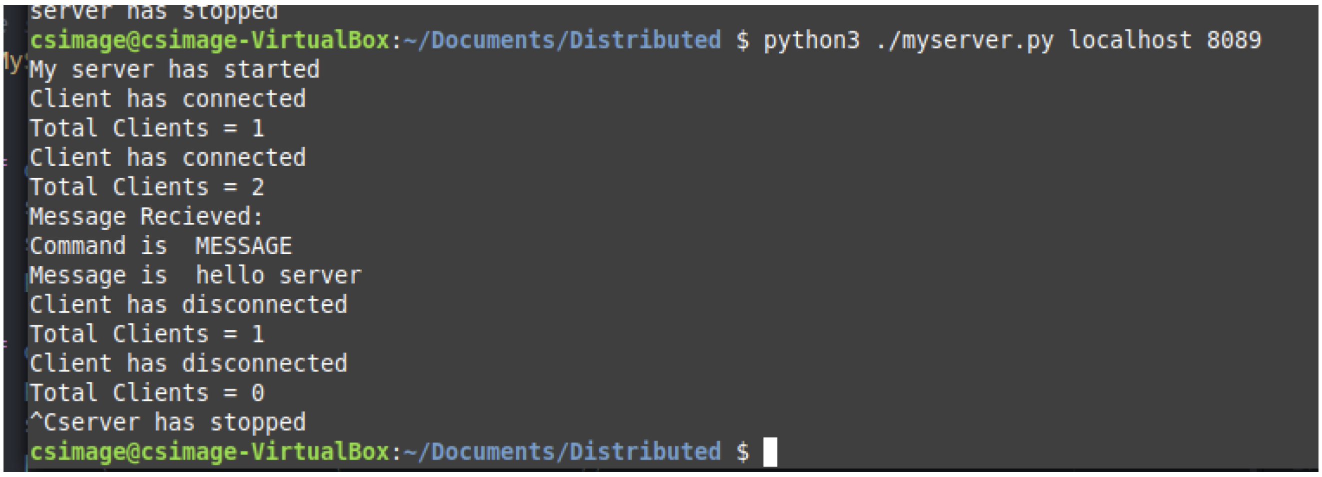
n79859am

Task 2.1 / 2.2 / 2.3

Please see myserver.py file for more information

Demo of server implementation



```
server has stopped
csimage@csimage-VirtualBox:~/Documents/Distributed $ python3 ./myserver.py localhost 8089
My server has started
Client has connected
Total Clients = 1
Client has connected
Total Clients = 2
Message Recieved:
Command is  MESSAGE
Message is  hello server
Client has disconnected
Total Clients = 1
Client has disconnected
Total Clients = 0
^Cserver has stopped
csimage@csimage-VirtualBox:~/Documents/Distributed $
```

Task 2.4 / 2.5

Starting with all the commands I will use and their functions

<NAME>	get name ; compare with others ; if unique set name
<MSGALL>	get msg ; get all users ; send to all users
<MSGIND>	send message to 1 client Must use next word as users name MSGIND <anna> hello what is up
<LIST>	iterate through the client name dictionary and print to client who requested
<CLOSE>	close connection for specified client

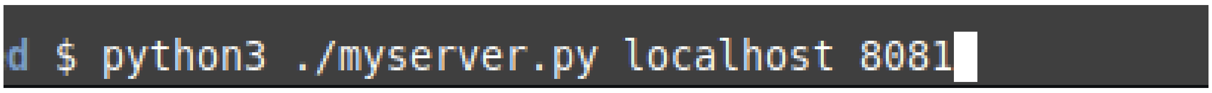
PseudoCode

Create variables

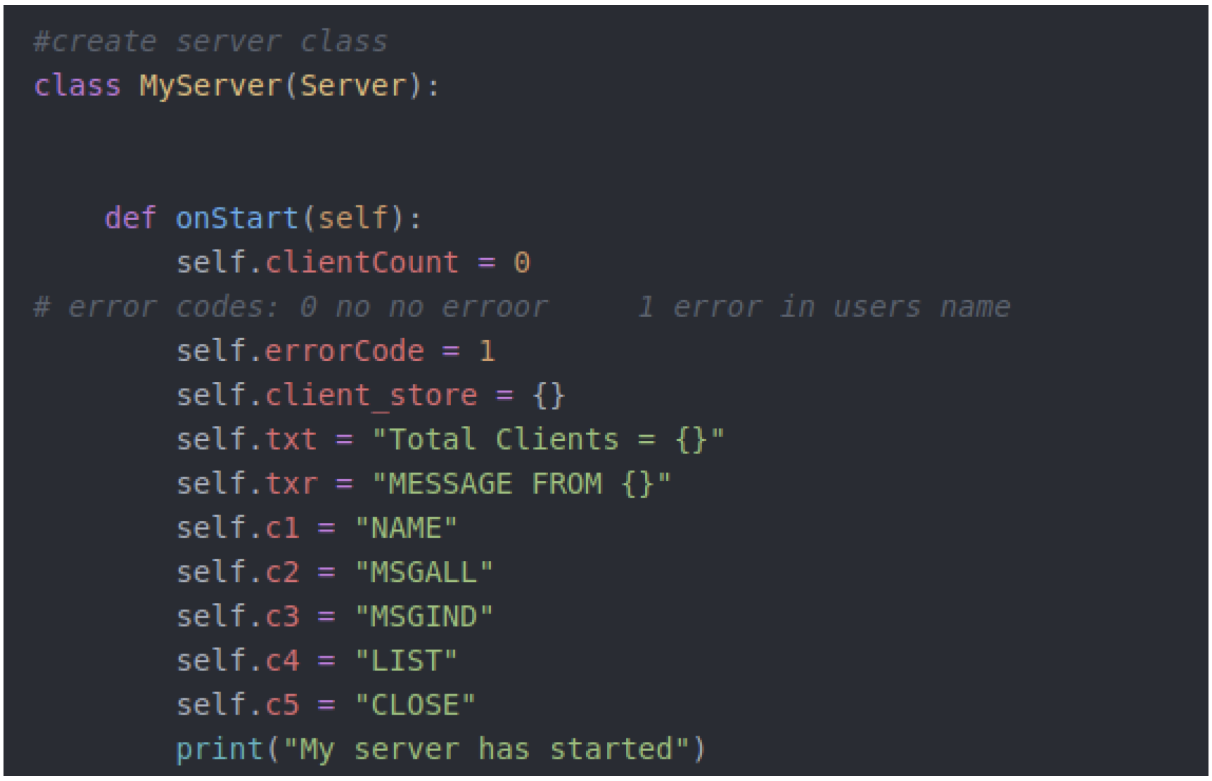
```
On message():
    Split message COMMAND parameter
    if(command == name)
        Register user into dictionary with naame and socket

    Else if(command == msgall)
        Loop through clients, send message too all but sender

    Else if(command == msgind)
        Split msg again into -user-parameter-
        Loop though clients find socket that matches -user- send msg there
        If user was not found report error msg to sender

    Else if(command == list)
        Loop through all clients and print their names out to sender who requested it

    Else if(command == close)
        Close the connection for this client

    Else
        Handle the errors -- tell sender to use a registered command
```

I will make a store in the ‘self.’ to store all these words and match them up the inputs with them. I will use an IF ELIF ELSE structure to implement. In terms of the clients I will use a dictionary store to allow me implement a key value pair storage system; each socket has with it the associated users name. I spent some time thinking about how I would implement all the functionality but I will describe my protocol here once I have built it. I will find it to be a cleaner and better thought out explanation.

Server initialisation



```
d $ python3 ./myserver.py localhost 8081
```

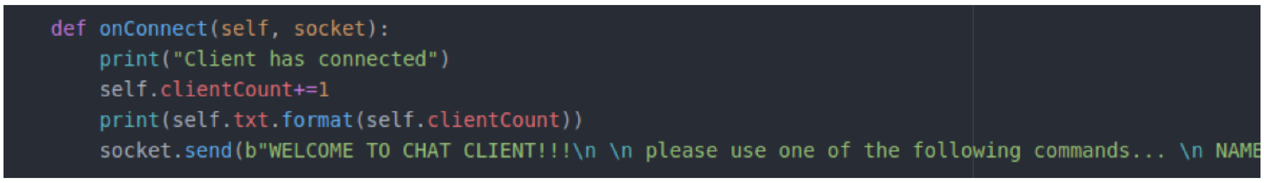


```
#create server class
class MyServer(Server):


    def onStart(self):
        self.clientCount = 0
# error codes: 0 no no erroor      1 error in users name
        self.errorCode = 1
        self.client_store = {}
        self.txt = "Total Clients = {}"
        self.txr = "MESSAGE FROM {}"
        self.c1 = "NAME"
        self.c2 = "MSGALL"
        self.c3 = "MSGIND"
        self.c4 = "LIST"
        self.c5 = "CLOSE"
        print("My server has started")
```

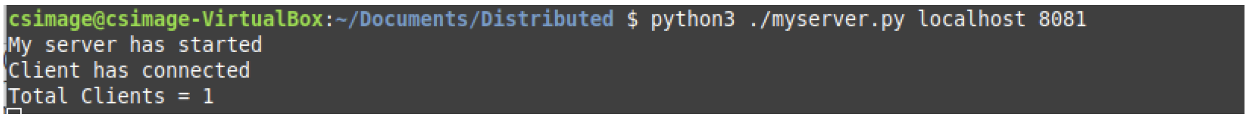
Once started I initialise all the variables that I will need (see above) including setting an error code and making a dictionary for clients names.

Update: error code has been moved to be set in the MSGIND code to ensure each error is caught not just first error.

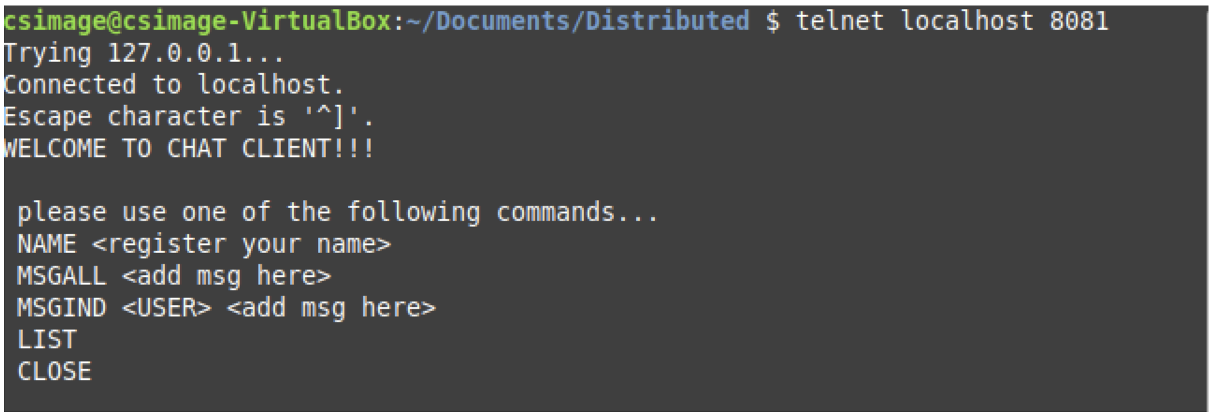
Client Initialisation
Upon the first client connecting, the server will increment the client count and send some useful information to the client to help them get started.



```
    def onConnect(self, socket):
        print("Client has connected")
        self.clientCount+=1
        print(self.txt.format(self.clientCount))
        socket.send(b"WELCOME TO CHAT CLIENT!!!\n \n please use one of the following commands... \n NAME
```



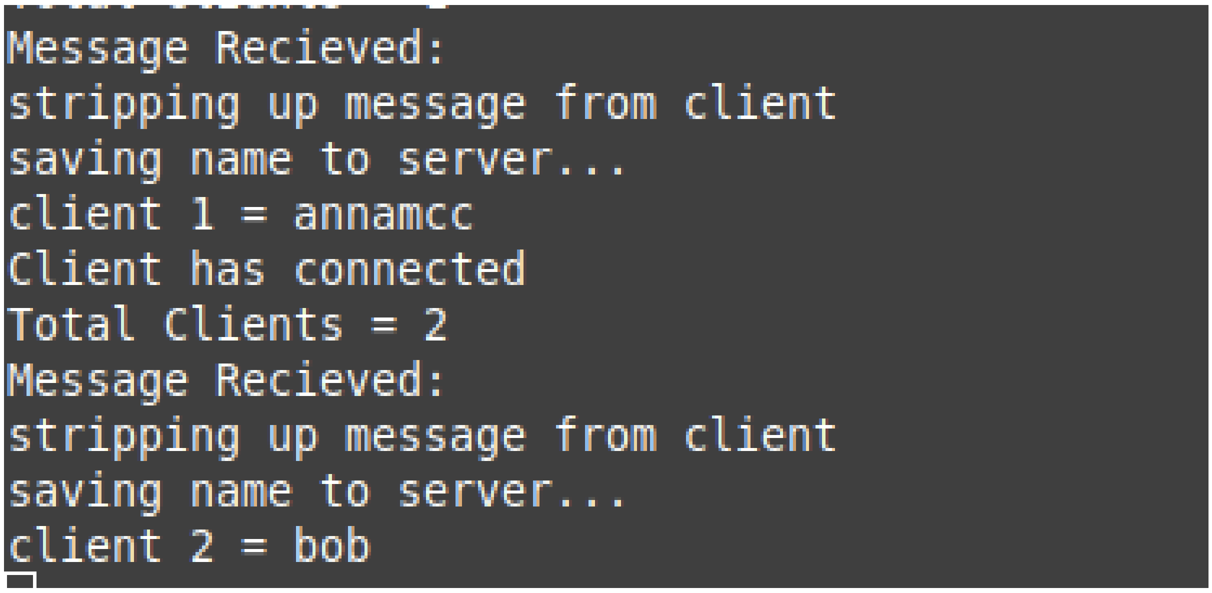
```
csimage@csimage-VirtualBox:~/Documents/Distributed $ python3 ./myserver.py localhost 8081
My server has started
Client has connected
Total Clients = 1
```



```
csimage@csimage-VirtualBox:~/Documents/Distributed $ telnet localhost 8081
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
WELCOME TO CHAT CLIENT!!!

 please use one of the following commands...
 NAME <register your name>
 MSGALL <add msg here>
 MSGIND <USER> <add msg here>
 LIST
 CLOSE
```

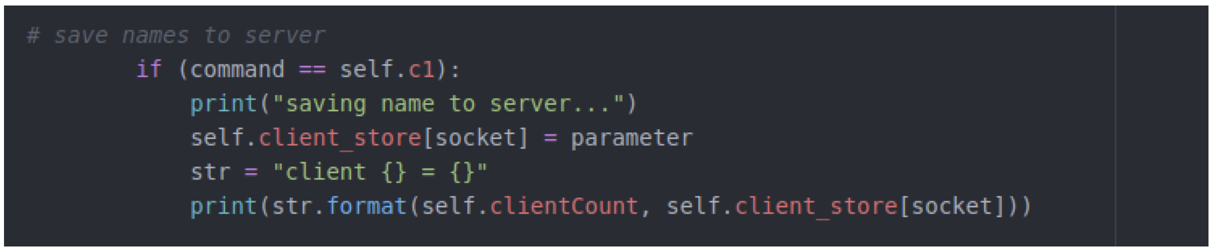
Registration of new users

A client can proceed and type <NAME anna> to be registered by the server.



```
Message Recieved:
stripping up message from client
saving name to server...
client 1 = annamcc
Client has connected
Total Clients = 2
Message Recieved:
stripping up message from client
saving name to server...
client 2 = bob
```

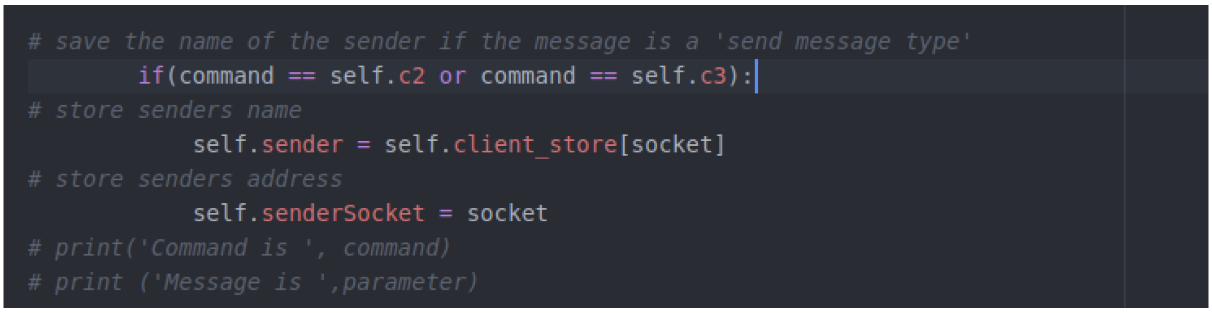
The users name (parameter) will be stored under the value of this clients socket



```
# save names to server
        if (command == self.c1):
            print("saving name to server...")
            self.client_store[socket] = parameter
            str = "client {} = {}"
            print(str.format(self.clientCount, self.client_store[socket]))
```

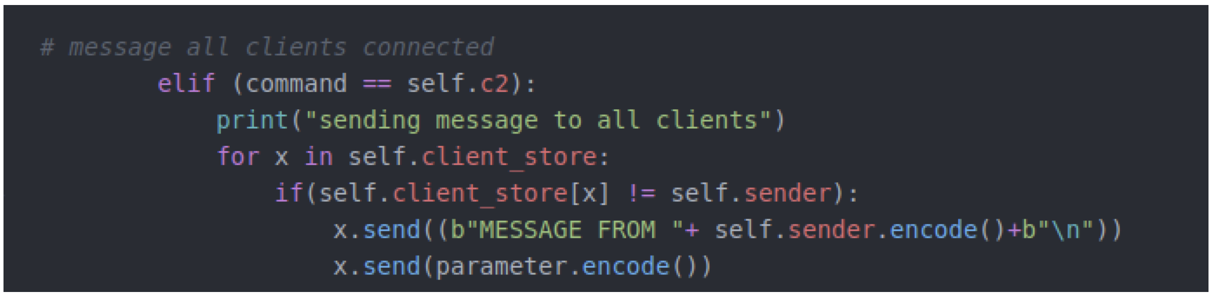
Message to all clients

Firstly I added some code to assist in sending message commands; it will store the senders information in an easy accessible place.

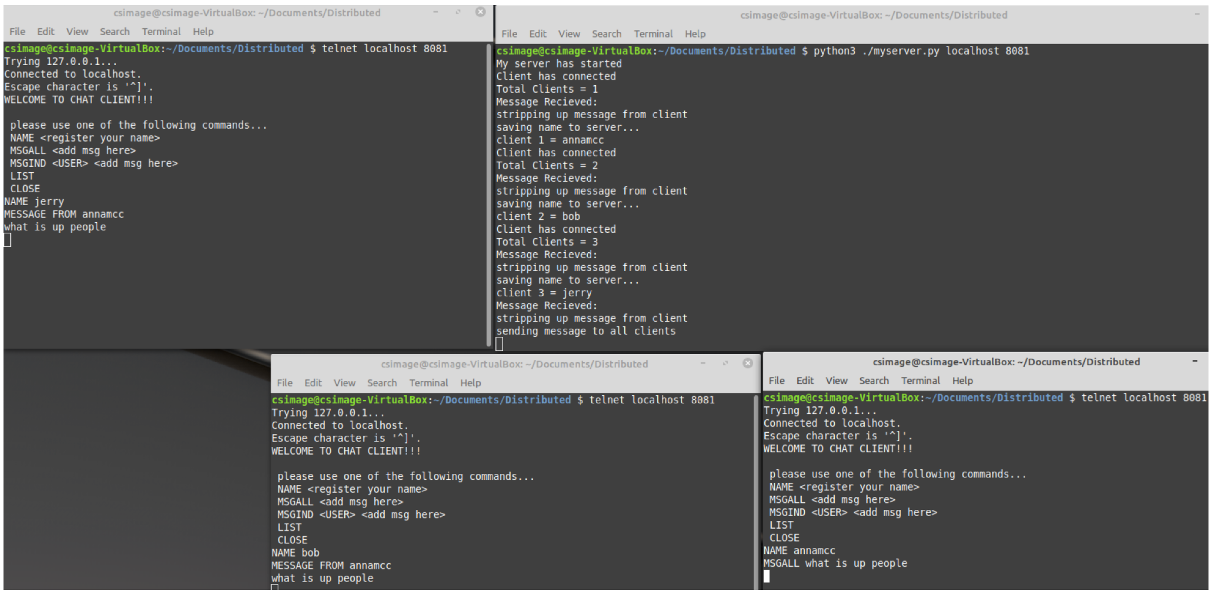


```
# save the name of the sender if the message is a 'send message type'
        if(command == self.c2 or command == self.c3):
# store senders name
            self.sender = self.client_store[socket]
# store senders address
            self.senderSocket = socket
# print('Command is ', command)
# print ('Message is ',parameter)
```

Next is code itself; it will cycle through all the sockets stored in client store and send the message out to all except from the sender itself.



```
# message all clients connected
        elif (command == self.c2):
            print("sending message to all clients")
            for x in self.client_store:
                if(self.client_store[x] != self.sender):
                    x.send((b"MESSAGE FROM "+ self.sender.encode()+b"\n"))
                    x.send(parameter.encode())
```



```
csimage@csimage-VirtualBox:~/Documents/Distributed $ telnet localhost 8081
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
WELCOME TO CHAT CLIENT!!!

 please use one of the following commands...
 NAME <register your name>
 MSGALL <add msg here>
 MSGIND <USER> <add msg here>
 LIST
 CLOSE
NAME jerry
MESSAGE FROM annamcc
what is up people
```

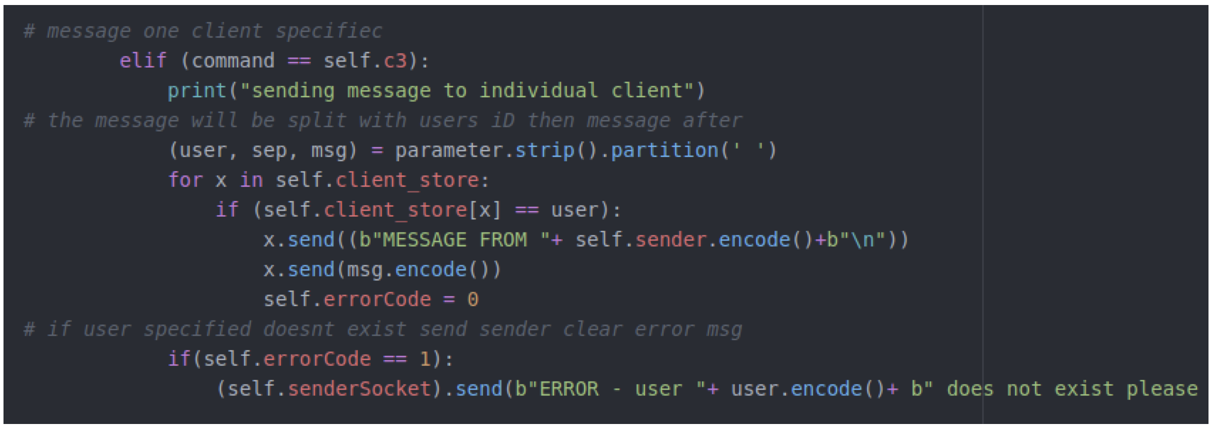
```
csimage@csimage-VirtualBox:~/Documents/Distributed $ python3 ./myserver.py localhost 8081
My server has started
Client has connected
Total Clients = 1
Message Recieved:
stripping up message from client
saving name to server...
client 1 = annamcc
Client has connected
Total Clients = 2
Message Recieved:
stripping up message from client
saving name to server...
client 2 = bob
Client has connected
Total Clients = 3
Message Recieved:
stripping up message from client
saving name to server...
client 3 = jerry
Message Recieved:
stripping up message from client
sending message to all clients
```

```
csimage@csimage-VirtualBox:~/Documents/Distributed $ telnet localhost 8081
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
WELCOME TO CHAT CLIENT!!!

 please use one of the following commands...
 NAME <register your name>
 MSGALL <add msg here>
 MSGIND <USER> <add msg here>
 LIST
 CLOSE
NAME bob
MESSAGE FROM annamcc
what is up people
```

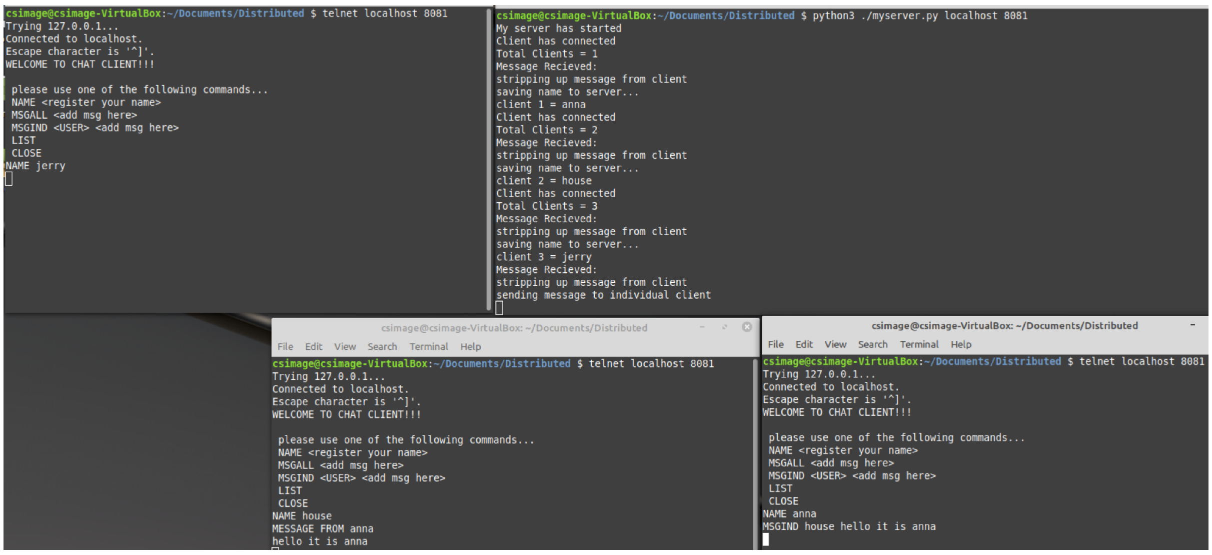
```
csimage@csimage-VirtualBox:~/Documents/Distributed $ telnet localhost 8081
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
WELCOME TO CHAT CLIENT!!!

 please use one of the following commands...
 NAME <register your name>
 MSGALL <add msg here>
 MSGIND <USER> <add msg here>
 LIST
 CLOSE
NAME annamcc
MSGALL what is up people
```

Message one client

To message one client I had to add some extra implementation. If the correct code is found, <MSGIND> then the message is split again taking the first word as the receivers name. It will cycle through all sockets in client store, find the name that matches the name given, send out the message, and update the error code. If the error code is not updated we know the user name specified contained an error. The server sends the 'sender' socket and error message to check the list of clients currently connected.



```python
# message one client specifiec
        elif (command == self.c3):
            print("sending message to individual client")
# the message will be split with users iD then message after
            (user, sep, msg) = parameter.strip().partition(' ')
            for x in self.client_store:
                if (self.client_store[x] == user):
                    x.send((b"MESSAGE FROM "+ self.sender.encode()+b"\n"))
                    x.send(msg.encode())
                    self.errorCode = 0
# if user specified doesnt exist send sender clear error msg
            if(self.errorCode == 1):
                (self.senderSocket).send(b"ERROR - user "+ user.encode()+ b" does not exist please
```



```
csimage@csimage-VirtualBox:~/Documents/Distributed $ telnet localhost 8081
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
WELCOME TO CHAT CLIENT!!!

 please use one of the following commands...
 NAME <register your name>
 MSGALL <add msg here>
 MSGIND <USER> <add msg here>
 LIST
 CLOSE
NAME jerry
```

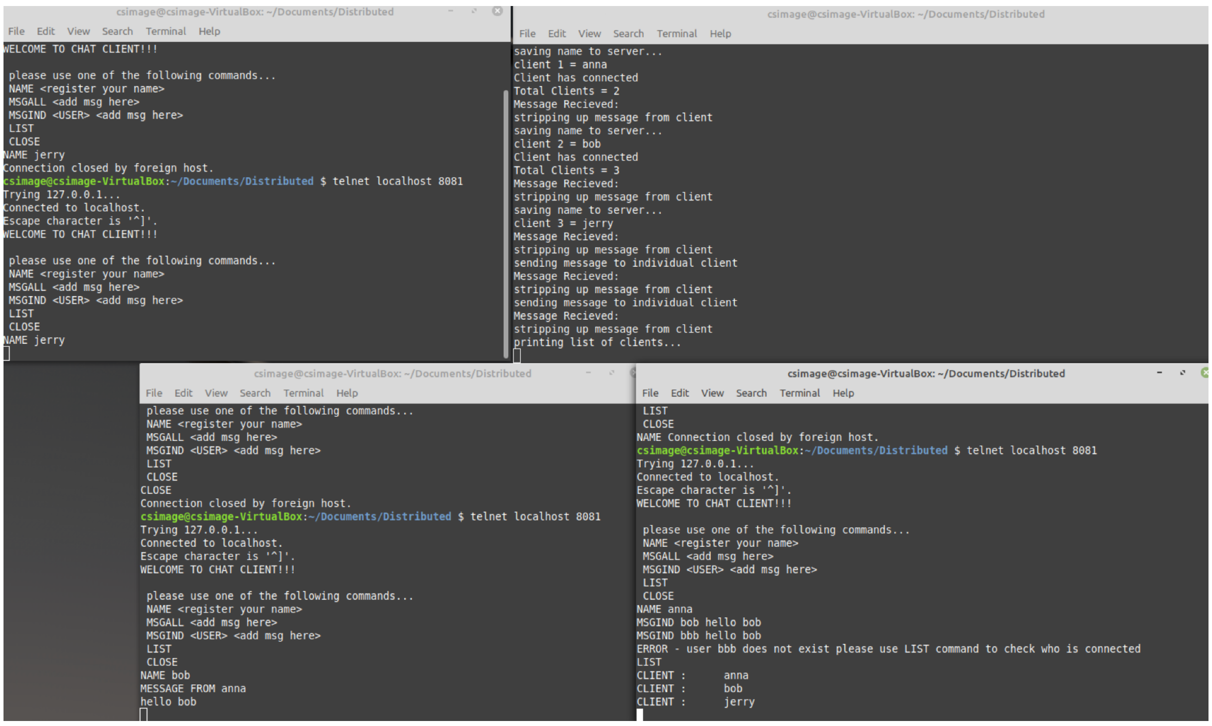
```
csimage@csimage-VirtualBox:~/Documents/Distributed $ python3 ./myserver.py localhost 8081
My server has started
Client has connected
Total Clients = 1
Message Recieved:
stripping up message from client
saving name to server...
client 1 = anna
Client has connected
Total Clients = 2
Message Recieved:
stripping up message from client
saving name to server...
client 2 = house
Client has connected
Total Clients = 3
Message Recieved:
stripping up message from client
saving name to server...
client 3 = jerry
Message Recieved:
stripping up message from client
sending message to individual client
```

csimage@csimage-VirtualBox: ~/Documents/Distributed
File Edit View Search Terminal Help

```
csimage@csimage-VirtualBox:~/Documents/Distributed $ telnet localhost 8081
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
WELCOME TO CHAT CLIENT!!!

 please use one of the following commands...
 NAME <register your name>
 MSGALL <add msg here>
 MSGIND <USER> <add msg here>
 LIST
 CLOSE
NAME house
MESSAGE FROM anna
hello it is anna
```

csimage@csimage-VirtualBox: ~/Documents/Distributed
File Edit View Search Terminal Help

```
csimage@csimage-VirtualBox:~/Documents/Distributed $ telnet localhost 8081
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
WELCOME TO CHAT CLIENT!!!

 please use one of the following commands...
 NAME <register your name>
 MSGALL <add msg here>
 MSGIND <USER> <add msg here>
 LIST
 CLOSE
NAME anna
MSGIND house hello it is anna
```



csimage@csimage-VirtualBox: ~/Documents/Distributed
File Edit View Search Terminal Help

```
WELCOME TO CHAT CLIENT!!!

 please use one of the following commands...
 NAME <register your name>
 MSGALL <add msg here>
 MSGIND <USER> <add msg here>
 LIST
 CLOSE
NAME jerry
Connection closed by foreign host.
csimage@csimage-VirtualBox:~/Documents/Distributed $ telnet localhost 8081
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
WELCOME TO CHAT CLIENT!!!

 please use one of the following commands...
 NAME <register your name>
 MSGALL <add msg here>
 MSGIND <USER> <add msg here>
 LIST
 CLOSE
NAME jerry
```

csimage@csimage-VirtualBox: ~/Documents/Distributed
File Edit View Search Terminal Help

```
saving name to server...
client 1 = anna
Client has connected
Total Clients = 2
Message Recieved:
stripping up message from client
saving name to server...
client 2 = bob
Client has connected
Total Clients = 3
Message Recieved:
stripping up message from client
saving name to server...
client 3 = jerry
Message Recieved:
stripping up message from client
sending message to individual client
Message Recieved:
stripping up message from client
sending message to individual client
Message Recieved:
stripping up message from client
printing list of clients...
```

csimage@csimage-VirtualBox: ~/Documents/Distributed
File Edit View Search Terminal Help

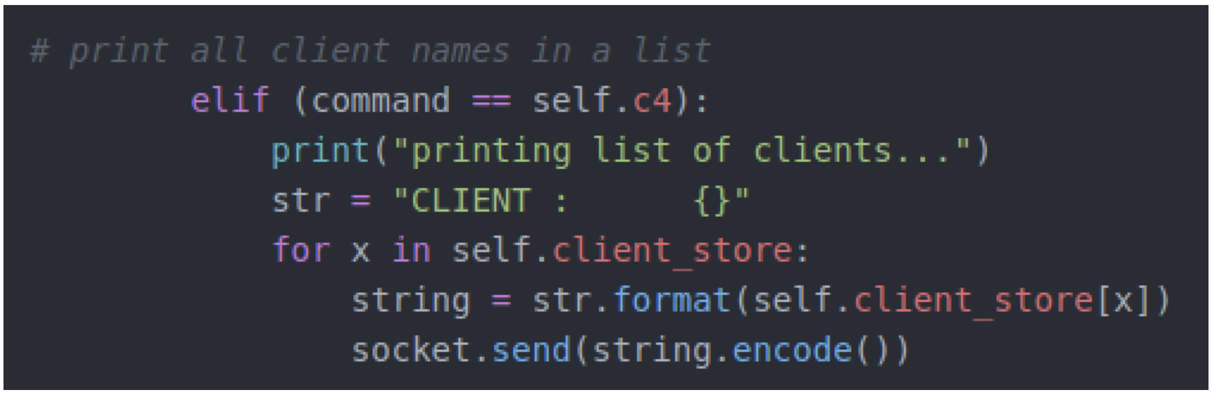
```
 please use one of the following commands...
 NAME <register your name>
 MSGALL <add msg here>
 MSGIND <USER> <add msg here>
 LIST
 CLOSE
CLOSE
Connection closed by foreign host.
csimage@csimage-VirtualBox:~/Documents/Distributed $ telnet localhost 8081
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
WELCOME TO CHAT CLIENT!!!

 please use one of the following commands...
 NAME <register your name>
 MSGALL <add msg here>
 MSGIND <USER> <add msg here>
 LIST
 CLOSE
NAME bob
MESSAGE FROM anna
hello bob
```

csimage@csimage-VirtualBox: ~/Documents/Distributed
File Edit View Search Terminal Help

```
 LIST
 CLOSE
NAME Connection closed by foreign host.
csimage@csimage-VirtualBox:~/Documents/Distributed $ telnet localhost 8081
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
WELCOME TO CHAT CLIENT!!!

 please use one of the following commands...
 NAME <register your name>
 MSGALL <add msg here>
 MSGIND <USER> <add msg here>
 LIST
 CLOSE
NAME anna
MSGIND bob hello bob
MSGIND bbb hello bob
ERROR - user bbb does not exist please use LIST command to check who is connected
LIST
CLIENT :      anna
CLIENT :      bob
CLIENT :      jerry
```

LIST

See list function in use in screenshot above. The list function I initially had printing the list of clients into the server terminal. I realised this is not very useful so adapted it to print into the clients window who requested to see the list.

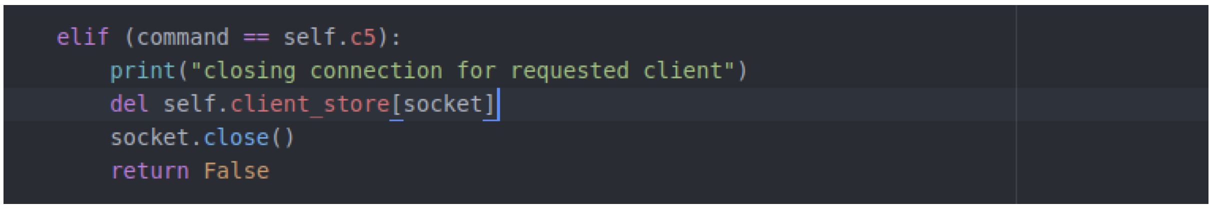


```
# print all client names in a list
        elif (command == self.c4):
            print("printing list of clients...")
            str = "CLIENT :       {}"
            for x in self.client_store:
                string = str.format(self.client_store[x])
                socket.send(string.encode())
```

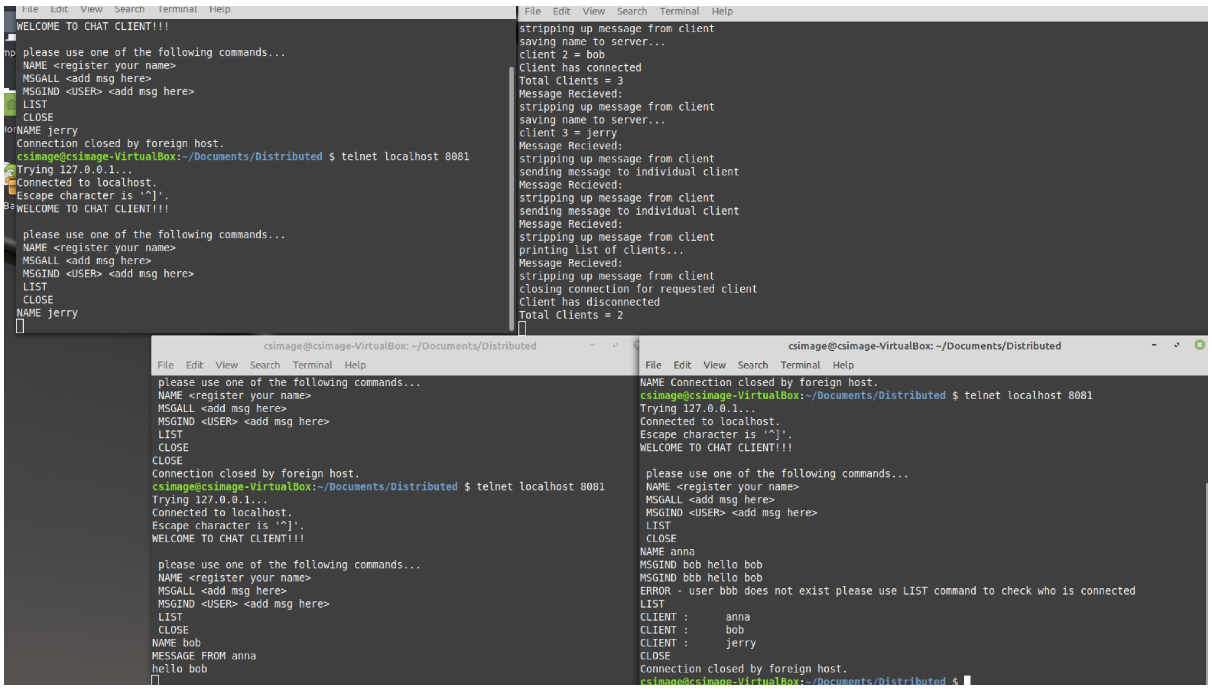
We cycles through all the sockets in the client store and print out there names in a neat format with encoding for safety.

CLOSE

Close the connection and return false to signify the client has left. (server wont keep checking for null client) Also delete the clients socket and name from server's dictionary. (see bottom right client types CLOSE)



```
    elif (command == self.c5):
        print("closing connection for requested client")
        del self.client_store[socket]
        socket.close()
        return False
```



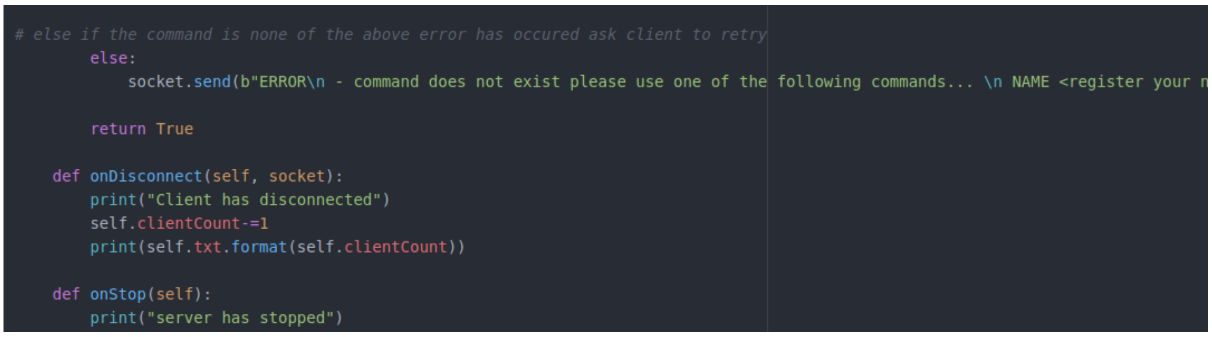
```
WELCOME TO CHAT CLIENT!!!

 please use one of the following commands...
 NAME <register your name>
 MSGALL <add msg here>
 MSGIND <USER> <add msg here>
 LIST
 CLOSE
NAME jerry
Connection closed by foreign host.
csimage@csimage-VirtualBox:~/Documents/Distributed $ telnet localhost 8081
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
WELCOME TO CHAT CLIENT!!!

 please use one of the following commands...
 NAME <register your name>
 MSGALL <add msg here>
 MSGIND <USER> <add msg here>
 LIST
 CLOSE
NAME jerry
```

```
stripping up message from client
saving name to server...
client 2 = bob
Client has connected
Total Clients = 3
Message Recieved:
stripping up message from client
saving name to server...
client 3 = jerry
Message Recieved:
stripping up message from client
sending message to individual client
Message Recieved:
stripping up message from client
sending message to individual client
Message Recieved:
stripping up message from client
printing list of clients...
Message Recieved:
stripping up message from client
closing connection for requested client
Client has disconnected
Total Clients = 2
```

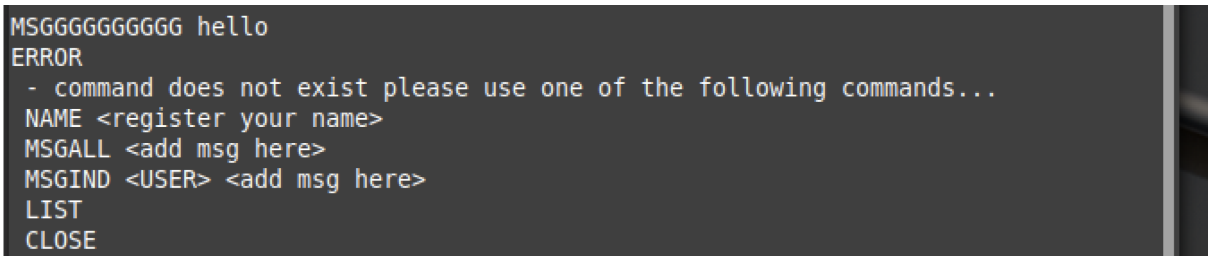
```
 please use one of the following commands...
 NAME <register your name>
 MSGALL <add msg here>
 MSGIND <USER> <add msg here>
 LIST
 CLOSE
CLOSE
Connection closed by foreign host.
csimage@csimage-VirtualBox:~/Documents/Distributed $ telnet localhost 8081
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
WELCOME TO CHAT CLIENT!!!

 please use one of the following commands...
 NAME <register your name>
 MSGALL <add msg here>
 MSGIND <USER> <add msg here>
 LIST
 CLOSE
NAME bob
MESSAGE FROM anna
hello bob
```

```
NAME Connection closed by foreign host.
csimage@csimage-VirtualBox:~/Documents/Distributed $ telnet localhost 8081
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
WELCOME TO CHAT CLIENT!!!

 please use one of the following commands...
 NAME <register your name>
 MSGALL <add msg here>
 MSGIND <USER> <add msg here>
 LIST
 CLOSE
NAME anna
MSGIND bob hello bob
MSGIND bbb hello bob
ERROR - user bbb does not exist please use LIST command to check who is connected
LIST
CLIENT :      anna
CLIENT :      bob
CLIENT :      jerry
CLOSE
Connection closed by foreign host.
csimage@csimage-VirtualBox:~/Documents/Distributed $
```

ERRORS
I have already covered how the MSGIND wrong user name error is covered. The other error would be if the command typed in was invalid. I use an ELSE as a last resort and reprint out the list of commands for this client to retry.



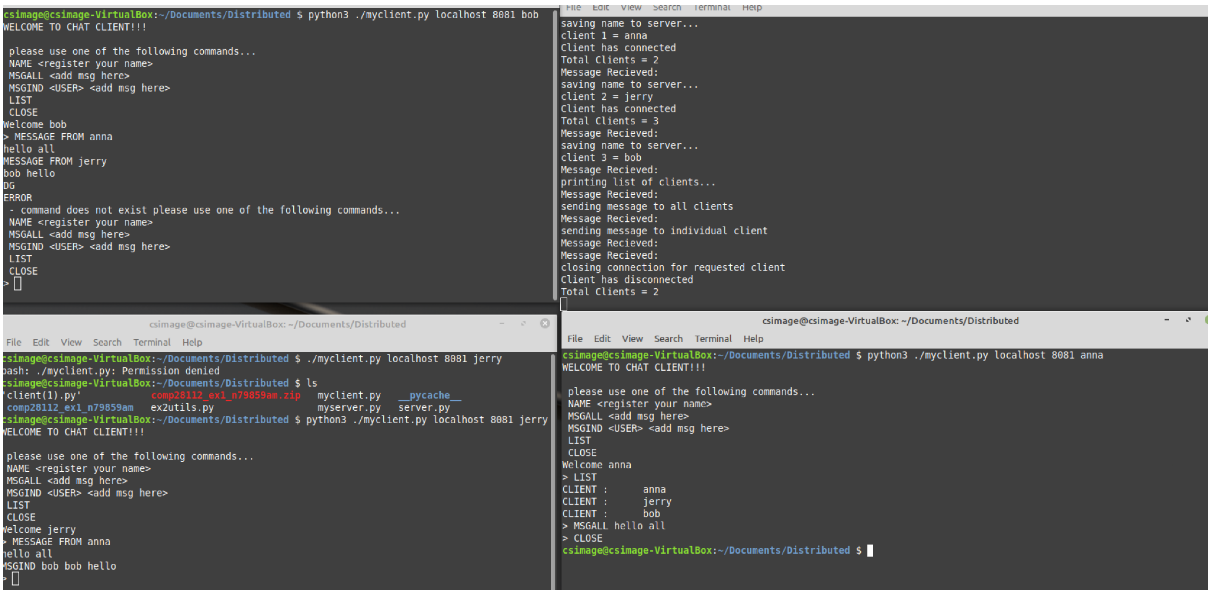
```
# else if the command is none of the above error has occured ask client to retry
        else:
            socket.send(b"ERROR\n - command does not exist please use one of the following commands... \n NAME <register your n

        return True

    def onDisconnect(self, socket):
        print("Client has disconnected")
        self.clientCount-=1
        print(self.txt.format(self.clientCount))

    def onStop(self):
        print("server has stopped")
```



```
MSGGGGGGGGGG hello
ERROR
 - command does not exist please use one of the following commands...
 NAME <register your name>
 MSGALL <add msg here>
 MSGIND <USER> <add msg here>
 LIST
 CLOSE
```

Task 2.6

The only lines of code i added here were
print (message) to the onMessage method. This is to display the output that was sent by server.
client.send(command.encode()) this is to send the clients request to the server.
I also added a welcome message for good measure.

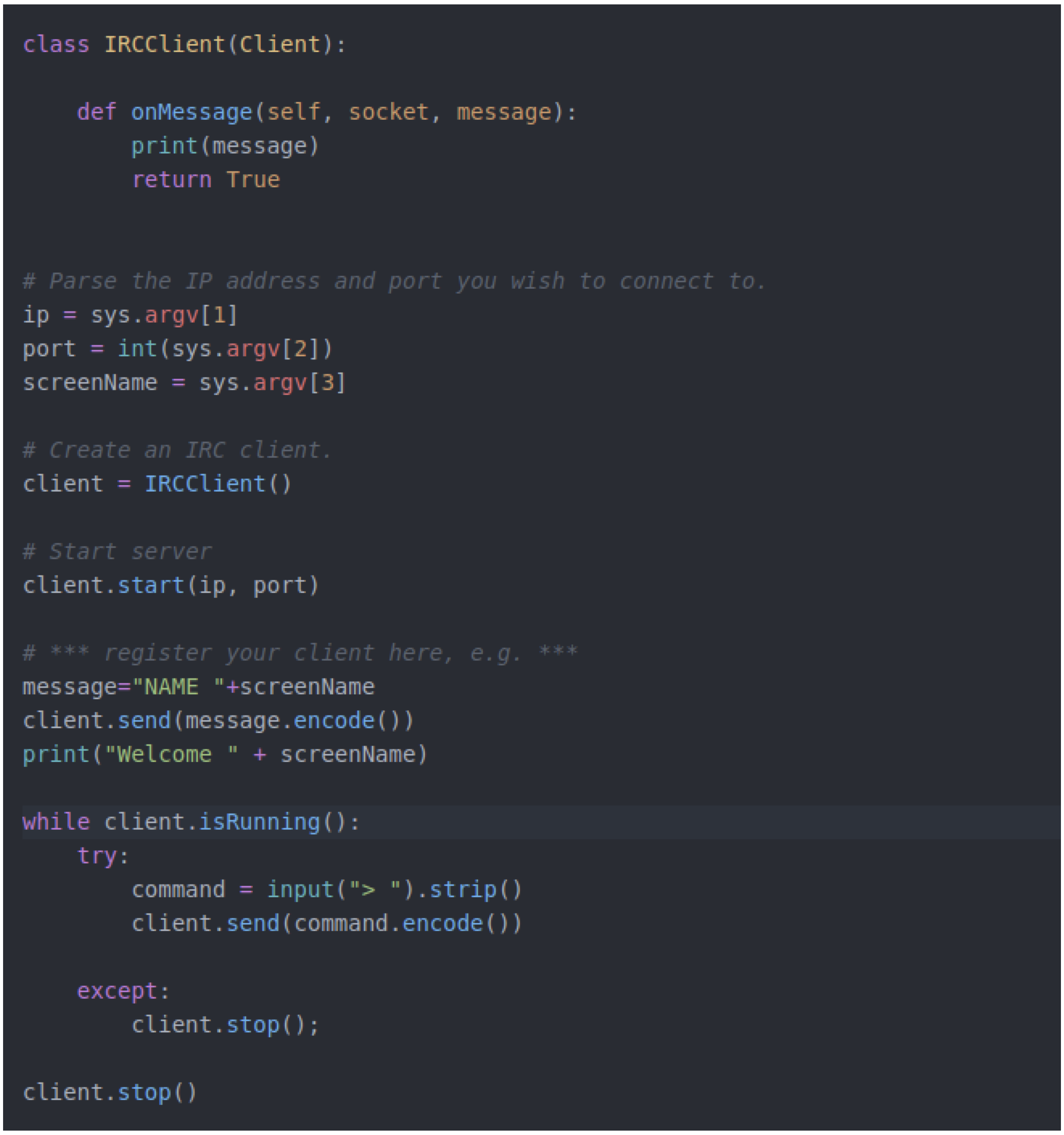


```
csimage@csimage-VirtualBox:~/Documents/Distributed $ python3 ./myclient.py localhost 8081 bob
WELCOME TO CHAT CLIENT!!!

 please use one of the following commands...
 NAME <register your name>
 MSGALL <add msg here>
 MSGIND <USER> <add msg here>
 LIST
 CLOSE
Welcome bob
> MESSAGE FROM anna
hello all
MESSAGE FROM jerry
bob hello
DG
ERROR
 - command does not exist please use one of the following commands...
 NAME <register your name>
 MSGALL <add msg here>
 MSGIND <USER> <add msg here>
 LIST
 CLOSE
> 
```

```
saving name to server...
client 1 = anna
Client has connected
Total Clients = 2
Message Recieved:
saving name to server...
client 2 = jerry
Client has connected
Total Clients = 3
Message Recieved:
saving name to server...
client 3 = bob
Message Recieved:
printing list of clients...
Message Recieved:
sending message to all clients
Message Recieved:
sending message to individual client
Message Recieved:
Message Recieved:
closing connection for requested client
Client has disconnected
Total Clients = 2
```

```
csimage@csimage-VirtualBox:~/Documents/Distributed $ ./myclient.py localhost 8081 jerry
bash: ./myclient.py: Permission denied
csimage@csimage-VirtualBox:~/Documents/Distributed $ ls
'client(1).py'          comp28112_ex1_n79859am.zip   myclient.py   __pycache__
 comp28112_ex1_n79859am   ex2utils.py                  myserver.py   server.py
csimage@csimage-VirtualBox:~/Documents/Distributed $ python3 ./myclient.py localhost 8081 jerry
WELCOME TO CHAT CLIENT!!!

 please use one of the following commands...
 NAME <register your name>
 MSGALL <add msg here>
 MSGIND <USER> <add msg here>
 LIST
 CLOSE
Welcome jerry
> MESSAGE FROM anna
hello all
MSGIND bob bob hello
> 
```

```
csimage@csimage-VirtualBox:~/Documents/Distributed $ python3 ./myclient.py localhost 8081 anna
WELCOME TO CHAT CLIENT!!!

 please use one of the following commands...
 NAME <register your name>
 MSGALL <add msg here>
 MSGIND <USER> <add msg here>
 LIST
 CLOSE
Welcome anna
> LIST
CLIENT :      anna
CLIENT :      jerry
CLIENT :      bob
> MSGALL hello all
> CLOSE
csimage@csimage-VirtualBox:~/Documents/Distributed $ 
```



```python
class IRCClient(Client):

    def onMessage(self, socket, message):
        print(message)
        return True


# Parse the IP address and port you wish to connect to.
ip = sys.argv[1]
port = int(sys.argv[2])
screenName = sys.argv[3]

# Create an IRC client.
client = IRCClient()

# Start server
client.start(ip, port)

# *** register your client here, e.g. ***
message="NAME "+screenName
client.send(message.encode())
print("Welcome " + screenName)

while client.isRunning():
    try:
        command = input("> ").strip()
        client.send(command.encode())

    except:
        client.stop();

client.stop()
```

Task 2.7

Is the protocol you created for this exercise stateless?

A stateful server is one which stores session information in order to identify clients over multiple subsequent requests. In this sense our server would be considered stateful instead of stateless as we store a dictionary of (key, value) pairs (socket , userName).

When the client changes state ie. disconnected/connected. The server stores this information through the client count.

In the opposite way a stateless server would receive a request, and send back the information from where it received the request. Ours is not like this as the name is stored between requests; therefore each request is not independent; if you want to send a message to someone you have to access the client store dictionary which has already been stored. One request **relies** on a previous request of having a socket related to a screen name.

This is not particularly relevant for our server as we need to store information in order for a chat client to work, there is no way around this! We are also storing the bare minimum needed only socket and name. There are however servers that could be stateless when they are in fact stateful and storing our information unnecessarily. We also perform safe practice and remove the clients socket and screenName from the dictionary when the disconnect request is carried out.

Describe how the current implementation could be extended/improved to allow subgroups of users to be created and messages to be sent only to members of a subgroup.

We can use the idea of a dictionary with key value pairs.
We will have method
<MAKEGROUP> <groupName> <all the users screen names we want to include>
This method will create the dictionary under self.groupName will then cycle through the client store, match the names up, and add the key value pairs (socket, name) to this new dictionary.
A second method
<SENDGROUP> <groupName> <the message here>
This method will locate the group dictionary in the server, cycle through all the sockets and send the message out to each one.