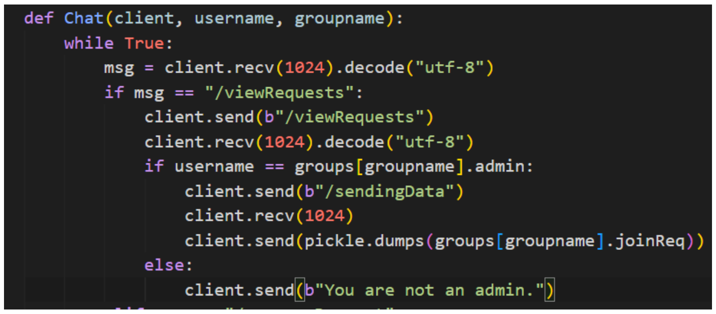
Group 7 - Lab SKJ CSA:

1. Almas Mirzandi Ramadhan - 21/472900/PA/20357
2. M. Rafi Syafrinaldi - 21/472772/PA/20335
3. Nashifa Ammara Fawziya Muchtar - 21/472666/PA/20319
4. Shreshta Adyaksa Hardono - 21/478740/PA/20767

To make it easier to store all the needed files, we uploaded everything to github: github-group7-csa

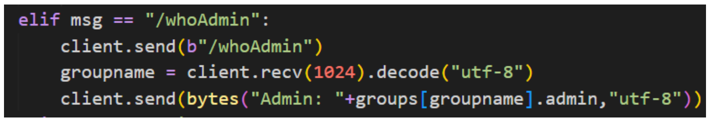
Server side:

1. **Server.py**
   - **Chat function:**
     This function handles all the messages being sent to/received from the server. Messages are sent in the form of bytes and so we used .decode() to decode it back into a string so the users are able to communicate with each other. There are also several trigger words that allows the users to perform a specific action such as playing rock paper scissors, viewing who the admin is, and kicking a user out of the group (for admins only). We also used subprocess to open another .py file to run our game.



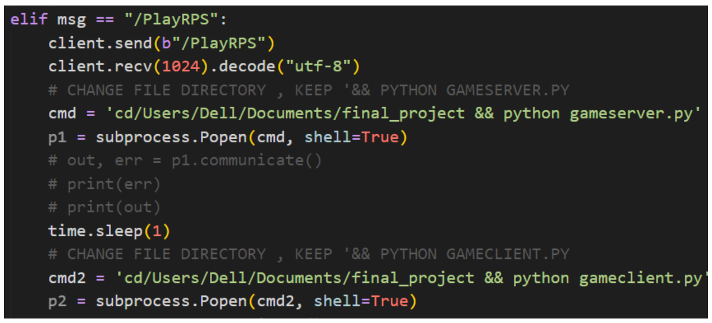
```python
def Chat(client, username, groupname):
    while True:
        msg = client.recv(1024).decode("utf-8")
        if msg == "/viewRequests":
            client.send(b"/viewRequests")
            client.recv(1024).decode("utf-8")
            if username == groups[groupname].admin:
                client.send(b"/sendingData")
                client.recv(1024)
                client.send(pickle.dumps(groups[groupname].joinReq))
            else:
                client.send(b"You are not an admin.")
```

The if statement checks whether or not the user is an admin and can use admin privileges



```python
elif msg == "/whoAdmin":
    client.send(b"/whoAdmin")
    groupname = client.recv(1024).decode("utf-8")
    client.send(bytes("Admin: "+groups[groupname].admin,"utf-8"))
```

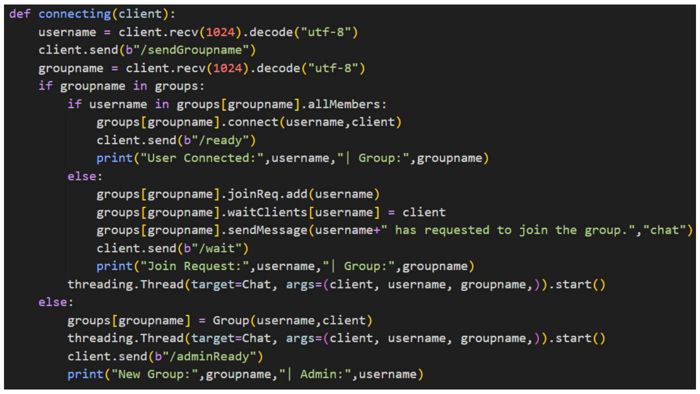
The phrase "/whoAdmin" sets off a trigger to check and return the username of the admin



```python
elif msg == "/PlayRPS":
    client.send(b"/PlayRPS")
    client.recv(1024).decode("utf-8")
    # CHANGE FILE DIRECTORY , KEEP '&& PYTHON GAMESERVER.PY
    cmd = 'cd/Users/Dell/Documents/final_project && python gameserver.py'
    p1 = subprocess.Popen(cmd, shell=True)
    # out, err = p1.communicate()
    # print(err)
    # print(out)
    time.sleep(1)
    # CHANGE FILE DIRECTORY , KEEP '&& PYTHON GAMECLIENT.PY
    cmd2 = 'cd/Users/Dell/Documents/final_project && python gameclient.py'
    p2 = subprocess.Popen(cmd2, shell=True)
```

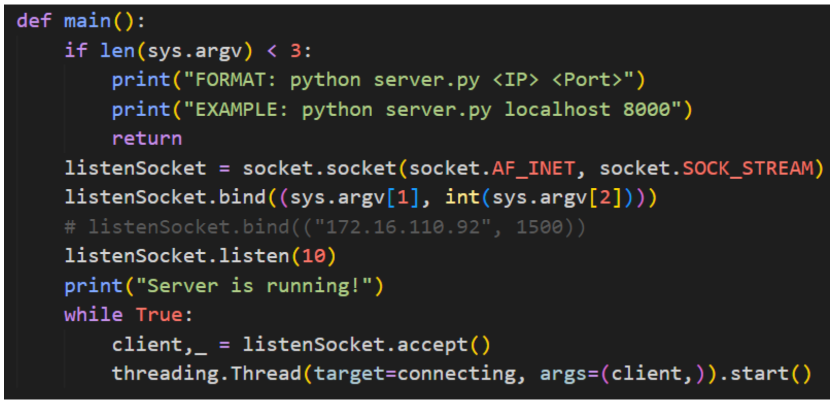
We use subprocess to open our .py files for our game. Unfortunately we were not able to figure out a way so we didn't have to access through our local directory.

- **Connecting function:**
  This function checks whether incoming users want to enter the same group/room or not. If a user wants to join an existing room, they must first be accepted by the admin (the first user to enter the room). If they are accepted by the admin, the user is then added to the room and can start chatting with other users in the room.



```python
def connecting(client):
    username = client.recv(1024).decode("utf-8")
    client.send(b"/sendGroupname")
    groupname = client.recv(1024).decode("utf-8")
    if groupname in groups:
        if username in groups[groupname].allMembers:
            groups[groupname].connect(username,client)
            client.send(b"/ready")
            print("User Connected:",username,"| Group:",groupname)
        else:
            groups[groupname].joinReq.add(username)
            groups[groupname].waitClients[username] = client
            groups[groupname].sendMessage(username+" has requested to join the group.","chat")
            client.send(b"/wait")
            print("Join Request:",username,"| Group:",groupname)
        threading.Thread(target=Chat, args=(client, username, groupname,)).start()
    else:
        groups[groupname] = Group(username,client)
        threading.Thread(target=Chat, args=(client, username, groupname,)).start()
        client.send(b"/adminReady")
        print("New Group:",groupname,"| Admin:",username)
```

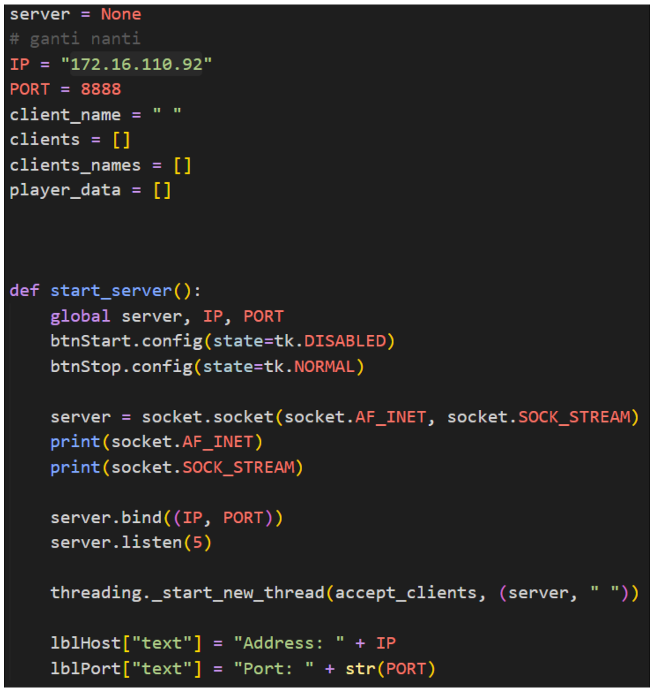
- **Main function:**
  The main function basically just takes the ip address and port that the user wants to use by asking the users to input their preferred ip address and port. The function then sends the ip address and port information to the rest of the functions to make our program run



```python
def main():
    if len(sys.argv) < 3:
        print("FORMAT: python server.py <IP> <Port>")
        print("EXAMPLE: python server.py localhost 8000")
        return
    listenSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    listenSocket.bind((sys.argv[1], int(sys.argv[2])))
    # listenSocket.bind(("172.16.110.92", 1500))
    listenSocket.listen(10)
    print("Server is running!")
    while True:
        client,_ = listenSocket.accept()
        threading.Thread(target=connecting, args=(client,)).start()
```

2. **gameServer.py**

This file basically sets up a server to let two users play a game of rock paper scissors and announces the winner after a set amount of rounds (in our case, initialised to 3 rounds). We also used a basic gui

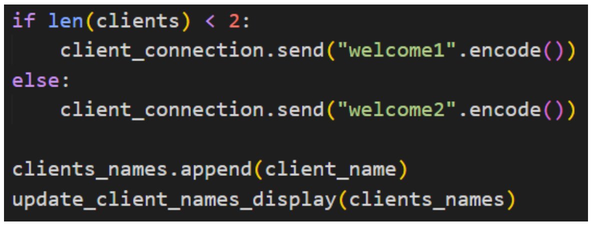


```
server = None
# ganti nanti
IP = "172.16.110.92"
PORT = 8888
client_name = " "
clients = []
clients_names = []
player_data = []



def start_server():
    global server, IP, PORT
    btnStart.config(state=tk.DISABLED)
    btnStop.config(state=tk.NORMAL)

    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    print(socket.AF_INET)
    print(socket.SOCK_STREAM)

    server.bind((IP, PORT))
    server.listen(5)

    threading._start_new_thread(accept_clients, (server, " "))

    lblHost["text"] = "Address: " + IP
    lblPort["text"] = "Port: " + str(PORT)
```

Here, we set the ip address and the port the match will be hosted. After we set the ip and port number, we wait for two users to join before starting the game. Once two users join, the game starts

---

The snippet below makes sure only two users are connected at a time



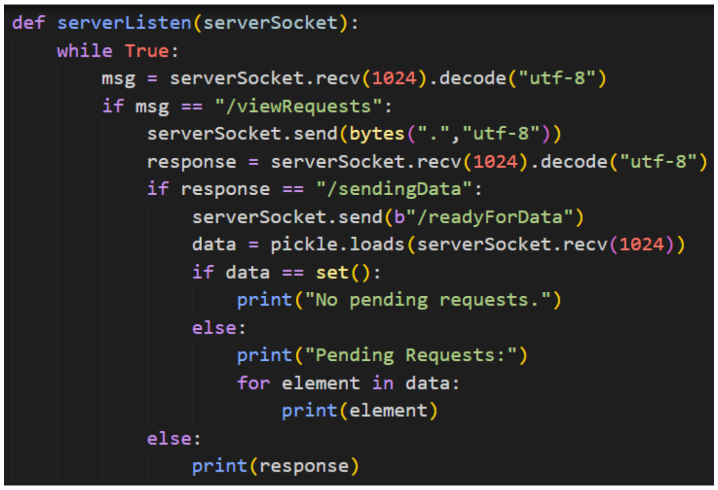
```
if len(clients) < 2:
    client_connection.send("welcome1".encode())
else:
    client_connection.send("welcome2".encode())

clients_names.append(client_name)
update_client_names_display(clients_names)
```

Client side:

1. **client.py**

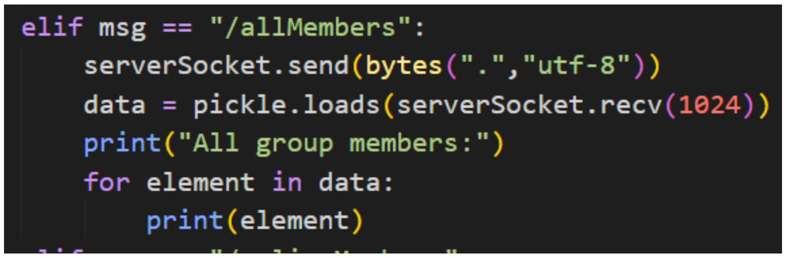
- **serverListen function:**

  This function is responsible for passing decisions and commands between the client and the server. Essentially, the way it works is more or less the same as server.py but for the client side. It makes sure the trigger words are properly coded so each trigger does the required action.



```
def serverListen(serverSocket):
    while True:
        msg = serverSocket.recv(1024).decode("utf-8")
        if msg == "/viewRequests":
            serverSocket.send(bytes(".","utf-8"))
            response = serverSocket.recv(1024).decode("utf-8")
            if response == "/sendingData":
                serverSocket.send(b"/readyForData")
                data = pickle.loads(serverSocket.recv(1024))
                if data == set():
                    print("No pending requests.")
                else:
                    print("Pending Requests:")
                    for element in data:
                        print(element)
            else:
                print(response)
```

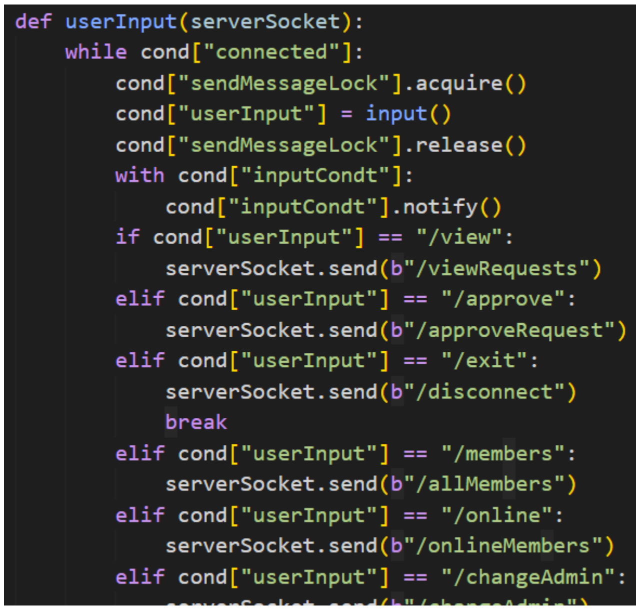
In the above snippet, the “/viewRequest” trigger is triggered. It then checks whether there are any pending requests that the admin needs to approve of.



```
elif msg == "/allMembers":
    serverSocket.send(bytes(".","utf-8"))
    data = pickle.loads(serverSocket.recv(1024))
    print("All group members:")
    for element in data:
        print(element)
```

In this snippet, the “/allMembers” trigger is triggered which prints out all the list of members in the groupchat

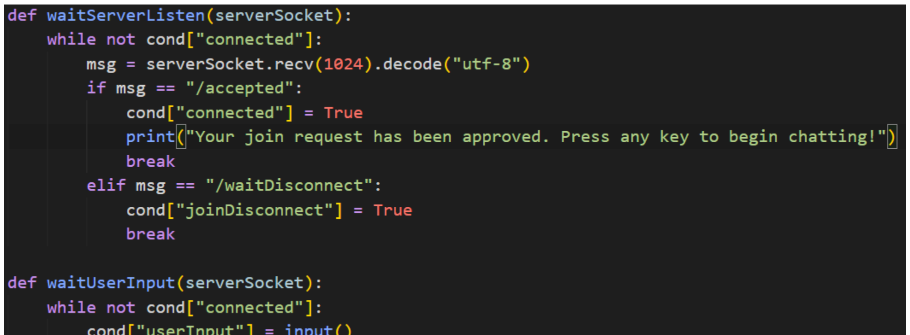
- **userInput function:**
  This functions reads the users’ inputs and then sends them to the server. They also look out for any trigger words and essentially “wires” the trigger word to their respective commands to send to the server.



```
def userInput(serverSocket):
    while cond["connected"]:
        cond["sendMessageLock"].acquire()
        cond["userInput"] = input()
        cond["sendMessageLock"].release()
        with cond["inputCondt"]:
            cond["inputCondt"].notify()
        if cond["userInput"] == "/view":
            serverSocket.send(b"/viewRequests")
        elif cond["userInput"] == "/approve":
            serverSocket.send(b"/approveRequest")
        elif cond["userInput"] == "/exit":
            serverSocket.send(b"/disconnect")
            break
        elif cond["userInput"] == "/members":
            serverSocket.send(b"/allMembers")
        elif cond["userInput"] == "/online":
            serverSocket.send(b"/onlineMembers")
        elif cond["userInput"] == "/changeAdmin":
```

- **waitServerListen and waitUserInput function:**

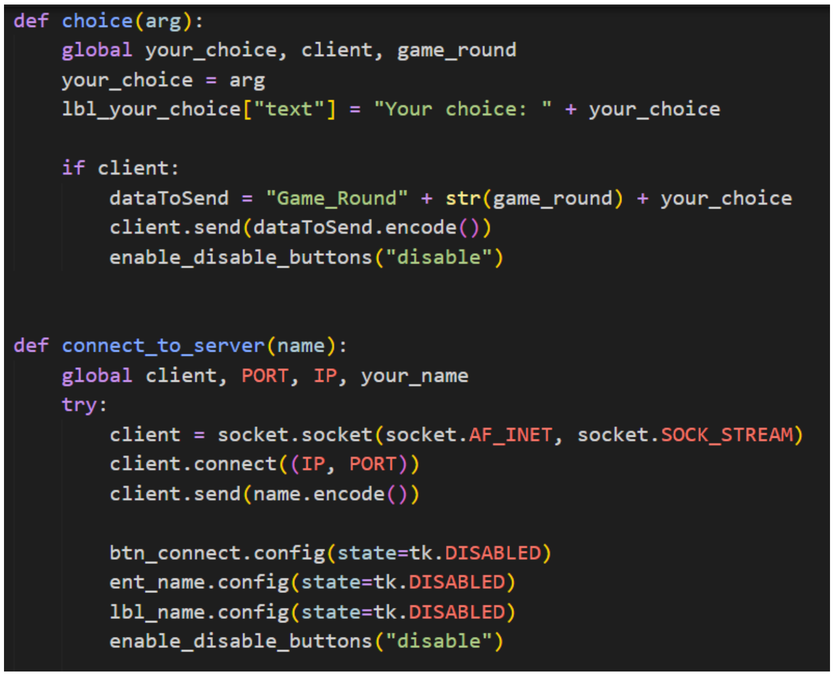
These functions essentially act as temporary buffer when needed such as when a user is waiting to be admitted into a room



```
def waitServerListen(serverSocket):
    while not cond["connected"]:
        msg = serverSocket.recv(1024).decode("utf-8")
        if msg == "/accepted":
            cond["connected"] = True
            print("Your join request has been approved. Press any key to begin chatting!")
            break
        elif msg == "/waitDisconnect":
            cond["joinDisconnect"] = True
            break

def waitUserInput(serverSocket):
    while not cond["connected"]:
        cond["userInput"] = input()
```

2. **gameClient.py**

This file acts as an interface between the users and server and allows them to play together

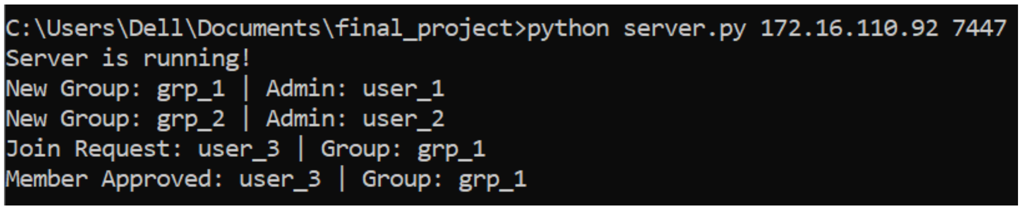


```
def choice(arg):
    global your_choice, client, game_round
    your_choice = arg
    lbl_your_choice["text"] = "Your choice: " + your_choice

    if client:
        dataToSend = "Game_Round" + str(game_round) + your_choice
        client.send(dataToSend.encode())
        enable_disable_buttons("disable")


def connect_to_server(name):
    global client, PORT, IP, your_name
    try:
        client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        client.connect((IP, PORT))
        client.send(name.encode())

        btn_connect.config(state=tk.DISABLED)
        ent_name.config(state=tk.DISABLED)
        lbl_name.config(state=tk.DISABLED)
        enable_disable_buttons("disable")
```

The snippet above saves your choice in the round and sends it to the server where it is then compared with your opponent's choice to determine who won the round. The connect_to_server function connects both user to the same server so that they can play together.

Demo:

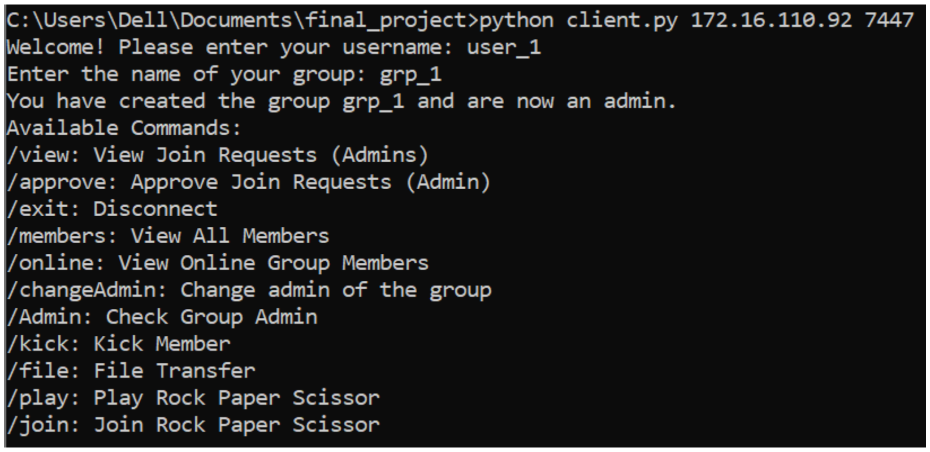
1. **Server**



```
C:\Users\Dell\Documents\final_project>python server.py 172.16.110.92 7447
Server is running!
New Group: grp_1 | Admin: user_1
New Group: grp_2 | Admin: user_2
Join Request: user_3 | Group: grp_1
Member Approved: user_3 | Group: grp_1
```

Here we can see the different users and their corresponding groups as well as request messages

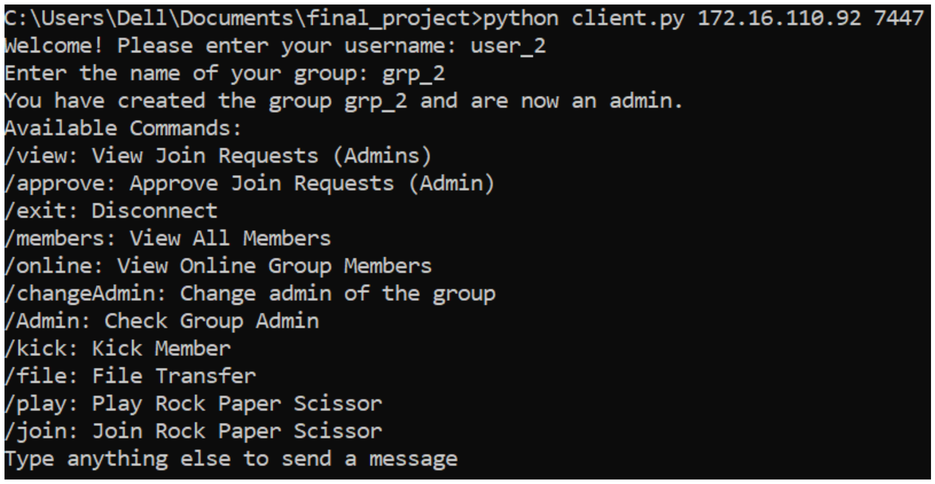
2. **Client 1**



```
C:\Users\Dell\Documents\final_project>python client.py 172.16.110.92 7447
Welcome! Please enter your username: user_1
Enter the name of your group: grp_1
You have created the group grp_1 and are now an admin.
Available Commands:
/view: View Join Requests (Admins)
/approve: Approve Join Requests (Admin)
/exit: Disconnect
/members: View All Members
/online: View Online Group Members
/changeAdmin: Change admin of the group
/Admin: Check Group Admin
/kick: Kick Member
/file: File Transfer
/play: Play Rock Paper Scissor
/join: Join Rock Paper Scissor
```

We can see that user_1 has successfully connected to the server and since it was the first user to enter a new room, it was set as an admin by default.

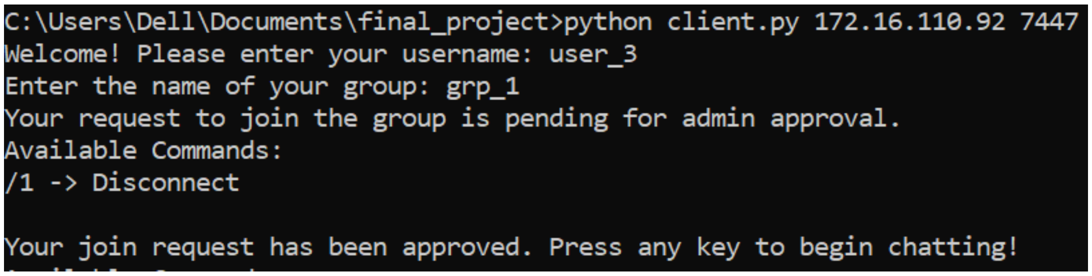
3. **Client 2**



```
C:\Users\Dell\Documents\final_project>python client.py 172.16.110.92 7447
Welcome! Please enter your username: user_2
Enter the name of your group: grp_2
You have created the group grp_2 and are now an admin.
Available Commands:
/view: View Join Requests (Admins)
/approve: Approve Join Requests (Admin)
/exit: Disconnect
/members: View All Members
/online: View Online Group Members
/changeAdmin: Change admin of the group
/Admin: Check Group Admin
/kick: Kick Member
/file: File Transfer
/play: Play Rock Paper Scissor
/join: Join Rock Paper Scissor
Type anything else to send a message
```

Same as user_1, user_2 as the first use to enter a new, room grp_2 and is therefore the admin of room grp_2

4. **Client 3**

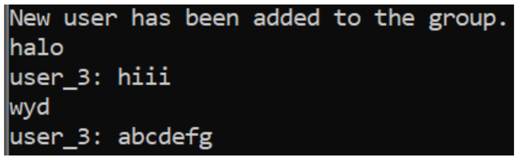


```
C:\Users\Dell\Documents\final_project>python client.py 172.16.110.92 7447
Welcome! Please enter your username: user_3
Enter the name of your group: grp_1
Your request to join the group is pending for admin approval.
Available Commands:
/1 -> Disconnect

Your join request has been approved. Press any key to begin chatting!
```

Here, user_3 is trying to join an already existing room, grp_1 and has to wait for the admin(user_2) to allow them into the room. While waiting for the admin to approve their request, user_3 is given the choice to disconnect now. After their request is approved, they are admitted into the room and can now start chatting

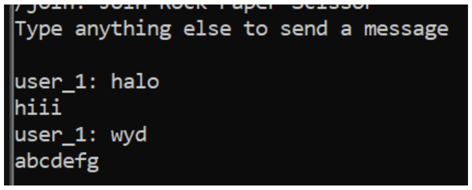
5. **Chatting in progress**

User_1 perspective:



```
New user has been added to the group.
halo
user_3: hiii
wyd
user_3: abcdefg
```

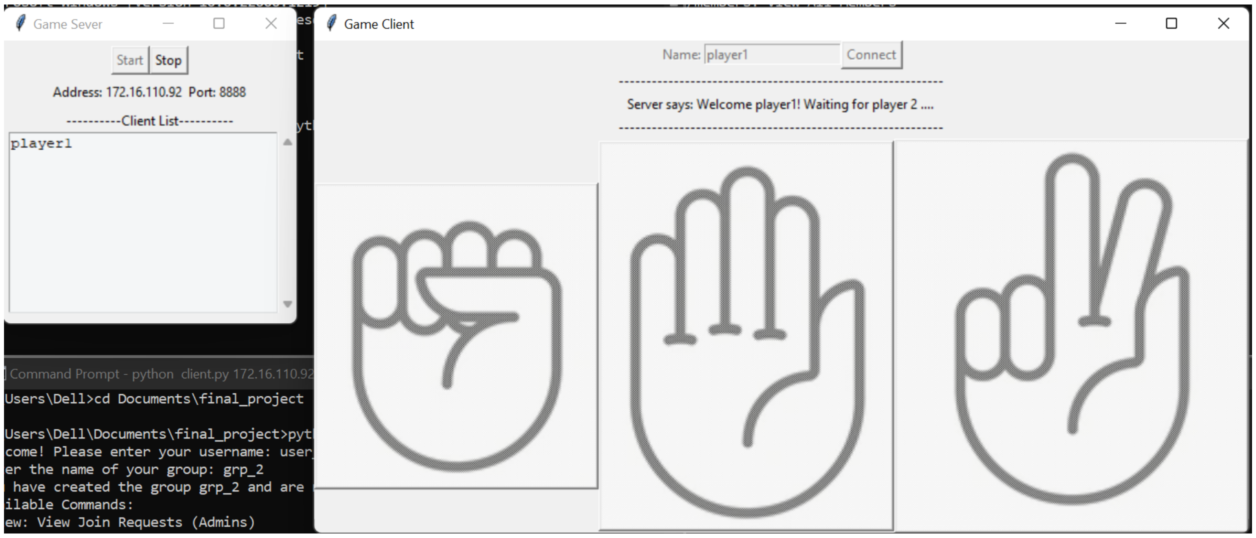
User_3 perspective:



```
Type anything else to send a message

user_1: halo
hiii
user_1: wyd
abcdefg
```

6. **Gameplay:**

When one of the user sends "/play", the game is triggered. The other player can then join using "/join"

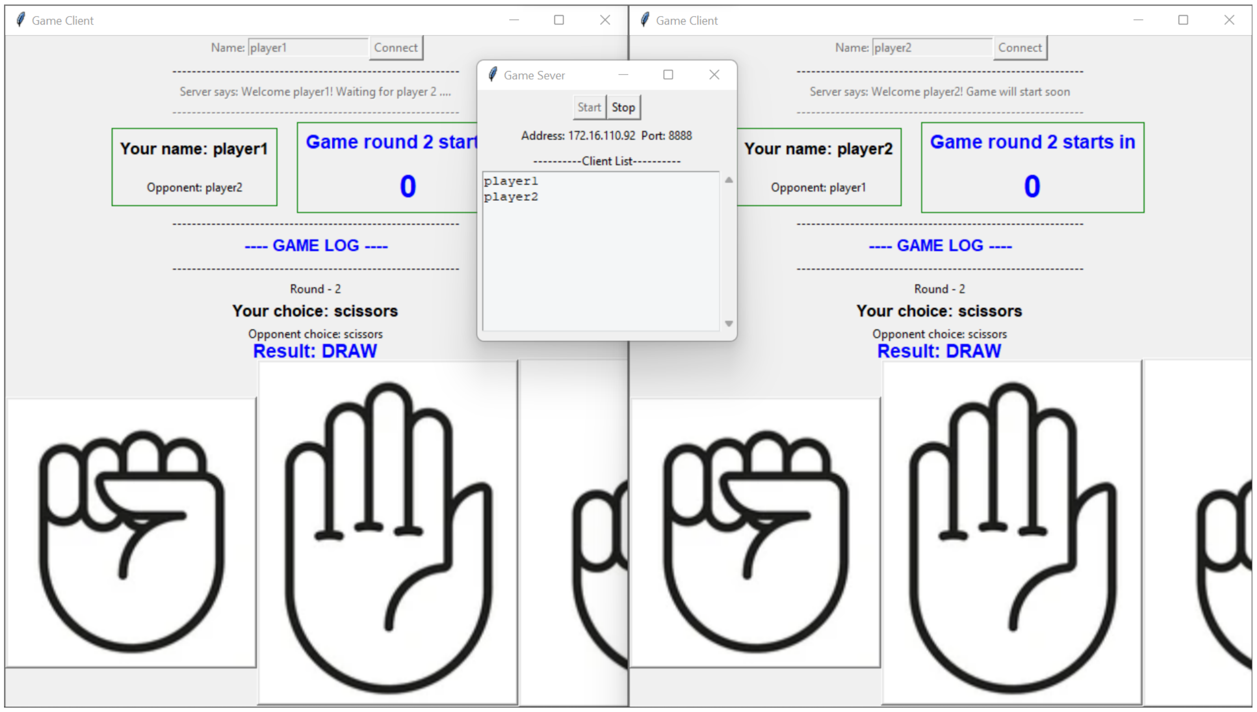


```
Address: 172.16.110.92  Port: 8888
----------Client List----------
player1
```

```
Name: player1
---------------------------------------------------------
Server says: Welcome player1! Waiting for player 2 ....
---------------------------------------------------------
```

```
Command Prompt - python  client.py 172.16.110.92
Users\Dell>cd Documents\final_project

Users\Dell\Documents\final_project>pyt
come! Please enter your username: user
er the name of your group: grp_2
 have created the group grp_2 and are
ilable Commands:
ew: View Join Requests (Admins)
```

After both players join:



Game Client

Name: player1 Connect

Server says: Welcome player1! Waiting for player 2 ....

Your name: player1

Opponent: player2

Game round 2 start

0

---- GAME LOG ----

Round - 2

Your choice: scissors

Opponent choice: scissors

Result: DRAW

Game Sever

Start Stop

Address: 172.16.110.92 Port: 8888

----------Client List----------

```
player1
player2
```

Game Client

Name: player2 Connect

Server says: Welcome player2! Game will start soon

Your name: player2

Opponent: player1

Game round 2 starts in

0

---- GAME LOG ----

Round - 2

Your choice: scissors

Opponent choice: scissors

Result: DRAW