**CSC 335 Data Communications and Networking I**

**Lab 5 Traffic Analysis with Socket Programming and Wireshark**

Student’s Name: **Michael Burns**

Date of Experiment: **5/4/23**

**1. Goal**

This lab will analyze the traffic with customized sockets using wireshark.

Deliverable: Please answer all questions and submit this lab report to D2L.

Note: You don’t have to use the same python file and project names as I do.

**2. Introduction and Background**

One’s understanding of network protocols can often be greatly deepened by “seeing protocols in action” and by “playing around with protocols” – observing the sequence of messages exchanged between two protocol entities, investigating the details of protocol operation, and causing protocols to perform certain actions and then observing these actions and their consequences. In this lab, you’ll be running various network applications in different scenarios using a computer on your desk, at home, or in a lab. You’ll observe the network protocols in your computer “in action,” interacting and exchanging messages with protocol entities executing elsewhere in the Internet.

In this lab, we will

(1) write a socket with python

(2) create a web service

(3) analyze network traffic using wireshark

**3. Getting Started with Python and Wireshark**

Here is a tutorial for install and setup python for Mac and Windows. It also goes over the basic of python. The video was recorded in 2017, and there many new python release since then. Please install the latest version of Python. <https://www.youtube.com/watch?v=YYXdXT2l-Gg>

If you don’t have a preference for python editor, I recommend PyCharm, which can be downloaded from https://www.jetbrains.com/pycharm/. The community version is free for everyone to use.

The Wireshark network protocol analyzer (former Wireshark) is a tool for capturing, displaying, and analyzing the frames, packets, and messages that are exchanged in a network. The Wireshark package can be downloaded from http://www.wireshark.org/download.html. Download the latest version. Note that in some computing environments, such as MS Windows, it is necessary to install a separate file capture utility (WinPcap for MS Windows). This utility is included to the latest version of the Wireshark installation package.

**4. Socket programming**

A socket represents one end of the connection between two endpoints. In the client-server mode, there are two basic types of sockets:

* A listening socket: is used by a server process to wait for connection requests from remote clients.
* A connected socket: is used by a client process to send and receive messages.

Socket programming is a way to connecting two nodes on a network to communicate with each other.

**4.1 Connecting to well-known server using sockets**

Step 0: create a socket programming project in pycharm by clicking “File->New Project”, then give a project name, such as “SocketProgramming”. Then, create client1.py file in “SocketProgramming” project by right clicking the project name, then “New -> Python File”, then give a file name such as server.

The entire code includes three parts: creating a client socket, identify the server you want to connect to, connect the socket to the server. Please include all three parts in the same python file.

Step 1: Creating a client socket. The command “socket.SOCK\_STREAM” means the socket is a TCP socket, which provide reliable communication. What command should we use to create a UDP socket?

**socket = new Datagram()**

Text

Description automatically generated

Step 2: Identify a server. The server that I will connect in the example is [www.google.com](http://www.google.com). Since I don’t know the IP address of [www.google.com](http://www.google.com), I use the function “gethostbyname”.

Text

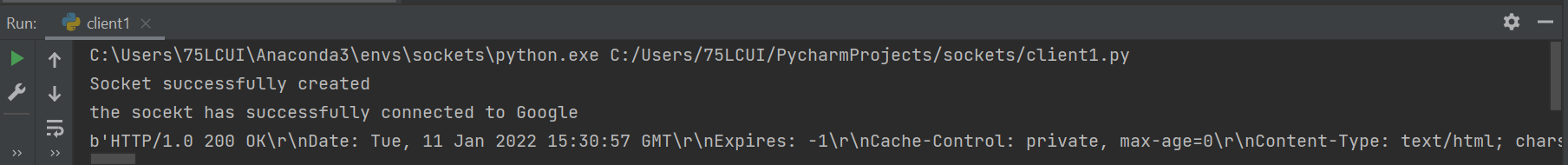
Description automatically generated

Step 3: Connect to the server, send a request to get the http file, and display the reply. The number 10000 indicate the longest message that this socket can receive in bytes.

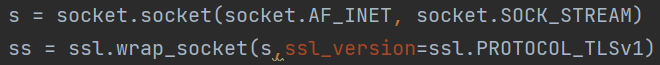
Text

Description automatically generated

When you run this program, it displays the following feedback. If you keep scroll to the right, you will see the http content “Google has many special features to help you find exactly what you’re looking for.”



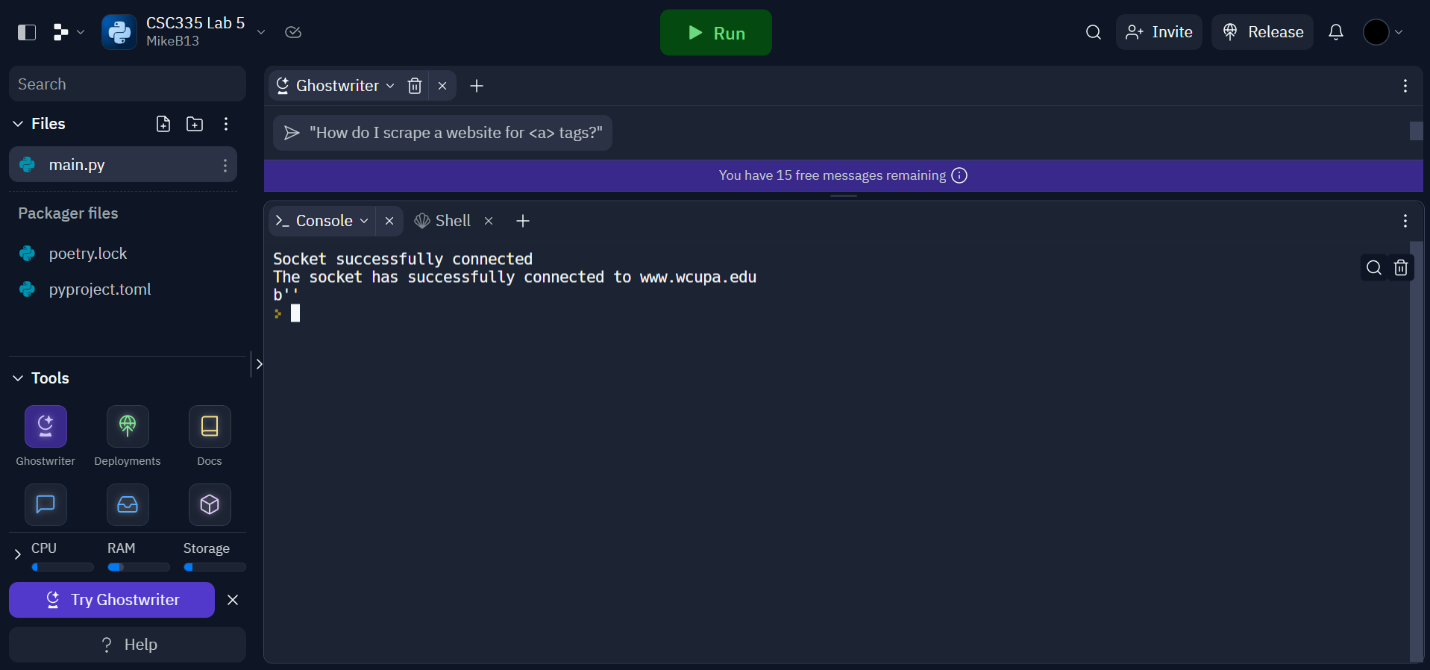
Now, it is your turn to create a socket and connect to a server. In the above example, I used port number 80, which is the default port number of http. As you may know, http is out since it is not secure enough. The current protocol that majority of the webpages use is https, which use port number 443. Https use ssl as the security measurement, so we need to create a socket that uses ssl. We could wrap the current socket s with the ssl to create the new one as shown below. Also, all data that will be send and receive from socket ss, not s.



When sending the request, we want to indicate the protocol and version as below.

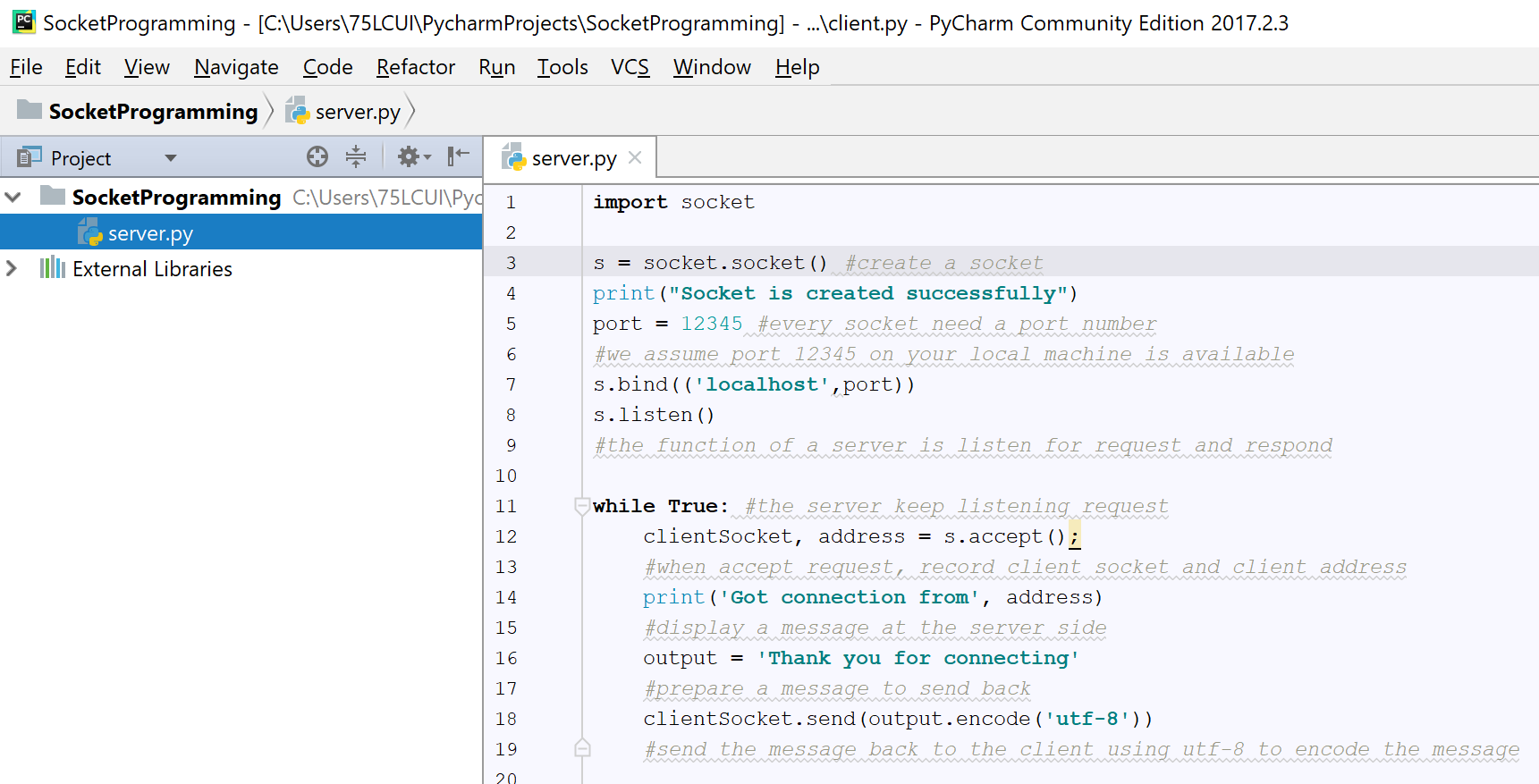


Please connect to [www.wcupa.edu](http://www.wcupa.edu) using port 443. Take a screenshot of the part of reply you got that clearly indicate it is a reply from wcu.

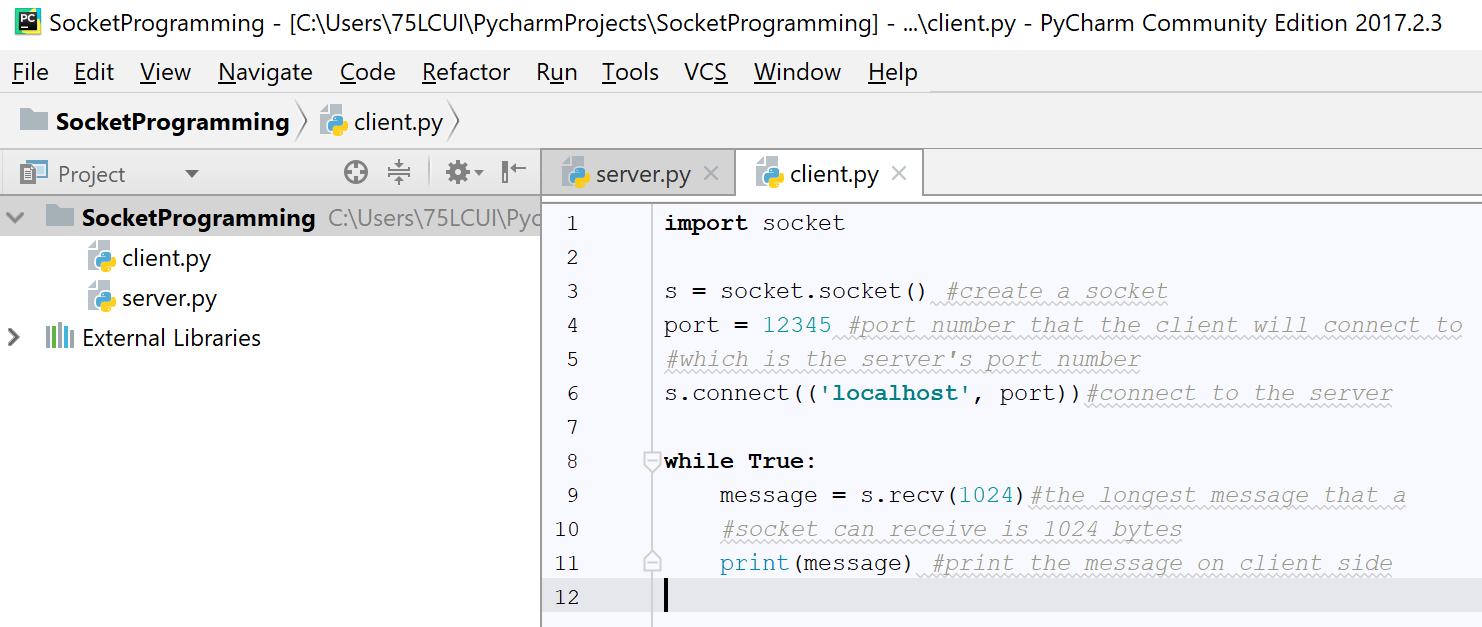


**4.2 Creating a customized server socket**

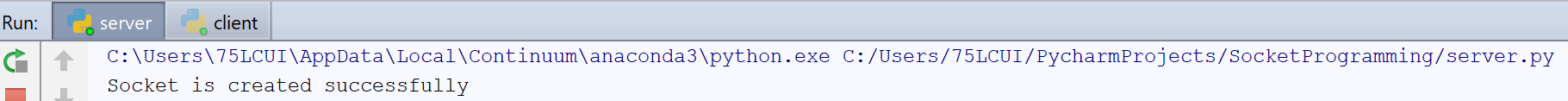
Step 4: create server.py file in “SocketProgramming” project by right clicking the project name, then “New -> Python File”, then give a file name such as server. Here is the program you need to write in server.py.



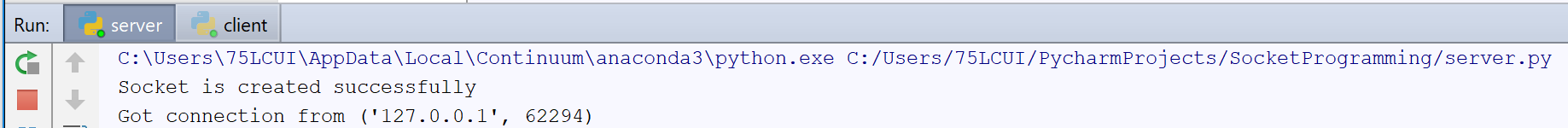
Step 5: create client.py file in “SocketProgramming” project by right clicking the project name, then “New -> Python File”, then give a file name such as client. Here is the program you need to write in client.py.



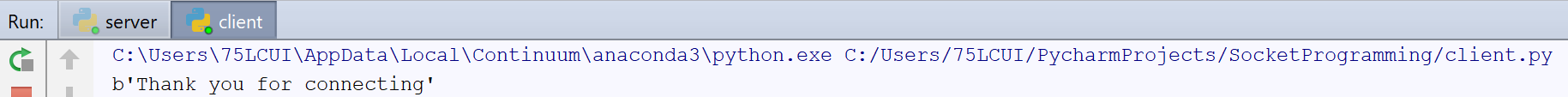
Step 6: run the client-server model. We first need to run the server, since we assume the server is “always on” and wait for request. If your server is setting up properly, you will see “Socket is created successfully as below. If it shows any error messages, please debug your program and try again.



Now, let’s run client.py. When the client.py is running, it tries to connect the server, so the server side will display the client’s information as expected. The message means the IP address of the client is 127.0.0.1, which is the IP address for localhost, and the client’s port number is 62294.



The client side displays the message it got from the server as below.



Please send a message, “What is the webpage for WCU CS department” from client to the server, and then the server replies “cs.wcupa.edu”. Please show the screenshot of the programs as well as client and server’s display.

Text

Description automatically generatedText

Description automatically generated

**4.3 Creating a web application**

Step 7: The Q&A in step 6 is not a real service, since the only reply from the server to the client is “cs.wcupa.edu” no matter what the client sends to the server. In this step, we will create a real web application, in which

(1) the client receives two number from the user using keyboard

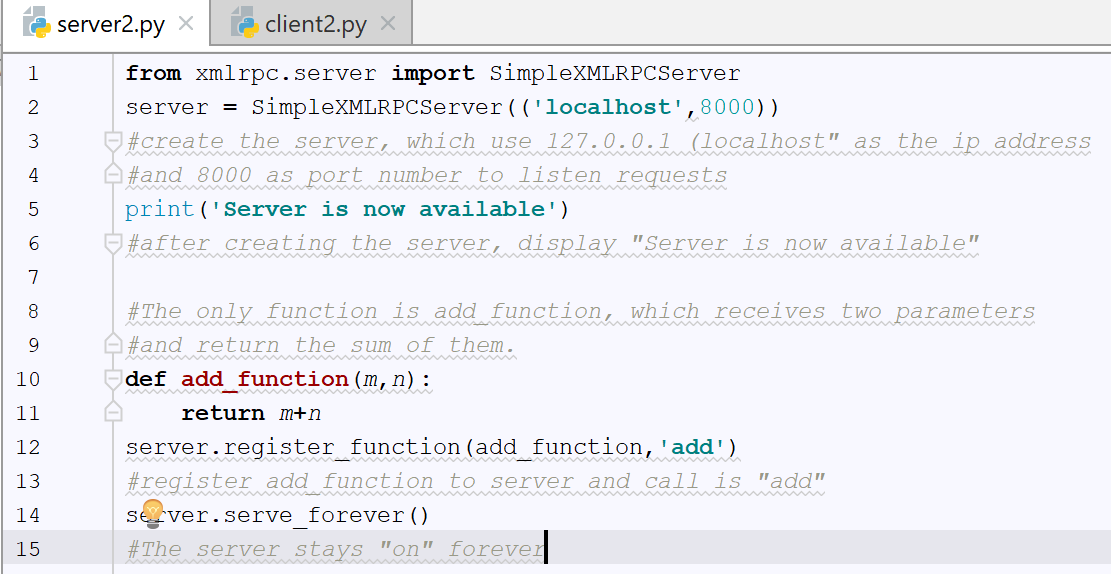
(2) the client sends the two number to the server

(3) the server calculates the sum of the two numbers and send the answer back to the client

In order to do so, we will create two more python programs, namely server2.py and client2.py.

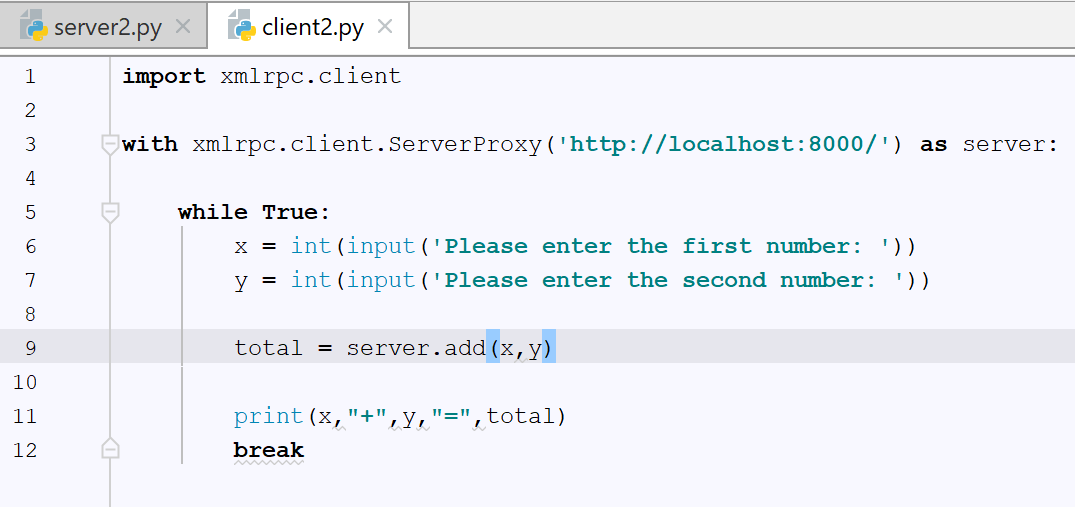
Step 8: server2.py

In this program, we will create a xmlrpc server. According to python doc (<https://docs.python.org/3/library/xmlrpc.server.html>), the “SimpleSMLRPCServer class is based on socketserver.TCPServer and provdes a means of creating simple, stand along XML-RPC servers. “



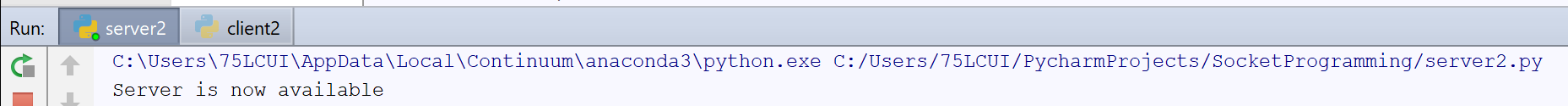
Step 9: client2.py

In this program, we create a xmlrpc client by import xmlrpc.client. Then, determine the server that the clients will talk to as “hptt://localhost:8000/”. Then, we run a loop to receive two numbers from the user. After that, the two numbers are send to server function “add” as “server.add(x,y)”. The answer that returns from the server will be stored in variable “total”. Finally, it displays the addition sentence with answer.

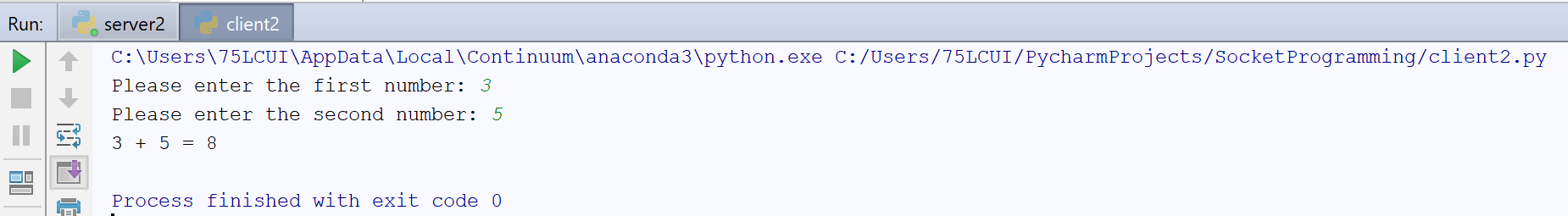


Step 10: run the web service

Please start from running server2.py. You will see a message as following figure.



Then, run the client2.py. It first asks for two numbers, then display the addition sentence with the answer.



Please write your own function in the server, which encrypts a message based on Caesar cipher. See the description and implementation of Caesar cipher here: <https://www.geeksforgeeks.org/caesar-cipher-in-cryptography/>. Here is what you need to do:

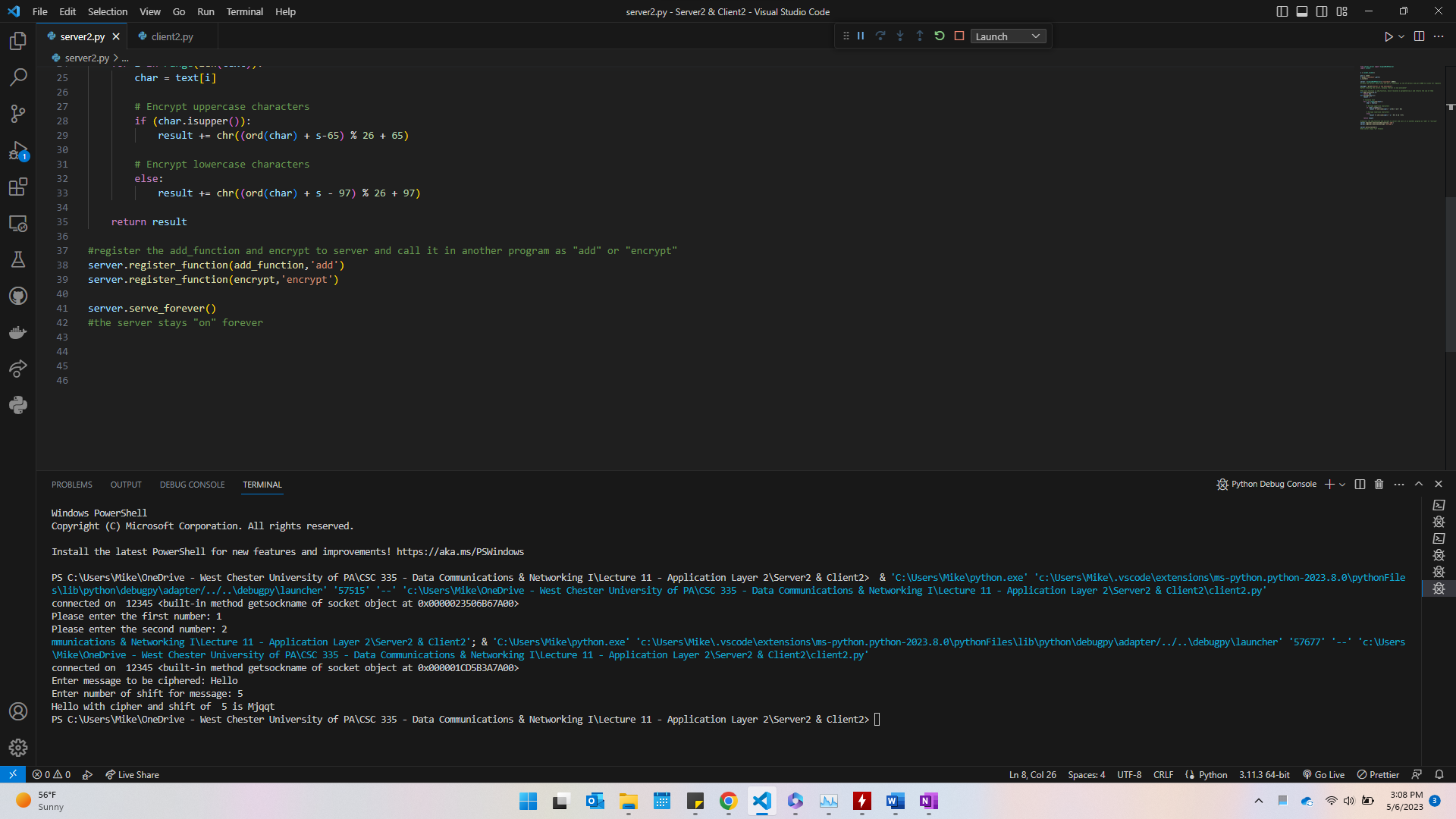
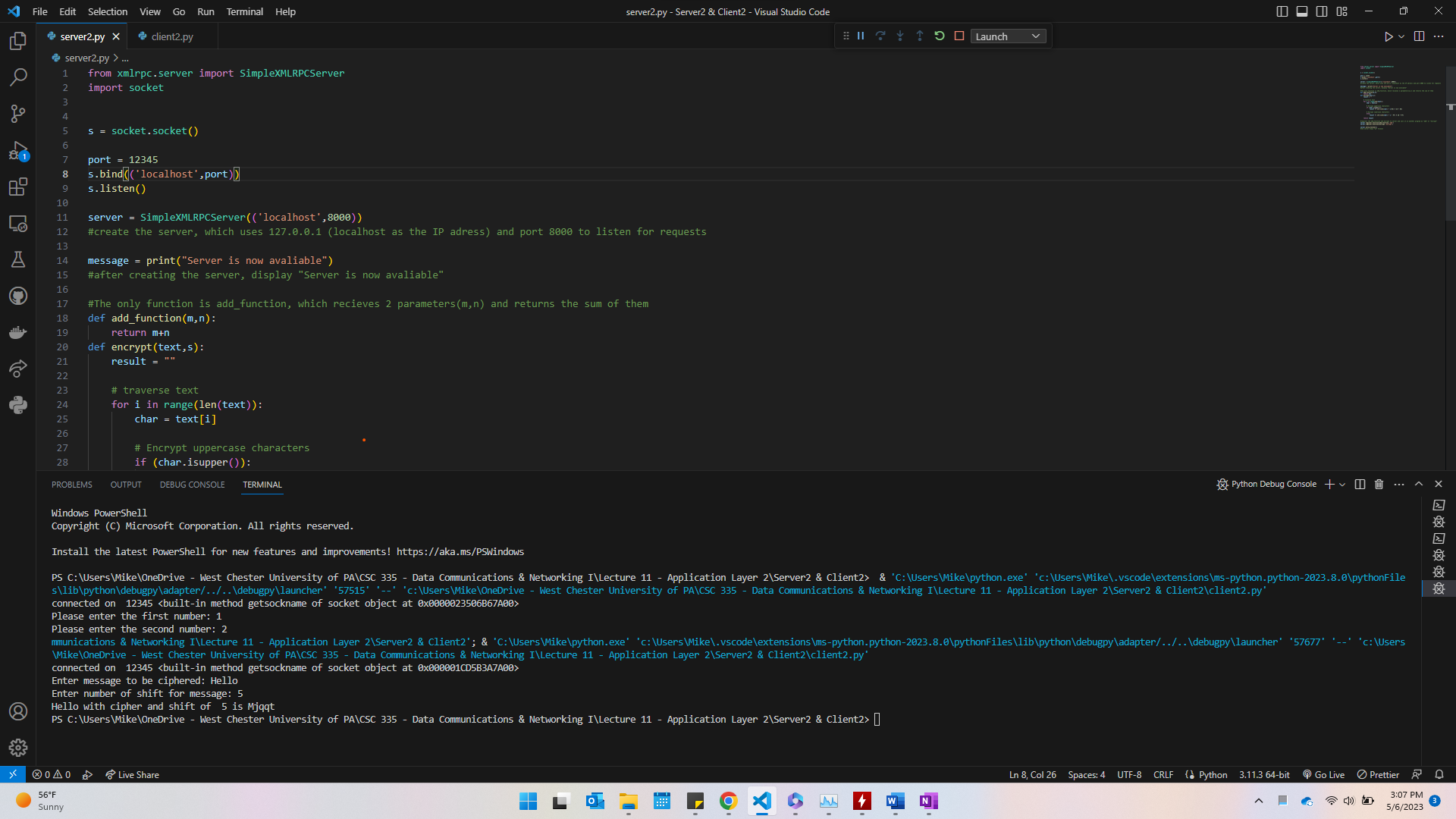
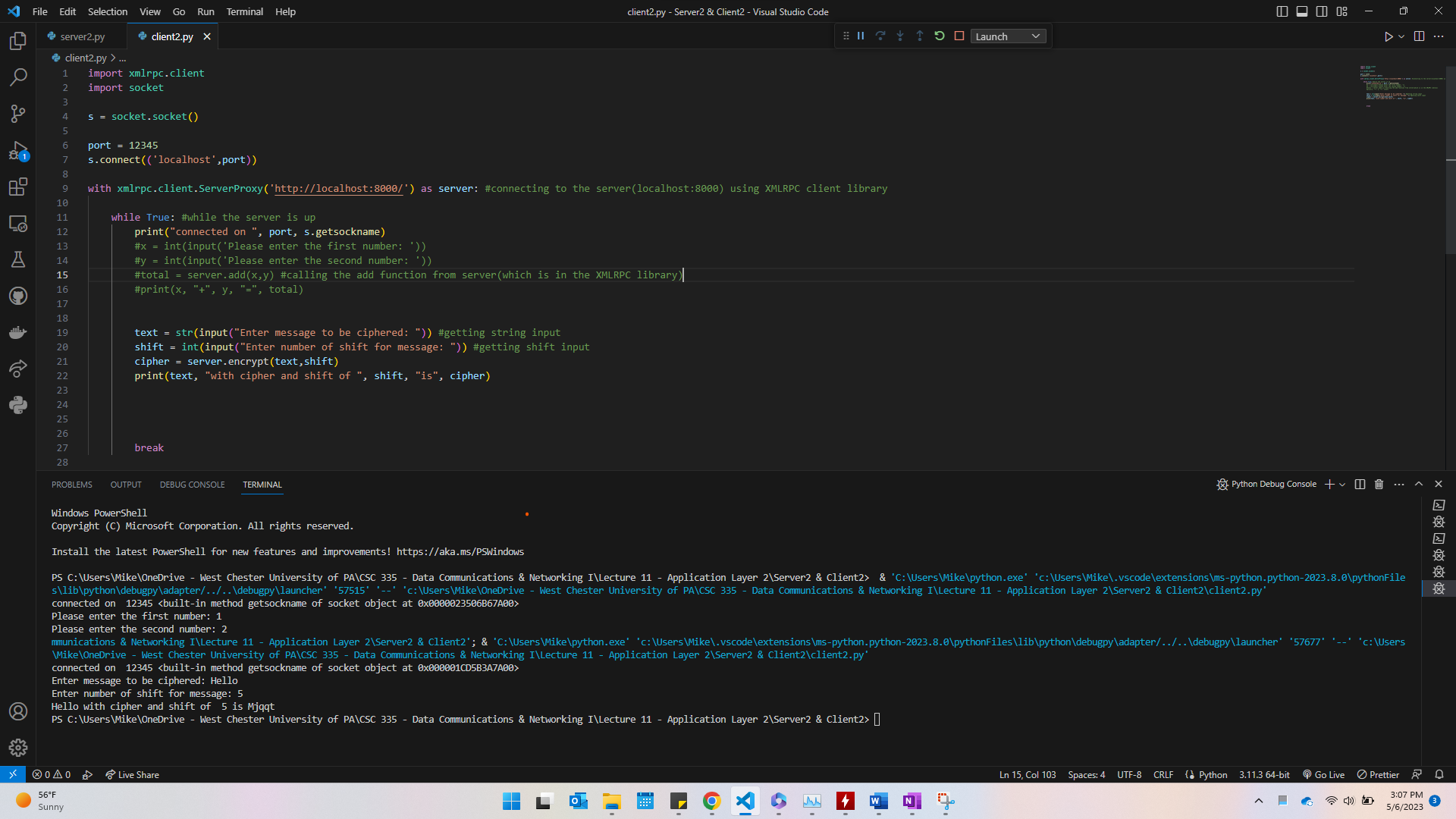
(1) The client receives a message from the user like “Transformers the last knight”. Please note that Caesar cipher can only handle letters.

(2) The client send the message to the server’s Caesar cipher function

(3) The server encrypts the message and send it back to the client

(4) The client display the encrypted message on the screen

Please attach the screenshots of your programs for both server and client as well as the outputs.



**5 Analyze Web Traffic with Wireshark**

Step 11: Capture network traffic on localhost. Now, let’s capture the communication between the server and client using Wireshark. Since both the client and server are on the same machine and we use the IP address 127.0.0.1 (localhost) for the server, please open Wireshark and choose “Adapter for loopback traffic capture”.

Step 12: Run your program. After you start capturing packets, please run server2.py and client2.py. Provide the required input and wait for the answer from the server. In my case, I will provide two numbers from keyboard and wait for the sum of the two numbers from the server.

Step 13: Stop packet capturing after you receive the answer from server.

Step 14: Filter the packets we want using term “tcp.port==8000”, since the server is listening on port 8000. If you set the listening port to other numbers, please change it accordingly.

Step 15: Analyze packets. Now, all packets you see here are relevant to our program. Before you dive into the details of this transmission, let’s review TCP first. As you may see, the transport layer protocol in use here is TCP, and the application layer protocol we use is HTTP (web service). On the server side, we determine it listens on port number 8000. On the client side, port number 55749 is a unused port that chosen by the system. There are many tags for TCP packets. Number 192, 193, and 194 ([SYN], [SYN, ACK], [ACK]) are the TCP three why handshakes, which happens before the actual data can be transmitted. As we mentioned in the lecture, you could find the sequence number (Seq), window size (Win), etc. Number 201, 202, and 203 ([FIN,ACK],[FIN],[FIN,ACK]) are used to tear down the connection. In number 201, the sequence number (Seq) is 258, and acknowledgment number (Ack) is 328. It means the server is sending its 258th packet and waiting for the 328th packet from the client. Therefore, in number 202, the client sending out the 328th packet (Seq = 328) and expecting the 259th packet from the server (Ack = 259).

Table

Description automatically generated

Step 16: Without encryption, we could see the actual contents that transmitted online. Please click “Analyze -> Follow -> HTTP Stream” to see the content. Here is what I got from sum two numbers. As you can see, the numbers I sent from the client are 3 and 4. Then, I got the answer 7 back.

Text

Description automatically generated with medium confidence

Please attack the screenshot of your HTTP stream and show the contents that exchanged between server and client.

Text

Description automatically generated

**6. Bonus (2 points)**

Currently, the server can only process one request at a time. Please modify the server2.py to accommodate multiple clients that sending request at the same time.

**7. References**

<http://web.mit.edu/6.005/www/fa15/classes/21-sockets-networking/>

<https://www.geeksforgeeks.org/socket-programming-python/>

<https://realpython.com/python-sockets/>

<https://docs.python.org/3/howto/sockets.html>

<https://www.coursera.org/learn/web-services-analysis-with-python-and-wireshark/home/welcome>