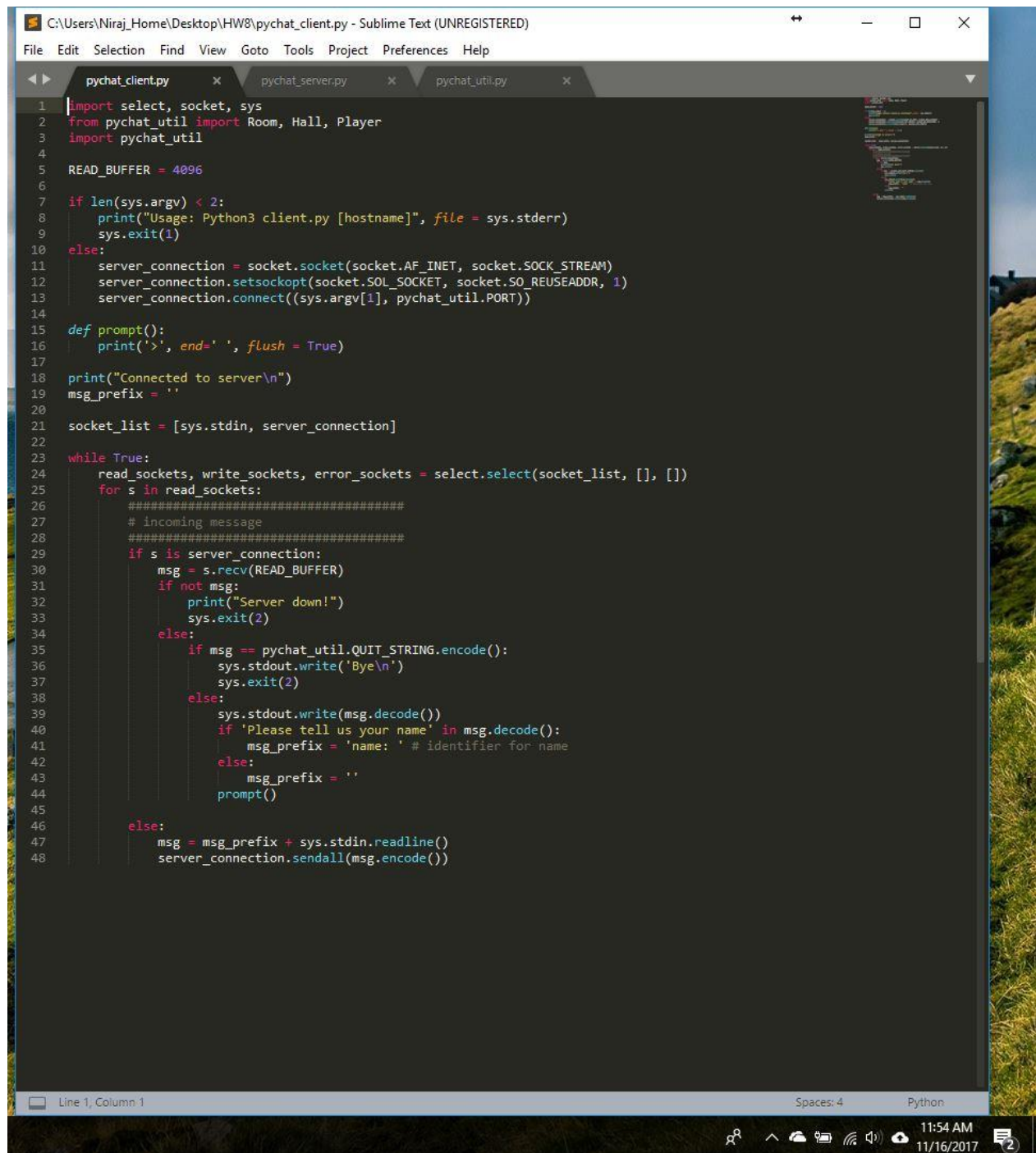


NAME : NIRAJ THANKI

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 x pychat_server.py x pychat_util.py x

1 import select, socket, sys
2 from pychat_util import Room, Hall, Player
3 import pychat_util
4
5 READ_BUFFER = 4096
6
7 if len(sys.argv) < 2:
8     print("Usage: Python3 client.py [hostname]", file = sys.stderr)
9     sys.exit(1)
10 else:
11     server_connection = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
12     server_connection.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
13     server_connection.connect((sys.argv[1], pychat_util.PORT))
14
15 def prompt():
16     print('>', end=' ', flush = True)
17
18 print("Connected to server\n")
19 msg_prefix = ''
20
21 socket_list = [sys.stdin, server_connection]
22
23 while True:
24     read_sockets, write_sockets, error_sockets = select.select(socket_list, [], [])
25     for s in read_sockets:
26         #####
27         # incoming message
28         #####
29         if s is server_connection:
30             msg = s.recv(READ_BUFFER)
31             if not msg:
32                 print("Server down!")
33                 sys.exit(2)
34             else:
35                 if msg == pychat_util.QUIT_STRING.encode():
36                     sys.stdout.write('Bye\n')
37                     sys.exit(2)
38                 else:
39                     sys.stdout.write(msg.decode())
40                     if 'Please tell us your name' in msg.decode():
41                         msg_prefix = 'name: ' # identifier for name
42                     else:
43                         msg_prefix = ''
44                     prompt()
45             else:
46                 msg = msg_prefix + sys.stdin.readline()
47                 server_connection.sendall(msg.encode())
48
```

Line 1, Column 1 Spaces: 4 Python

11:54 AM 11/16/2017

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 x pychat_server.py x pychat_util.py x

1 #####
2 # implementing 3-tier structure: Hall --> Room --> Clients;
3 # 14-Jun-2013
4 #####
5
6 import select, socket, sys, pdb
7 from pychat_util import Hall, Room, Player
8 import pychat_util
9
10 READ_BUFFER = 4096
11
12 host = sys.argv[1] if len(sys.argv) >= 2 else ''
13 listen_sock = pychat_util.create_socket((host, pychat_util.PORT))
14
15 hall = Hall()
16 connection_list = []
17 connection_list.append(listen_sock)
18
19 while True:
20     #####
21     # Player.fileno()
22     # select - Waiting for I/O completion
23     #####
24     read_players, write_players, error_sockets = select.select(connection_list, [], [])
25     for player in read_players:
26         #####
27         # new connection, player is a socket
28         #####
29         if player is listen_sock:
30             new_socket, add = player.accept()
31             new_player = Player(new_socket)
32             connection_list.append(new_player)
33             hall.welcome_new(new_player)
34
35         #####
36         # new message
37         #####
38         else:
39             msg = player.socket.recv(READ_BUFFER)
40             if msg:
41                 msg = msg.decode().lower()
42                 hall.handle_msg(player, msg)
43             else:
44                 player.socket.close()
45                 connection_list.remove(player)
46
47         #####
48         # close error sockets
49         #####
50     for sock in error_sockets:
51         sock.close()
52         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 x pychat_server.py x pychat_util.py

1 #####
2 # implementing 3-tier structure:
3 # Hall --> Room --> Clients;
4 # 14-Jun-2013
5 #####
6 import socket, pdb
7 import json
8
9 MAX_CLIENTS = 30
10 PORT = 22222
11 QUIT_STRING = '<quit$>'
12
13
14 def create_socket(address):
15     s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
16     s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
17     s.setblocking(0)
18     s.bind(address)
19     s.listen(MAX_CLIENTS)
20     print("Now listening at ", address)
21     return s
22
23 class Hall:
24     def __init__(self):
25         self.rooms = {} # {room_name: Room}
26         self.room_player_map = {} # {playerName: roomName}
27
28     def welcome_new(self, new_player):
29         new_player.socket.sendall(b'Welcome to pychat.\nPlease tell us your name:\n')
30
31
32
33     def list_rooms(self, player):
34
35         if len(self.rooms) == 0:
36             msg = 'Oops, no active rooms currently. Create your own!\n' \
37                 + 'Use [<join> room_name] to create a room.\n'
38             player.socket.sendall(msg.encode())
39         else:
40             msg = 'Listing current rooms...\n'
41             for room in self.rooms:
42                 msg += room + ": " + str(len(self.rooms[room].players)) + " player(s)\n"
43
44             player.socket.sendall(msg.encode())
45
46     def handle_msg(self, player, msg):
47         instructions = b'Instructions:\n'\
48             + b' [<list>] to list all rooms\n'\
49             + b' [<join> room_name] to join/create/switch to a room\n' \
50             + b' [<manual>] to show instructions\n' \
51             + b' [<quit>] to quit\n' \
52             + b' Otherwise start typing and enjoy!' \
53             + b'\n'
54
55
56         print(player.name + " says: " + msg)
57         if "name:" in msg:
58             name = msg.split()[1]
59             player.name = name
60             print("New connection from:", player.name)
61             player.socket.sendall(instructions)
62
```

Line 54, Column 5 Spaces: 4 Python 11:56 AM 11/16/2017



```
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 x pychat_server.py x pychat_util.py

62
63     elif "<join>" in msg:
64         same_room = False
65         if len(msg.split()) >= 2: # error check
66             room_name = msg.split()[1]
67             if player.name in self.room_player_map: # switching?
68                 if self.room_player_map[player.name] == room_name:
69                     player.socket.sendall(b'You are already in room: ' + room_name.encode())
70                     same_room = True
71                 else: # switch
72                     old_room = self.room_player_map[player.name]
73                     self.rooms[old_room].remove_player(player)
74             if not same_room:
75                 if not room_name in self.rooms: # new room:
76                     new_room = Room(room_name)
77                     self.rooms[room_name] = new_room
78                     self.rooms[room_name].players.append(player)
79                     self.rooms[room_name].welcome_new(player)
80                     self.room_player_map[player.name] = room_name
81             else:
82                 player.socket.sendall(instructions)
83
84     elif "<list>" in msg:
85         self.list_rooms(player)
86
87     elif "<manual>" in msg:
88         player.socket.sendall(instructions)
89
90     elif "<quit>" in msg:
91         player.socket.sendall(QUIT_STRING.encode())
92         self.remove_player(player)
93     elif "<json>" in msg:
94         json_string = msg.split(" ", 1)[1]
95         print(json_string)
96         print(msg)
97         parsed_json = json.loads(str(json_string))
98         print(parsed_json)
99         message = json.dumps(parsed_json) + '\n'
100         player.socket.sendall(msg.encode())
101
102
103
104
105
106     else:
107         #####
108         # check if in a room or not first
109         #####
110         if player.name in self.room_player_map:
111             self.rooms[self.room_player_map[player.name]].broadcast(player, msg.encode())
112         else:
113             msg = 'You are currently not in any room! \n' \
114                 + 'Use [<list>] to see available rooms! \n' \
115                 + 'Use [<join> room_name] to join a room! \n'
116             player.socket.sendall(msg.encode())
117
118     def remove_player(self, player):
119         if player.name in self.room_player_map:
120             self.rooms[self.room_player_map[player.name]].remove_player(player)
121             del self.room_player_map[player.name]
122             print("Player: " + player.name + " has left\n")
123
124 class Room:
125
Line 54, Column 5 Spaces: 4 Python
```



```
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 x pychat_server.py x pychat_util.py

105
106     else:
107         #####
108         # check if in a room or not first
109         #####
110         if player.name in self.room_player_map:
111             self.rooms[self.room_player_map[player.name]].broadcast(player, msg.encode())
112         else:
113             msg = 'You are currently not in any room! \n' \
114                 + 'Use [<list>] to see available rooms! \n' \
115                 + 'Use [<join> room_name] to join a room! \n'
116             player.socket.sendall(msg.encode())
117
118     def remove_player(self, player):
119         if player.name in self.room_player_map:
120             self.rooms[self.room_player_map[player.name]].remove_player(player)
121             del self.room_player_map[player.name]
122             print("Player: " + player.name + " has left\n")
123
124     class Room:
125     def __init__(self, name):
126         self.players = [] # a list of sockets
127         self.name = name
128
129     def welcome_new(self, from_player):
130         msg = self.name + " welcomes: " + from_player.name + '\n'
131         for player in self.players:
132             player.socket.sendall(msg.encode())
133
134     def broadcast(self, from_player, msg):
135         msg = from_player.name.encode() + b": " + msg
136         for player in self.players:
137             player.socket.sendall(msg)
138
139     def remove_player(self, player):
140         self.players.remove(player)
141         leave_msg = player.name.encode() + b"has left the room\n"
142         self.broadcast(player, leave_msg)
143
144     class Player:
145     def __init__(self, socket, name = "new"):
146         socket.setblocking(0)
147         self.socket = socket
148         self.name = name
149
150     def fileno(self):
151         return self.socket.fileno()
152
153
Line 54, Column 5 Spaces: 4 Python 11:58 AM 11/16/2017
```

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!\n' \
        + 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:

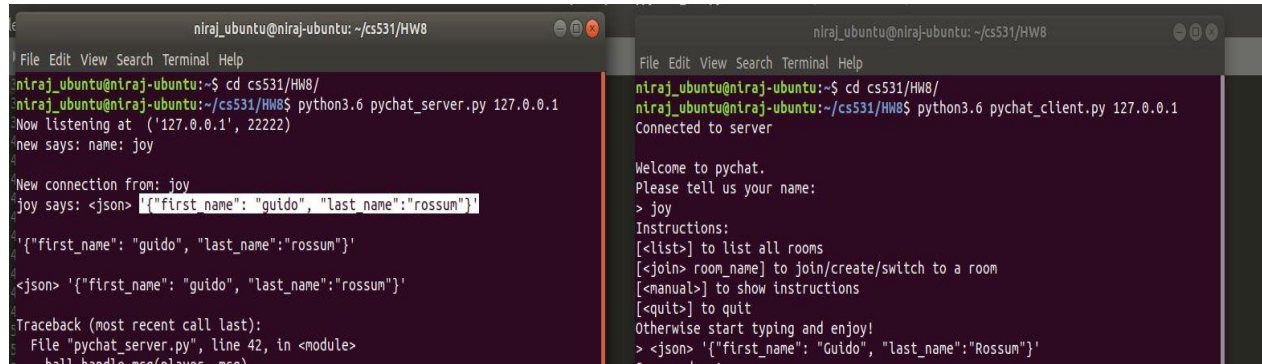
```
    def __init__(self, socket, name = "new"):
        socket.setblocking(0)

        self.socket = socket

        self.name = name
```



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



The image shows two terminal windows side-by-side. The left window is titled 'niraj\_ubuntu@niraj-ubuntu: ~/cs531/HW8' and shows the execution of a Python chat server. The right window is also titled 'niraj\_ubuntu@niraj-ubuntu: ~/cs531/HW8' and shows the execution of a Python chat client.

```
niraj_ubuntu@niraj-ubuntu: ~/cs531/HW8  
File Edit View Search Terminal Help  
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>  
    ball.handle_new_players_msg()
```

```
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"}'
```