Marcos Valdez
valdemar
CS 372 Sec 400
12/03/2022

# Project 4 – Client / Server Chat

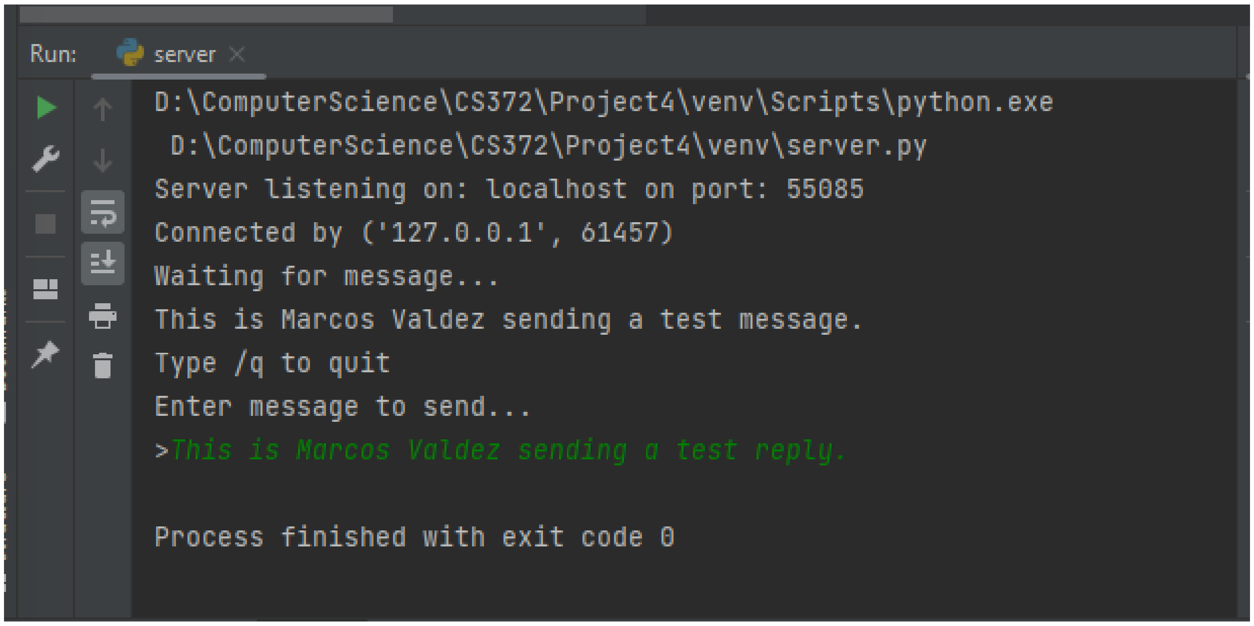
## A. Instructions for Running Program

Note: Programs are written in Python 3 and are run by running server.py and client.py as scripts simultaneously in two separate IDE consoles. The files are dependent on each other as well as config.py. Output is formatted to mimic the example screenshots in the assignment instructions. Extra credit functionality was not implemented.

1. Ensure Python 3 is installed on the device you are using.
2. Place all 3 .py files in a single folder/project.
3. Open the folder/project in your IDE of choice.
4. Ensure the run configuration of the folder/project uses the Python3 interpreter.
5. Open both server.py and client.py in the IDE.
6. Run server.py as a script.
7. Run client.py as a script.
8. Optional: If the IDE supports it, display the consoles for each file side by side. If you don't do this step, you'll have to tab between the consoles for each message.
9. In the client console, type a message and press Enter to send.
10. In the server console, view the message. In the same console, type a reply and press Enter to send.
11. In the client console, view the reply.
12. Continue chatting to yourself across client and sever by repeating steps 9-11. The last line of the active console will always display a ">". To close the connection at any time, send the message "/q", without quotes, from the active console. No confirmation is displayed by the code but the IDE should show an exit code in both consoles.

B. Code Output Screenshots

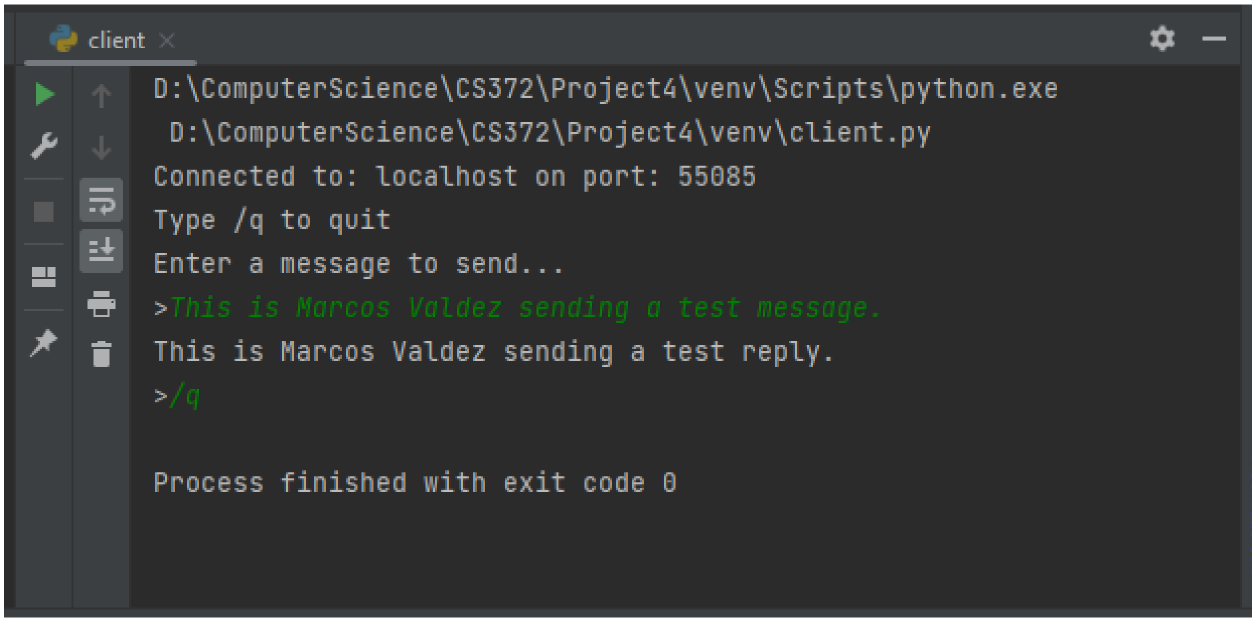
1. Server Side: Listens, accepts, receives, sends, and closes (when client quits).



```
D:\ComputerScience\CS372\Project4\venv\Scripts\python.exe
 D:\ComputerScience\CS372\Project4\venv\server.py
Server listening on: localhost on port: 55085
Connected by ('127.0.0.1', 61457)
Waiting for message...
This is Marcos Valdez sending a test message.
Type /q to quit
Enter message to send...
>This is Marcos Valdez sending a test reply.

Process finished with exit code 0
```

2. Client Side: Connects, sends, receives, and quits (resulting in closing).



```
D:\ComputerScience\CS372\Project4\venv\Scripts\python.exe
 D:\ComputerScience\CS372\Project4\venv\client.py
Connected to: localhost on port: 55085
Type /q to quit
Enter a message to send...
>This is Marcos Valdez sending a test message.
This is Marcos Valdez sending a test reply.
>/q

Process finished with exit code 0
```

## C. Comments

Sources employed included one of the tutorials provided in the assignment instructions and some logic from my submission of Project 1 in this class. Both are cited in the code files.

Port Hanging:

In testing, attempting to connect from the client port (rather than the server port, as noted in the assignment instructions) would occasionally throw the following error:

[WinError 10048] Only one usage of each socket address (protocol/network address/port) is normally permitted

I was unable to solve the problem in a similar manner to the suggestion for the server port so I made the client port random each time the script runs.

Handling Received Data:

When one host or the other closes the connection, I elected to send no information to the other party because the wording of the instructions seem to indicate this is the desired behavior. As such, I had each side check the size, in bytes, of the return value of .recv() called on the open socket to determine if nothing was sent. Testing revealed an empty response was 17 bytes. Thus, I check responses against the constant in config.py of MIN_RESP =  18.

This is similar to my (inelegant) solution to the same issue in my large_file.py submission for Project 1 in which the size of a .recv() return value with no HTML data was 33 bytes.

With additional time, this would be the primary issue I'd like to fix. It's also possible I'm overthinking it and I could have just sent the "/q" and had the other party respond appropriately.

While my default buffer size of 4096 bytes is probably more than enough for a text-only chat program, my solution would presumably truncate anything larger as it does not include logic to keep checking for more data.

User Experience:

I duplicated the output format of the examples from the assignment instructions to save time. The readability could be improved. Furthermore, running two scripts in two IDE consoles is clunky and could be corrected with a bit of work.