COMP2190 Project

on

Socket Programming



Name: Jon-Daniel Coombs (620142113)

Course Code: COMP2190

Documentation

Contents

[**Server File** 3](#_Toc85889051)

[Figure 1. Showing the Binding and Setup of Server socket 4](#_Toc85889052)

[Figure 2. Listening for incoming connections 5](#_Toc85889053)

[Figure 3a. Client-handling function Part 1 6](#_Toc85889054)

[Figure 3b. Client-handling function Part 2 7](#_Toc85889055)

[Figure 4. Server Log 8](#_Toc85889056)

[**Client File** 9](#_Toc85889057)

[Figure 5a. Client-side setup 10](#_Toc85889058)

[Figure 5b. Client-side setup continued 11](#_Toc85889059)

[Figure 5c. Client-side setup continued 12](#_Toc85889060)

[Figure 6. Client Log 13](#_Toc85889061)

[**Verify File** 14](#_Toc85889062)

[Figure 7a. Verify File info 15](#_Toc85889063)

[Figure 7b. Verify file info continued 16](#_Toc85889064)

[Figure 7c Verify file info continued 17](#_Toc85889065)

[**Errors** 18](#_Toc85889066)

[Figure 8 Server response to incorrect connection code been given 19](#_Toc85889067)

[Figure 9 Client response to incorrect connection code been given 20](#_Toc85889068)

[Figure 10 Server response to incorrect answer to question been given 21](#_Toc85889069)

[Figure 11 Client response to incorrect answer to question been given 22](#_Toc85889070)

# **Server File**

Figure 1


## Figure 1. Showing the Binding and Setup of Server socket

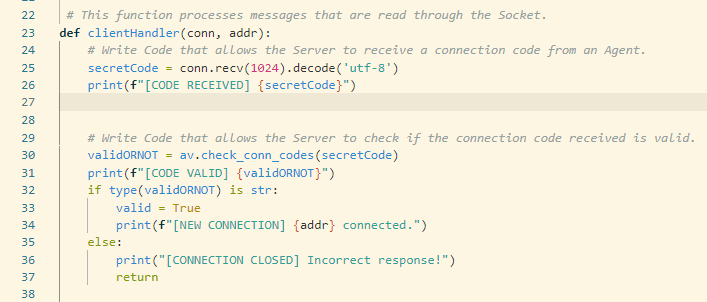
As depicted in Figure 1, to setup the socket the Port and IP Address was required to create the Server Address which would be the central point to which all the clients would be connected to. The variable FORMAT was initialized as it will be of use further in the code to be used for encoding and decoding purposes.

Graphical user interface, text

Description automatically generated

## Figure 2. Listening for incoming connections

As depicted in Figure 2, after creating the socket the next line of business was to make that socket be able to accept incoming connections.



## Figure 3a. Client-handling function Part 1

Figure 3 shows the introduction to the way the server will deal with each connection it receives.

When the client connects to the server. The server will expect is secret code from the client. This secret code will then be decoded using the UTF 8 format. This happens because for the client to send the code over, it must encode it into bytes. Once it reaches over to the server, the server will then convert it back to the usual string using that format. Once completed it will print a message saying CODE RECEIVED and then show secret code that was given by the client in the server log.

Once this code is received, it will then check the verify file to see if this code is valid. If it is valid, then it will say that it's the client is CONNECTED and then show the address of the client and then continue into the process. If it is not valid it show in the server INCORRRECT RESPONSE and it will close the connection to that client.



## Figure 3b. Client-handling function Part 2

Figure 3B continues in the client handling function Here it shows that if the client entered the right connection code The client today and receive a secret question, as can be seen in line 43. As can be seen in line 50, the secret question is then sent over to the client. The secret question is encoded using the format as mentioned before. Once the question is sent the client will then send a responsive answer to this question. The answer that was sent will then be shown in the server log. After this the answer that was sent by the agent will then be compared to the correct answer, which would be the second part of the secret question before. As can be seen in line 60, the if statement is then used to compare both answers. If the correct answer is given, the server will get the message that the correct answer is being given and once that is done it will then send a welcome message which entails both the date and time and the agent’s name. How it knows the agent’s name will be explained further in the verified file. If this if condition is completed, so if the correct answer is given, this will be done, if it is not correct, the server will print a CONNECTION CLOSED message and the connection will be cut.

Graphical user interface, text, application, email

Description automatically generated

## Figure 4. Server Log

# **Client File**

Graphical user interface, text, application

Description automatically generated

## Figure 5a. Client-side setup

For the client, know instead of. Binding we're connecting it to the server. Normally we would not use this **socket.gethostbyname** rather, we would use the public IP address of the client is it’s an external server, but since I am using connecting the client on the same device or the localhost as the server that is running the **socket.gethostbyname** is used.

Graphical user interface, text, application, email

Description automatically generated

## Figure 5b. Client-side setup continued

There are a variety of function that have been used on this client side. At Line 16 we see the **send** function which is used basically to tell the client how it should send information to the server, which is by taking whatever message it intends to send and encoding it using the “utf-8” format.

At Line 22, the **getConnCode** function is created which is there in order to accept the connection code of the Agent.

At Line 29, the **getAnswer** function is located. This function prints out to the client, the random question that was picked by the Server. After printing it out, it asks the Agent to input their answer.

Line 36 shows the assigning of the connection code that was given by the Agent to a variable called **connCode.**

Line 39 shows the use to **send** function to transfer the connection code from the client side to the server side.

Graphical user interface, text, application

Description automatically generated

## Figure 5c. Client-side setup continued

Line 42 shows the question variable which contains the Secret Question that was sent by the Server for the Agent to answer. It shows how the socket is receiving that data with the buffer size of 1024 and decoding the data.

Line 46 show the answer for the Secret Question that was given by the Agent being assigned to the answer variable.

At Line 49 the answer is then being sent off to the Server for processing

Line 53 and 54 shows the result if the answer given by the Agent was correct. It will print out a message telling the specific Agent, Welcome and also state the date and time of the login.

Graphical user interface, text, application, email

Description automatically generated

## Figure 6. Client Log

# **Verify File**

Graphical user interface, website

Description automatically generated

## Figure 7a. Verify File info

In this depiction the necessary information to verify an Agent’s identity is provided. Each Agent is given a specific number and when this is combined with the predefined list of codes, it creates the required connection codes to login into the server.

Graphical user interface, text, application

Description automatically generated

## Figure 7b. Verify file info continued

As can be seen in the for loop in the figure above, the respective codes for each Agent were created by joining both the predefined code and the agent numbers and placing them in the specific list.

Below that loop the list of tuples containing the Secret Questions and their answers are present. By manipulating the indexes, the questions can be extracted to show to the client as could be seen in Figure 3b, Line 45-51. The answers also were extracted to be used.

Graphical user interface, text, application, Teams

Description automatically generated

## Figure 7c Verify file info continued

In the image above two functions can be seen. Firstly, the **getSecretQuestion** function, which the **random** function is used to randomly pick a tuple from the list called **question** as seen in Figure 7b Line 25 and returns this specific tuple. The tuple is then sent over to the Server side as can be seen in Figure 3b line 43 to be manipulated.

The second function is called **check\_conn\_code** which takes in the connection code that was given by the Agent and what it does is first scan the first batch of codes for the first Agent to see if anyone of them matches with he given connection code. If there aren’t any matches it then loops through the send batch of Agent codes to see if it matches.

If it finds a match what happens the is that the function returns name of the Agent whose batch of Agent codes that connection code was found in. For example, if it matches with a code from the Agent A batch the function tells the server that “the code that was given is a code corresponding with Agent A” and same is the case for Agent B. These values are returned as a string to fit the criteria for the validation section as seen in Figure 3a, Line 30 – 37.

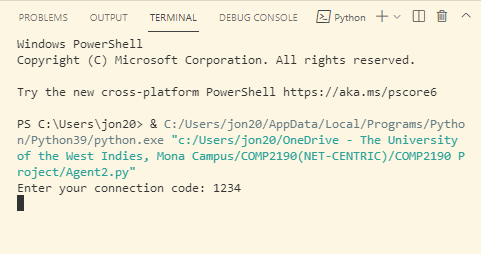
If it does not find any matches it returns the integer -1 which is then used as a signal by the server to terminate the connection.

# **Errors**



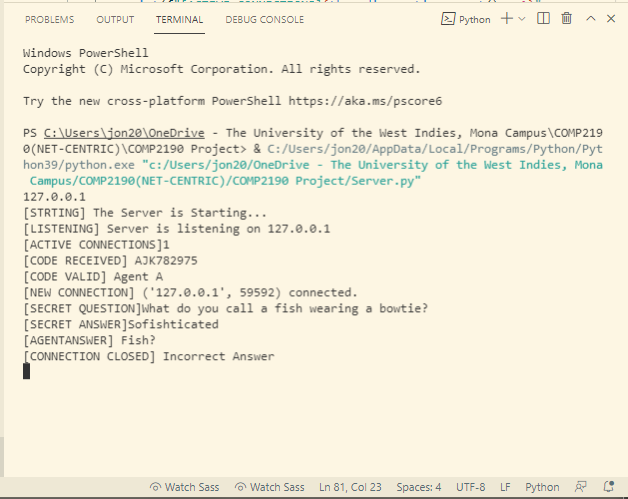
## Figure 8 Server response to incorrect connection code been given

As can be seen in the image above, the client sent the Connection Code (1234) to the server. In response to this the server check to see if code is correct by means of the verify file and receives the massage (-1) which means that the code is invalid. Once the server realizes this it ends the connection.



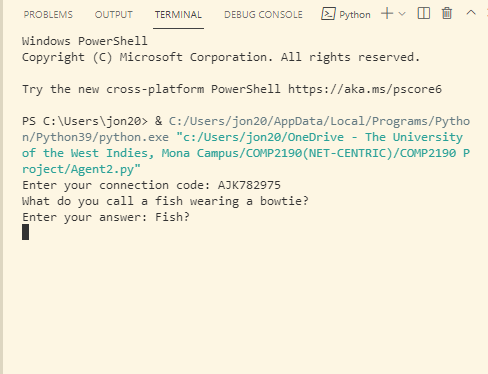
## Figure 9 Client response to incorrect connection code been given

Once the incorrect response was given by the client side and the server terminates the connection as can be seen in Figure 8, on the client side once the connection is terminated it will not allow any more inputs to be made



## Figure 10 Server response to incorrect answer to question been given

As show in Figure 10, once the incorrect answer is given the connection is terminated.



## Figure 11 Client response to incorrect answer to question been given

Once the incorrect response was given by the client side and the server terminates the connection as can be seen in Figure 10, on the client side once the connection is terminated it will not allow any more inputs to be made