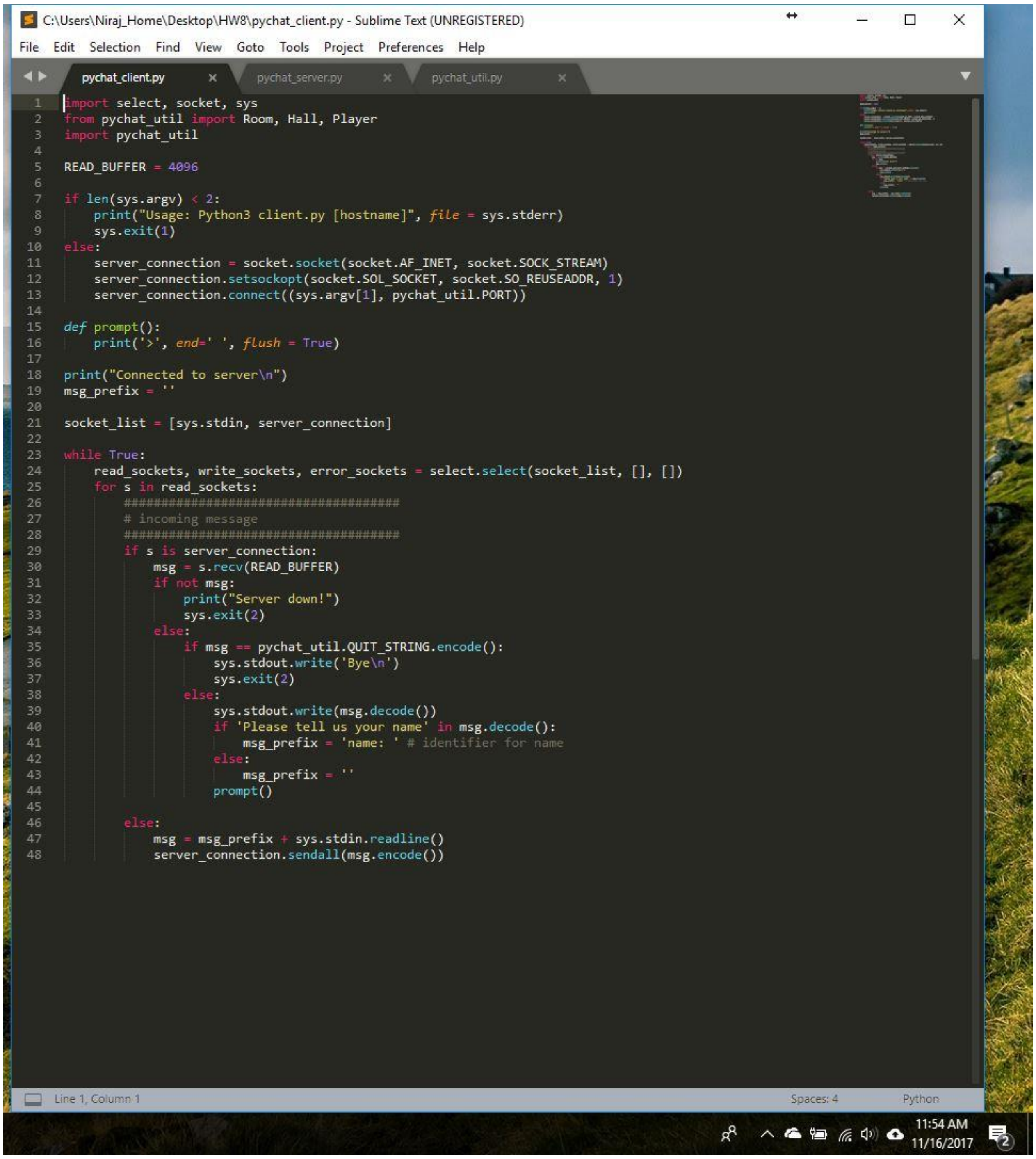
NAME : NIRAJ THANKI SID : 19376 CLASS: CS531

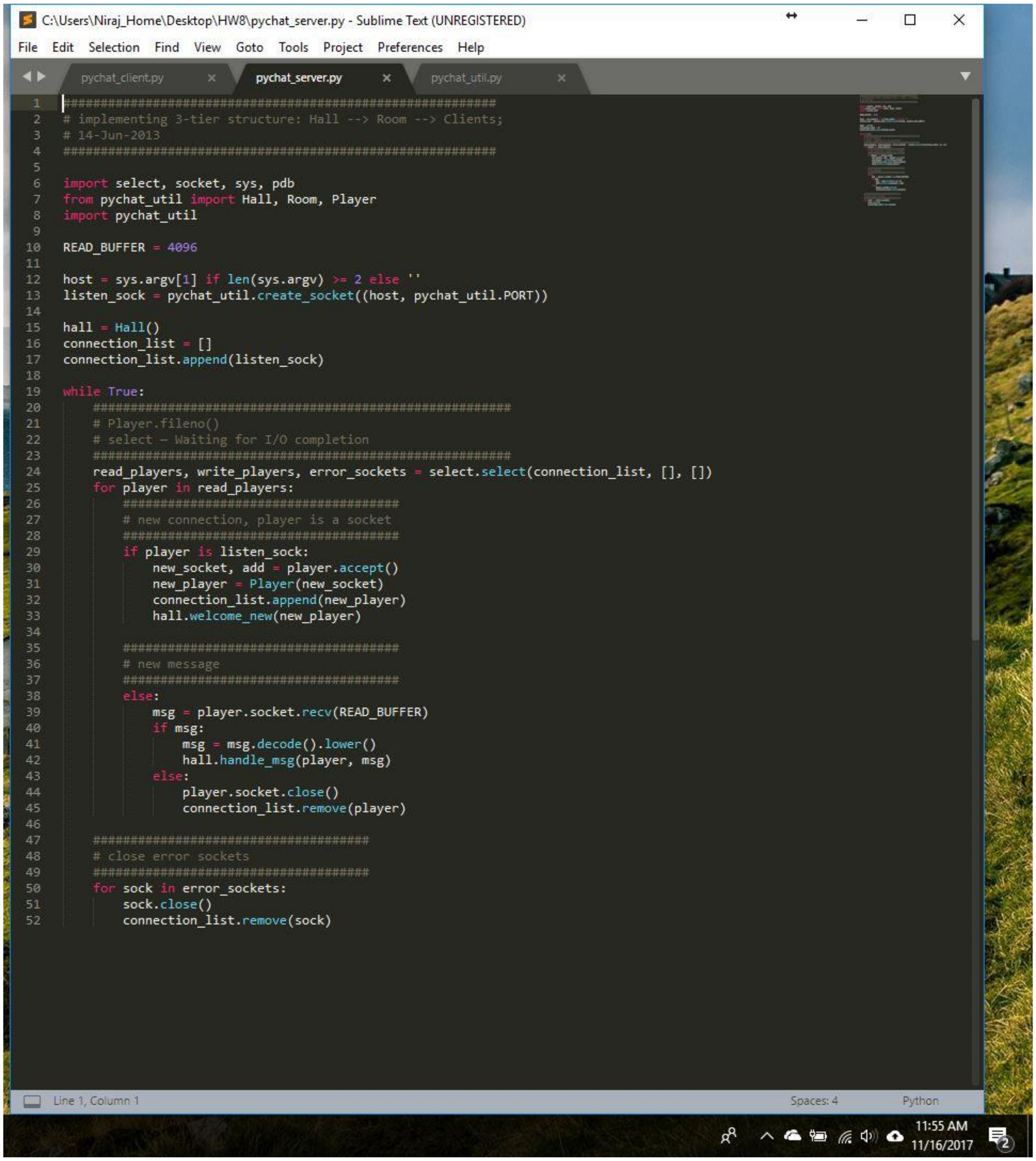

C:\Users\Niraj_Home\Desktop\HW8\pychat_client.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
pychat_client.py
pychat_server.py
pychat_util.py
Line 1, Column 1
Spaces: 4
Python
11:54 AM
11/16/2017

```
import select, socket, sys
from pychat_util import Room, Hall, Player
import pychat_util

READ_BUFFER = 4096

if len(sys.argv) < 2:
    print("Usage: Python3 client.py [hostname]", file = sys.stderr)
    sys.exit(1)
else:
    server_connection = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_connection.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    server_connection.connect((sys.argv[1], pychat_util.PORT))

def prompt():
    print('>', end=' ', flush = True)

print("Connected to server\n")
msg_prefix = ''

socket_list = [sys.stdin, server_connection]

while True:
    read_sockets, write_sockets, error_sockets = select.select(socket_list, [], [])
    for s in read_sockets:
        #####################################
        # incoming message
        #####################################
        if s is server_connection:
            msg = s.recv(READ_BUFFER)
            if not msg:
                print("Server down!")
                sys.exit(2)
            else:
                if msg == pychat_util.QUIT_STRING.encode():
                    sys.stdout.write('Bye\n')
                    sys.exit(2)
                else:
                    sys.stdout.write(msg.decode())
                    if 'Please tell us your name' in msg.decode():
                        msg_prefix = 'name: ' # identifier for name
                    else:
                        msg_prefix = ''
                    prompt()

        else:
            msg = msg_prefix + sys.stdin.readline()
            server_connection.sendall(msg.encode())
```

Source Code :

import select, socket, sys

```
from pychat_util import Room, Hall, Player
import pychat_util

READ_BUFFER = 4096

if len(sys.argv) < 2:
    print("Usage: Python3 client.py [hostname]", file = sys.stderr)
    sys.exit(1)
else:
    server_connection = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_connection.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    server_connection.connect((sys.argv[1], pychat_util.PORT))

def prompt():
    print('>', end=' ', flush = True)

print("Connected to server\n")
msg_prefix = ''

socket_list = [sys.stdin, server_connection]

while True:
    read_sockets, write_sockets, error_sockets = select.select(socket_list, [], [])
    for s in read_sockets:
        ######################################
        # incoming message
        ######################################
        if s is server_connection:
            msg = s.recv(READ_BUFFER)
            if not msg:
                print("Server down!")
                sys.exit(2)
            else:
                if msg == pychat_util.QUIT_STRING.encode():
                    sys.stdout.write('Bye\n')
                    sys.exit(2)
                else:
                    sys.stdout.write(msg.decode())
                    if 'Please tell us your name' in msg.decode():
                        msg_prefix = 'name: ' # identifier for name
                    else:
                        msg_prefix = ''
                    prompt()

        else:
            msg = msg_prefix + sys.stdin.readline()
            server_connection.sendall(msg.encode())
```


C:\Users\Niraj_Home\Desktop\HW8\pychat_server.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
pychat_client.py
pychat_server.py
pychat_util.py
Line 1, Column 1
Spaces: 4
Python
11:55 AM
11/16/2017

```python
##########################################################
# implementing 3-tier structure: Hall --> Room --> Clients;
# 14-Jun-2013
##########################################################

import select, socket, sys, pdb
from pychat_util import Hall, Room, Player
import pychat_util

READ_BUFFER = 4096

host = sys.argv[1] if len(sys.argv) >= 2 else ''
listen_sock = pychat_util.create_socket((host, pychat_util.PORT))

hall = Hall()
connection_list = []
connection_list.append(listen_sock)

while True:
    ########################################################
    # Player.fileno()
    # select – Waiting for I/O completion
    ########################################################
    read_players, write_players, error_sockets = select.select(connection_list, [], [])
    for player in read_players:
        #####################################
        # new connection, player is a socket
        #####################################
        if player is listen_sock:
            new_socket, add = player.accept()
            new_player = Player(new_socket)
            connection_list.append(new_player)
            hall.welcome_new(new_player)

        #####################################
        # new message
        #####################################
        else:
            msg = player.socket.recv(READ_BUFFER)
            if msg:
                msg = msg.decode().lower()
                hall.handle_msg(player, msg)
            else:
                player.socket.close()
                connection_list.remove(player)

    #####################################
    # close error sockets
    #####################################
    for sock in error_sockets:
        sock.close()
        connection_list.remove(sock)
```

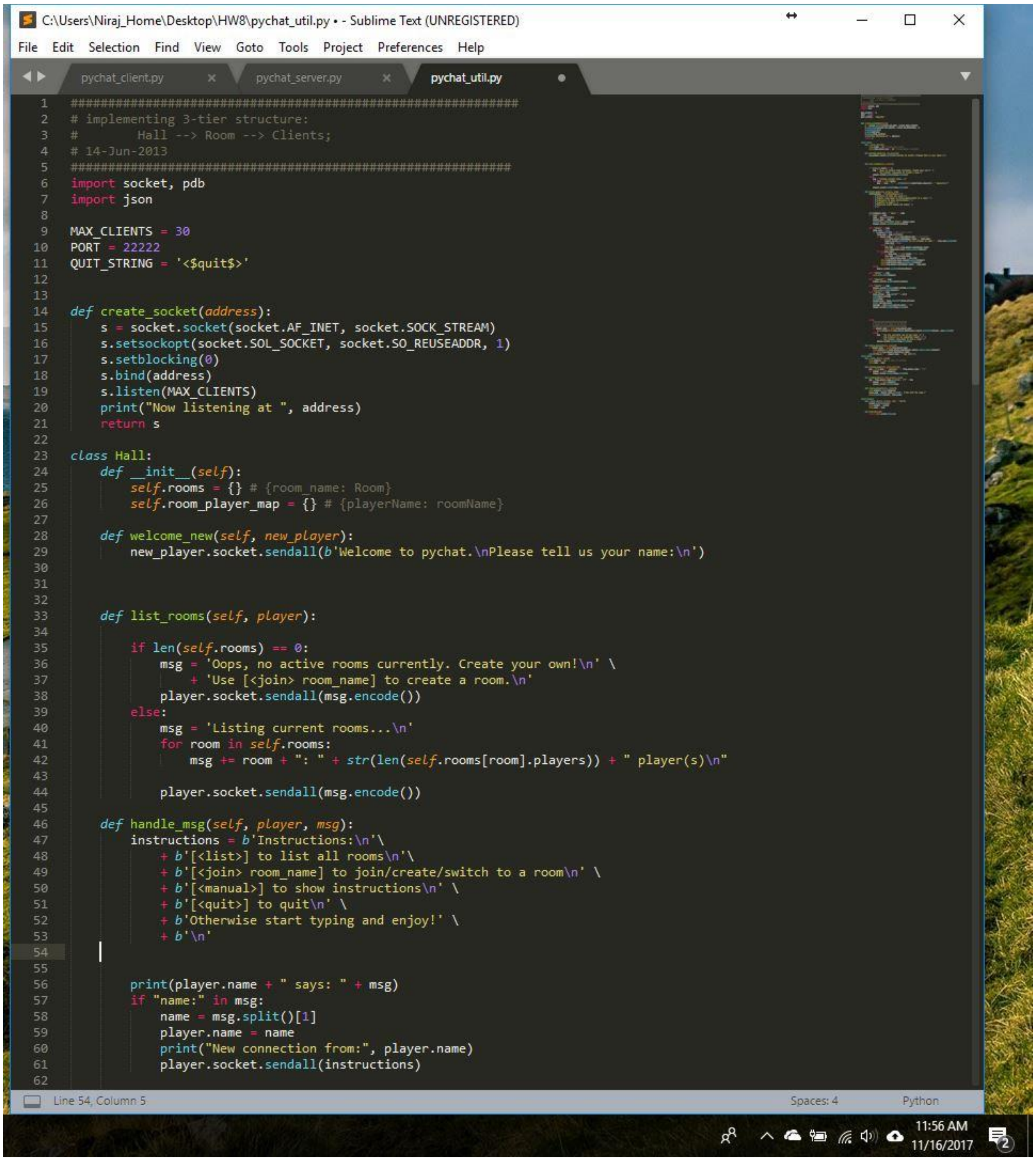
Source Code:

##

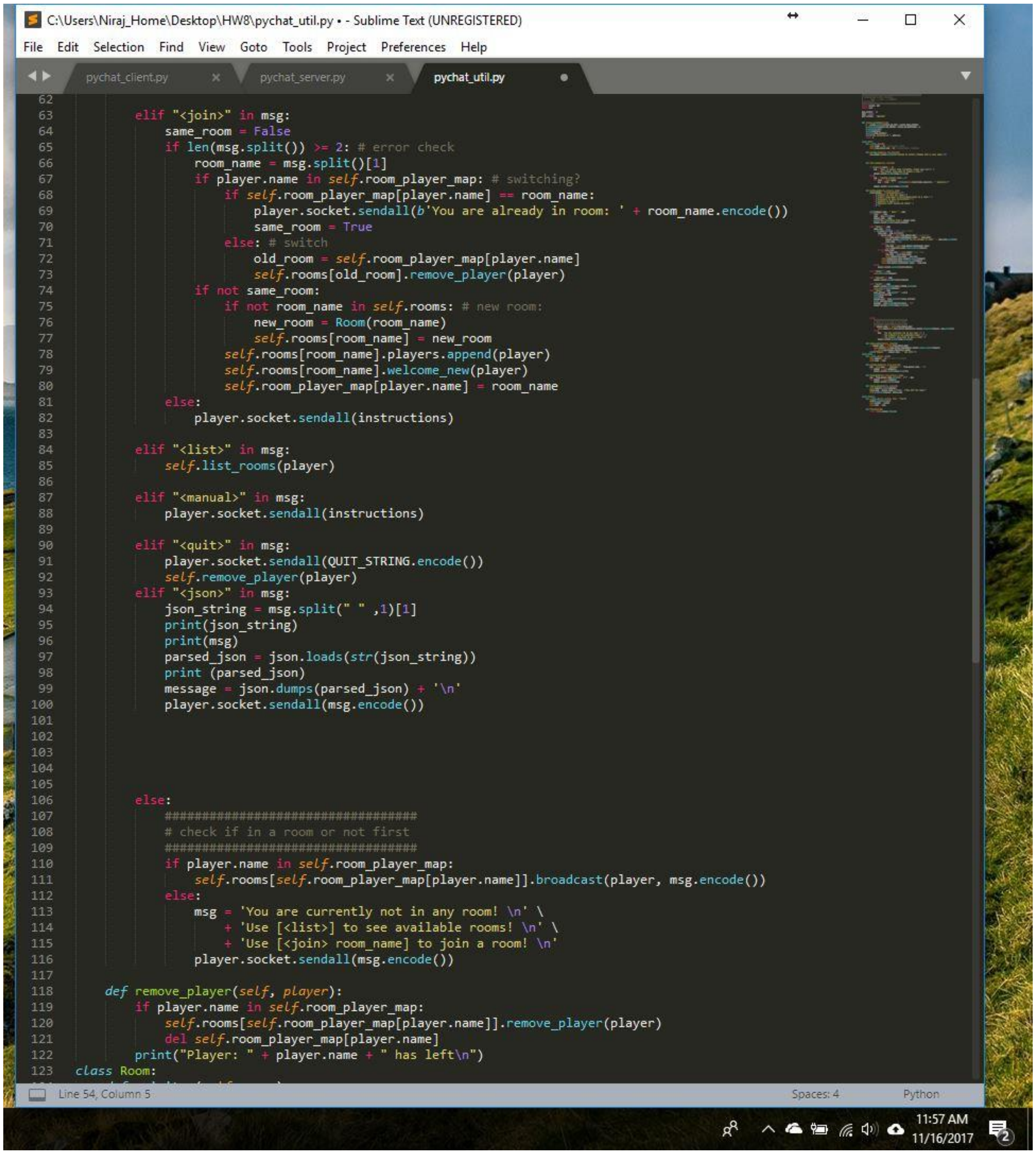
implementing 3-tier structure: Hall --> Room --> Clients;

14-Jun-2013

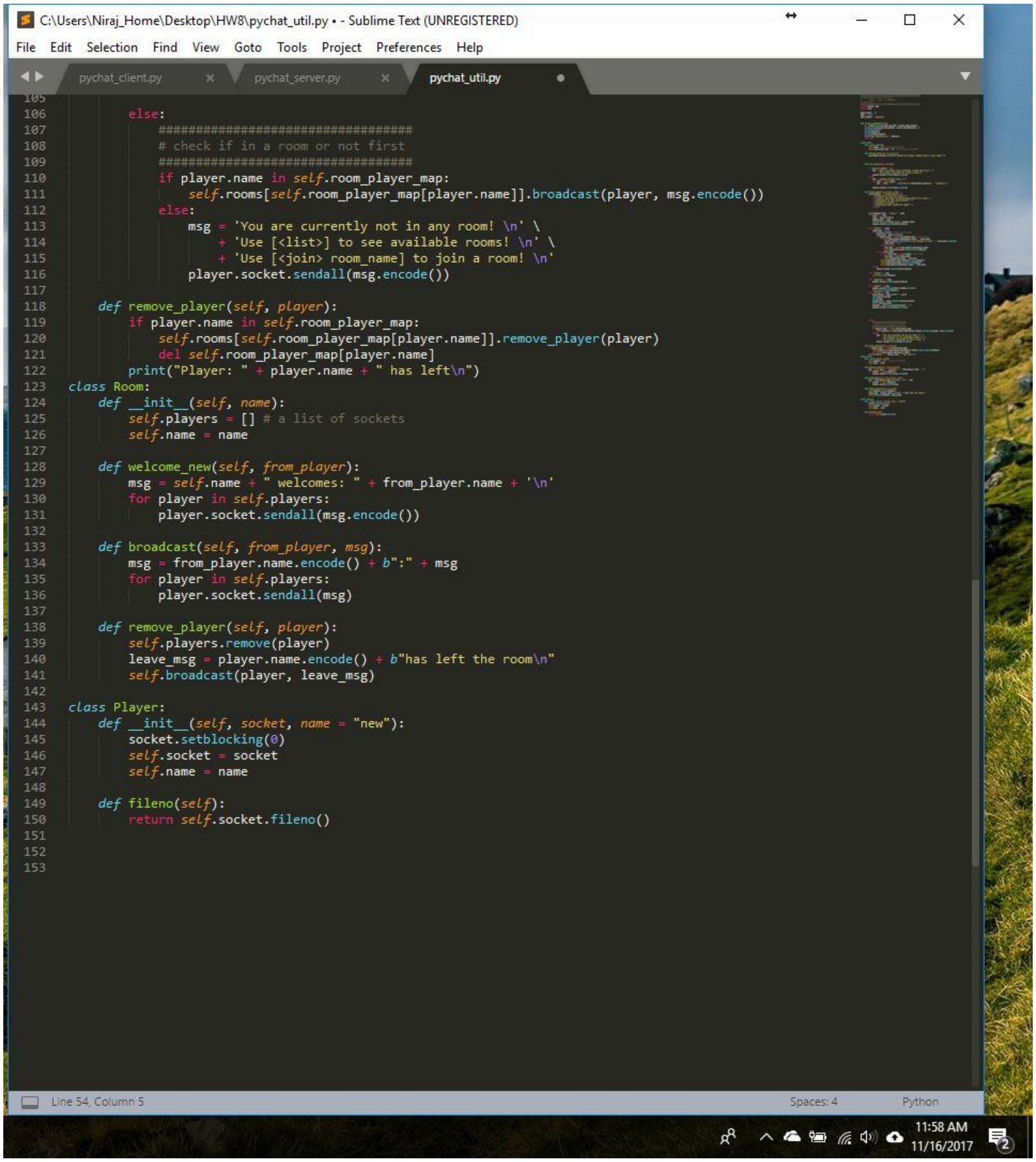
```
############################################################

import select, socket, sys, pdb
from pychat_util import Hall, Room, Player
import pychat_util

READ_BUFFER = 4096

host = sys.argv[1] if len(sys.argv) >= 2 else ''
listen_sock = pychat_util.create_socket((host, pychat_util.PORT))

hall = Hall()
connection_list = []
connection_list.append(listen_sock)

while True:
    ##########################################################
    # Player.fileno()
    # select — Waiting for I/O completion
    ##########################################################
    read_players, write_players, error_sockets = select.select(connection_list, [], [])
    for player in read_players:
        ######################################
        # new connection, player is a socket
        ######################################
        if player is listen_sock:
            new_socket, add = player.accept()
            new_player = Player(new_socket)
            connection_list.append(new_player)
            hall.welcome_new(new_player)

        #####################################
        # new message
        #####################################
        else:
            msg = player.socket.recv(READ_BUFFER)
            if msg:
                msg = msg.decode().lower()
                hall.handle_msg(player, msg)
            else:
                player.socket.close()
                connection_list.remove(player)

    #####################################
    # close error sockets
    #####################################
    for sock in error_sockets:
        sock.close()
        connection_list.remove(sock)
```


C:\Users\Niraj_Home\Desktop\HW8\pychat_util.py • - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
pychat_client.py
pychat_server.py
pychat_util.py
Line 54, Column 5
Spaces: 4
Python
11:56 AM
11/16/2017

```python
############################################################
# implementing 3-tier structure:
#         Hall --> Room --> Clients;
# 14-Jun-2013
###########################################################
import socket, pdb
import json

MAX_CLIENTS = 30
PORT = 22222
QUIT_STRING = '<$quit$>'


def create_socket(address):
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    s.setblocking(0)
    s.bind(address)
    s.listen(MAX_CLIENTS)
    print("Now listening at ", address)
    return s

class Hall:
    def __init__(self):
        self.rooms = {} # {room_name: Room}
        self.room_player_map = {} # {playerName: roomName}

    def welcome_new(self, new_player):
        new_player.socket.sendall(b'Welcome to pychat.\nPlease tell us your name:\n')



    def list_rooms(self, player):

        if len(self.rooms) == 0:
            msg = 'Oops, no active rooms currently. Create your own!\n' \
                + 'Use [<join> room_name] to create a room.\n'
            player.socket.sendall(msg.encode())
        else:
            msg = 'Listing current rooms...\n'
            for room in self.rooms:
                msg += room + ": " + str(len(self.rooms[room].players)) + " player(s)\n"

            player.socket.sendall(msg.encode())

    def handle_msg(self, player, msg):
        instructions = b'Instructions:\n'\
            + b'[<list>] to list all rooms\n'\
            + b'[<join> room_name] to join/create/switch to a room\n' \
            + b'[<manual>] to show instructions\n' \
            + b'[<quit>] to quit\n' \
            + b'Otherwise start typing and enjoy!' \
            + b'\n'


        print(player.name + " says: " + msg)
        if "name:" in msg:
            name = msg.split()[1]
            player.name = name
            print("New connection from:", player.name)
            player.socket.sendall(instructions)

```


C:\Users\Niraj_Home\Desktop\HW8\pychat_util.py • - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
pychat_client.py
pychat_server.py
pychat_util.py
Line 54, Column 5
Spaces: 4
Python
11:57 AM
11/16/2017

```python

        elif "<join>" in msg:
            same_room = False
            if len(msg.split()) >= 2: # error check
                room_name = msg.split()[1]
                if player.name in self.room_player_map: # switching?
                    if self.room_player_map[player.name] == room_name:
                        player.socket.sendall(b'You are already in room: ' + room_name.encode())
                        same_room = True
                    else: # switch
                        old_room = self.room_player_map[player.name]
                        self.rooms[old_room].remove_player(player)
                if not same_room:
                    if not room_name in self.rooms: # new room:
                        new_room = Room(room_name)
                        self.rooms[room_name] = new_room
                    self.rooms[room_name].players.append(player)
                    self.rooms[room_name].welcome_new(player)
                    self.room_player_map[player.name] = room_name
            else:
                player.socket.sendall(instructions)

        elif "<list>" in msg:
            self.list_rooms(player)

        elif "<manual>" in msg:
            player.socket.sendall(instructions)

        elif "<quit>" in msg:
            player.socket.sendall(QUIT_STRING.encode())
            self.remove_player(player)
        elif "<json>" in msg:
            json_string = msg.split(" " ,1)[1]
            print(json_string)
            print(msg)
            parsed_json = json.loads(str(json_string))
            print (parsed_json)
            message = json.dumps(parsed_json) + '\n'
            player.socket.sendall(msg.encode())





        else:
            ##################################
            # check if in a room or not first
            ##################################
            if player.name in self.room_player_map:
                self.rooms[self.room_player_map[player.name]].broadcast(player, msg.encode())
            else:
                msg = 'You are currently not in any room! \n' \
                    + 'Use [<list>] to see available rooms! \n' \
                    + 'Use [<join> room_name] to join a room! \n'
                player.socket.sendall(msg.encode())

    def remove_player(self, player):
        if player.name in self.room_player_map:
            self.rooms[self.room_player_map[player.name]].remove_player(player)
            del self.room_player_map[player.name]
        print("Player: " + player.name + " has left\n")
class Room:
```


C:\Users\Niraj_Home\Desktop\HW8\pychat_util.py • - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
pychat_client.py
pychat_server.py
pychat_util.py
Line 54, Column 5
Spaces: 4
Python
11:58 AM
11/16/2017

```python
        else:
            ##################################
            # check if in a room or not first
            ##################################
            if player.name in self.room_player_map:
                self.rooms[self.room_player_map[player.name]].broadcast(player, msg.encode())
            else:
                msg = 'You are currently not in any room! \n' \
                    + 'Use [<list>] to see available rooms! \n' \
                    + 'Use [<join> room_name] to join a room! \n'
                player.socket.sendall(msg.encode())

    def remove_player(self, player):
        if player.name in self.room_player_map:
            self.rooms[self.room_player_map[player.name]].remove_player(player)
            del self.room_player_map[player.name]
        print("Player: " + player.name + " has left\n")
class Room:
    def __init__(self, name):
        self.players = [] # a list of sockets
        self.name = name

    def welcome_new(self, from_player):
        msg = self.name + " welcomes: " + from_player.name + '\n'
        for player in self.players:
            player.socket.sendall(msg.encode())

    def broadcast(self, from_player, msg):
        msg = from_player.name.encode() + b":" + msg
        for player in self.players:
            player.socket.sendall(msg)

    def remove_player(self, player):
        self.players.remove(player)
        leave_msg = player.name.encode() + b"has left the room\n"
        self.broadcast(player, leave_msg)

class Player:
    def __init__(self, socket, name = "new"):
        socket.setblocking(0)
        self.socket = socket
        self.name = name

    def fileno(self):
        return self.socket.fileno()
```

Source Code:

###

# implementing 3-tier structure:

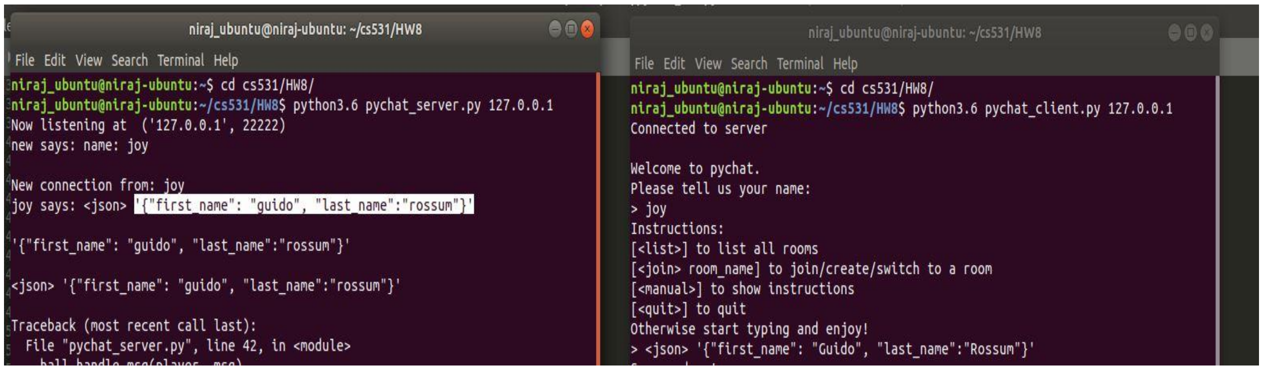
#   Hall --> Room --> Clients;

```
# 14-Jun-2013
#############################################################
import socket, pdb
import json


MAX_CLIENTS = 30
PORT = 22222
QUIT_STRING = '<$quit$>'



def create_socket(address):
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    s.setblocking(0)
    s.bind(address)
    s.listen(MAX_CLIENTS)
    print("Now listening at ", address)
    return s


class Hall:
    def __init__(self):
        self.rooms = {} # {room_name: Room}
        self.room_player_map = {} # {playerName: roomName}


    def welcome_new(self, new_player):
        new_player.socket.sendall(b'Welcome to pychat.\nPlease tell us your name:\n')
```

```
def list_rooms(self, player):

    if len(self.rooms) == 0:
        msg = 'Oops, no active rooms currently. Create your own!\n' \
            + 'Use [<join> room_name] to create a room.\n'
        player.socket.sendall(msg.encode())
    else:
        msg = 'Listing current rooms...\n'
        for room in self.rooms:
            msg += room + ": " + str(len(self.rooms[room].players)) + " player(s)\n"

        player.socket.sendall(msg.encode())

def handle_msg(self, player, msg):
    instructions = b'Instructions:\n'\
        + b'[<list>] to list all rooms\n'\
        + b'[<join> room_name] to join/create/switch to a room\n' \
        + b'[<manual>] to show instructions\n' \
        + b'[<quit>] to quit\n' \
        + b'Otherwise start typing and enjoy!' \
        + b'\n'


    print(player.name + " says: " + msg)
    if "name:" in msg:
        name = msg.split()[1]
        player.name = name
        print("New connection from:", player.name)
        player.socket.sendall(instructions)
```

```
elif "<join>" in msg:
    same_room = False
    if len(msg.split()) >= 2: # error check
        room_name = msg.split()[1]
        if player.name in self.room_player_map: # switching?
            if self.room_player_map[player.name] == room_name:
                player.socket.sendall(b'You are already in room: ' + room_name.encode())
                same_room = True
            else: # switch
                old_room = self.room_player_map[player.name]
                self.rooms[old_room].remove_player(player)
        if not same_room:
            if not room_name in self.rooms: # new room:
                new_room = Room(room_name)
                self.rooms[room_name] = new_room
            self.rooms[room_name].players.append(player)
            self.rooms[room_name].welcome_new(player)
            self.room_player_map[player.name] = room_name
    else:
        player.socket.sendall(instructions)

elif "<list>" in msg:
    self.list_rooms(player)

elif "<manual>" in msg:
    player.socket.sendall(instructions)

elif "<quit>" in msg:
        player.socket.sendall(QUIT_STRING.encode())
        self.remove_player(player)
    elif "<json>" in msg:
        json_string = msg.split(" " ,1)[1]
        print(json_string)
        print(msg)
        parsed_json = json.loads(str(json_string))
        print (parsed_json)
        message = json.dumps(parsed_json) + '\n'
        player.socket.sendall(msg.encode())






    else:
        ####################################
        # check if in a room or not first
        ####################################
        if player.name in self.room_player_map:
            self.rooms[self.room_player_map[player.name]].broadcast(player, msg.encode())
        else:
            msg = 'You are currently not in any room! \n' \
                + 'Use [<list>] to see available rooms! \n' \
                + 'Use [<join> room_name] to join a room! \n'
            player.socket.sendall(msg.encode())

def remove_player(self, player):
    if player.name in self.room_player_map:
            self.rooms[self.room_player_map[player.name]].remove_player(player)
            del self.room_player_map[player.name]
        print("Player: " + player.name + " has left\n")
class Room:
    def __init__(self, name):
        self.players = [] # a list of sockets
        self.name = name

    def welcome_new(self, from_player):
        msg = self.name + " welcomes: " + from_player.name + '\n'
        for player in self.players:
            player.socket.sendall(msg.encode())

    def broadcast(self, from_player, msg):
        msg = from_player.name.encode() + b":" + msg
        for player in self.players:
            player.socket.sendall(msg)

    def remove_player(self, player):
        self.players.remove(player)
        leave_msg = player.name.encode() + b"has left the room\n"
        self.broadcast(player, leave_msg)

class Player:
    def __init__(self, socket, name = "new"):
        socket.setblocking(0)
        self.socket = socket
        self.name = name
```

```
def fileno(self):
    return self.socket.fileno()
```



niraj_ubuntu@niraj-ubuntu: ~/cs531/HW8

```
niraj_ubuntu@niraj-ubuntu:~$ cd cs531/HW8/
niraj_ubuntu@niraj-ubuntu:~/cs531/HW8$ python3.6 pychat_server.py 127.0.0.1
Now listening at  ('127.0.0.1', 22222)
new says: name: joy

New connection from: joy
joy says: <json> '{"first_name": "guido", "last_name":"rossum"}'

'{"first_name": "guido", "last_name":"rossum"}'

<json> '{"first_name": "guido", "last_name":"rossum"}'

Traceback (most recent call last):
  File "pychat_server.py", line 42, in <module>
```

niraj_ubuntu@niraj-ubuntu: ~/cs531/HW8

```
niraj_ubuntu@niraj-ubuntu:~$ cd cs531/HW8/
niraj_ubuntu@niraj-ubuntu:~/cs531/HW8$ python3.6 pychat_client.py 127.0.0.1
Connected to server

Welcome to pychat.
Please tell us your name:
> joy
Instructions:
[<list>] to list all rooms
[<join> room_name] to join/create/switch to a room
[<manual>] to show instructions
[<quit>] to quit
Otherwise start typing and enjoy!
> <json> '{"first_name": "Guido", "last_name":"Rossum"}'
```