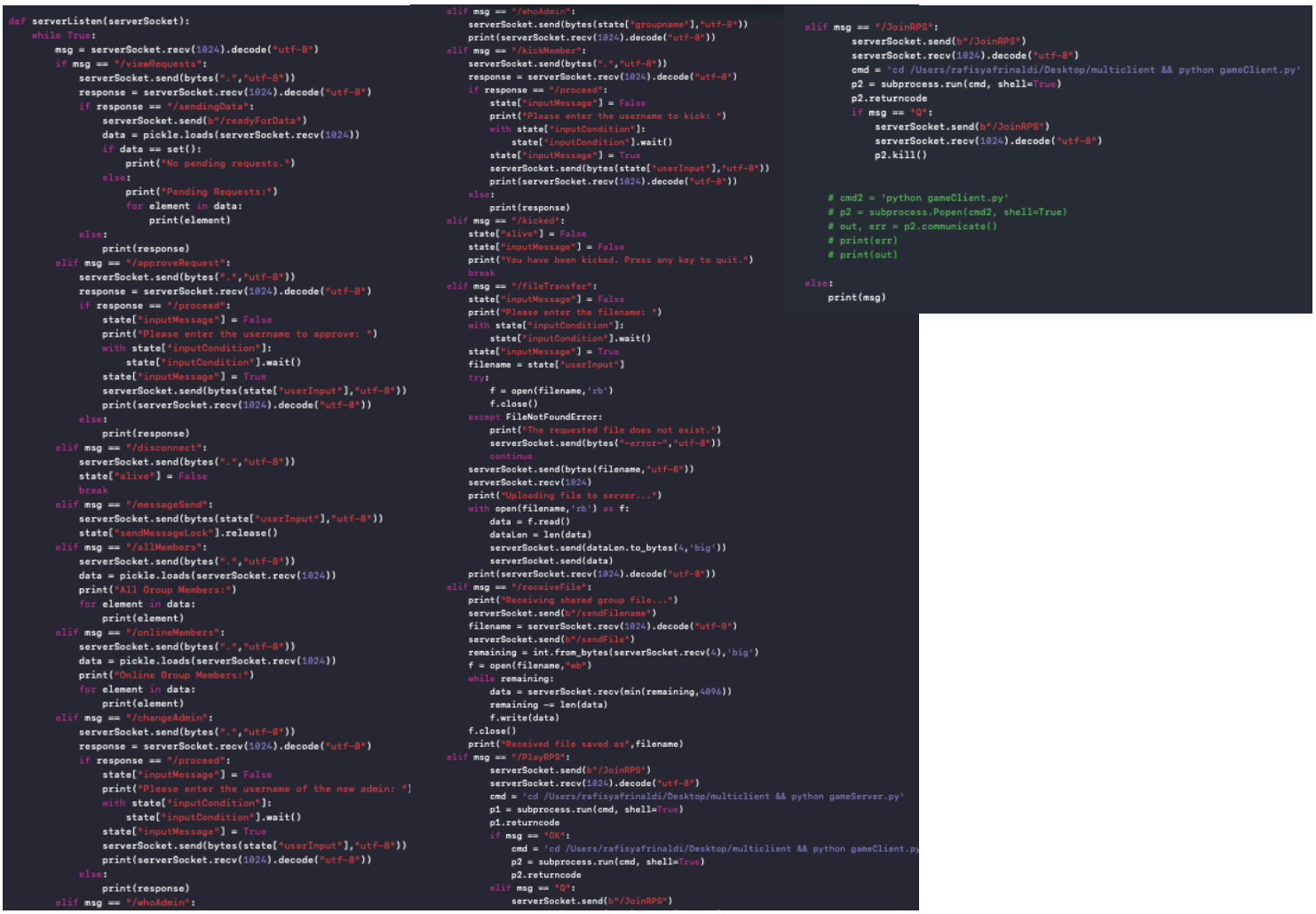
Client.py

these methods just like its name, it is for passing the decisions from each command that client make and get encoded to pass it to the server so that it can be shown and run on both parties. And of course, with each command it got run successfully if the conditions are met, the way it is written is basically the same as the server's method but it's using different sending medium which for the client using serverSocket, and as for the server using client.



```
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)
        elif msg == "/approveRequest":
            serverSocket.send(bytes(".","utf-8"))
            response = serverSocket.recv(1024).decode("utf-8")
            if response == "/proceed":
                state["inputMessage"] = False
                print("Please enter the username to approve: ")
                with state["inputCondition"]:
                    state["inputCondition"].wait()
                state["inputMessage"] = True
                serverSocket.send(bytes(state["userInput"],"utf-8"))
                print(serverSocket.recv(1024).decode("utf-8"))
            else:
                print(response)
        elif msg == "/disconnect":
            serverSocket.send(bytes(".","utf-8"))
            state["alive"] = False
            break
        elif msg == "/messageSend":
            serverSocket.send(bytes(state["userInput"],"utf-8"))
            state["sendMessageLock"].release()
        elif msg == "/allMembers":
            serverSocket.send(bytes(".","utf-8"))
            data = pickle.loads(serverSocket.recv(1024))
            print("All Group Members:")
            for element in data:
                print(element)
        elif msg == "/onlineMembers":
            serverSocket.send(bytes(".","utf-8"))
            data = pickle.loads(serverSocket.recv(1024))
            print("Online Group Members:")
            for element in data:
                print(element)
        elif msg == "/changeAdmin":
            serverSocket.send(bytes(".","utf-8"))
            response = serverSocket.recv(1024).decode("utf-8")
            if response == "/proceed":
                state["inputMessage"] = False
                print("Please enter the username of the new admin: ")
                with state["inputCondition"]:
                    state["inputCondition"].wait()
                state["inputMessage"] = True
                serverSocket.send(bytes(state["userInput"],"utf-8"))
                print(serverSocket.recv(1024).decode("utf-8"))
            else:
                print(response)
        elif msg == "/whoAdmin":
            serverSocket.send(bytes(state["groupname"],"utf-8"))
            print(serverSocket.recv(1024).decode("utf-8"))
        elif msg == "/kickMember":
            serverSocket.send(bytes(".","utf-8"))
            response = serverSocket.recv(1024).decode("utf-8")
            if response == "/proceed":
                state["inputMessage"] = False
                print("Please enter the username to kick: ")
                with state["inputCondition"]:
                    state["inputCondition"].wait()
                state["inputMessage"] = True
                serverSocket.send(bytes(state["userInput"],"utf-8"))
                print(serverSocket.recv(1024).decode("utf-8"))
            else:
                print(response)
        elif msg == "/kicked":
            state["alive"] = False
            state["inputMessage"] = False
            print("You have been kicked. Press any key to quit.")
            break
        elif msg == "/fileTransfer":
            state["inputMessage"] = False
            print("Please enter the filename: ")
            with state["inputCondition"]:
                state["inputCondition"].wait()
            state["inputMessage"] = True
            filename = state["userInput"]
            try:
                f = open(filename,'rb')
                f.close()
            except FileNotFoundError:
                print("The requested file does not exist.")
                serverSocket.send(bytes("~error~","utf-8"))
                continue
            serverSocket.send(bytes(filename,"utf-8"))
            serverSocket.recv(1024)
            print("Uploading file to server...")
            with open(filename,'rb') as f:
                data = f.read()
                dataLen = len(data)
                serverSocket.send(dataLen.to_bytes(4,'big'))
                serverSocket.send(data)
            print(serverSocket.recv(1024).decode("utf-8"))
        elif msg == "/receiveFile":
            print("Receiving shared group file...")
            serverSocket.send(b"/sendFilename")
            filename = serverSocket.recv(1024).decode("utf-8")
            serverSocket.send(b"/sendFile")
            remaining = int.from_bytes(serverSocket.recv(4),'big')
            f = open(filename,"wb")
            while remaining:
                data = serverSocket.recv(min(remaining,4096))
                remaining -= len(data)
                f.write(data)
            f.close()
            print("Received file saved as",filename)
        elif msg == "/PlayRPS":
                serverSocket.send(b"/JoinRPS")
                serverSocket.recv(1024).decode("utf-8")
                cmd = 'cd /Users/rafisyafrinaldi/Desktop/multiclient && python gameServer.py'
                p1 = subprocess.run(cmd, shell=True)
                p1.returncode
                if msg == "OK":
                    cmd = 'cd /Users/rafisyafrinaldi/Desktop/multiclient && python gameClient.py'
                    p2 = subprocess.run(cmd, shell=True)
                    p2.returncode
                elif msg == "Q":
                    serverSocket.send(b"/JoinRPS")
        elif msg == "/JoinRPS":
                serverSocket.send(b"/JoinRPS")
                serverSocket.recv(1024).decode("utf-8")
                cmd = 'cd /Users/rafisyafrinaldi/Desktop/multiclient && python gameClient.py'
                p2 = subprocess.run(cmd, shell=True)
                p2.returncode
                if msg == "Q":
                    serverSocket.send(b"/JoinRPS")
                    serverSocket.recv(1024).decode("utf-8")
                    p2.kill()


            # cmd2 = 'python gameClient.py'
            # p2 = subprocess.Popen(cmd2, shell=True)
            # out, err = p2.communicate()
            # print(err)
            # print(out)

        else:
            print(msg)
```

The userInput method lets us to translate the input string to trigger action to the previous method's commands. The waitServerListen method is used for the client waiting to be accepted to the group chat and it will work side by side with the waitUserInput where if it's the client who is not in the chat cannot access the certain command both these methods are using the alive statement to indicate the readiness of the client.
Last but not least, the main method is used for the UI that will be shown to the client first in the shell. Displaying client's requirement ot enter the groupchat, the menu of commands, etc. And as for the indicating the readiness as in the client is in the server or not can be used for the disconnect or department of the client when they leave the room.

```
def userInput(serverSocket):
    while state["alive"]:
        state["sendMessageLock"].acquire()
        state["userInput"] = input()
        state["sendMessageLock"].release()
        with state["inputCondition"]:
            state["inputCondition"].notify()
        if state["userInput"] == "/1":
            serverSocket.send(b"/viewRequests")
        elif state["userInput"] == "/2":
            serverSocket.send(b"/approveRequest")
        elif state["userInput"] == "/3":
            serverSocket.send(b"/disconnect")
            break
        elif state["userInput"] == "/4":
            serverSocket.send(b"/allMembers")
        elif state["userInput"] == "/5":
            serverSocket.send(b"/onlineMembers")
        elif state["userInput"] == "/6":
            serverSocket.send(b"/changeAdmin")
        elif state["userInput"] == "/7":
            serverSocket.send(b"/whoAdmin")
        elif state["userInput"] == "/8":
            serverSocket.send(b"/kickMember")
        elif state["userInput"] == "/9":
            serverSocket.send(b"/fileTransfer")
        elif state["userInput"] == "/10":
            serverSocket.send(b"/PlayRPS")
        elif state["userInput"] == "/11":
            serverSocket.send(b"/JoinRPS")


        elif state["inputMessage"]:
            state["sendMessageLock"].acquire()
            serverSocket.send(b"/messageSend")

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

def waitUserInput(serverSocket):
    while not state["alive"]:
        state["userInput"] = input()
        if state["userInput"] == "/1" and not state["alive"]:
            serverSocket.send(b"/waitDisconnect")
            break
```

```
def main():
    if len(sys.argv) < 3:
        print("USAGE: python client.py <IP> <Port>")
        print("EXAMPLE: python client.py localhost 8000")
        return
    serverSocket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
    serverSocket.connect((sys.argv[1], int(sys.argv[2])))
    state["inputCondition"] = threading.Condition()
    state["sendMessageLock"] = threading.Lock()
    state["username"] = input("Welcome to PyconChat! Please enter your username: ")
    state["groupname"] = input("Please enter the name of the group: ")
    state["alive"] = False
    state["joinDisconnect"] = False
    state["inputMessage"] = True
    serverSocket.send(bytes(state["username"],"utf-8"))
    serverSocket.recv(1024)
    serverSocket.send(bytes(state["groupname"],"utf-8"))
    response = serverSocket.recv(1024).decode("utf-8")
    if response == "/adminReady":
        print("You have created the group",state["groupname"],"and are now an admin.")
        state["alive"] = True
    elif response == "/ready":
        print("You have joined the group",state["groupname"])
        state["alive"] = True
    elif response == "/wait":
        print("Your request to join the group is pending admin approval.")
        print("Available Commands:\n/1 -> Disconnect\n")
    waitUserInputThread = threading.Thread(target=waitUserInput,args=(serverSocket,))
    waitServerListenThread = threading.Thread(target=waitServerListen,args=(serverSocket,))
    userInputThread = threading.Thread(target=userInput,args=(serverSocket,))
    serverListenThread = threading.Thread(target=serverListen,args=(serverSocket,))
    waitUserInputThread.start()
    waitServerListenThread.start()
    while True:
        if state["alive"] or state["joinDisconnect"]:
            break
    if state["alive"]:
        print("Available Commands:\n/1 -> View Join Requests (Admins)\n/2 -> Approve Join Requests (Admin)\n/3 -> Disconnect\n/4 -> View All Members\n/5 -> View Online Group Members\n/6 -> Transfer
            Adminship\n/7 -> Check Group Admin\n/8 -> Kick Member\n/9 -> File Transfer\n/10 -> Play Rock Paper Scissor \n/11 -> Join Rock Paper Scissor \nType anything else to send a message")
        waitUserInputThread.join()
        waitServerListenThread.join()
```