

Socket Programming: Group chat room mini project

Developed by:

6030226121 Thaworn Kangwansinghanat 28

6030609921 Suchut Sapsathien 28

6031045021 Bhoomie Suitcharit 28

6031061021 Anan Methasate 28

Github repo: https://github.com/daidew/group_chat_web_socket

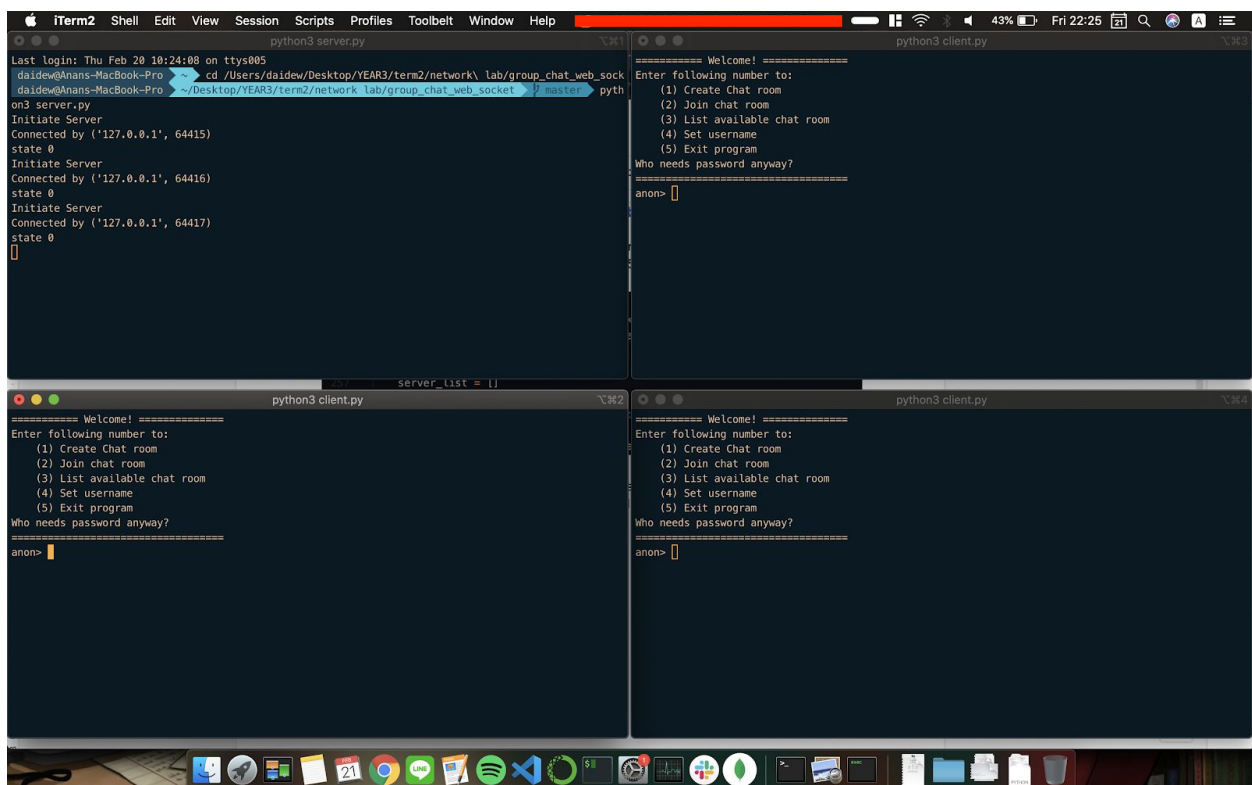
How to use it?

Before running, configure the number of clients you want your server to be able to handle by specifying in `s.listen(n)` and `for i in range(n)` in line 256 and 260 respectively. By default, the server will be able to accept a maximum of 5 clients.

```
252 if __name__ == '__main__':
253     s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
254     s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
255     s.bind(('', 50006))
256     s.listen(5)
257     server_list = []
258     # conn, addr = s.accept()
259     # server_list.append(server(conn, addr))
260     for i in range(5):
261         threading.Thread(target=server, args=(s.accept(),)).start()
```

Then, start the server by executing command `python3 server.py` (in project directory) for initiating server.

Start the client by executing `python3 client.py`. For this prototype, we have not implemented ngrok tunneling yet, so it can only be used locally (local network). As an example, I have created 3 clients and 1 server as shown below. (top left terminal is the host server)

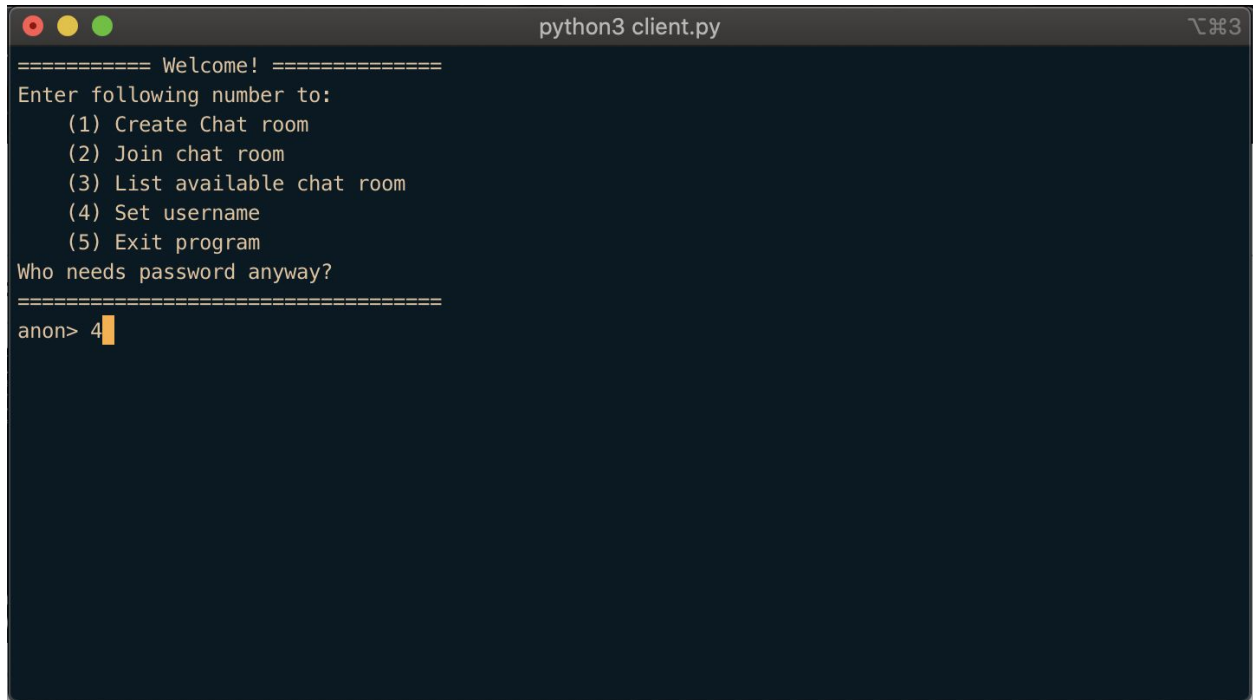


The screenshot displays four terminal windows from an iTerm2 application. The top-left window, titled 'python3 server.py', shows the server's execution. It logs the start time, the directory path, and the command to run the server. It then shows three successful connections from clients at IP addresses 127.0.0.1 with ports 64415, 64416, and 64417. The top-right window, titled 'python3 client.py', shows a client's menu with options: (1) Create Chat room, (2) Join chat room, (3) List available chat room, (4) Set username, and (5) Exit program. The client has entered 'anon' for the username. The bottom-left and bottom-right windows also show the same client menu and 'anon' username input. The macOS dock is visible at the bottom of the screen.

After starting the client program, we will see the main menu rendered through the terminal screen. For clients, there are 5 options available;

1. Create chat room
2. Join chat room
3. List available chat room
4. Set username (default is "anon")
5. Exit program

To set username, type `4` then enter.



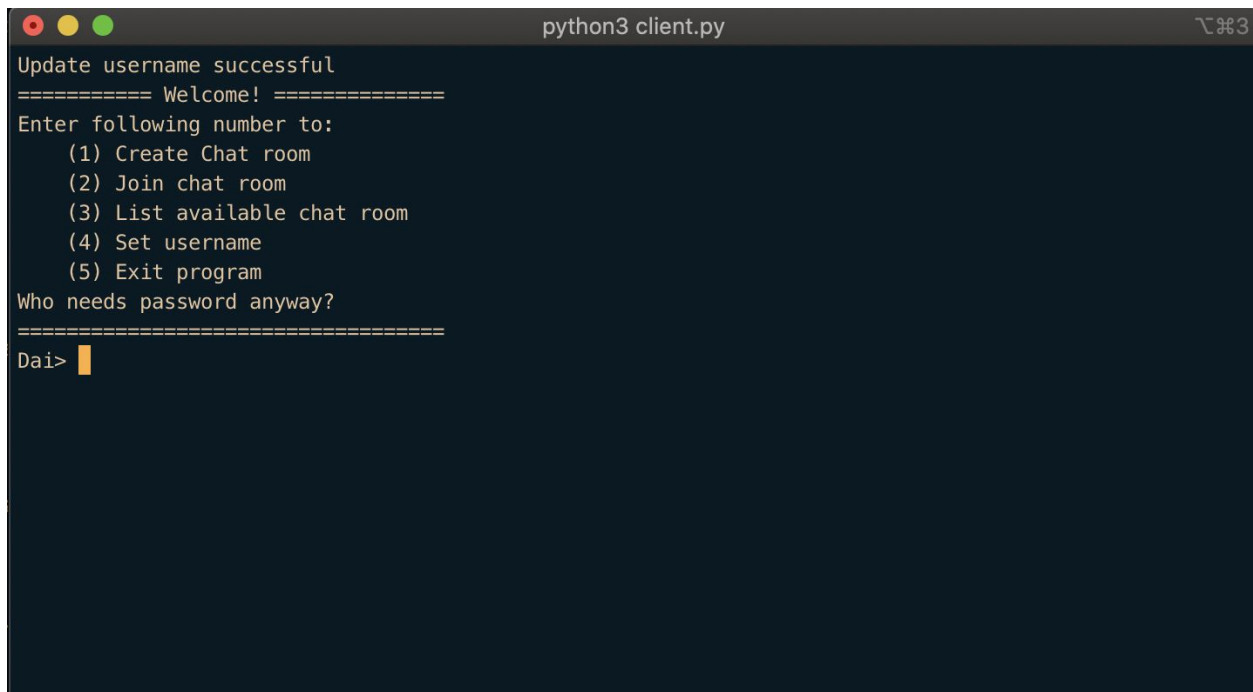
```
python3 client.py
===== Welcome! =====
Enter following number to:
  (1) Create Chat room
  (2) Join chat room
  (3) List available chat room
  (4) Set username
  (5) Exit program
Who needs password anyway?
=====
anon> 4
```

When prompted, enter your desired name and enter.



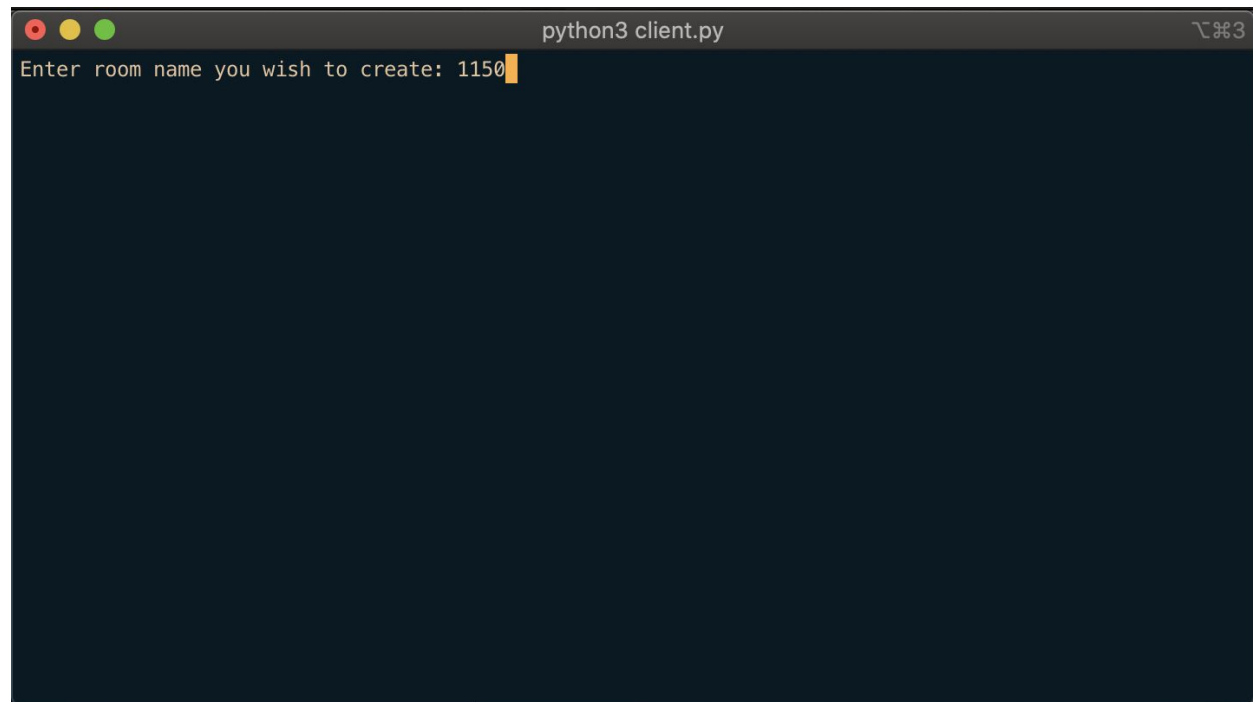
```
python3 client.py
Please enter your new username: Dai
```

If the process went successfully, you will be able to see your handle changed from anon to your newly entered name as shown below.

A terminal window titled 'python3 client.py' with a dark background. It displays the following text: 'Update username successful', '==== Welcome! =====', 'Enter following number to:', a numbered list (1) Create Chat room, (2) Join chat room, (3) List available chat room, (4) Set username, (5) Exit program, 'Who needs password anyway?', '====', and a prompt 'Dai>' with a yellow cursor.

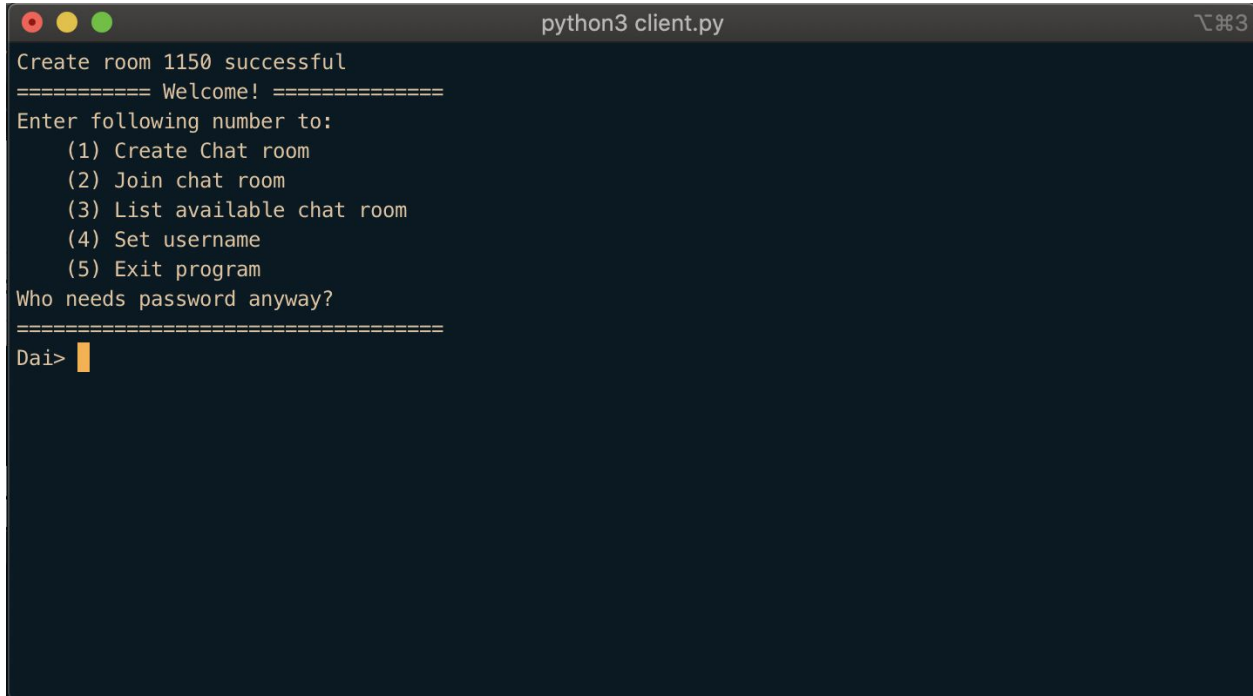
```
python3 client.py
Update username successful
==== Welcome! =====
Enter following number to:
  (1) Create Chat room
  (2) Join chat room
  (3) List available chat room
  (4) Set username
  (5) Exit program
Who needs password anyway?
=====
Dai> |
```

Next, we will try to create a new chat room by typing `1` and enter.

A terminal window titled 'python3 client.py' with a dark background. It displays the text 'Enter room name you wish to create: 1150' with a yellow cursor at the end of the input.

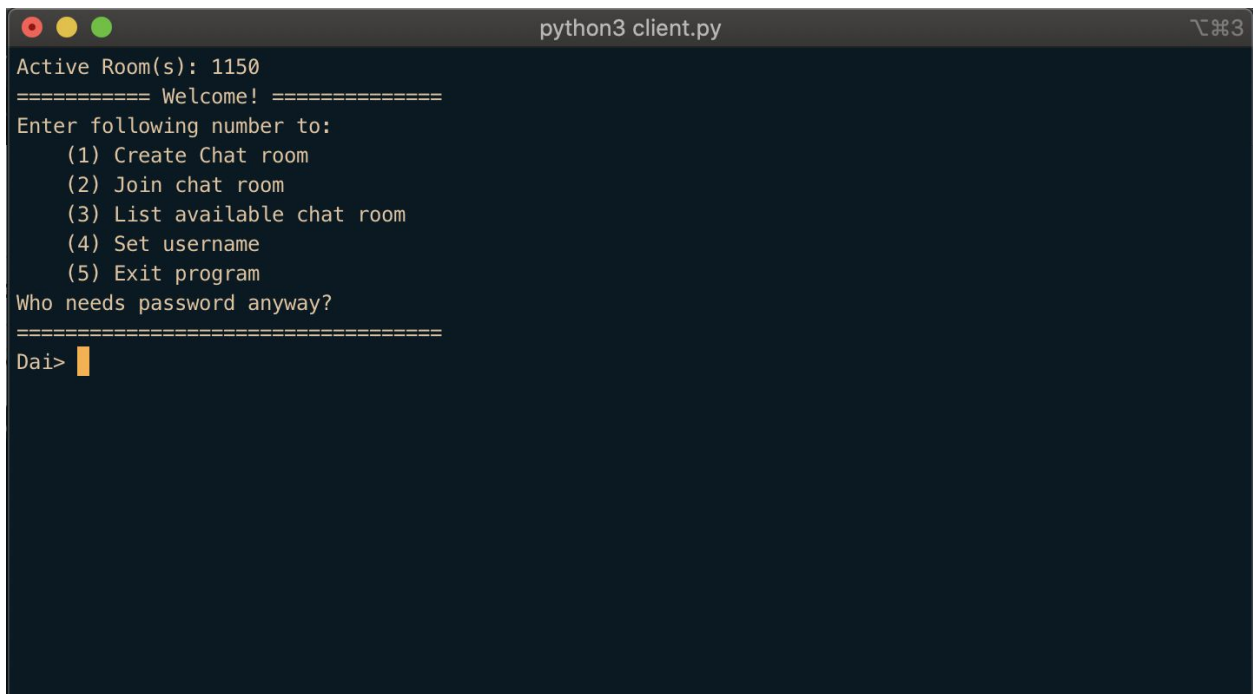
```
python3 client.py
Enter room name you wish to create: 1150|
```

If successful, the message will be returned as `Create room XXXX successful`.



```
python3 client.py
Create room 1150 successful
===== Welcome! =====
Enter following number to:
    (1) Create Chat room
    (2) Join chat room
    (3) List available chat room
    (4) Set username
    (5) Exit program
Who needs password anyway?
=====
Dai> 
```

We can also list all created chat rooms by typing `3` and enter.



```
python3 client.py
Active Room(s): 1150
===== Welcome! =====
Enter following number to:
    (1) Create Chat room
    (2) Join chat room
    (3) List available chat room
    (4) Set username
    (5) Exit program
Who needs password anyway?
=====
Dai> 
```

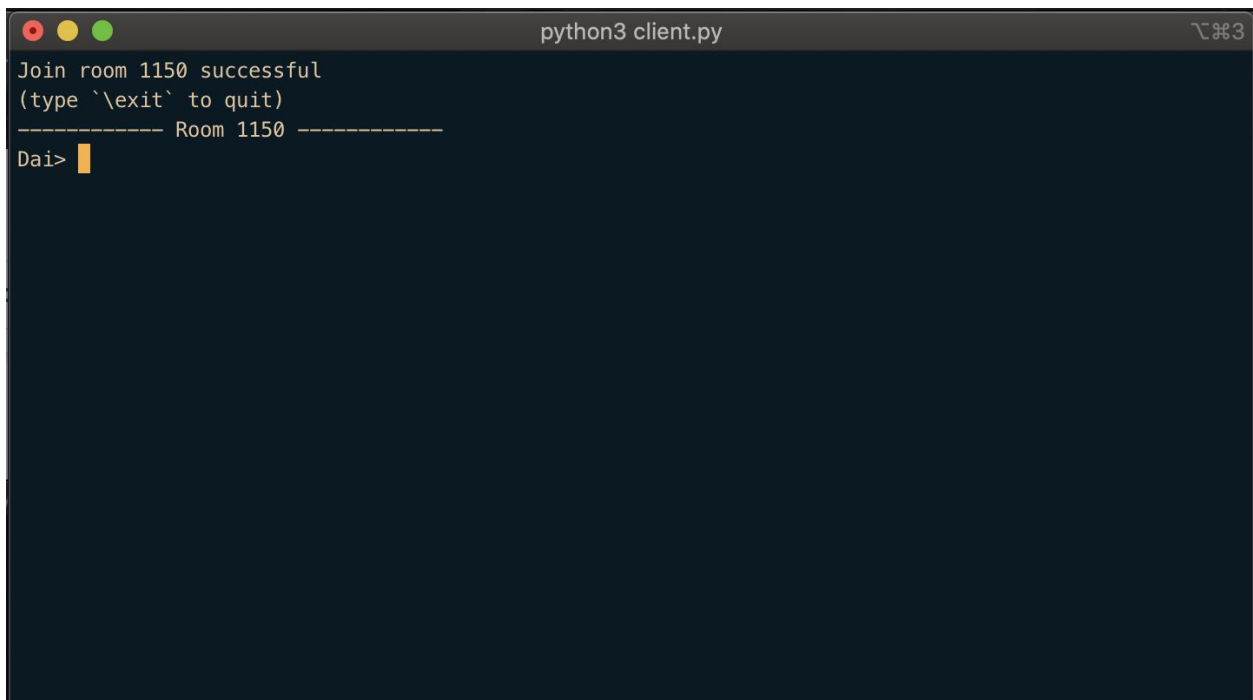
If we want to join a chat room, type `2` and enter.

Then enter a room number you wish to join.



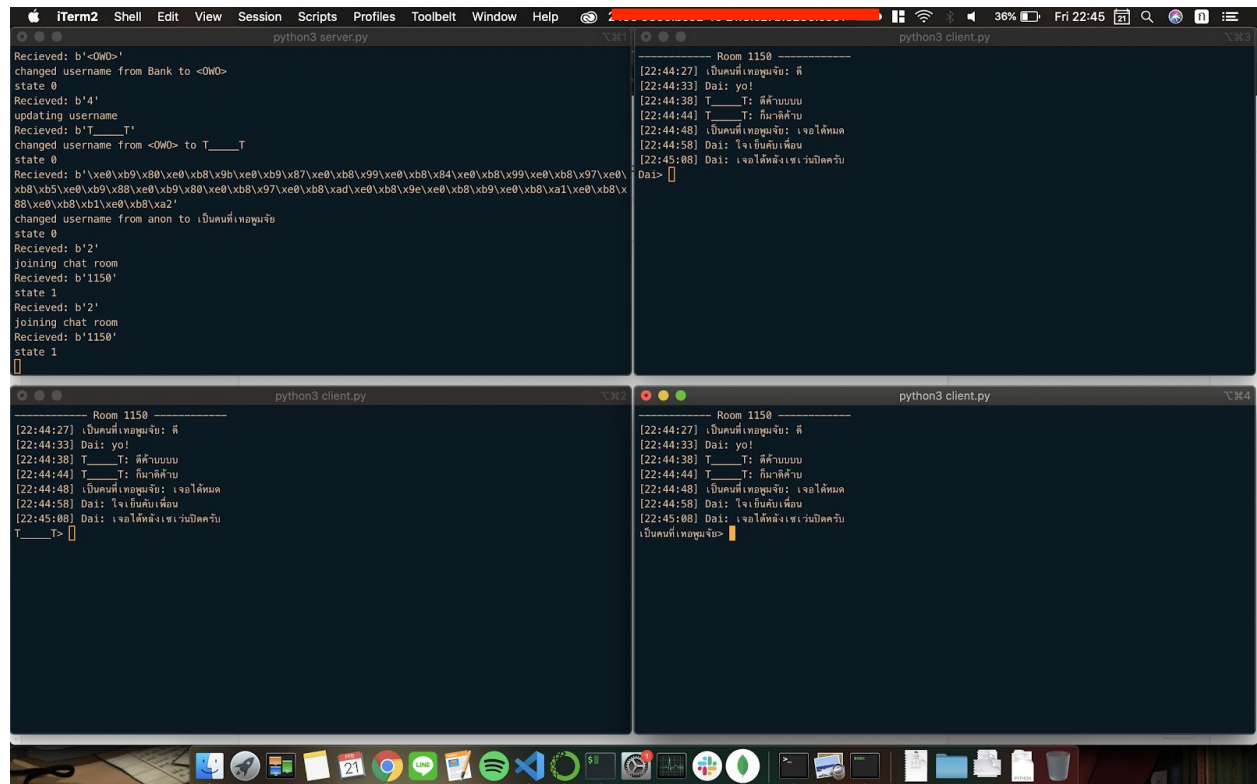
```
python3 client.py
Please enter room number: 1150
```

If successful, the screen will now look something like below. You are now in a chat room.
Hooray!



```
python3 client.py
Join room 1150 successful
(type '\exit' to quit)
----- Room 1150 -----
Dai>
```

This is an example of a simulated chat scenario where 3 people are talking to each other. Each terminal screen is updated in real time. (to exit chat room, type `exit`)



```
python3 server.py
Recieved: b'<0w0>'
changed username from Bank to <0w0>
state 0
Recieved: b'4'
updating username
Recieved: b'T____T'
changed username from <0w0> to T____T
state 0
Recieved: b'\xe0\xb9\x80\xe0\xb8\x9b\xe0\xb9\x87\xe0\xb8\x99\xe0\xb8\x84\xe0\xb8\x99\xe0\xb8\x97\xe0\x
xb8\xb5\xe0\xb9\x88\xe0\xb9\x80\xe0\xb8\x97\xe0\xb8\xad\xe0\xb8\x9e\xe0\xb9\xe0\xb8\xad\xe0\xb8\x
88\xe0\xb8\xb1\xe0\xb8\xa2'
changed username from anon to เป็นคนที่เพ่ขุญจิง
state 0
Recieved: b'2'
joining chat room
Recieved: b'1150'
state 1
Recieved: b'2'
joining chat room
Recieved: b'1150'
state 1

python3 client.py
Room 1150 -----
[22:44:27] เป็นคนที่เพ่ขุญจิง: คี
[22:44:33] Dai: yo!
[22:44:38] T____T: คีค่านนน
[22:44:44] T____T: คีมาคิค่าน
[22:44:48] เป็นคนที่เพ่ขุญจิง: เจอได้หนค
[22:44:58] Dai: โจเย็นคึนเพื่อน
[22:45:08] Dai: เจอได้คึงเซเว่นคึคัรบ
T____T>

python3 client.py
Room 1150 -----
[22:44:27] เป็นคนที่เพ่ขุญจิง: คี
[22:44:33] Dai: yo!
[22:44:38] T____T: คีค่านนน
[22:44:44] T____T: คีมาคิค่าน
[22:44:48] เป็นคนที่เพ่ขุญจิง: เจอได้หนค
[22:44:58] Dai: โจเย็นคึนเพื่อน
[22:45:08] Dai: เจอได้คึงเซเว่นคึคัรบ
เป็นคนที่เพ่ขุญจิง>

python3 client.py
Room 1150 -----
[22:44:27] เป็นคนที่เพ่ขุญจิง: คี
[22:44:33] Dai: yo!
[22:44:38] T____T: คีค่านนน
[22:44:44] T____T: คีมาคิค่าน
[22:44:48] เป็นคนที่เพ่ขุญจิง: เจอได้หนค
[22:44:58] Dai: โจเย็นคึนเพื่อน
[22:45:08] Dai: เจอได้คึงเซเว่นคึคัรบ
เป็นคนที่เพ่ขุญจิง>
```

Algorithm

I have used a state machine to track the state of each client, where the session is maintained in different function threads in a server side. The client is almost like an input/output relay. The server does rendering and processing like a 1st tier architecture.

Code structure design is OOP based.

Problems Encountered

My major problem was to deal with input handling, where the client must receive the data from both stdin and socket at the same time, so some kind of polling technique was used to process input. The second problem was the screen is a terminal, so when clearing the screen (to update chat), the typed input will also get cleared as well. I have tried to alleviate this by capturing input without waiting for newline characters, but till this day I have not managed to be able to fix it.