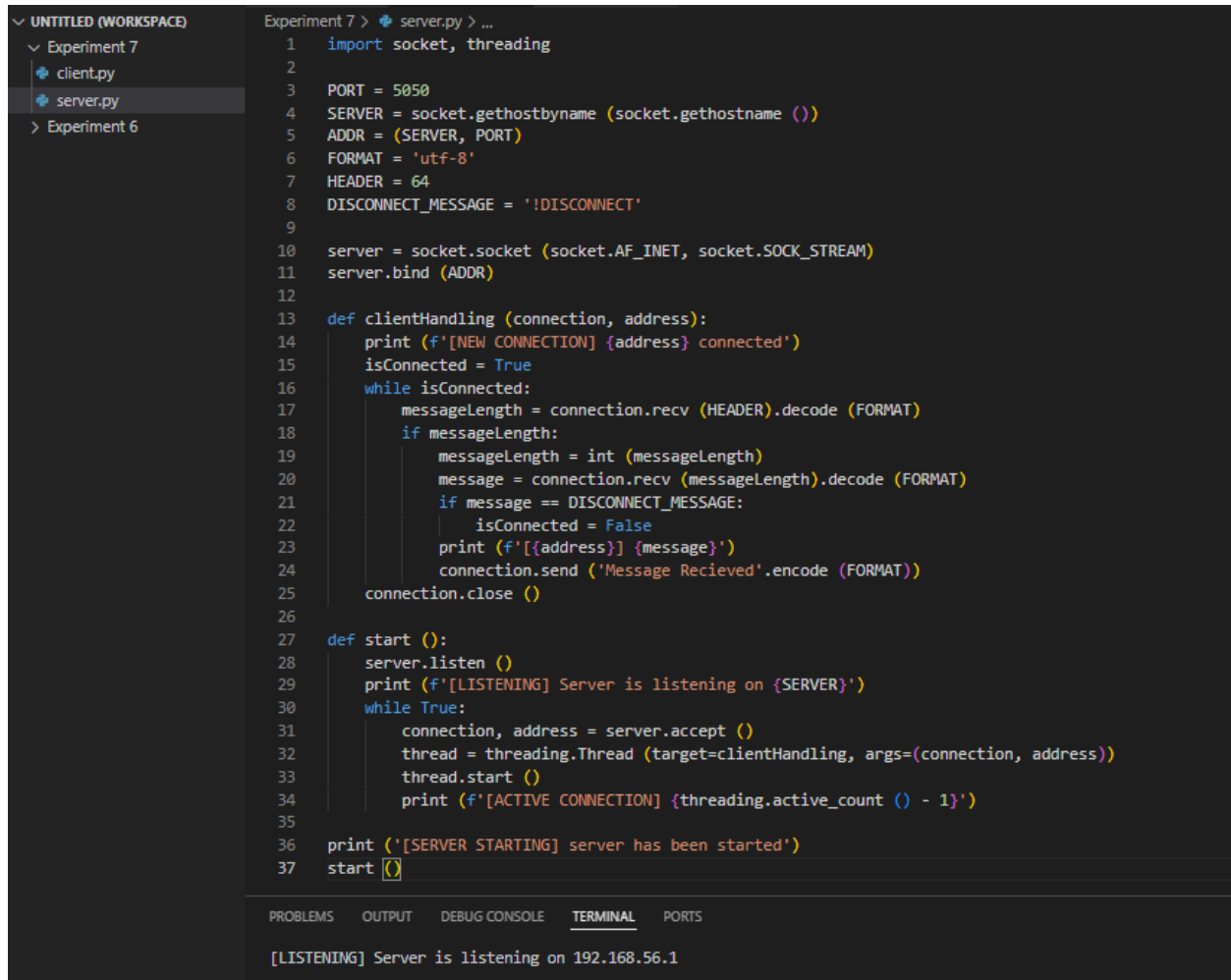


Sumeet Kadam Roll No.23



The image shows a code editor interface with a file explorer on the left and a code editor on the right. The file explorer shows a workspace named 'UNTITLED (WORKSPACE)' containing two files: 'client.py' and 'server.py'. The 'server.py' file is selected and its code is displayed in the editor. The code is a Python script for a multi-threaded server. It imports 'socket' and 'threading', sets a port of 5050, and gets the host name. It then binds a socket to the host and port. A function 'clientHandling' is defined to process incoming connections, including a loop to receive messages and a check for a disconnect message. A 'start' function is defined to listen for connections and handle them in separate threads. The script prints messages for new connections, active connections, and when the server starts. The terminal at the bottom shows the output of the script, indicating it is listening on 192.168.56.1.

```
Experiment 7 > server.py > ...
1  import socket, threading
2
3  PORT = 5050
4  SERVER = socket.gethostname (socket.gethostname ())
5  ADDR = (SERVER, PORT)
6  FORMAT = 'utf-8'
7  HEADER = 64
8  DISCONNECT_MESSAGE = '!DISCONNECT'
9
10 server = socket.socket (socket.AF_INET, socket.SOCK_STREAM)
11 server.bind (ADDR)
12
13 def clientHandling (connection, address):
14     print (f'[NEW CONNECTION] {address} connected')
15     isConnected = True
16     while isConnected:
17         messageLength = connection.recv (HEADER).decode (FORMAT)
18         if messageLength:
19             messageLength = int (messageLength)
20             message = connection.recv (messageLength).decode (FORMAT)
21             if message == DISCONNECT_MESSAGE:
22                 isConnected = False
23             print (f'[{address}] {message}')
24             connection.send ('Message Recieved'.encode (FORMAT))
25     connection.close ()
26
27 def start ():
28     server.listen ()
29     print (f'[LISTENING] Server is listening on {SERVER}')
30     while True:
31         connection, address = server.accept ()
32         thread = threading.Thread (target=clientHandling, args=(connection, address))
33         thread.start ()
34         print (f'[ACTIVE CONNECTION] {threading.active_count () - 1}')
35
36 print ('[SERVER STARTING] server has been started')
37 start ()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

[LISTENING] Server is listening on 192.168.56.1

UNTITLED (WORKSPACE)

Experiment 7

client.py

server.py

Experiment 6

Experiment 7 > client.py > ...

```
1  import socket
2
3  PORT = 5050
4  SERVER = '192.168.56.1'
5  ADDR = (SERVER, PORT)
6  FORMAT = 'utf-8'
7  HEADER = 64
8  DISCONNECT_MESSAGE = '!DISCONNECT'
9
10 client = socket.socket (socket.AF_INET, socket.SOCK_STREAM)
11 client.connect (ADDR)
12
13 def send (messages): (function) encode: Any
14     message = messages.encode (FORMAT)
15     messageLength = len (message)
16     sendLength = str (messageLength).encode (FORMAT)
17     sendLength += b' ' * (HEADER - len (sendLength))
18     client.send (sendLength)
19     client.send (message)
20     print (client.recv (2045).decode (FORMAT))
21
22 if __name__ == '__main__':
23     ### Pass the Hello World message as an input
24     # ... Here
25     send("Hello World")
26
27     send (DISCONNECT_MESSAGE)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
[LISTENING] Server is listening on 192.168.56.1
[NEW CONNECTION] ('192.168.56.1', 50792) connected
[ACTIVE CONNECTION] 1
[('192.168.56.1', 50792)] Hello World
[('192.168.56.1', 50792)] !DISCONNECT
```