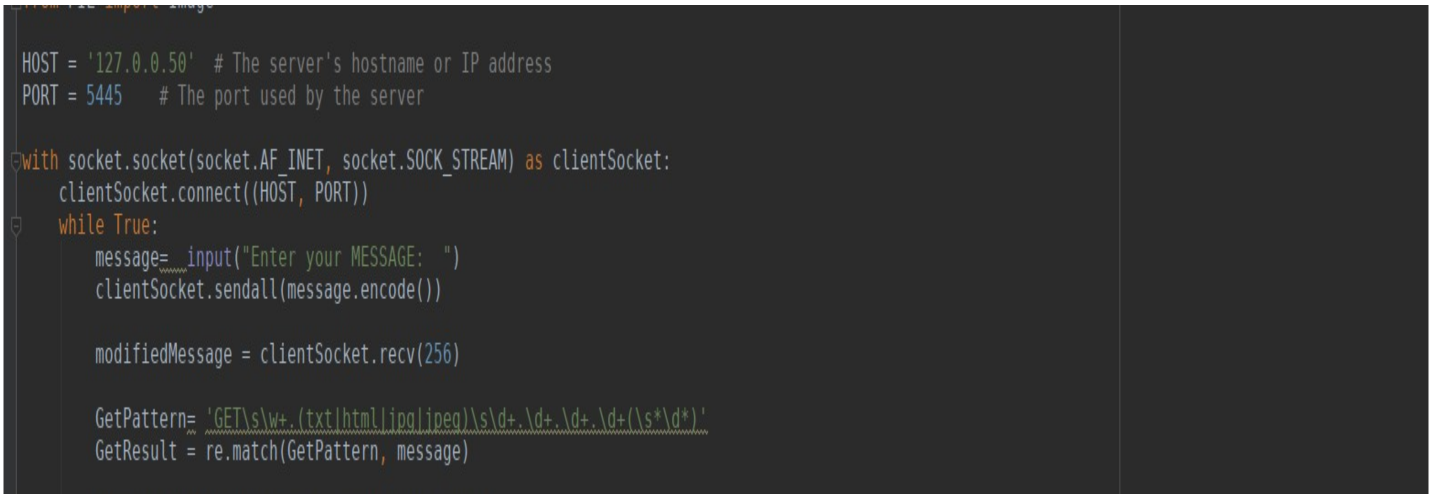
Computer Networks
Project 1
Socket Programming

<u>Reported by:</u>

Omar Wagdy Salah	4095
Amr Ahmed Saleh	4023
Ziad Ashraf Shreet	3947

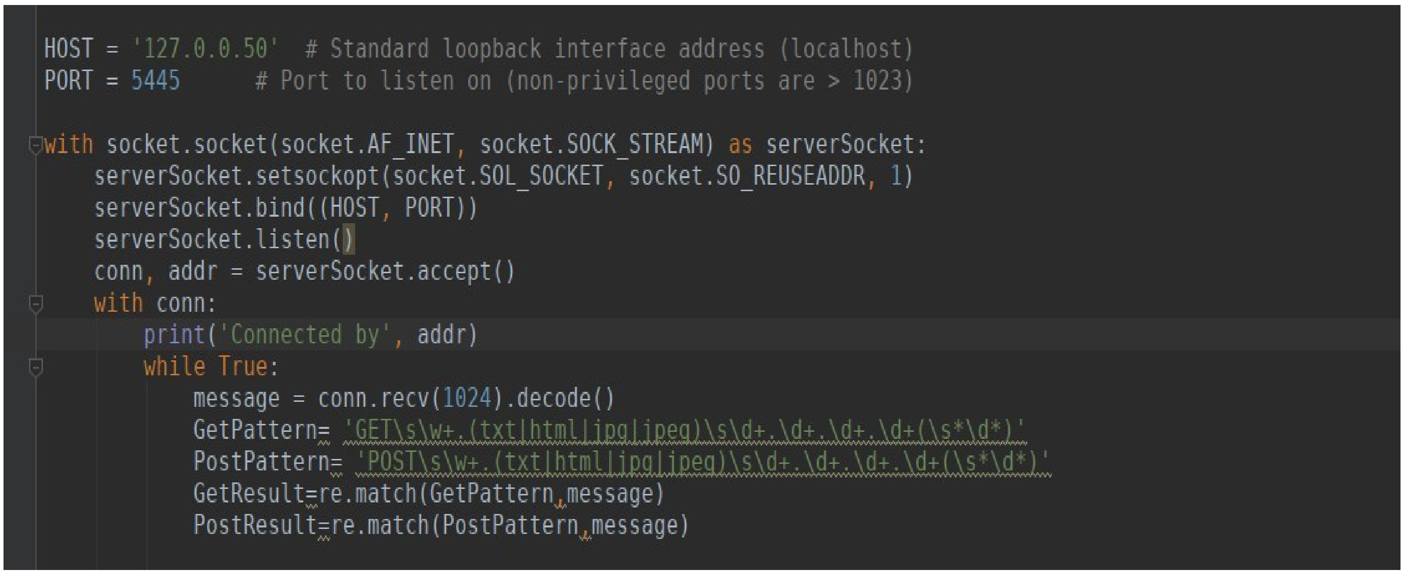
We use sockets to implement a simple web client that communicates with a web server.

Client Socket



```
HOST = '127.0.0.50'  # The server's hostname or IP address
PORT = 5445    # The port used by the server

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as clientSocket:
    clientSocket.connect((HOST, PORT))
    while True:
        message= input("Enter your MESSAGE:  ")
        clientSocket.sendall(message.encode())

        modifiedMessage = clientSocket.recv(256)

        GetPattern= 'GET\s\w+.(txt|html|jpg|jpeg)\s\d+.\d+.\d+.\d+(\s*\d*)'
        GetResult = re.match(GetPattern, message)
```

Server Socket



```
HOST = '127.0.0.50'  # Standard loopback interface address (localhost)
PORT = 5445        # Port to listen on (non-privileged ports are > 1023)

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as serverSocket:
    serverSocket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    serverSocket.bind((HOST, PORT))
    serverSocket.listen()
    conn, addr = serverSocket.accept()
    with conn:
        print('Connected by', addr)
        while True:
            message = conn.recv(1024).decode()
            GetPattern= 'GET\s\w+.(txt|html|jpg|jpeg)\s\d+.\d+.\d+.\d+(\s*\d*)'
            PostPattern= 'POST\s\w+.(txt|html|jpg|jpeg)\s\d+.\d+.\d+.\d+(\s*\d*)'
            GetResult=re.match(GetPattern,message)
            PostResult=re.match(PostPattern,message)
```

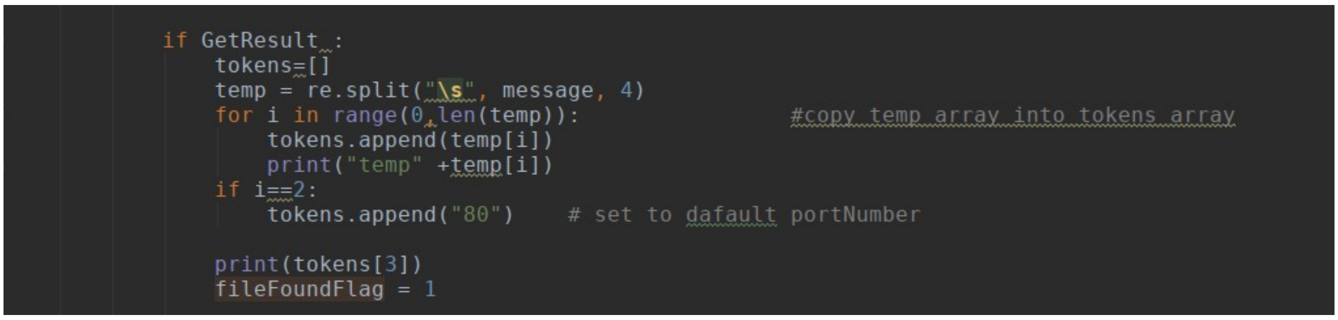
We made web server that accept incoming GET and POST requests. We then look if it is a GET request and pickup the name of the requested file. In case of a POST request, the server sends OK message to the client and wait for the uploaded file from the client.

The server side:

- Listen for connection.
- Accept new connection from incoming clients multiprocesses.
- Parse HTTP request and determine whether the request is POST or GET.
- Determine if target file exists then return OK or do not exist and return ERROR NOT FOUND in case of GET request
- Transmit the content of file, in case of GET; or Receive in case of POST.
- Close the connection after each transaction.

In case of a GET request:

We tokenise the message request and get the file name, whether it is a **.txt**, **.jpg**, **.jpeg**, or **.html** extension, and check the requested file exists or not. If the message doesn't include a port number, it is set to be 80 by default.



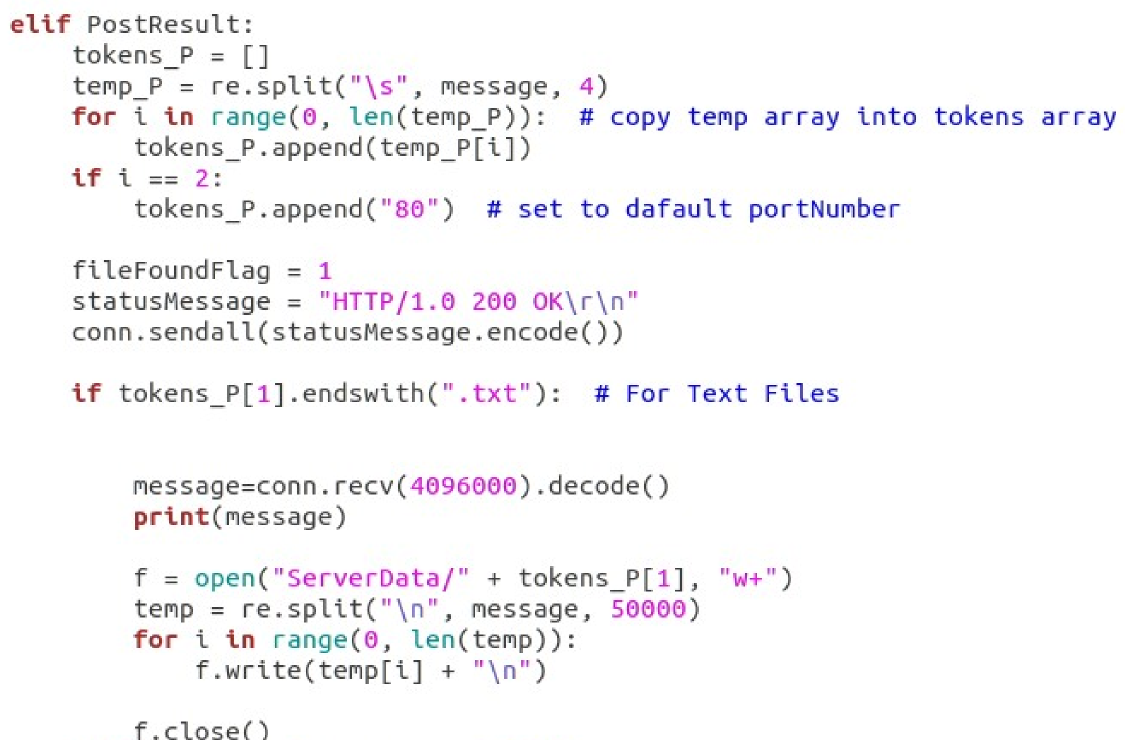
```
if GetResult :
    tokens=[]
    temp = re.split("\s", message, 4)
    for i in range(0,len(temp)):                #copy temp array into tokens array
        tokens.append(temp[i])
        print("temp" +temp[i])
    if i==2:
        tokens.append("80")    # set to dafault portNumber

    print(tokens[3])
    fileFoundFlag = 1
```

We deal with each file type separately:

- encode message
- send file content

In case of a POST request:

Server receives message request from client side, decode it, and get file content in the ServerDate from ClientData.



```
elif PostResult:
    tokens_P = []
    temp_P = re.split("\s", message, 4)
    for i in range(0, len(temp_P)):  # copy temp array into tokens array
        tokens_P.append(temp_P[i])
    if i == 2:
        tokens_P.append("80")  # set to dafault portNumber

    fileFoundFlag = 1
    statusMessage = "HTTP/1.0 200 OK\r\n"
    conn.sendall(statusMessage.encode())

    if tokens_P[1].endswith(".txt"):  # For Text Files


        message=conn.recv(4096000).decode()
        print(message)

        f = open("ServerData/" + tokens_P[1], "w+")
        temp = re.split("\n", message, 50000)
        for i in range(0, len(temp)):
            f.write(temp[i] + "\n")

        f.close()
```

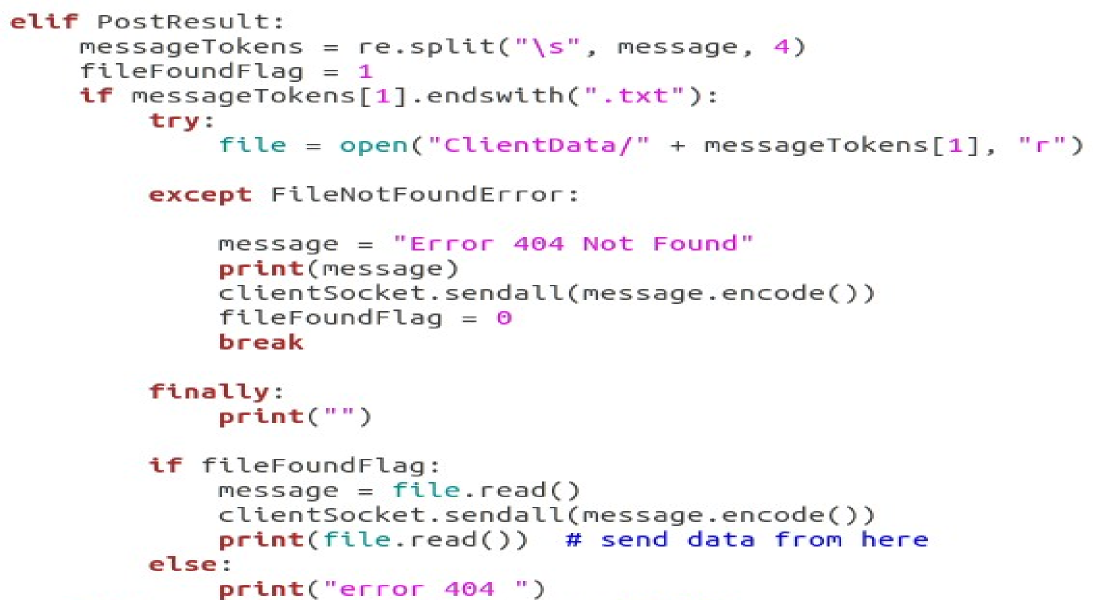
The client side:

- Create a TCP connection and wait for permission from the server.
- Send next requests to the server.
- Receives data from the server, in case of GET; or send date in case of a POST request.
- Close the connection.

We first check on the modified message whether is is 200 OK or ERROR 404 NOT FOUND.

In case of a POST request:

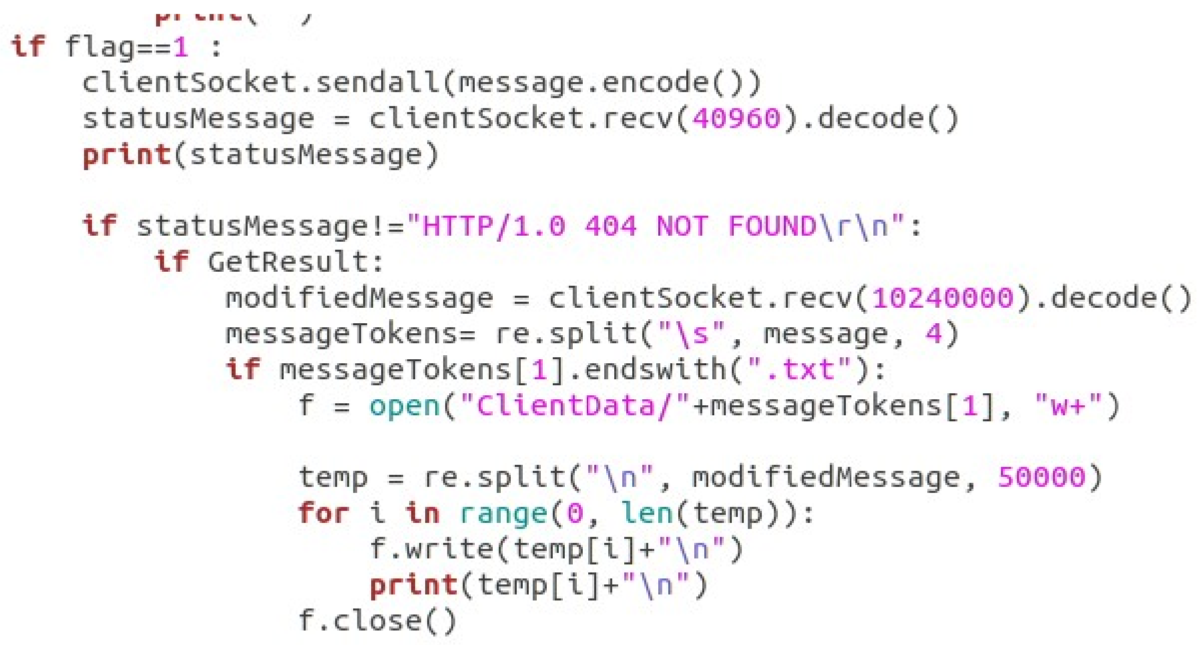
We tokenise the message, get file name, and search for the file in the ClientData if exists or not; and if exists, it is then encoded and sent to the server through the client socket.



```
elif PostResult:
    messageTokens = re.split("\s", message, 4)
    fileFoundFlag = 1
    if messageTokens[1].endswith(".txt"):
        try:
            file = open("ClientData/" + messageTokens[1], "r")

        except FileNotFoundError:

            message = "Error 404 Not Found"
            print(message)
            clientSocket.sendall(message.encode())
            fileFoundFlag = 0
            break

        finally:
            print("")

        if fileFoundFlag:
            message = file.read()
            clientSocket.sendall(message.encode())
            print(file.read())   # send data from here
        else:
            print("error 404 ")
```

In case of a GET request:

Client socket receive decoded modified message from server and get file content from ServerData side.



```
if flag==1 :
    clientSocket.sendall(message.encode())
    statusMessage = clientSocket.recv(40960).decode()
    print(statusMessage)

    if statusMessage!="HTTP/1.0 404 NOT FOUND\r\n":
        if GetResult:
            modifiedMessage = clientSocket.recv(10240000).decode()
            messageTokens= re.split("\s", message, 4)
            if messageTokens[1].endswith(".txt"):
                f = open("ClientData/"+messageTokens[1], "w+")

                temp = re.split("\n", modifiedMessage, 50000)
                for i in range(0, len(temp)):
                    f.write(temp[i]+"\n")
                    print(temp[i]+"\n")
                f.close()
```