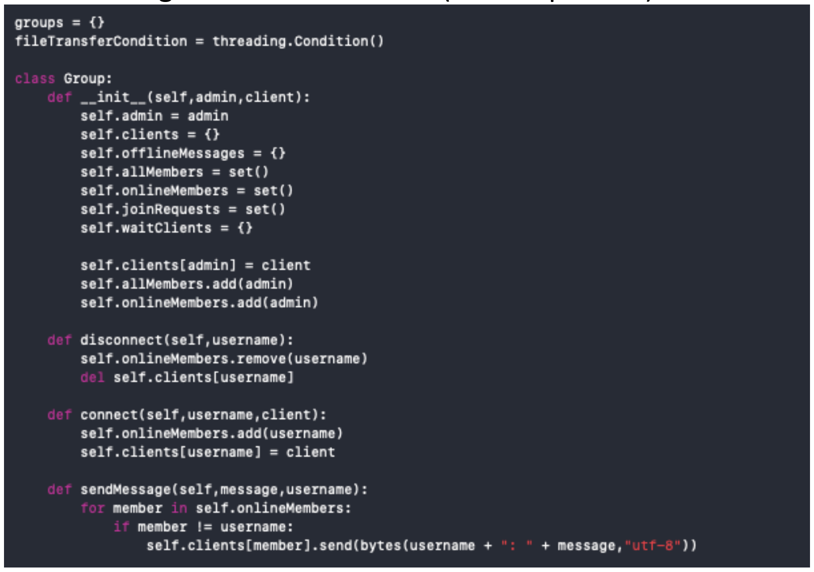
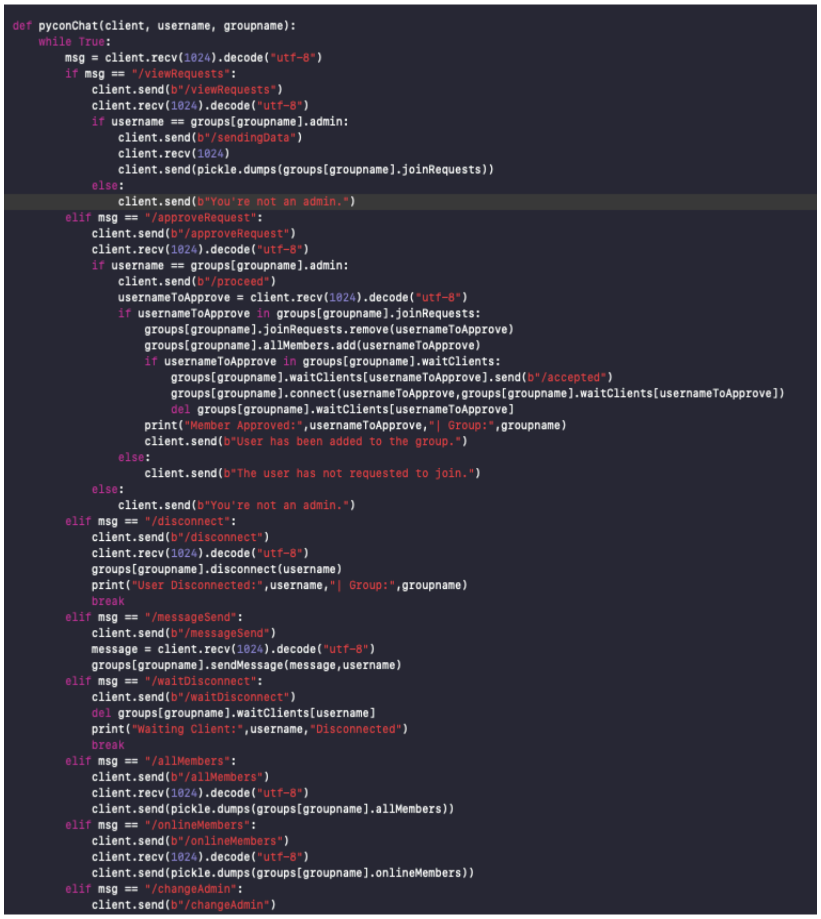
SERVER.PY

For declaring the methods to use (later explained):

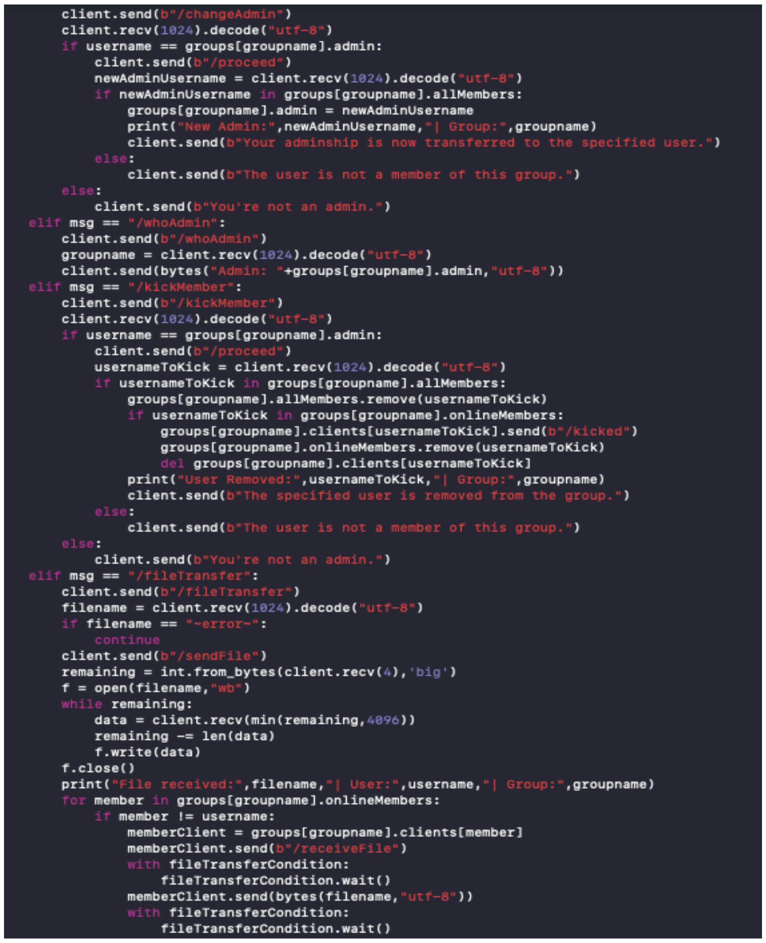


```
groups = {}
fileTransferCondition = threading.Condition()

class Group:
    def __init__(self,admin,client):
        self.admin = admin
        self.clients = {}
        self.offlineMessages = {}
        self.allMembers = set()
        self.onlineMembers = set()
        self.joinRequests = set()
        self.waitClients = {}

        self.clients[admin] = client
        self.allMembers.add(admin)
        self.onlineMembers.add(admin)

    def disconnect(self,username):
        self.onlineMembers.remove(username)
        del self.clients[username]

    def connect(self,username,client):
        self.onlineMembers.add(username)
        self.clients[username] = client

    def sendMessage(self,message,username):
        for member in self.onlineMembers:
            if member != username:
                self.clients[member].send(bytes(username + ": " + message,"utf-8"))
```

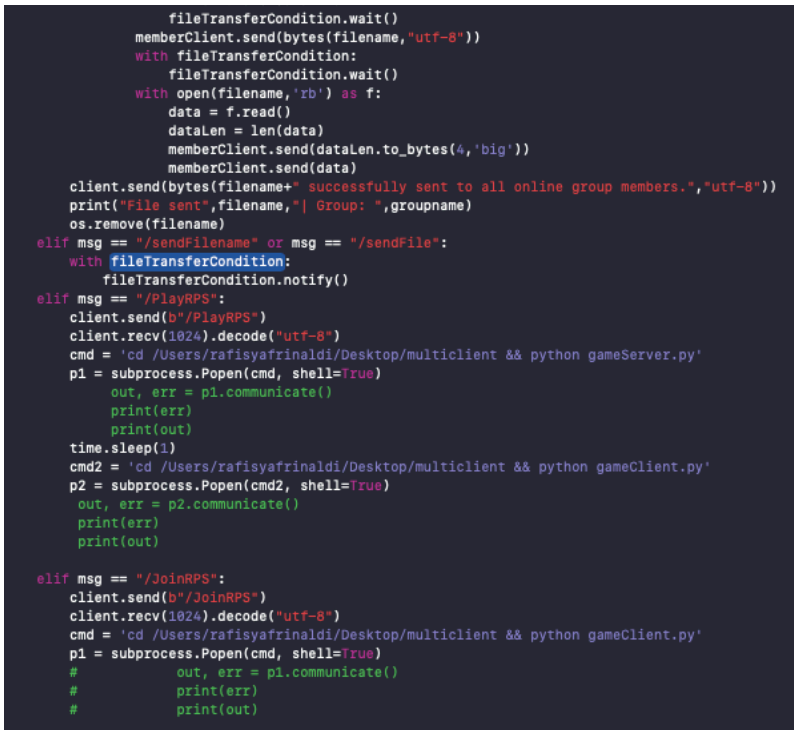
This function is used to contain the commands that this app offer to the clients to use, where in each conditions provided with decoding and encoding process of the message sent from the clients and with the help of the packages imported to the source code it can help to pass the informations based on the essential socket or client informations. And specifically for the method to play the game the suprocess package helps us to run another .py file on another .py file.



```python
def pyconChat(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].joinRequests))
            else:
                client.send(b"You're not an admin.")
        elif msg == "/approveRequest":
            client.send(b"/approveRequest")
            client.recv(1024).decode("utf-8")
            if username == groups[groupname].admin:
                client.send(b"/proceed")
                usernameToApprove = client.recv(1024).decode("utf-8")
                if usernameToApprove in groups[groupname].joinRequests:
                    groups[groupname].joinRequests.remove(usernameToApprove)
                    groups[groupname].allMembers.add(usernameToApprove)
                    if usernameToApprove in groups[groupname].waitClients:
                        groups[groupname].waitClients[usernameToApprove].send(b"/accepted")
                        groups[groupname].connect(usernameToApprove,groups[groupname].waitClients[usernameToApprove])
                        del groups[groupname].waitClients[usernameToApprove]
                    print("Member Approved:",usernameToApprove,"| Group:",groupname)
                    client.send(b"User has been added to the group.")
                else:
                    client.send(b"The user has not requested to join.")
            else:
                client.send(b"You're not an admin.")
        elif msg == "/disconnect":
            client.send(b"/disconnect")
            client.recv(1024).decode("utf-8")
            groups[groupname].disconnect(username)
            print("User Disconnected:",username,"| Group:",groupname)
            break
        elif msg == "/messageSend":
            client.send(b"/messageSend")
            message = client.recv(1024).decode("utf-8")
            groups[groupname].sendMessage(message,username)
        elif msg == "/waitDisconnect":
            client.send(b"/waitDisconnect")
            del groups[groupname].waitClients[username]
            print("Waiting Client:",username,"Disconnected")
            break
        elif msg == "/allMembers":
            client.send(b"/allMembers")
            client.recv(1024).decode("utf-8")
            client.send(pickle.dumps(groups[groupname].allMembers))
        elif msg == "/onlineMembers":
            client.send(b"/onlineMembers")
            client.recv(1024).decode("utf-8")
            client.send(pickle.dumps(groups[groupname].onlineMembers))
        elif msg == "/changeAdmin":
            client.send(b"/changeAdmin")
```

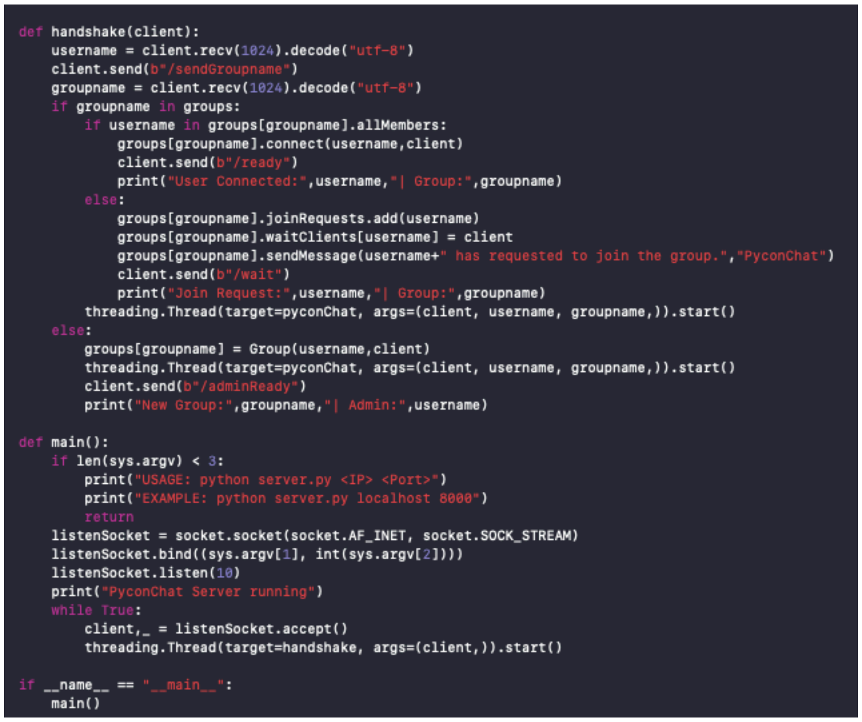


```python
            client.send(b"/changeAdmin")
            client.recv(1024).decode("utf-8")
            if username == groups[groupname].admin:
                client.send(b"/proceed")
                newAdminUsername = client.recv(1024).decode("utf-8")
                if newAdminUsername in groups[groupname].allMembers:
                    groups[groupname].admin = newAdminUsername
                    print("New Admin:",newAdminUsername,"| Group:",groupname)
                    client.send(b"Your adminship is now transferred to the specified user.")
                else:
                    client.send(b"The user is not a member of this group.")
            else:
                client.send(b"You're not an admin.")
        elif msg == "/whoAdmin":
            client.send(b"/whoAdmin")
            groupname = client.recv(1024).decode("utf-8")
            client.send(bytes("Admin: "+groups[groupname].admin,"utf-8"))
        elif msg == "/kickMember":
            client.send(b"/kickMember")
            client.recv(1024).decode("utf-8")
            if username == groups[groupname].admin:
                client.send(b"/proceed")
                usernameToKick = client.recv(1024).decode("utf-8")
                if usernameToKick in groups[groupname].allMembers:
                    groups[groupname].allMembers.remove(usernameToKick)
                    if usernameToKick in groups[groupname].onlineMembers:
                        groups[groupname].clients[usernameToKick].send(b"/kicked")
                        groups[groupname].onlineMembers.remove(usernameToKick)
                        del groups[groupname].clients[usernameToKick]
                    print("User Removed:",usernameToKick,"| Group:",groupname)
                    client.send(b"The specified user is removed from the group.")
                else:
                    client.send(b"The user is not a member of this group.")
            else:
                client.send(b"You're not an admin.")
        elif msg == "/fileTransfer":
            client.send(b"/fileTransfer")
            filename = client.recv(1024).decode("utf-8")
            if filename == "~error~":
                continue
            client.send(b"/sendFile")
            remaining = int.from_bytes(client.recv(4),'big')
            f = open(filename,"wb")
            while remaining:
                data = client.recv(min(remaining,4096))
                remaining -= len(data)
                f.write(data)
            f.close()
            print("File received:",filename,"| User:",username,"| Group:",groupname)
            for member in groups[groupname].onlineMembers:
                if member != username:
                    memberClient = groups[groupname].clients[member]
                    memberClient.send(b"/receiveFile")
                    with fileTransferCondition:
                        fileTransferCondition.wait()
                    memberClient.send(bytes(filename,"utf-8"))
                    with fileTransferCondition:
                        fileTransferCondition.wait()
```



```python
                        fileTransferCondition.wait()
                    memberClient.send(bytes(filename,"utf-8"))
                    with fileTransferCondition:
                        fileTransferCondition.wait()
                    with open(filename,'rb') as f:
                        data = f.read()
                        dataLen = len(data)
                        memberClient.send(dataLen.to_bytes(4,'big'))
                        memberClient.send(data)
            client.send(bytes(filename+" successfully sent to all online group members.","utf-8"))
            print("File sent",filename,"| Group: ",groupname)
            os.remove(filename)
        elif msg == "/sendFilename" or msg == "/sendFile":
            with fileTransferCondition:
                fileTransferCondition.notify()
        elif msg == "/PlayRPS":
            client.send(b"/PlayRPS")
            client.recv(1024).decode("utf-8")
            cmd = 'cd /Users/rafisyafrinaldi/Desktop/multiclient && python gameServer.py'
            p1 = subprocess.Popen(cmd, shell=True)
                 out, err = p1.communicate()
                 print(err)
                 print(out)
            time.sleep(1)
            cmd2 = 'cd /Users/rafisyafrinaldi/Desktop/multiclient && python gameClient.py'
            p2 = subprocess.Popen(cmd2, shell=True)
             out, err = p2.communicate()
             print(err)
             print(out)

        elif msg == "/JoinRPS":
            client.send(b"/JoinRPS")
            client.recv(1024).decode("utf-8")
            cmd = 'cd /Users/rafisyafrinaldi/Desktop/multiclient && python gameClient.py'
            p1 = subprocess.Popen(cmd, shell=True)
            #            out, err = p1.communicate()
            #            print(err)
            #            print(out)
```

Lastly, the handshake method is used to compact the clients into one groupchat using the condtions if met based on whether the client has entered the informations, i.e username and group name.
And as for the server main method it basically will display us the first informations on the server shell if got compiled.



```python
def handshake(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].joinRequests.add(username)
            groups[groupname].waitClients[username] = client
            groups[groupname].sendMessage(username+" has requested to join the group.","PyconChat")
            client.send(b"/wait")
            print("Join Request:",username,"| Group:",groupname)
        threading.Thread(target=pyconChat, args=(client, username, groupname,)).start()
    else:
        groups[groupname] = Group(username,client)
        threading.Thread(target=pyconChat, args=(client, username, groupname,)).start()
        client.send(b"/adminReady")
        print("New Group:",groupname,"| Admin:",username)

def main():
    if len(sys.argv) < 3:
        print("USAGE: 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.listen(10)
    print("PyconChat Server running")
    while True:
        client,_ = listenSocket.accept()
        threading.Thread(target=handshake, args=(client,)).start()

if __name__ == "__main__":
    main()
```