Abhinay Lavu
4/4/2024
CS4390
Dr. Ding

Design of 2-way Networking Project

Project Overview:

This project involves creating two Python scripts for simulating communication between a client and a server. The server is initiated first, waiting for connections from clients. Once the client script is executed and connects to the server with the proper port number, it sends the user's name and a number between 1 and 100 chosen by the user.

The server generates a random number within the same range, adds it to the client's number, and sends the result back to the client. This process repeats until the client inputs a number outside the range of 1 to 100. At that point, both the server and client terminate their connection.

Design Choices:

I was given the choice to code in a couple languages; Python was chosen as the primary language for implementing the client-server communication due to its large support, my prior experience with Python, its simplicity, and cross-platform compatibility.

The project follows a client-server architecture, where one program (the server) listens for incoming connections and processes requests, while another program (the client) initiates connections and sends requests to the server.

Both the client and server utilize Python's socket module for network communication. Socket programming provides a low-level interface for network communication, allowing the client and server to establish connections, send data, and receive responses. By using sockets, the project achieves platform-independent communication over TCP/IP, enabling interoperability across different devices and operating systems.

Abhinay Lavu
4/4/2024
CS4390
Dr. Ding

Challenges:

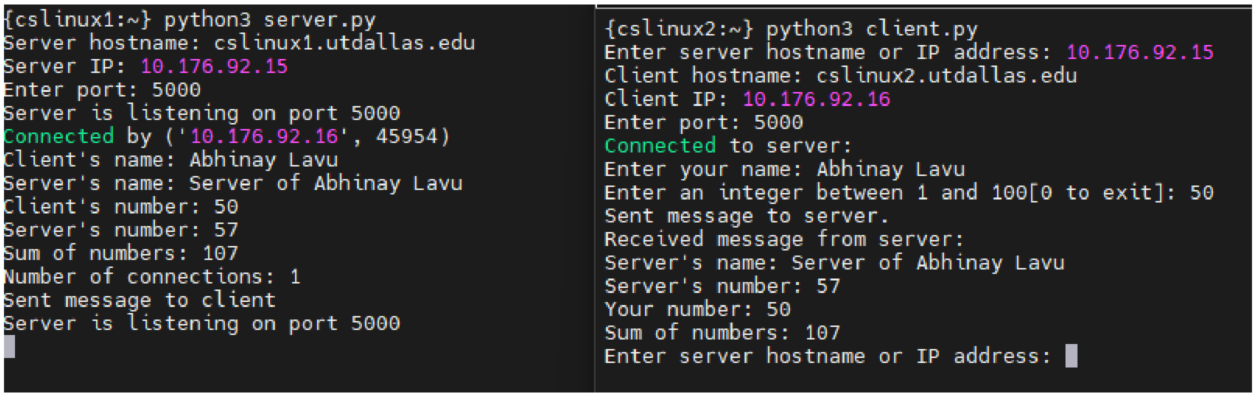
My largest issue came from not being able to connect to the proper IP Address of the server. I was focused on trying to have the client connect to the server automatically, however, I realized that this was not possible since the client would have no way of knowing the server's IP address unless it was hardcoded. I was able to solve this by having the client ask for the input of the server's IP address, and then connecting to it if the correct port number was available.

I also wanted to incorporate input validation, ensuring that the client would not end unless it sends a number outside the range of 1 to 100 to the server. I specified "0 to exit" because entering 0 is much faster than a number greater than 100, but any number outside of the range does the task. Having input validation for the port number proved to be challenging.

Another challenge was ensuring the data was able to be parsed properly by the client and server. There were difficulties when the client inputted a name with spaces in it, for example: "John Smith". I was able to solve this issue by incorporating regular expressions to filter out the name and number of the client and server.

Sample Images:

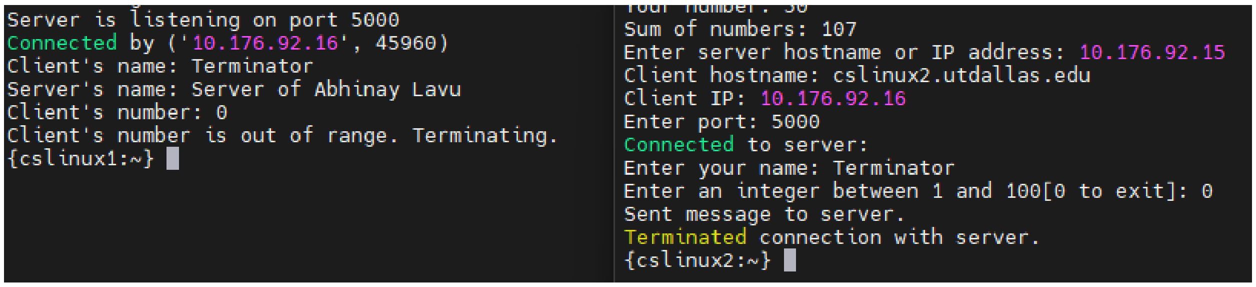
Proper Connection between two different machines:



```
{cslinux1:~} python3 server.py
Server hostname: cslinux1.utdallas.edu
Server IP: 10.176.92.15
Enter port: 5000
Server is listening on port 5000
Connected by ('10.176.92.16', 45954)
Client's name: Abhinay Lavu
Server's name: Server of Abhinay Lavu
Client's number: 50
Server's number: 57
Sum of numbers: 107
Number of connections: 1
Sent message to client
Server is listening on port 5000
```

```
{cslinux2:~} python3 client.py
Enter server hostname or IP address: 10.176.92.15
Client hostname: cslinux2.utdallas.edu
Client IP: 10.176.92.16
Enter port: 5000
Connected to server:
Enter your name: Abhinay Lavu
Enter an integer between 1 and 100[0 to exit]: 50
Sent message to server.
Received message from server:
Server's name: Server of Abhinay Lavu
Server's number: 57
Your number: 50
Sum of numbers: 107
Enter server hostname or IP address:
```

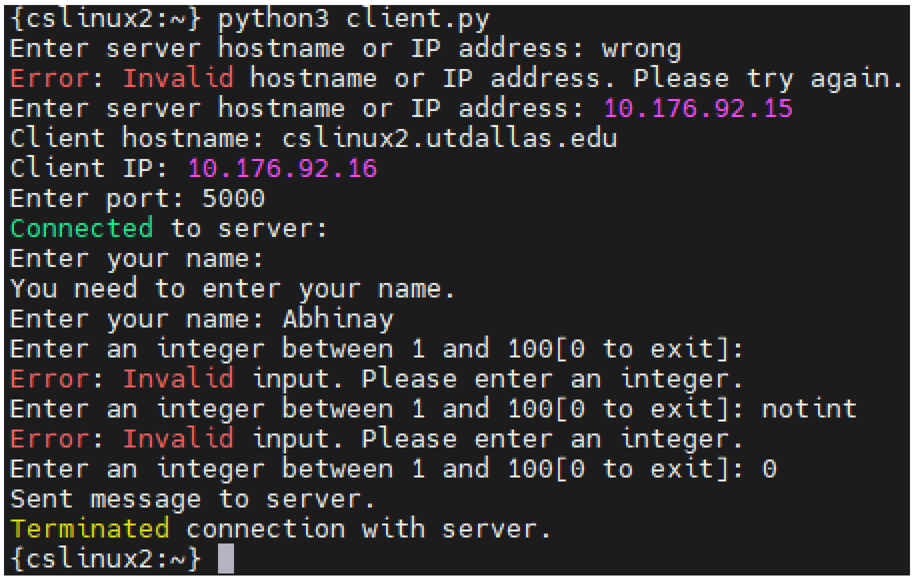
Abhinay Lavu
4/4/2024
CS4390
Dr. Ding
Termination:



```
Server is listening on port 5000
Connected by ('10.176.92.16', 45960)
Client's name: Terminator
Server's name: Server of Abhinay Lavu
Client's number: 0
Client's number is out of range. Terminating.
{cslinux1:~}
```

```
Sum of numbers: 107
Enter server hostname or IP address: 10.176.92.15
Client hostname: cslinux2.utdallas.edu
Client IP: 10.176.92.16
Enter port: 5000
Connected to server:
Enter your name: Terminator
Enter an integer between 1 and 100[0 to exit]: 0
Sent message to server.
Terminated connection with server.
{cslinux2:~}
```

Input Validation:



```
{cslinux2:~} python3 client.py
Enter server hostname or IP address: wrong
Error: Invalid hostname or IP address. Please try again.
Enter server hostname or IP address: 10.176.92.15
Client hostname: cslinux2.utdallas.edu
Client IP: 10.176.92.16
Enter port: 5000
Connected to server:
Enter your name:
You need to enter your name.
Enter your name: Abhinay
Enter an integer between 1 and 100[0 to exit]:
Error: Invalid input. Please enter an integer.
Enter an integer between 1 and 100[0 to exit]: notint
Error: Invalid input. Please enter an integer.
Enter an integer between 1 and 100[0 to exit]: 0
Sent message to server.
Terminated connection with server.
{cslinux2:~}
```