EXP:12	End –End Communication at Transport Layer

Aim

To implement an echo client-server using TCP/UDP sockets and to implement a chat program using socket programming.

Algorithm / Procedure

1. **Implement** a basic **TCP Echo Server** that listens for connections, receives a message, and sends the same message back to the client.
2. **Implement** a basic **TCP Echo Client** that connects to the server, sends a message, and prints the echoed response.
3. **Implement** a simple **Chat Program** using TCP sockets, allowing two separate programs (client and server) to send and receive messages interactively.
4. *(Optional)* Implement the same Echo Client-Server using **UDP** sockets to compare connection-oriented vs. connectionless communication.

Code:

tcp_client.py

```
import socket


s = socket.socket()

s.connect(('localhost', 12345))


while True:

    msg = input("Client: ")

    if msg.lower() == 'exit':

        break

    s.send(msg.encode())

    data = s.recv(1024).decode()

    print("From Server:", data)
```

```
s.close()
```

tcp_server.py

```
import socket

s = socket.socket()
s.bind(('localhost', 12345))
s.listen(1)
print("Server ready, waiting for connection...")
conn, addr = s.accept()
print("Connected with", addr)

while True:
    data = conn.recv(1024).decode()
    if not data:
        break
    print("From Client:", data)
    conn.send(data.encode())

conn.close()
```

Client_server.py

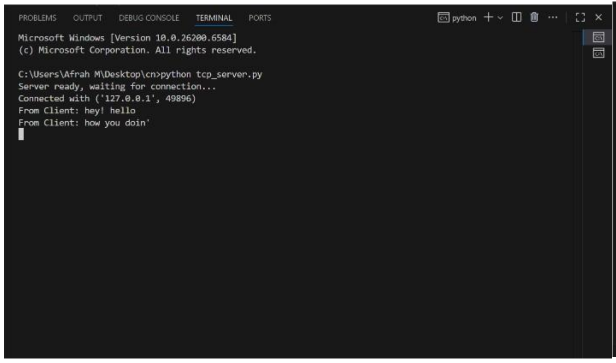
```
import socket

s = socket.socket()
s.bind(('localhost', 12345))
```

```
s.listen(1)
print("Chat Server waiting for connection...")
conn, addr = s.accept()
print("Connected with", addr)

while True:
    msg = conn.recv(1024).decode()
    if msg.lower() == 'bye':
        print("Client ended chat.")
        break
    print("Client:", msg)
    reply = input("Server: ")
    conn.send(reply.encode())

conn.close()
```

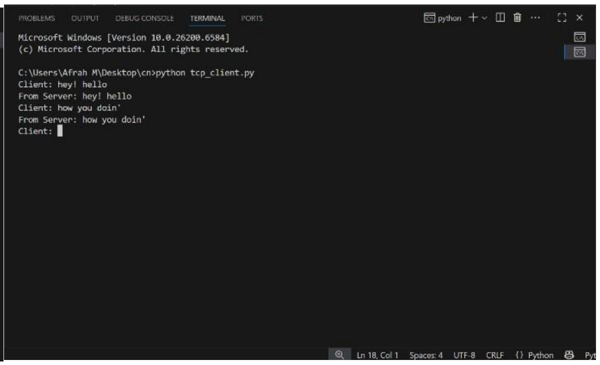
client.py

```
import socket

s = socket.socket()
s.connect(('localhost', 12345))
print("Connected to server. Type 'bye' to end.")

while True:
    msg = input("Client: ")
    s.send(msg.encode())
    if msg.lower() == 'bye':
        break
    reply = s.recv(1024).decode()
    print("Server:", reply)

s.close()
```

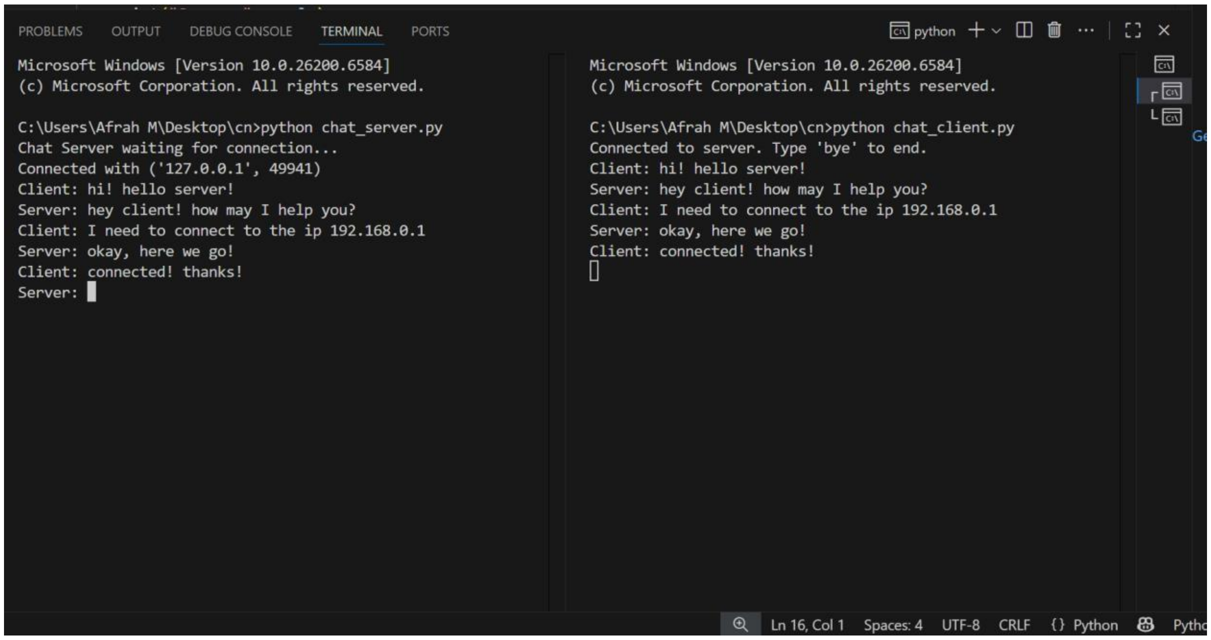
Output:



```
Microsoft Windows [Version 10.0.26200.6584]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Afrah M\Desktop\cn>python tcp_server.py
Server ready, waiting for connection...
Connected with ('127.0.0.1', 49896)
From Client: hey! hello
From Client: how you doin'
```



```
Microsoft Windows [Version 10.0.26200.6584]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Afrah M\Desktop\cn>python tcp_client.py
Client: hey! hello
From Server: hey! hello
Client: how you doin'
From Server: how you doin'
Client:
```



```
Microsoft Windows [Version 10.0.26200.6584]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Afrah M\Desktop\cn>python chat_server.py
Chat Server waiting for connection...
Connected with ('127.0.0.1', 49941)
Client: hi! hello server!
Server: hey client! how may I help you?
Client: I need to connect to the ip 192.168.0.1
Server: okay, here we go!
Client: connected! thanks!
Server:
```

```
Microsoft Windows [Version 10.0.26200.6584]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Afrah M\Desktop\cn>python chat_client.py
Connected to server. Type 'bye' to end.
Client: hi! hello server!
Server: hey client! how may I help you?
Client: I need to connect to the ip 192.168.0.1
Server: okay, here we go!
Client: connected! thanks!
```

Result

An echo client-server and a chat program were successfully implemented using socket programming. The experiment demonstrated end-to-end communication at the Transport Layer using TCP sockets.