Terna Engineering College
Computer Engineering Department
Program: Sem VIII

Course: Distributed Computing Lab (CSL802)

Faculty: Rohini Patil

Experiment No. 2

A.1 Aim: To Implement Group Communication as a Chat application using socket programming.

PART B
(PART B: TO BE COMPLETED BY STUDENTS)

Roll No. 50	**Name:** AMEY MAHENDRA THAKUR
Class: BE COMPS B 50	**Batch:** B3
Date of Experiment: 20-01-2022	**Date of Submission:** 20-01-2022
Grade:	

B.1 Software Code written by student:

- **Server.py**

```
import time, socket, sys

new_socket = socket.socket()
host_name = socket.gethostname()
s_ip = socket.gethostbyname(host_name)

port = 8080

print("Welcome to the Chat Room\n")
new_socket.bind((host_name, port))
print("Binding Successful!")
print("This is your IP: ", s_ip)

name = input('Enter name: ')
```

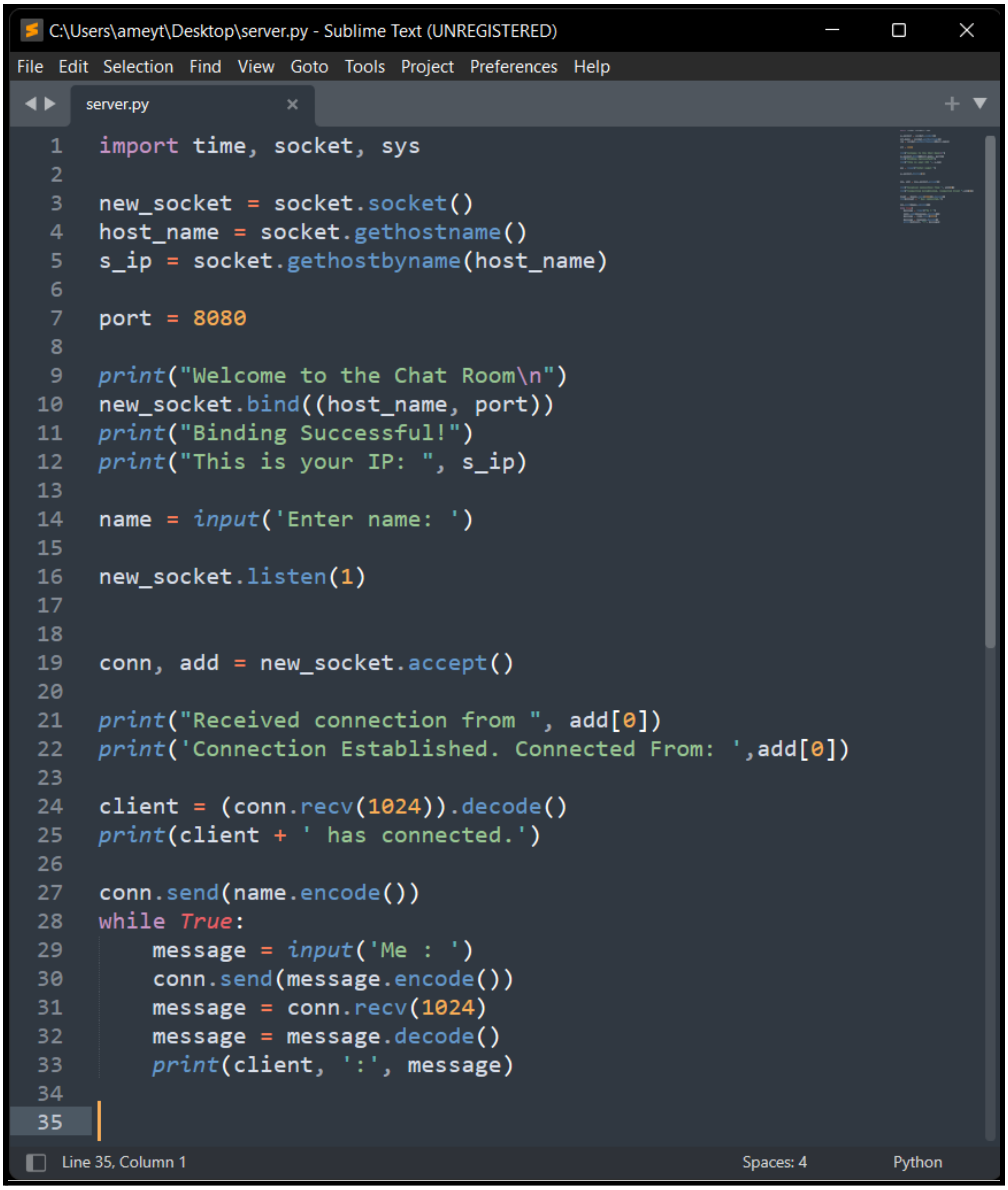
```
new_socket.listen(1)
conn, add = new_socket.accept()

print("Received connection from ", add[0])
print('Connection Established. Connected From: ',add[0])
client = (conn.recv(1024)).decode()
print(client + ' has connected.')

conn.send(name.encode())
while True:
    message = input('Me : ')
    conn.send(message.encode())
    message = conn.recv(1024)
    message = message.decode()
    print(client, ':', message)
```

- **Client.py**

```
import time, socket, sys

socket_server = socket.socket()
server_host = socket.gethostname()
ip = socket.gethostbyname(server_host)
sport = 8080

print("Welcome to the Chat Room\n")
server_host = input('Enter friend\'s IP address:')
name = input('Enter Friend\'s name: ')

socket_server.connect((server_host, sport))

socket_server.send(name.encode())
server_name = socket_server.recv(1024)
server_name = server_name.decode()

print(server_name,' has joined...')
while True:
    message = (socket_server.recv(1024)).decode()
    print(server_name, ":", message)
    message = input("Me : ")
    socket_server.send(message.encode())
```

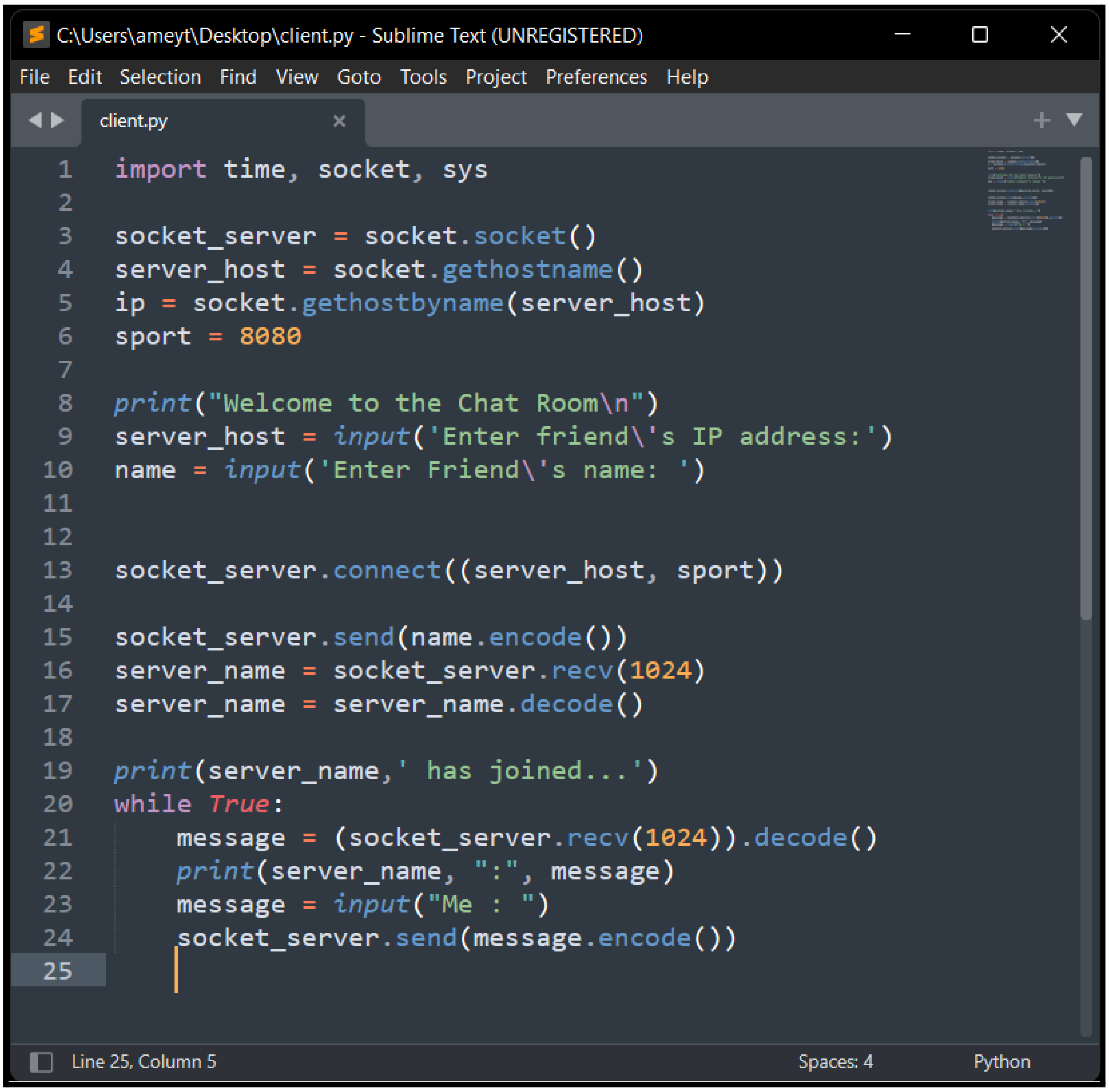
B.2 Input and Output:

- **<u>Server.py</u>**


C:\Users\ameyt\Desktop\server.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
server.py
Line 35, Column 1
Spaces: 4
Python

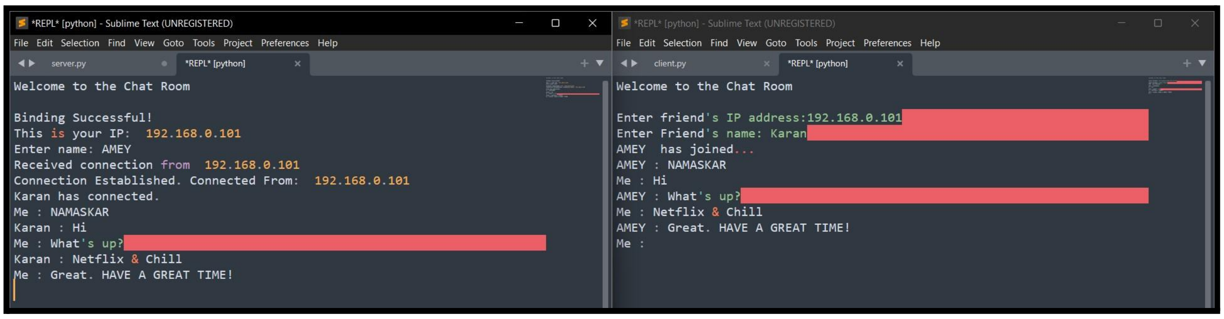
```python
import time, socket, sys

new_socket = socket.socket()
host_name = socket.gethostname()
s_ip = socket.gethostbyname(host_name)

port = 8080

print("Welcome to the Chat Room\n")
new_socket.bind((host_name, port))
print("Binding Successful!")
print("This is your IP: ", s_ip)

name = input('Enter name: ')

new_socket.listen(1)


conn, add = new_socket.accept()

print("Received connection from ", add[0])
print('Connection Established. Connected From: ',add[0])

client = (conn.recv(1024)).decode()
print(client + ' has connected.')

conn.send(name.encode())
while True:
    message = input('Me : ')
    conn.send(message.encode())
    message = conn.recv(1024)
    message = message.decode()
    print(client, ':', message)

```

- **Client.py**



C:\Users\ameyt\Desktop\client.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
client.py
Line 25, Column 5
Spaces: 4
Python

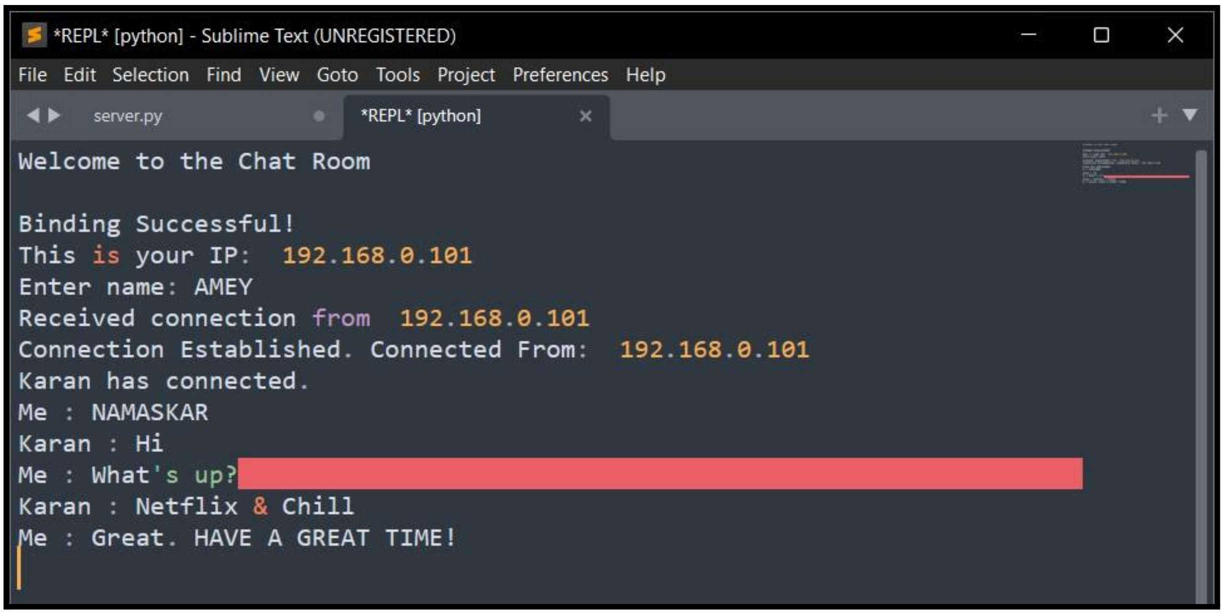
```python
import time, socket, sys

socket_server = socket.socket()
server_host = socket.gethostname()
ip = socket.gethostbyname(server_host)
sport = 8080

print("Welcome to the Chat Room\n")
server_host = input('Enter friend\'s IP address:')
name = input('Enter Friend\'s name: ')


socket_server.connect((server_host, sport))

socket_server.send(name.encode())
server_name = socket_server.recv(1024)
server_name = server_name.decode()

print(server_name,' has joined...')
while True:
    message = (socket_server.recv(1024)).decode()
    print(server_name, ":", message)
    message = input("Me : ")
    socket_server.send(message.encode())
    
```



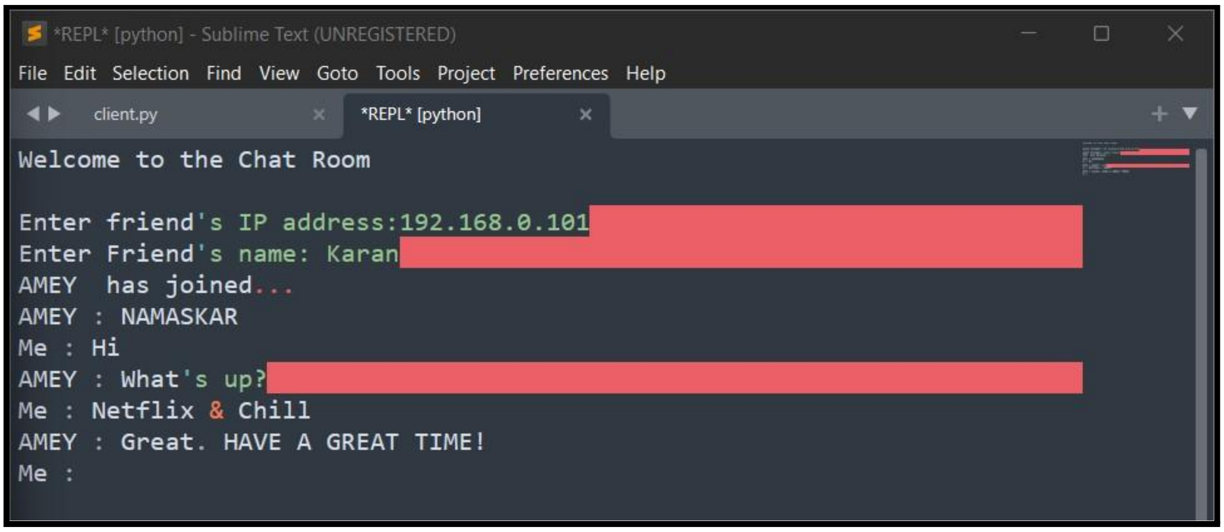
REPL [python] - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
server.py
REPL [python]
REPL [python] - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
client.py
REPL [python]

```
Welcome to the Chat Room

Binding Successful!
This is your IP:  192.168.0.101
Enter name: AMEY
Received connection from  192.168.0.101
Connection Established. Connected From:  192.168.0.101
Karan has connected.
Me : NAMASKAR
Karan : Hi
Me : What's up?
Karan : Netflix & Chill
Me : Great. HAVE A GREAT TIME!
```

```
Welcome to the Chat Room

Enter friend's IP address:192.168.0.101
Enter Friend's name: Karan
AMEY  has joined...
AMEY : NAMASKAR
Me : Hi
AMEY : What's up?
Me : Netflix & Chill
AMEY : Great. HAVE A GREAT TIME!
Me :
```


REPL [python] - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
server.py
REPL [python]

```
Welcome to the Chat Room

Binding Successful!
This is your IP:  192.168.0.101
Enter name: AMEY
Received connection from  192.168.0.101
Connection Established. Connected From:  192.168.0.101
Karan has connected.
Me : NAMASKAR
Karan : Hi
Me : What's up?
Karan : Netflix & Chill
Me : Great. HAVE A GREAT TIME!
```


REPL [python] - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
client.py
REPL [python]

```
Welcome to the Chat Room

Enter friend's IP address:192.168.0.101
Enter Friend's name: Karan
AMEY  has joined...
AMEY : NAMASKAR
Me : Hi
AMEY : What's up?
Me : Netflix & Chill
AMEY : Great. HAVE A GREAT TIME!
Me :
```

B.3 Observations and learning:

In this experiment, we learnt socket programming to establish a server-client network and implemented a chat application in python.

B.4 Conclusion:

We successfully implemented a chat application using socket programming in python.

B.5 Question of Curiosity.

Q1: Server Socket consists of port address and IP address or only port address.

ANS:

- A server socket has a port address as well as an IP address. Every device on a TCP/IP network requires an IP address. The gadget is identified by its IP address. A socket is associated with a port number so that the TCP layer can identify the application to which data is being transmitted.

Q2: Compare UDP socket programming and TCP socket programming
ANS:

Feature	TCP	UDP
Connection status	Requires an established connection to transmit data (connection should be closed once transmission is complete)	Connectionless protocol with no requirements for opening, maintaining, or terminating a connection
Data sequencing	Able to sequence	Unable to sequence
Guaranteed delivery	Can guarantee delivery of data to the destination router	Cannot guarantee delivery of data to the destination
Speed	Slower than UDP	Faster than TCP
Broadcasting	Does not support Broadcasting	Does support Broadcasting
Optimal use	Used by HTTPS, HTTP, SMTP, POP, FTP, etc	Video conferencing, streaming, DNS, VoIP, etc
Error checking	Extensive error checking and acknowledgment of data	Basic error checking mechanism using checksums

Q3. The distributed system uses a ___________architecture to break down the complexity of system design. The ______________ is the distributed software that drives the distributed system, while providing transparency of heterogeneity at the platform level.

1. Layered, Middleware
2. Message-passing, CORBA
3. Tree, RPC
4. Loosely coupled, MPI

ANS: 1. Layered Middleware

Q4: The ___________________allows the server and client to authenticate each other and to negotiate an encryption algorithm and cryptographic keys before the application protocol transmits or receives its first byte of data.

1. SSL record protocol
2. SSL TCP segment
3. SSL handshake protocol
4. None of these

ANS: 3. SSL handshake protocol