, I got some results.

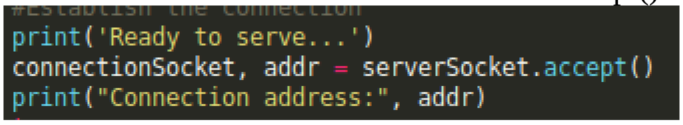
CODE EXPLANATION

Firstly, I determined the tcp port which is 8000.
Then it create socket, bind it with bind() function. Then the socket has been put into listening mode with listen() function.



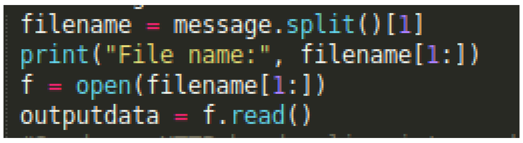
```
portNumber = 8000
serverSocket.bind(('', portNumber))
serverSocket.listen(1)
```

If the connection has been established with accept() function, it will listen requests.



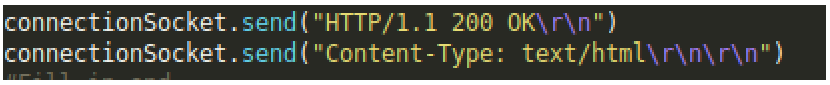
```
print('Ready to serve...')
connectionSocket, addr = serverSocket.accept()
print("Connection address:", addr)
```

After that the requested file is read with using read() function.



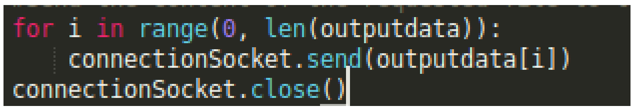
```
filename = message.split()[1]
print("File name:", filename[1:])
f = open(filename[1:])
outputdata = f.read()
```

An HTTP header line is sent to the socket with using send() function. At this stage, the connection is tested.



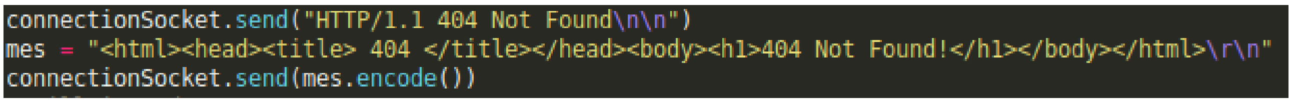
```
connectionSocket.send("HTTP/1.1 200 OK\r\n")
connectionSocket.send("Content-Type: text/html\r\n\r\n")
```

The content of the requested file is sent to the client with using send() function.

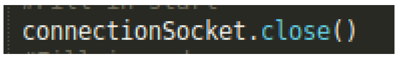


```
for i in range(0, len(outputdata)):
    connectionSocket.send(outputdata[i])
connectionSocket.close()
```

If the file cannot be found, the error message is sent with send() function.



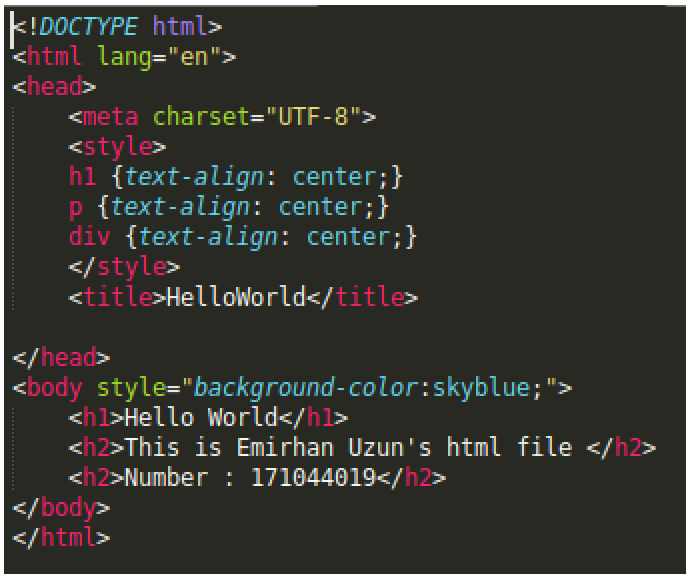
```
connectionSocket.send("HTTP/1.1 404 Not Found\n\n")
mes = "<html><head><title> 404 </title></head><body><h1>404 Not Found!</h1></body></html>\r\n"
connectionSocket.send(mes.encode())
```



```
connectionSocket.close()
```

Client socket is closed with using close function() at the end.

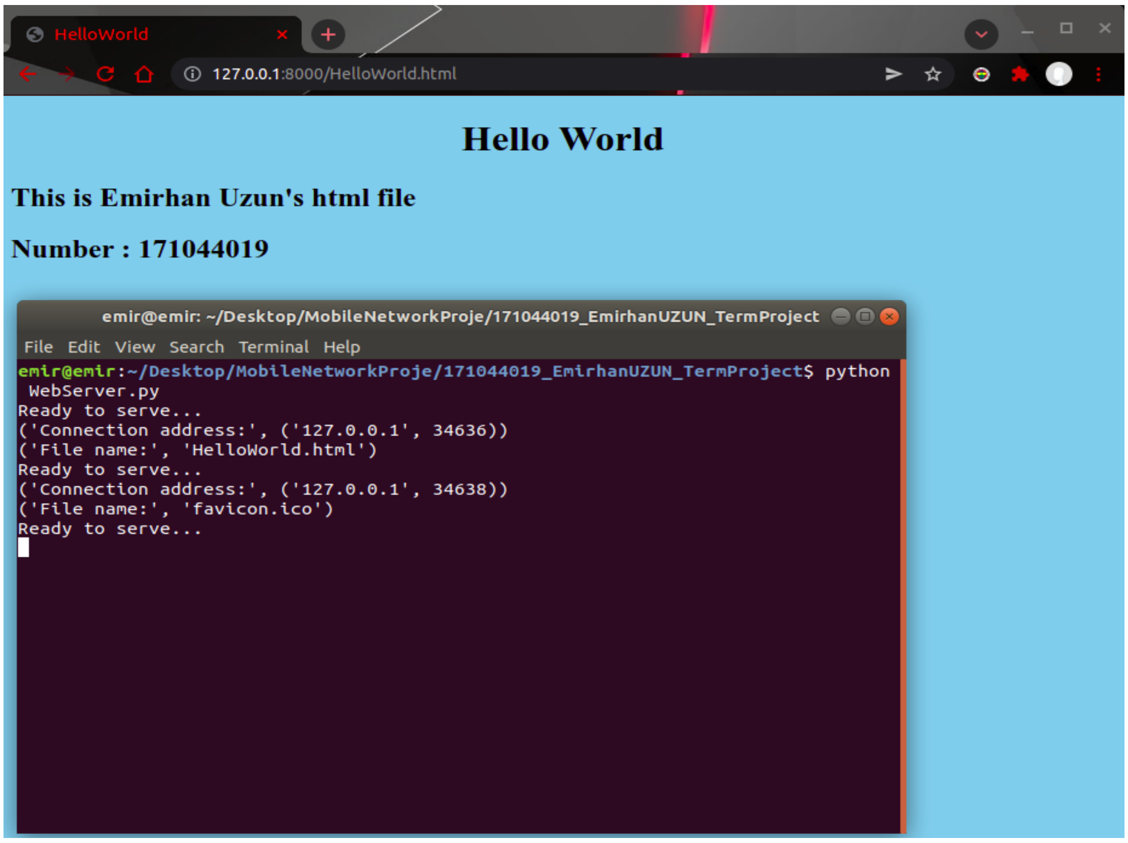
HTML CODE EXPLANATION



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <style>
    h1 {text-align: center;}
    p {text-align: center;}
    div {text-align: center;}
    </style>
    <title>HelloWorld</title>

</head>
<body style="background-color:skyblue;">
    <h1>Hello World</h1>
    <h2>This is Emirhan Uzun's html file </h2>
    <h2>Number : 171044019</h2>
</body>
</html>
```

RESULTS

This is the terminal which I run WebServer.py and browser. I got these results.


HelloWorld
127.0.0.1:8000/HelloWorld.html
emir@emir: ~/Desktop/MobileNetworkProje/171044019_EmirhanUZUN_TermProject
File Edit View Search Terminal Help

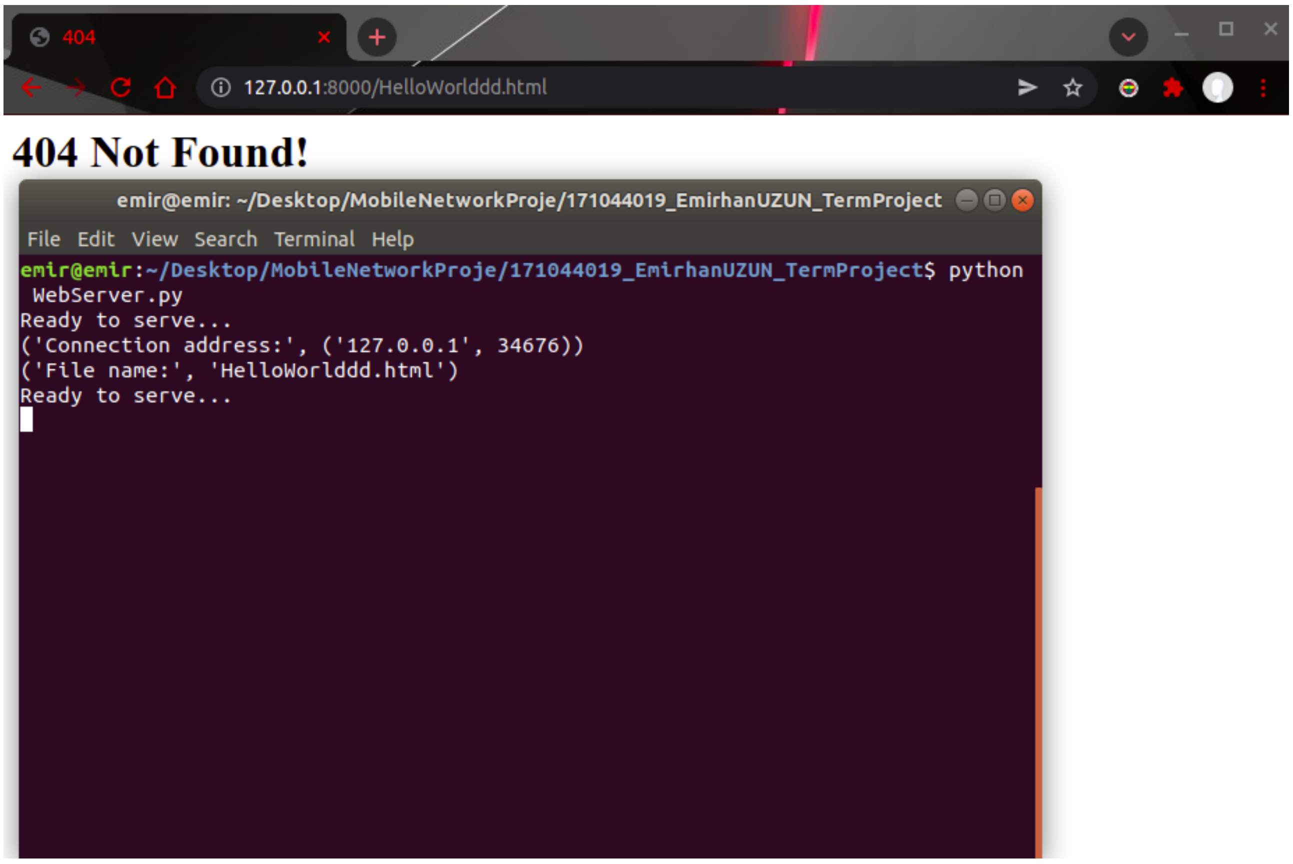
Hello World

This is Emirhan Uzun's html file

Number : 171044019

```
emir@emir:~/Desktop/MobileNetworkProje/171044019_EmirhanUZUN_TermProject$ python
 WebServer.py
Ready to serve...
('Connection address:', ('127.0.0.1', 34636))
('File name:', 'HelloWorld.html')
Ready to serve...
('Connection address:', ('127.0.0.1', 34638))
('File name:', 'favicon.ico')
Ready to serve...
```

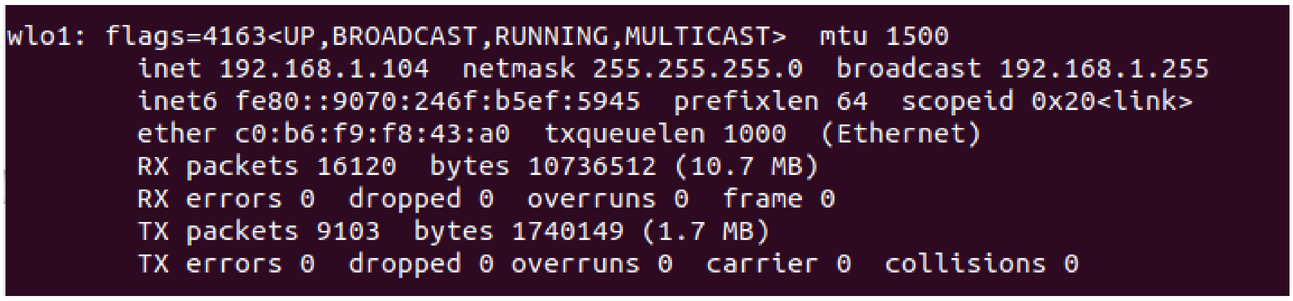
If I run the server with file which is not found file, it prints the "404 Not Found!" message.


404
127.0.0.1:8000/HelloWorlddd.html
emir@emir: ~/Desktop/MobileNetworkProje/171044019_EmirhanUZUN_TermProject
File Edit View Search Terminal Help

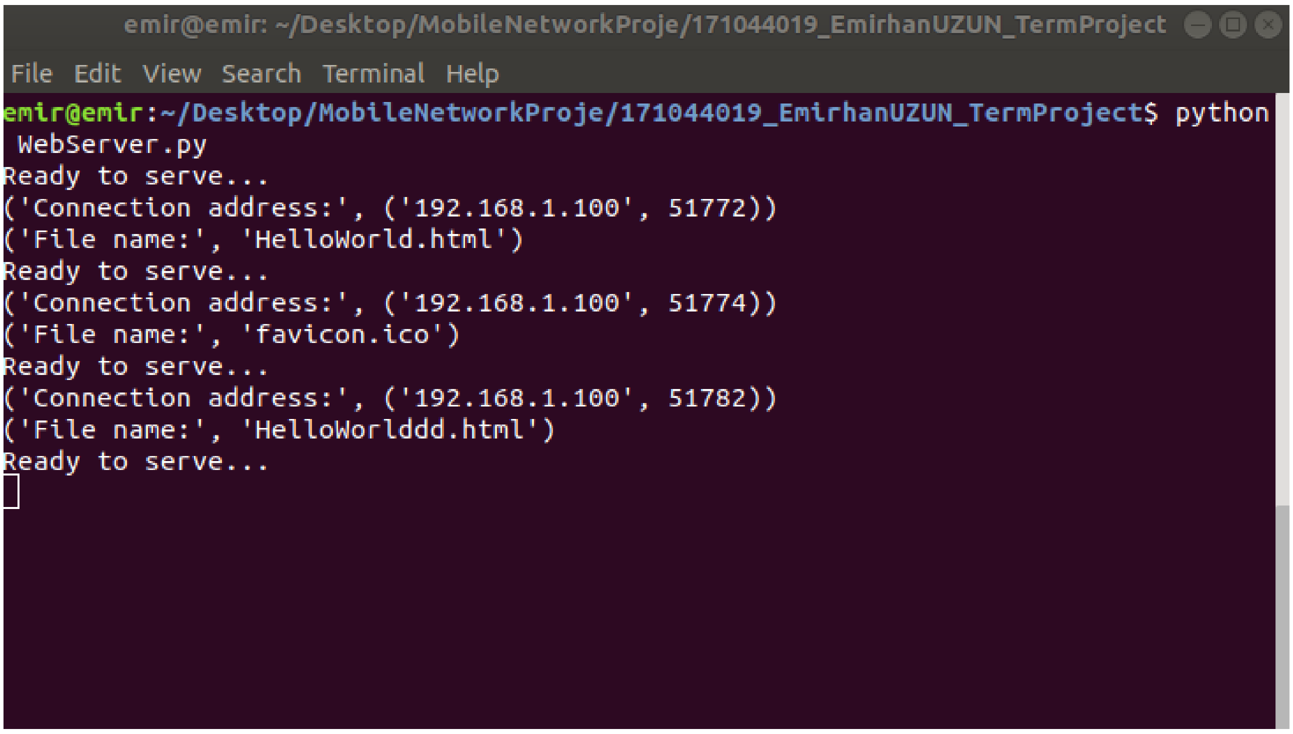
404 Not Found!

```
emir@emir:~/Desktop/MobileNetworkProje/171044019_EmirhanUZUN_TermProject$ python
 WebServer.py
Ready to serve...
('Connection address:', ('127.0.0.1', 34676))
('File name:', 'HelloWorlddd.html')
Ready to serve...
```

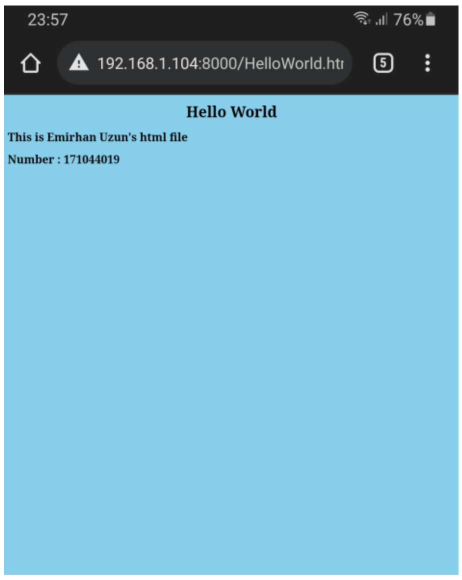
Also I tried this with my IPv4 address and my phone. My address is 192.168.1.104.
It works correctly ! Again I gave wrong file name to try, it gave an error.



```
wlo1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.104  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::9070:246f:b5ef:5945  prefixlen 64  scopeid 0x20<link>
        ether c0:b6:f9:f8:43:a0  txqueuelen 1000  (Ethernet)
        RX packets 16120  bytes 10736512 (10.7 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 9103  bytes 1740149 (1.7 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```



```
emir@emir: ~/Desktop/MobileNetworkProje/171044019_EmirhanUZUN_TermProject
File  Edit  View  Search  Terminal  Help
emir@emir:~/Desktop/MobileNetworkProje/171044019_EmirhanUZUN_TermProject$ python
 WebServer.py
Ready to serve...
('Connection address:', ('192.168.1.100', 51772))
('File name:', 'HelloWorld.html')
Ready to serve...
('Connection address:', ('192.168.1.100', 51774))
('File name:', 'favicon.ico')
Ready to serve...
('Connection address:', ('192.168.1.100', 51782))
('File name:', 'HelloWorlddd.html')
Ready to serve...
```

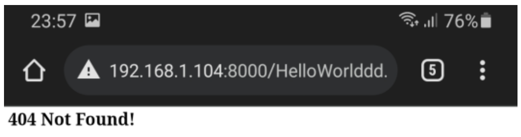


23:57 76%
192.168.1.104:8000/HelloWorld.htı

Hello World

This is Emirhan Uzun's html file

Number : 171044019

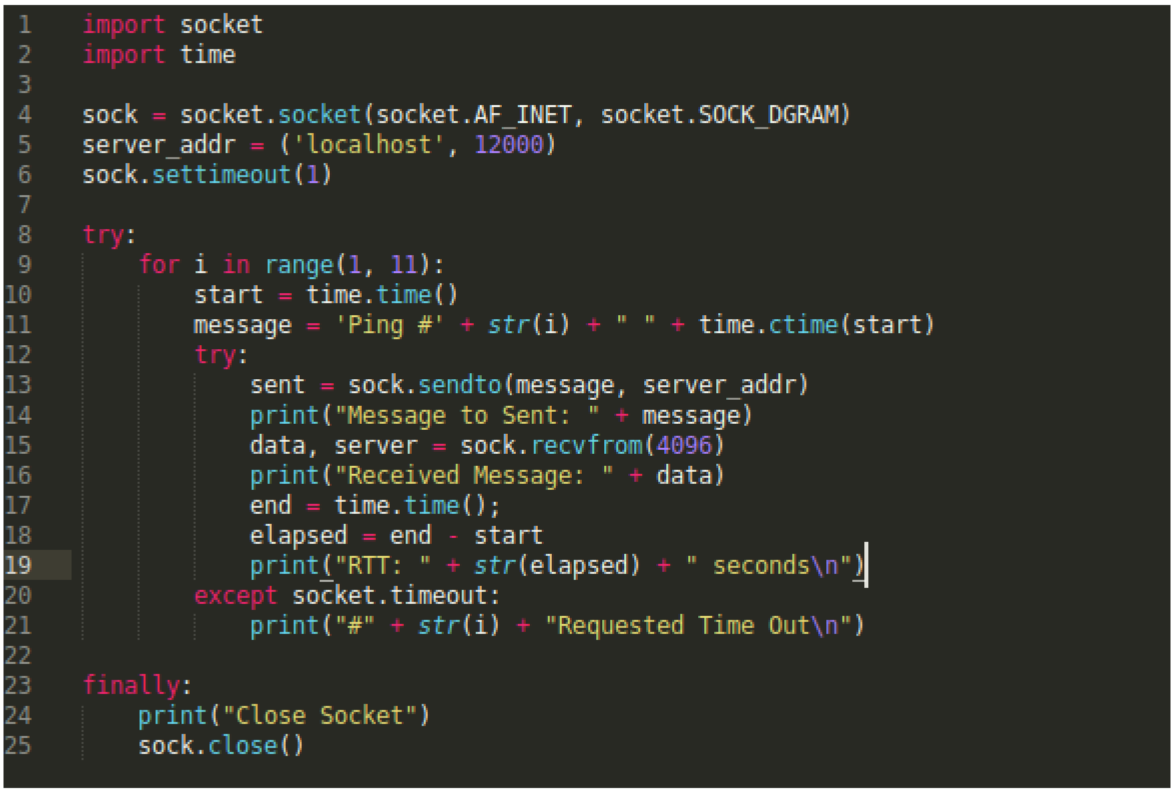


23:57 76%
192.168.1.104:8000/HelloWorlddd.

404 Not Found!

Assignment 2 : UDP Pinger

For Assignment2, we are asked to write a client ping program with using UDP between client and server. I did some researches about this assignment and I copy the UDPPingerServer.py skeleton code which was given from our teacher.
Then I started implementing UDPPingerClient.py code.
Firstly, I create a UDP socket on localhost.
After that, using setTimeout() function, I set the timeout value on a datagram socket.
Client wait up to once second for a reply from server.
This client code sends 10 ping messages using UDP.
It prints the response message if there is a response from the server. Time information format is like day of the week, month, day, hour, minute, second, year.
The ping message is sent to the server address with sendto() function. The client gets the response message from the socket using the recvfrom() function.
And it calculates and prints the round trip time.
To calculate RTT time, I used time() function to get instant time data.
The difference of stored time is Round Trip Time.
If the response are not received from server, it prints request timed out message.
At the end, close the client socket.
For the seeing results firstly I run the server program after that I run the client program and I got some results. I will show these outputs.

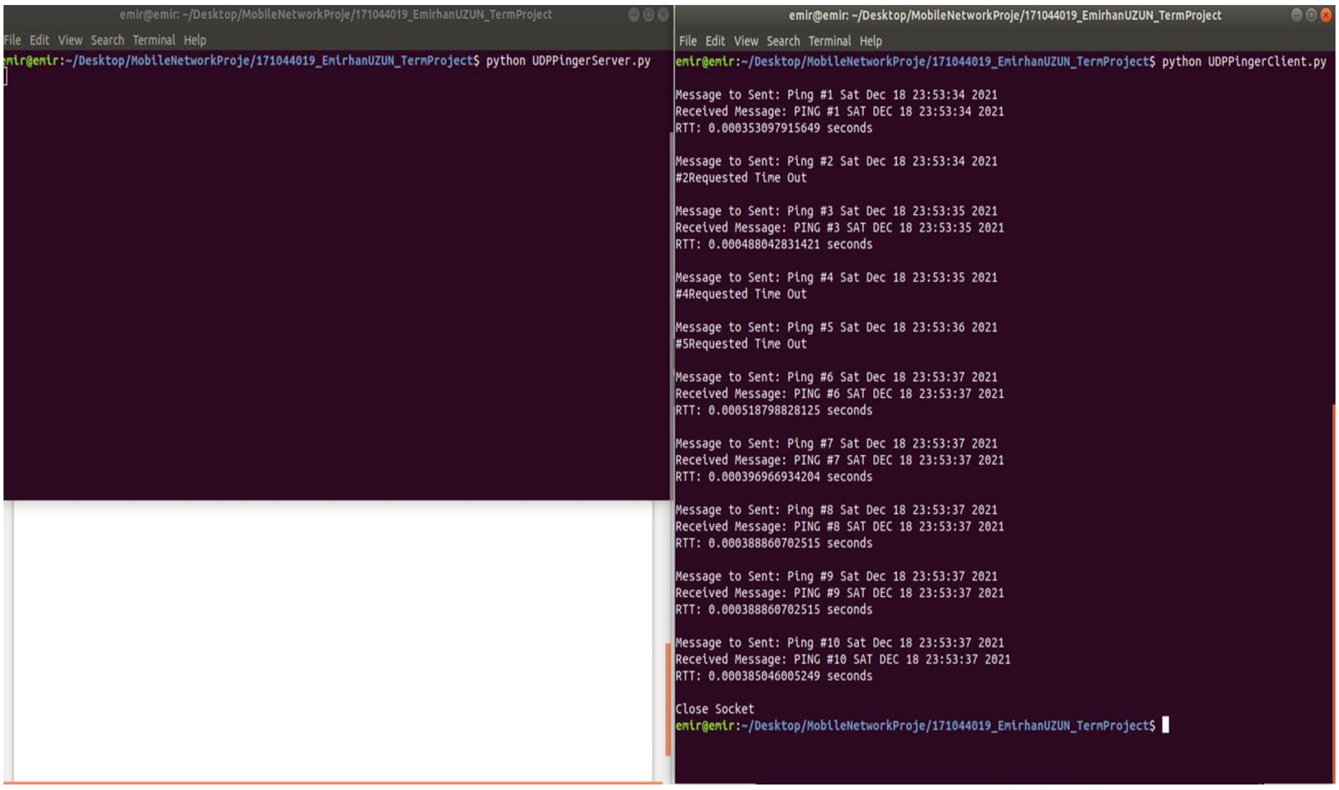


```python
import socket
import time

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
server_addr = ('localhost', 12000)
sock.settimeout(1)

try:
    for i in range(1, 11):
        start = time.time()
        message = 'Ping #' + str(i) + " " + time.ctime(start)
        try:
            sent = sock.sendto(message, server_addr)
            print("Message to Sent: " + message)
            data, server = sock.recvfrom(4096)
            print("Received Message: " + data)
            end = time.time();
            elapsed = end - start
            print("RTT: " + str(elapsed) + " seconds\n")
        except socket.timeout:
            print("#" + str(i) + "Requested Time Out\n")

finally:
    print("Close Socket")
    sock.close()
```

RESULTS

-First Result

For instance, in ping 2,4 and 5, the response aren’t received from server. In that cases, it prints timed out message.

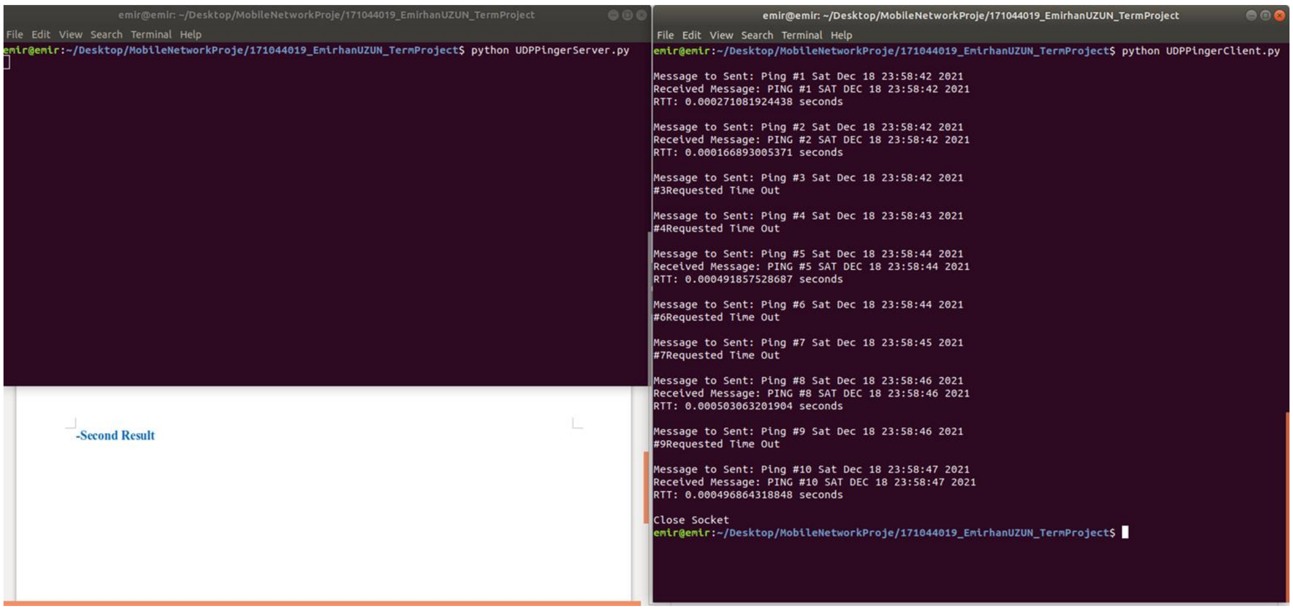


```
emir@emir:~/Desktop/MobileNetworkProje/171044019_EmirhanUZUN_TermProject$ python UDPPingerServer.py
```

```
emir@emir:~/Desktop/MobileNetworkProje/171044019_EmirhanUZUN_TermProject$ python UDPPingerClient.py

Message to Sent: Ping #1 Sat Dec 18 23:53:34 2021
Received Message: PING #1 SAT DEC 18 23:53:34 2021
RTT: 0.000353097915649 seconds

Message to Sent: Ping #2 Sat Dec 18 23:53:34 2021
#2Requested Time Out

Message to Sent: Ping #3 Sat Dec 18 23:53:35 2021
Received Message: PING #3 SAT DEC 18 23:53:35 2021
RTT: 0.000488042831421 seconds

Message to Sent: Ping #4 Sat Dec 18 23:53:35 2021
#4Requested Time Out

Message to Sent: Ping #5 Sat Dec 18 23:53:36 2021
#5Requested Time Out

Message to Sent: Ping #6 Sat Dec 18 23:53:37 2021
Received Message: PING #6 SAT DEC 18 23:53:37 2021
RTT: 0.000518798828125 seconds

Message to Sent: Ping #7 Sat Dec 18 23:53:37 2021
Received Message: PING #7 SAT DEC 18 23:53:37 2021
RTT: 0.000396966934204 seconds

Message to Sent: Ping #8 Sat Dec 18 23:53:37 2021
Received Message: PING #8 SAT DEC 18 23:53:37 2021
RTT: 0.000388860702515 seconds

Message to Sent: Ping #9 Sat Dec 18 23:53:37 2021
Received Message: PING #9 SAT DEC 18 23:53:37 2021
RTT: 0.000388860702515 seconds

Message to Sent: Ping #10 Sat Dec 18 23:53:37 2021
Received Message: PING #10 SAT DEC 18 23:53:37 2021
RTT: 0.000385046005249 seconds

Close Socket
emir@emir:~/Desktop/MobileNetworkProje/171044019_EmirhanUZUN_TermProject$
```

-Second Result

In this example, in ping 3,4,6,7 and 9 , there is no response from server.



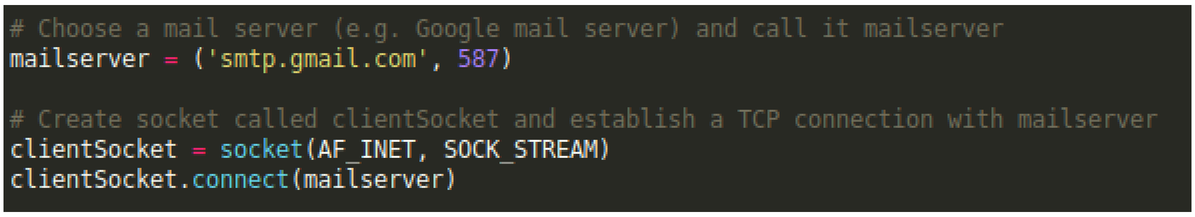
```
emir@emir:~/Desktop/MobileNetworkProje/171044019_EmirhanUZUN_TermProject$ python UDPPingerServer.py
```

```
emir@emir:~/Desktop/MobileNetworkProje/171044019_EmirhanUZUN_TermProject$ python UDPPingerClient.py

Message to Sent: Ping #1 Sat Dec 18 23:58:42 2021
Received Message: PING #1 SAT DEC 18 23:58:42 2021
RTT: 0.000271081924438 seconds

Message to Sent: Ping #2 Sat Dec 18 23:58:42 2021
Received Message: PING #2 SAT DEC 18 23:58:42 2021
RTT: 0.000166893005371 seconds

Message to Sent: Ping #3 Sat Dec 18 23:58:42 2021
#3Requested Time Out

Message to Sent: Ping #4 Sat Dec 18 23:58:43 2021
#4Requested Time Out

Message to Sent: Ping #5 Sat Dec 18 23:58:44 2021
Received Message: PING #5 SAT DEC 18 23:58:44 2021
RTT: 0.000491857528687 seconds

Message to Sent: Ping #6 Sat Dec 18 23:58:44 2021
#6Requested Time Out

Message to Sent: Ping #7 Sat Dec 18 23:58:45 2021
#7Requested Time Out

Message to Sent: Ping #8 Sat Dec 18 23:58:46 2021
Received Message: PING #8 SAT DEC 18 23:58:46 2021
RTT: 0.000503063201904 seconds

Message to Sent: Ping #9 Sat Dec 18 23:58:46 2021
#9Requested Time Out

Message to Sent: Ping #10 Sat Dec 18 23:58:47 2021
Received Message: PING #10 SAT DEC 18 23:58:47 2021
RTT: 0.000496864318848 seconds

Close Socket
emir@emir:~/Desktop/MobileNetworkProje/171044019_EmirhanUZUN_TermProject$
```

Assignment 3: Mail Client

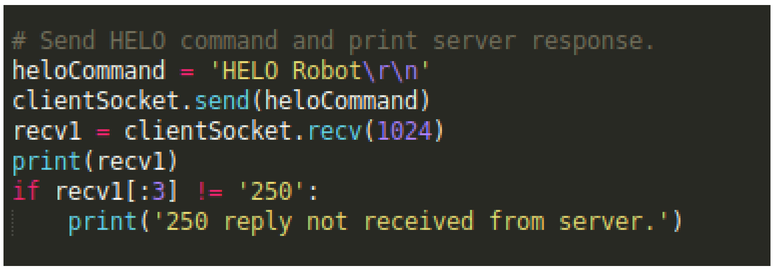
In this assignment, we are asked to develop a simple mail client that sends e-mail to any recipient. Our client will need to connect to a mail server, dialogue with the mail server using the SMTP protocol, and send an email message to the mail server. We have skeleton code for the client. I completed the code. I will explain this code in below :

Firstly, I choose gmail server which is smpt.gmail.com. Gmail port is 587.
After that, I create a socket called clientSocket and establis a TCP connection with mail server which is gmail server.



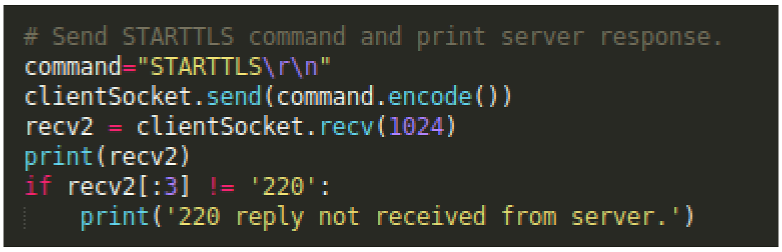
```
# Choose a mail server (e.g. Google mail server) and call it mailserver
mailserver = ('smtp.gmail.com', 587)

# Create socket called clientSocket and establish a TCP connection with mailserver
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect(mailserver)
```

Send HELO command and print server response. In this part, this helo message is sent to the mail server with the send() function. Then print server response message. If no response is received, error message is printed.



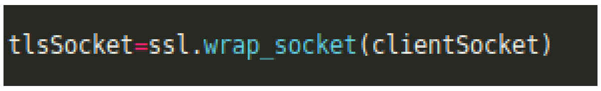
```
# Send HELO command and print server response.
heloCommand = 'HELO Robot\r\n'
clientSocket.send(heloCommand)
recv1 = clientSocket.recv(1024)
print(recv1)
if recv1[:3] != '250':
    print('250 reply not received from server.')
```

In STARTTLS part, it sends an TLS command. It is important to connect gmail server because of securtiy. Also this message is encrypted with encode() function. Then it is sent to the mail server with the send() function. Then print server response again. If no response is received, the error message is printed.



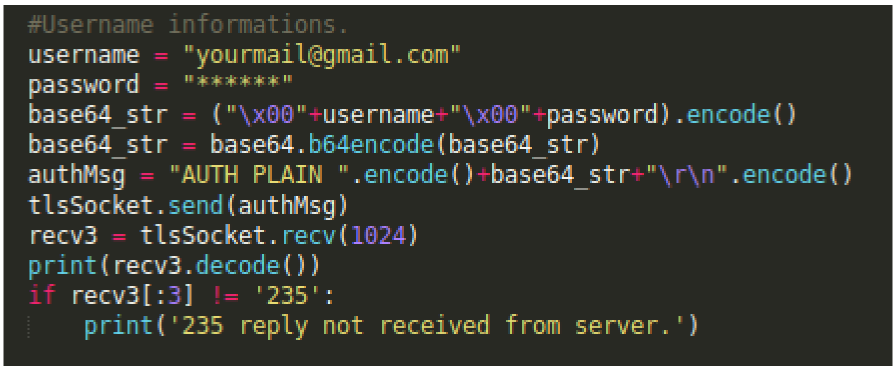
```
# Send STARTTLS command and print server response.
command="STARTTLS\r\n"
clientSocket.send(command.encode())
recv2 = clientSocket.recv(1024)
print(recv2)
if recv2[:3] != '220':
    print('220 reply not received from server.')
```

The wrap_socket part makes an ssl socket that wrapped client socket. Also it is important for security.



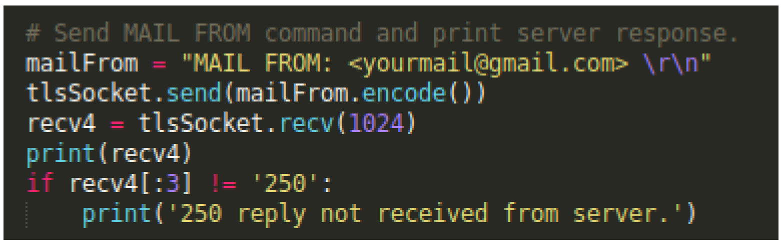
```
tlsSocket=ssl.wrap_socket(clientSocket)
```

For run this program you should enter your mail addresses and necessary information (such as your mail accounts and password). The email address and password is encrypted with the encode() function. Then it is encrypted with b64encode() function. After that, it is sent to the mail server with the send() function. To terminal, it prints the server response message. If no response is received, the error message is printed.



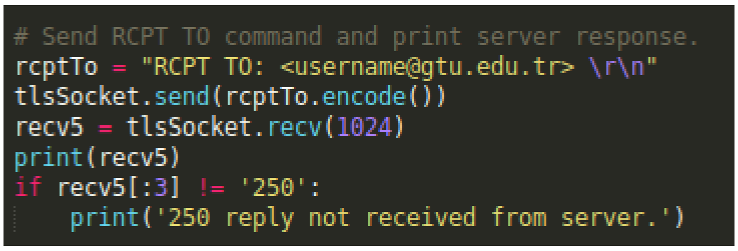
```python
#Username informations.
username = "yourmail@gmail.com"
password = "******"
base64_str = ("\x00"+username+"\x00"+password).encode()
base64_str = base64.b64encode(base64_str)
authMsg = "AUTH PLAIN ".encode()+base64_str+"\r\n".encode()
tlsSocket.send(authMsg)
recv3 = tlsSocket.recv(1024)
print(recv3.decode())
if recv3[:3] != '235':
    print('235 reply not received from server.')
```

In send mail from command part, it creates a command. This command carries the sender information. Also again I encrypted the message with encode() function. Then it is sent to the mail server with the send() function. If no response message is received, the error message is printed.



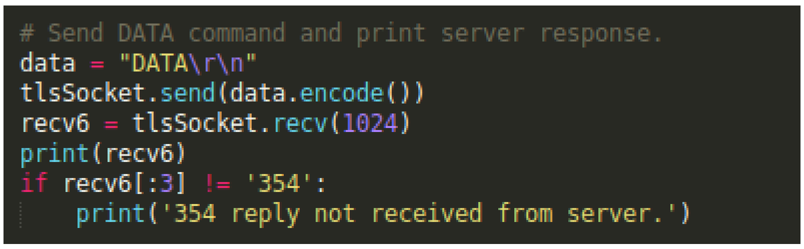
```python
# Send MAIL FROM command and print server response.
mailFrom = "MAIL FROM: <yourmail@gmail.com> \r\n"
tlsSocket.send(mailFrom.encode())
recv4 = tlsSocket.recv(1024)
print(recv4)
if recv4[:3] != '250':
    print('250 reply not received from server.')
```

In receipt to part, it creates a command. This command carries the receiver information. Also again I encrypted the message with encode() function. Then it is sent to the mail server with the send() function. If no response message is received, the error message is printed.



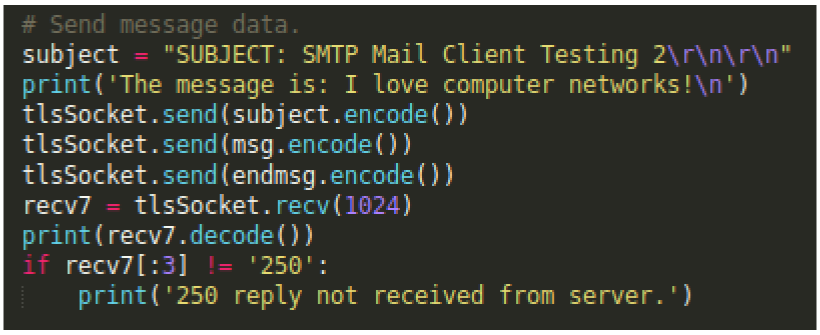
```python
# Send RCPT TO command and print server response.
rcptTo = "RCPT TO: <username@gtu.edu.tr> \r\n"
tlsSocket.send(rcptTo.encode())
recv5 = tlsSocket.recv(1024)
print(recv5)
if recv5[:3] != '250':
    print('250 reply not received from server.')
```

In data part, it creates a data command. This message is encrypted with the encode() function. Then it is sent to the mail server with send() function. If no response message is received, the error message is printed.



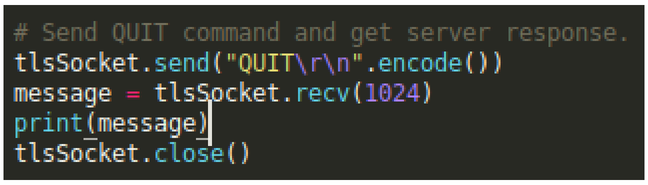
```python
# Send DATA command and print server response.
data = "DATA\r\n"
tlsSocket.send(data.encode())
recv6 = tlsSocket.recv(1024)
print(recv6)
if recv6[:3] != '354':
    print('354 reply not received from server.')
```

The send message part contains the information of the mail. This message encrypted with the encode() function. Then it is sent to the mail server with send() function. If no response message is received, the error message is printed.



```
# Send message data.
subject = "SUBJECT: SMTP Mail Client Testing 2\r\n\r\n"
print('The message is: I love computer networks!\n')
tlsSocket.send(subject.encode())
tlsSocket.send(msg.encode())
tlsSocket.send(endmsg.encode())
recv7 = tlsSocket.recv(1024)
print(recv7.decode())
if recv7[:3] != '250':
    print('250 reply not received from server.')
```

In the last part, it send a quit command. This command is encrypted with the encode() function. Then it is sent to the mail server with send() function. At the end, connection will be closed with close() function.



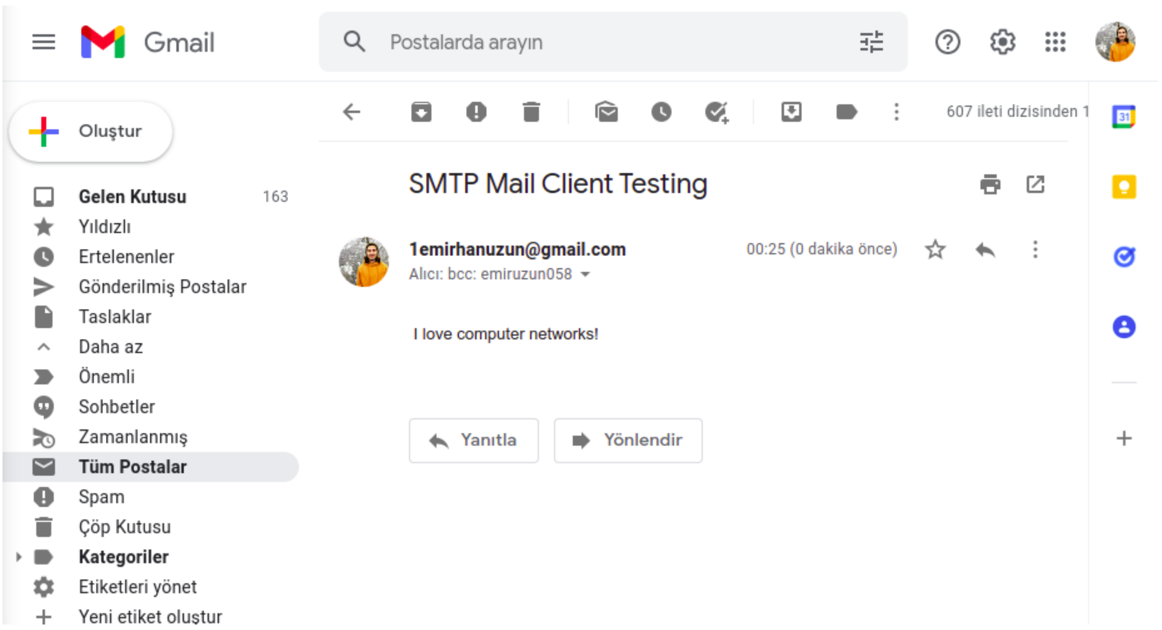
```
# Send QUIT command and get server response.
tlsSocket.send("QUIT\r\n".encode())
message = tlsSocket.recv(1024)
print(message)
tlsSocket.close()
```

RESULTS

-First Result

Sender : 1emirhanuzun@gmail.com
Receiver : emiruzun058@gmail.com

Sender part : 1emirhanuzun@gmail.com

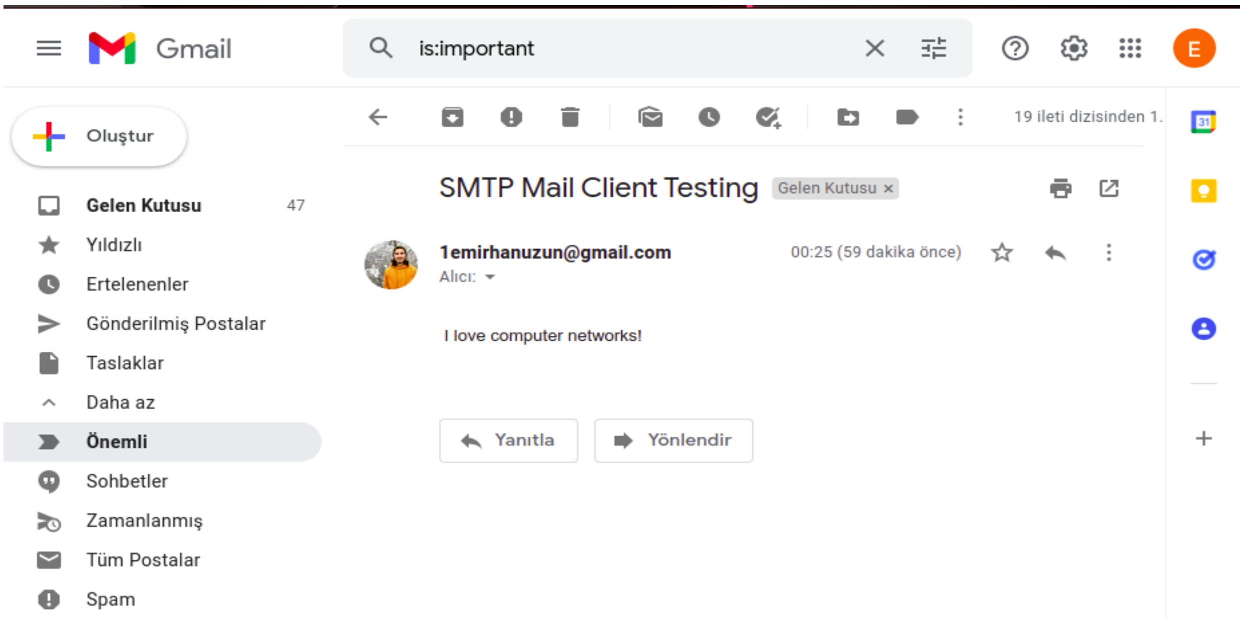


SMTP Mail Client Testing

1emirhanuzun@gmail.com 00:25 (0 dakika önce)
Alıcı: bcc: emiruzun058

I love computer networks!

Receiver part : emiruzun058@gmail.com

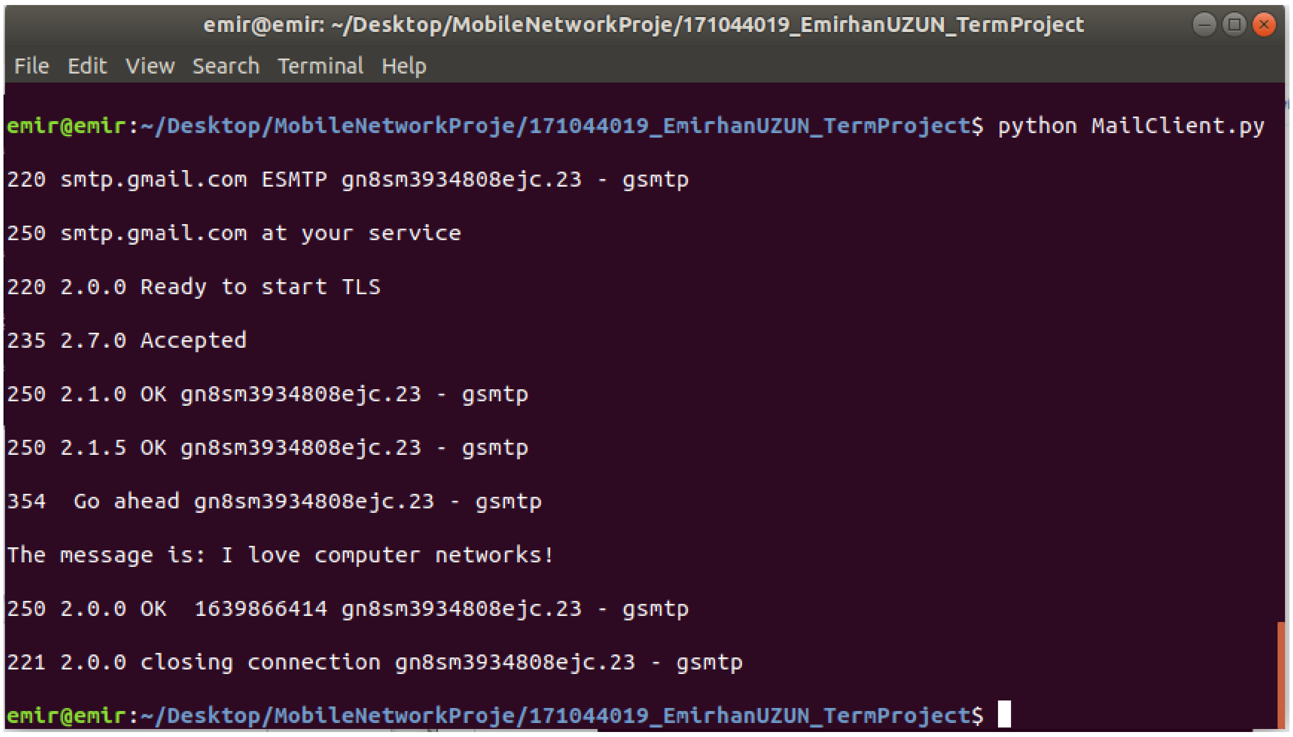


SMTP Mail Client Testing

1emirhanuzun@gmail.com 00:25 (59 dakika önce)
Alıcı:

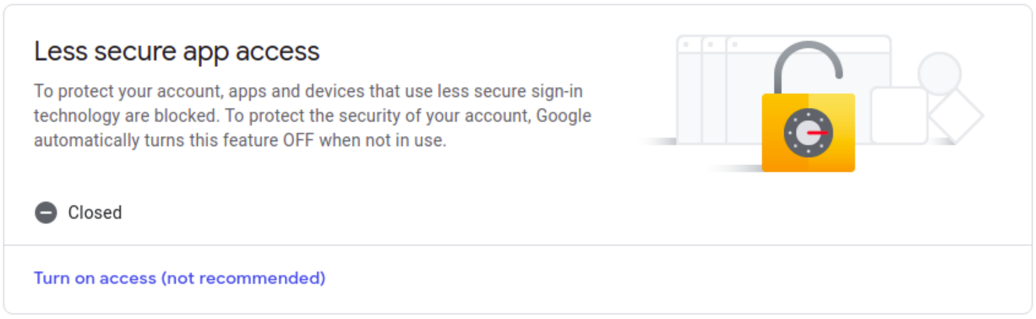
I love computer networks!

Terminal :



```
emir@emir: ~/Desktop/MobileNetworkProje/171044019_EmirhanUZUN_TermProject
File  Edit  View  Search  Terminal  Help
emir@emir:~/Desktop/MobileNetworkProje/171044019_EmirhanUZUN_TermProject$ python MailClient.py
220 smtp.gmail.com ESMTP gn8sm3934808ejc.23 - gsmtp
250 smtp.gmail.com at your service
220 2.0.0 Ready to start TLS
235 2.7.0 Accepted
250 2.1.0 OK gn8sm3934808ejc.23 - gsmtp
250 2.1.5 OK gn8sm3934808ejc.23 - gsmtp
354  Go ahead gn8sm3934808ejc.23 - gsmtp
The message is: I love computer networks!
250 2.0.0 OK  1639866414 gn8sm3934808ejc.23 - gsmtp
221 2.0.0 closing connection gn8sm3934808ejc.23 - gsmtp
emir@emir:~/Desktop/MobileNetworkProje/171044019_EmirhanUZUN_TermProject$
```

First time, I got an error message. Although my email and password are correct, it said that the connection could not be established. At the same time, an email was reached to me and it said that an entry was blocked from a device. I researched this situation and found that it is necessary to allow for IMAP and POP. However, less reliable devices should also be allowed. It works fine after these settings.

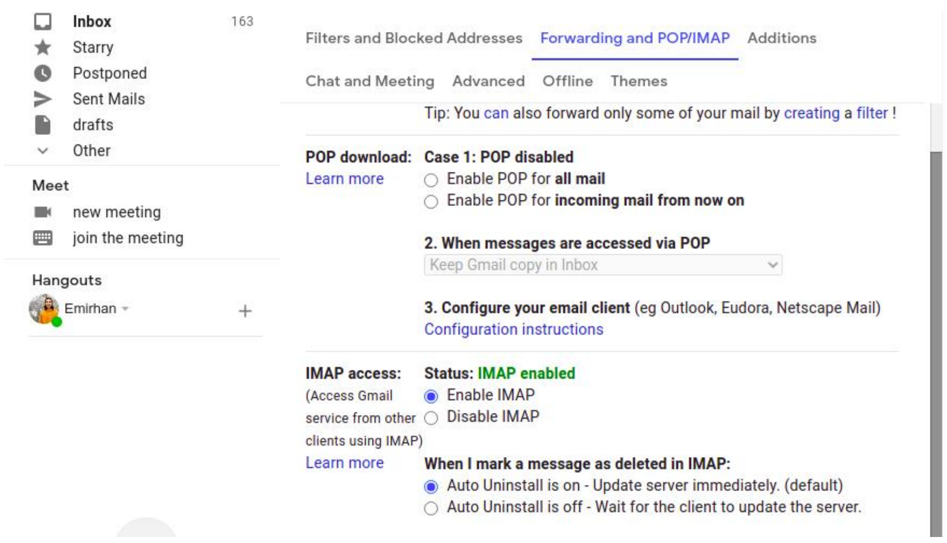


Less secure app access

To protect your account, apps and devices that use less secure sign-in technology are blocked. To protect the security of your account, Google automatically turns this feature OFF when not in use.

Closed

Turn on access (not recommended)



Tip: You can also forward only some of your mail by creating a filter !

POP download: Learn more
Case 1: POP disabled
○ Enable POP for **all mail**
○ Enable POP for **incoming mail from now on**

2. When messages are accessed via POP
Keep Gmail copy in Inbox

3. Configure your email client (eg Outlook, Eudora, Netscape Mail)
Configuration instructions

IMAP access: (Access Gmail service from other clients using IMAP) Learn more
Status: IMAP enabled
[✓ Enable IMAP]
○ Disable IMAP

When I mark a message as deleted in IMAP:
[✓ Auto Uninstall is on - Update server immediately. (default)]
○ Auto Uninstall is off - Wait for the client to update the server.