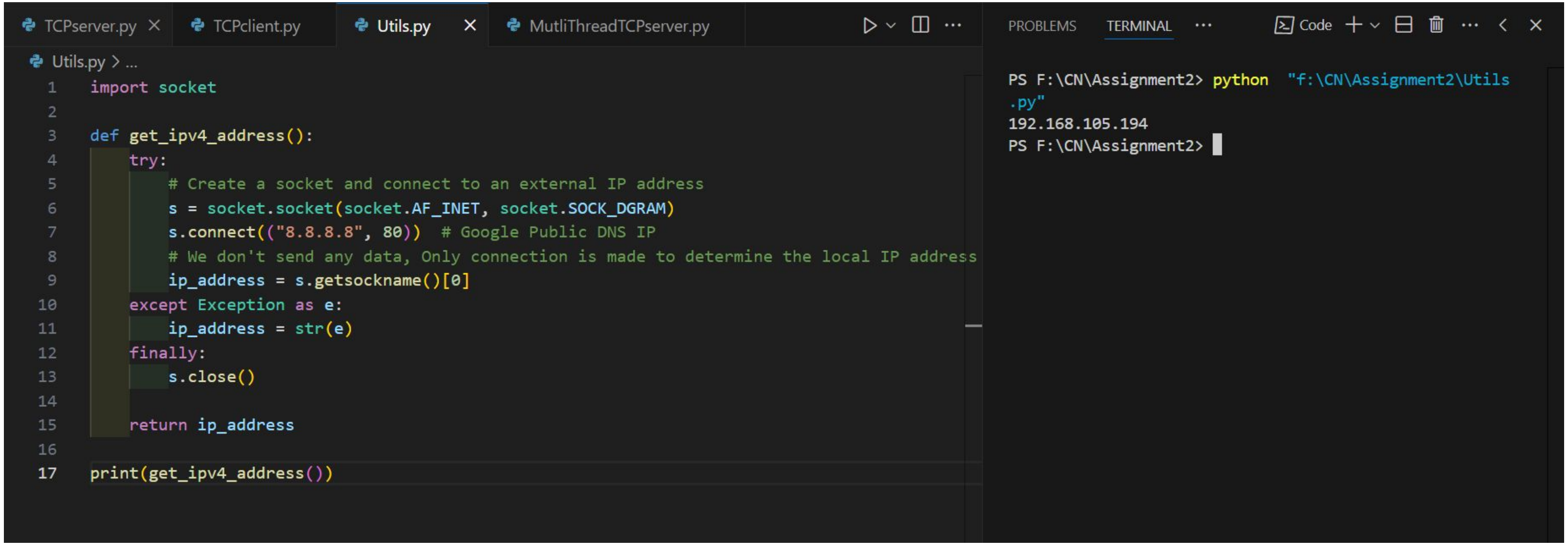
CSE 232 Section B, Computer Networks; Programming Assignment 2: TCP based Web application Lab

Harsh Rajput (2022201)
Aryan Singla (2022112)

Utils.get_ipv4_address

This function determines and returns the local machine's IPv4 address by creating a UDP socket connection to an external IP address (Google DNS) without sending data.

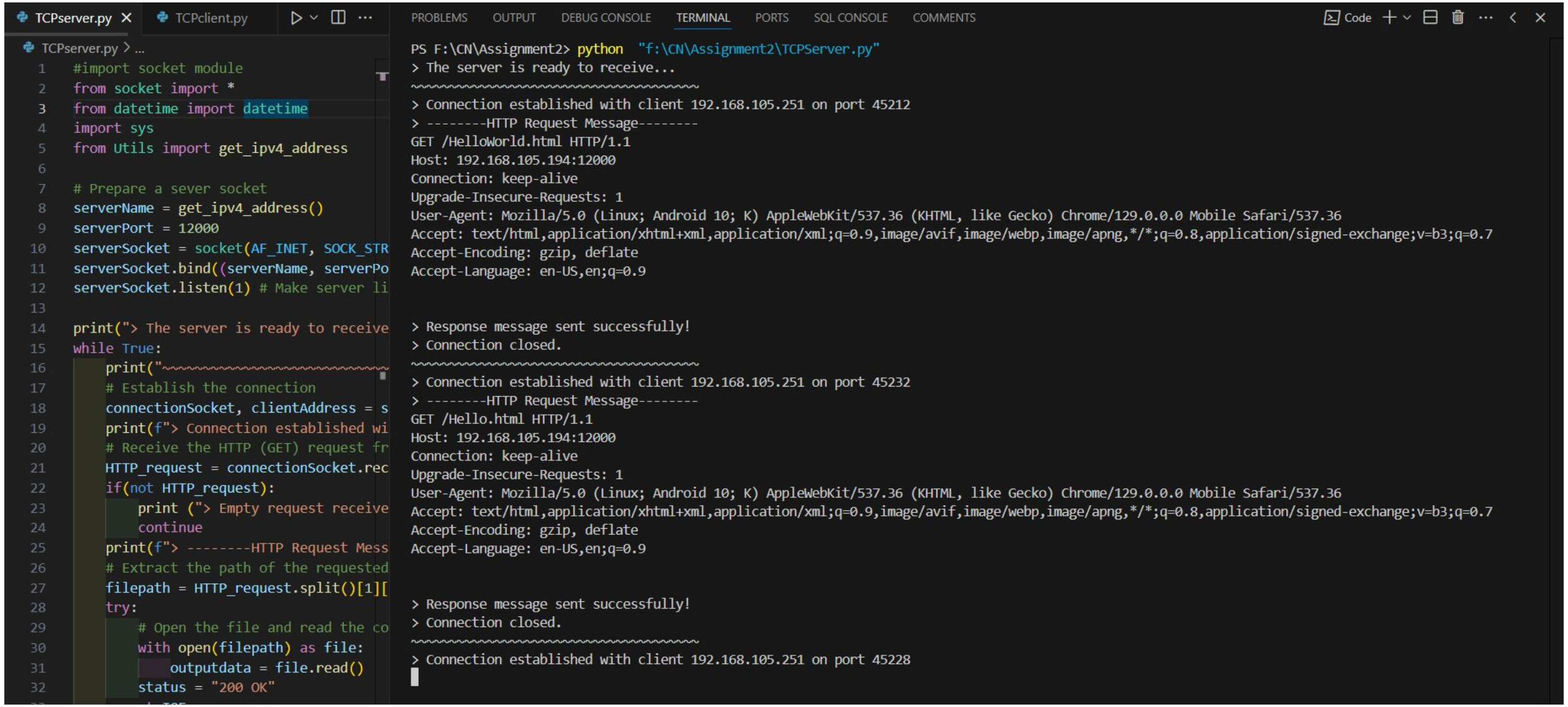


```python
import socket

def get_ipv4_address():
    try:
        # Create a socket and connect to an external IP address
        s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        s.connect(("8.8.8.8", 80))  # Google Public DNS IP
        # We don't send any data, Only connection is made to determine the local IP address
        ip_address = s.getsockname()[0]
    except Exception as e:
        ip_address = str(e)
    finally:
        s.close()

    return ip_address

print(get_ipv4_address())
```

```
PS F:\CN\Assignment2> python  "f:\CN\Assignment2\Utils
.py"
192.168.105.194
PS F:\CN\Assignment2>
```

TCPserver.py

The `TCPserver.py` script sets up a TCP server that listens for client connections, processes a single HTTP GET request at a time, and serves the requested file or returns a 404 error if the file is not found.



```python
#import socket module
from socket import *
from datetime import datetime
import sys
from Utils import get_ipv4_address

# Prepare a sever socket
serverName = get_ipv4_address()
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STR
serverSocket.bind((serverName, serverPo
serverSocket.listen(1) # Make server li

print("> The server is ready to receive
while True:
    print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    # Establish the connection
    connectionSocket, clientAddress = s
    print(f"> Connection established wi
    # Receive the HTTP (GET) request fr
    HTTP_request = connectionSocket.rec
    if(not HTTP_request):
        print ("> Empty request receive
        continue
    print(f"> --------HTTP Request Mess
    # Extract the path of the requested
    filepath = HTTP_request.split()[1][
    try:
        # Open the file and read the co
        with open(filepath) as file:
            outputdata = file.read()
        status = "200 OK"
```

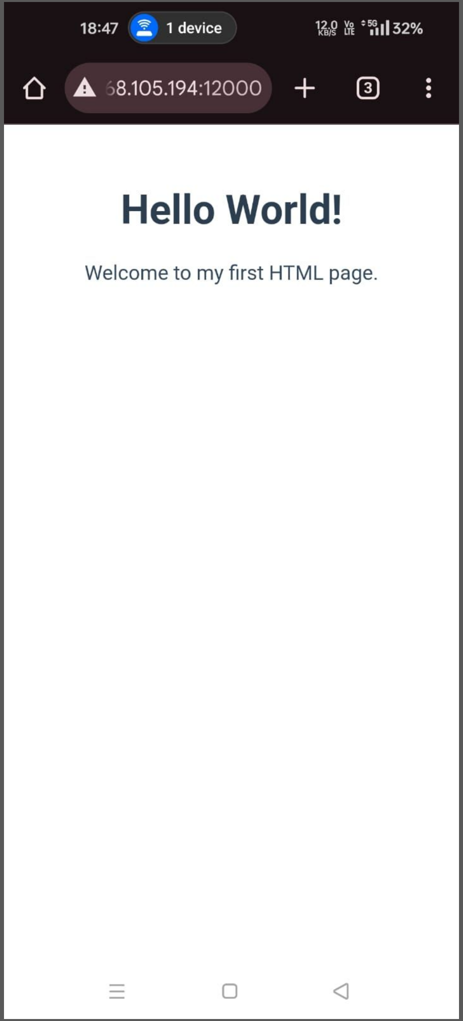
```
PS F:\CN\Assignment2> python  "f:\CN\Assignment2\TCPServer.py"
> The server is ready to receive...
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
> Connection established with client 192.168.105.251 on port 45212
> --------HTTP Request Message--------
GET /HelloWorld.html HTTP/1.1
Host: 192.168.105.194:12000
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/129.0.0.0 Mobile 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
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9


> Response message sent successfully!
> Connection closed.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
> Connection established with client 192.168.105.251 on port 45232
> --------HTTP Request Message--------
GET /Hello.html HTTP/1.1
Host: 192.168.105.194:12000
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/129.0.0.0 Mobile 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
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9


> Response message sent successfully!
> Connection closed.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
> Connection established with client 192.168.105.251 on port 45228
```

Request 1

```
http://192.168.105.194:12000/HelloWorld.html
```

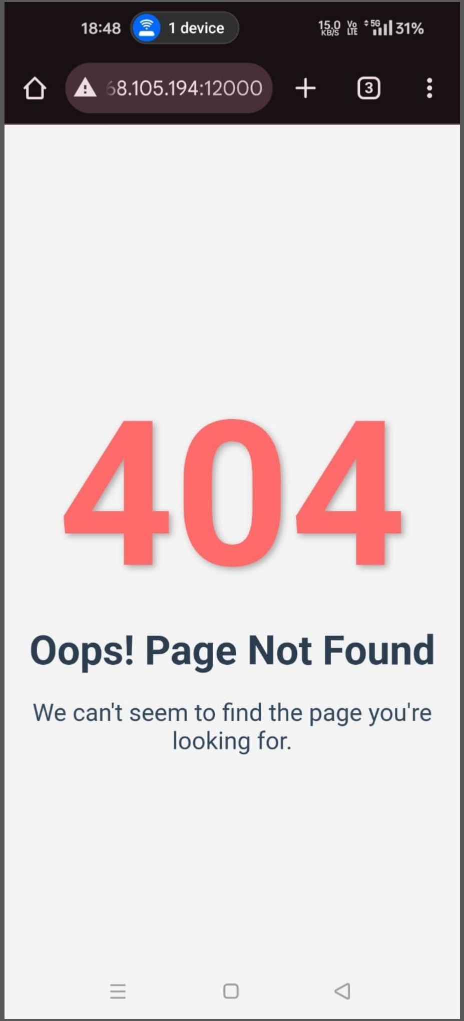
Request 2

```
http://192.168.105.194:12000/Hello.html
```



Hello World!

Welcome to my first HTML page.



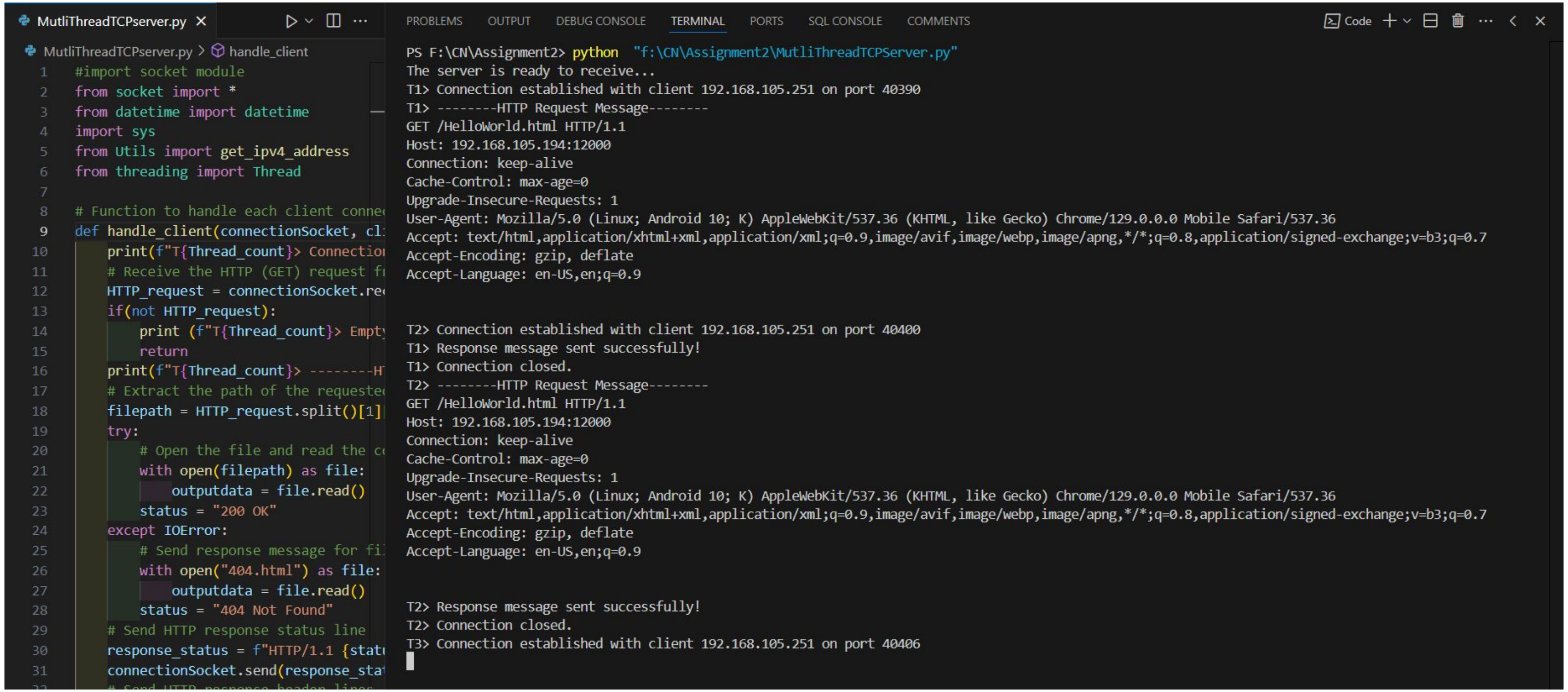
404

Oops! Page Not Found

We can't seem to find the page you're looking for.

MutliThreadTCPserver.py

The `multithreadedTCPserver.py` script sets up a TCP server that uses multithreading to handle multiple client connections concurrently, processing HTTP GET requests and serving requested files or returning a 404 error if the file is not found.

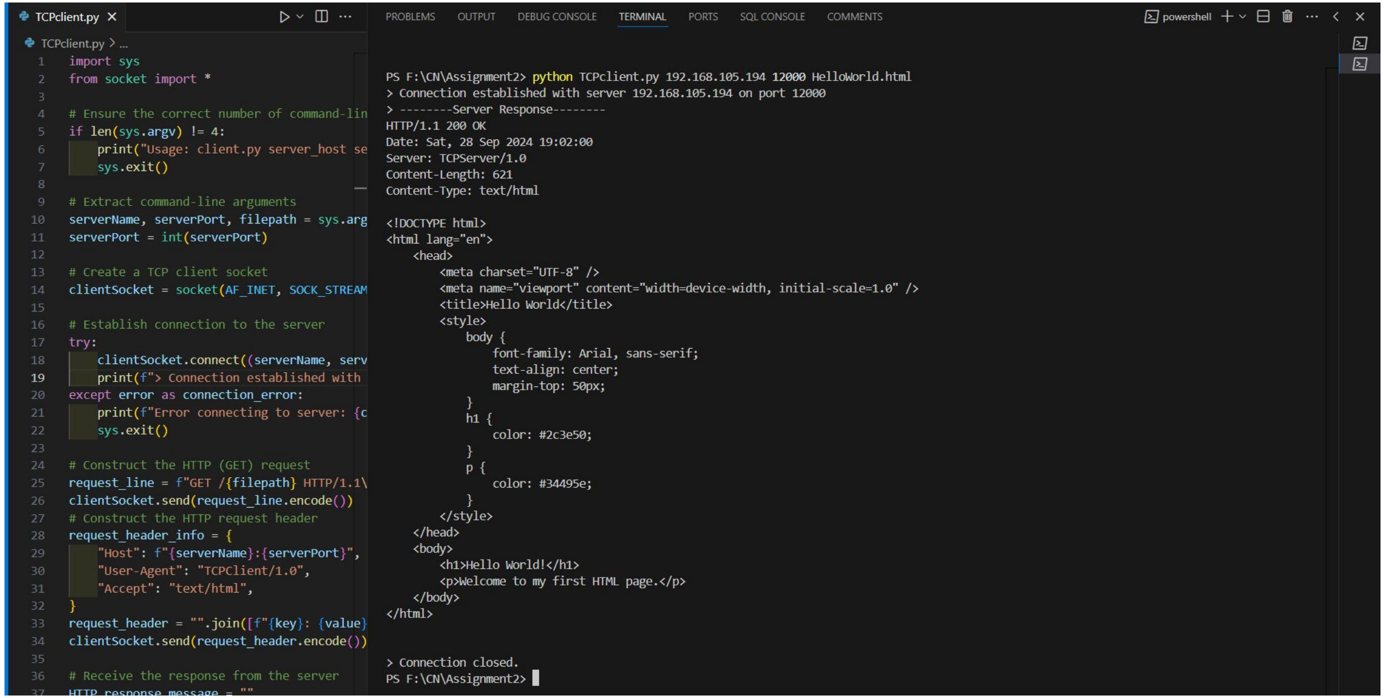


```python
#import socket module
from socket import *
from datetime import datetime
import sys
from Utils import get_ipv4_address
from threading import Thread

# Function to handle each client conne
def handle_client(connectionSocket, cl
    print(f"T{Thread_count}> Connectio
    # Receive the HTTP (GET) request fr
    HTTP_request = connectionSocket.re
    if(not HTTP_request):
        print (f"T{Thread_count}> Empty
        return
    print(f"T{Thread_count}> --------H
    # Extract the path of the requeste
    filepath = HTTP_request.split()[1]
    try:
        # Open the file and read the co
        with open(filepath) as file:
            outputdata = file.read()
        status = "200 OK"
    except IOError:
        # Send response message for fil
        with open("404.html") as file:
            outputdata = file.read()
        status = "404 Not Found"
    # Send HTTP response status line
    response_status = f"HTTP/1.1 {statu
    connectionSocket.send(response_stat
```

```
PS F:\CN\Assignment2> python "f:\CN\Assignment2\MutliThreadTCPServer.py"
The server is ready to receive...
T1> Connection established with client 192.168.105.251 on port 40390
T1> --------HTTP Request Message--------
GET /HelloWorld.html HTTP/1.1
Host: 192.168.105.194:12000
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/129.0.0.0 Mobile 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
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9


T2> Connection established with client 192.168.105.251 on port 40400
T1> Response message sent successfully!
T1> Connection closed.
T2> --------HTTP Request Message--------
GET /HelloWorld.html HTTP/1.1
Host: 192.168.105.194:12000
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/129.0.0.0 Mobile 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
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9


T2> Response message sent successfully!
T2> Connection closed.
T3> Connection established with client 192.168.105.251 on port 40406
```

TCPclient.py

The `TCPclient.py` script connects to a TCP server, sends an HTTP GET request for a specified file, receives the server's response, and displays it.



```python
import sys
from socket import *

# Ensure the correct number of command-lin
if len(sys.argv) != 4:
    print("Usage: client.py server_host se
    sys.exit()

# Extract command-line arguments
serverName, serverPort, filepath = sys.arg
serverPort = int(serverPort)

# Create a TCP client socket
clientSocket = socket(AF_INET, SOCK_STREAM

# Establish connection to the server
try:
    clientSocket.connect((serverName, serv
    print(f"> Connection established with
except error as connection_error:
    print(f"Error connecting to server: {c
    sys.exit()

# Construct the HTTP (GET) request
request_line = f"GET /{filepath} HTTP/1.1\
clientSocket.send(request_line.encode())
# Construct the HTTP request header
request_header_info = {
    "Host": f"{serverName}:{serverPort}",
    "User-Agent": "TCPClient/1.0",
    "Accept": "text/html",
}
request_header = "".join([f"{key}: {value}
clientSocket.send(request_header.encode())

# Receive the response from the server
HTTP_response_message = ""
```

```
PS F:\CN\Assignment2> python TCPclient.py 192.168.105.194 12000 HelloWorld.html
> Connection established with server 192.168.105.194 on port 12000
> --------Server Response--------
HTTP/1.1 200 OK
Date: Sat, 28 Sep 2024 19:02:00
Server: TCPServer/1.0
Content-Length: 621
Content-Type: text/html

<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <title>Hello World</title>
        <style>
            body {
                font-family: Arial, sans-serif;
                text-align: center;
                margin-top: 50px;
            }
            h1 {
                color: #2c3e50;
            }
            p {
                color: #34495e;
            }
        </style>
    </head>
    <body>
        <h1>Hello World!</h1>
        <p>Welcome to my first HTML page.</p>
    </body>
</html>


> Connection closed.
PS F:\CN\Assignment2>
```