Advanced Socket Exercise - Multi-User Chat Writing - Barak Gonen

Explanatory video for the exercise: https://youtu.be/bWU6TfHbf1g

You need to write a server and client to allow chat between multiple users. Submit the server file, client file, and protocol file. Check that the server works with multiple clients communicating simultaneously. Go over the example run and make sure your programs runs the same.

Commands supported by the client

NAME <name> will set name. The server will reply with an error if the name already exists
GET_NAMES will get all client names
MSG <NAME><message> will send <message> to the client NAME
EXIT will close the client and connection

Explanation

The NAME command allows you to determine a name for our customer. For example, NAME A will set the name A for the client, so that messages intended for A are now forwarded to it by the server. The server will answer it with HELLO with the specified name. For example, HELLO A. If the name A is already occupied by another client, the server will reply that the name is taken.

A GET_NAMES command will cause the server to send the client a list of all client names that are connected to the server (don't forget to remove disconnected clients)

An MSG command will send a message to the customer based on their name.

For example, MSG A HELLO will send HELLO to CLIENT A. The server will first add the sender's name; for example, if B sent the message to A, then A will receive B SENT HELLO.

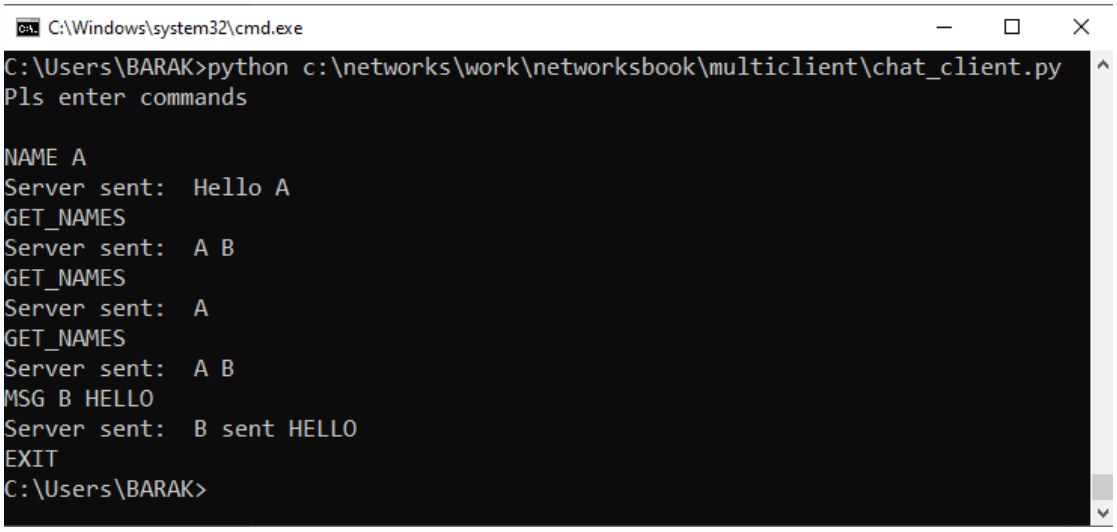
An EXIT order will cause the customer to be disconnected closing the socket.

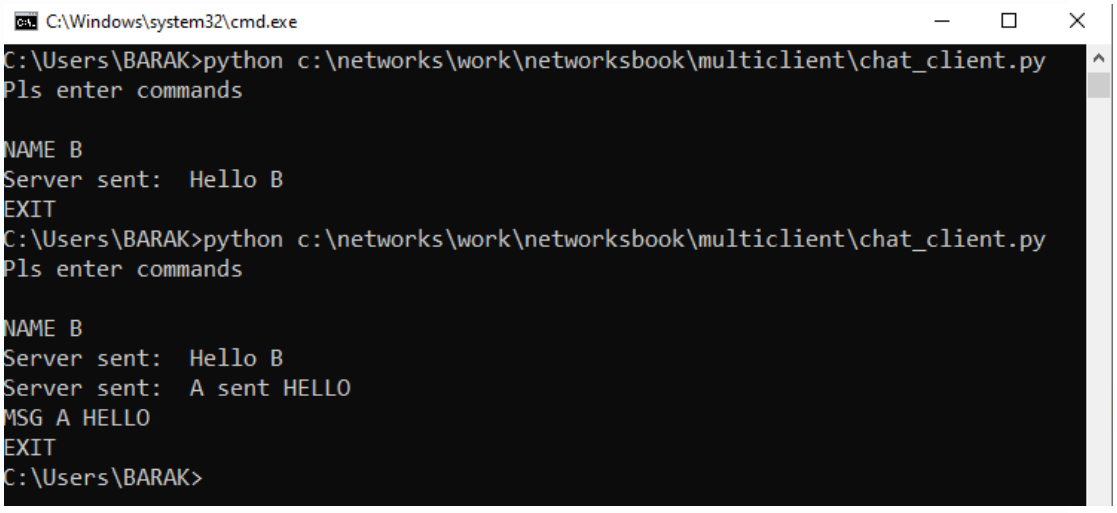
Example run

The following screens show the following:

1. Client connects
2. The client sets the name A
3. Another Client connects
4. The client sets the name B
5. Client A checked which clients exist and received A B

6. Client B disconnected
7. Client A checked which customers exist and received A
8. Another Client connects
9. The client set the name B
10. Client A checked which clients exist and received A B
11. Client A sent a HELLO message to B
12. Client B received the message along with a message from the server that the message was sent from A
13. Client B sent a HELLO message to A
14. Client B disconnects.
15. Client A disconnects.





General guidelines

Like the basic server exercise, you need to use a protocol file, which will include a length field. The server and the client will use the length field in communicating (and not just *recv*(1024)).

If the server receives an unknown command, it will return an appropriate message to the client that sent the message.

If the server is prompted to forward a message to a client that does not exist, it will return an appropriate message to the client that sent the message.

Server writing guidance
Using the server code in the textbook, the main difference is that you need to write a function that handles messages coming from the client. It is recommended to have a data structure that will link a socket to a client name; a **Python dictionary** can be useful.

Client writing guidance
As you may recall, the ***recv*** and ***input*** functions are blocking, meaning that the customer will wait for information before moving on. This will prevent the program from running properly because the client may not be able to receive user input until they receive a notification, and vice versa – it will not be possible to receive messages until the user has finished entering input.
The solution consists of two parts.

Instead of the *recv* function, the client will use the ***select*** function. You can set the **timeout** in seconds.
Instead of the *input* function, we import the **msvcrt** module (an important part of our skill is self-study of topics, explanations and examples can be found on the Internet, so I am not attaching an explanation) Create code that receives a character at a time from the user using the ***kbhit*** and ***getch*** functions, prints to the screen and after receiving a line feed character (Enter) sends the message to the server.
Tip: When printing to the screen, use the ***flush=True*** parameter so that each character prints immediately to the screen.

Good luck!