



**CSE 328**

**Computer Networks**

**Project 1**

**Socket Programming**

**18/3/2023**

**Submitted by**

**Youssef Ashraf El-Harty – CSE 2 – 120200090**

**Ahmed Hagag Abulwafa – CSE 2 – 120200101**

**to**

**Dr. Ahmed Fares**

## Server File Code:

This Python file is a server program that listens for incoming connections on a specified IP address and port number, and handles client requests.

The socket and threading modules are imported at the beginning of the file.

The count variable is initialized to 0.

The `handle_client` function takes four parameters: `conn` (the connection object), `addr` (the address of the client), `server_name` (a string representing the name of the server), and `server_number` (an integer representing a number associated with the server).

Within the function, the incoming client request is received and decoded. If the data is empty, the function breaks out of the loop. Otherwise, the `client_name` and `client_number` are extracted from the data string. If the `client_number` is greater than 100 or less than 0, the function breaks out of the loop.

Next, the function prints a message indicating the client has connected to the server, and prints the client's number. It then calculates the sum of the `client_number` and `server_number`, and prints this sum. The modified message is then encoded and sent back to the client.

The `start_server` function initializes the server by setting the IP address, port number, `server_name`, and `server_number` values. A socket object is then created, bound to the specified IP address and port number, and set to listen for incoming connections. The function then enters a loop that continuously accepts incoming client connections, creating a new thread to handle each connection.

Finally, the `if __name__ == "__main__":` block executes the `start_server` function if the script is run directly (rather than imported as a module).

## Client File Code:

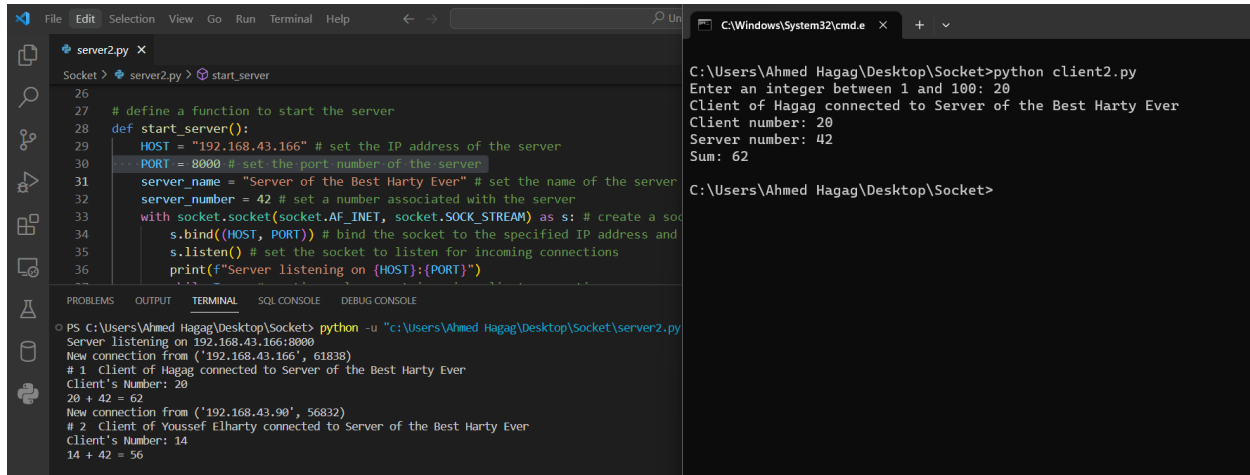
This Python script defines a function called `start_client` that establishes a socket connection to a remote server and sends data to the server.

The script starts by importing the socket module, which provides low-level network communication. The `start_client` function sets the IP address and port number of the server to connect to, creates a socket object, and connects to the server using the `connect` method.

The function then enters an infinite loop that prompts the user to enter an integer between 1 and 100. The integer and a client name are combined into a message and sent to the server using the `send` method. The client then receives data from the server using the `recv` method, which is decoded from bytes to a string using the `decode` method.

The received data contains the server name and number, which are extracted from the message using the split method. The client then calculates the sum of the server and client numbers and prints the client number, server number, and their sum. Finally, the client socket connection is closed using the close method.

If the script is run directly, the start\_client function is called, initiating the process of connecting to the server and sending and receiving data.



The screenshot displays a code editor on the left and a terminal window on the right. The code editor shows a Python script named `server2.py` with the following content:

```
26
27 # define a function to start the server
28 def start_server():
29     HOST = "192.168.43.166" # set the IP address of the server
30     PORT = 8000 # set the port number of the server
31     server_name = "Server of the Best Harty Ever" # set the name of the server
32     server_number = 42 # set a number associated with the server
33     with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s: # create a socket
34         s.bind((HOST, PORT)) # bind the socket to the specified IP address and
35         s.listen() # set the socket to listen for incoming connections
36         print(f"Server listening on (HOST):(PORT)")
```

The terminal window on the right shows the execution of the server script. It prompts the user to enter an integer between 1 and 100, and then displays the output for two clients:

```
C:\Users\Ahmed Hagag\Desktop\Socket>python client2.py
Enter an integer between 1 and 100: 20
Client of Hagag connected to Server of the Best Harty Ever
Client number: 20
Server number: 42
Sum: 62

C:\Users\Ahmed Hagag\Desktop\Socket>
```

The terminal window on the left shows the output of the server script, indicating that it is listening on 192.168.43.166:8000 and has received two connections from different clients:

```
PS C:\Users\Ahmed Hagag\Desktop\Socket> python -u "c:\Users\Ahmed Hagag\Desktop\Socket\server2.py"
Server listening on 192.168.43.166:8000
New connection from ('192.168.43.166', 61838)
# 1 Client of Hagag connected to Server of the Best Harty Ever
Client's Number: 20
20 + 42 = 62
New connection from ('192.168.43.90', 56832)
# 2 Client of Youssef Elharty connected to Server of the Best Harty Ever
Client's Number: 14
14 + 42 = 56
```

This screenshot shows two clients with different IP addresses and ports.