**Instruction :-**

**Certainly! Here's how to complete the instructions for running the client script and finalize the instruction file:**

**---**

**## Client-Server Application using TCP Sockets in Python**

**### Project Overview**

**This project demonstrates a simple implementation of a client-server architecture using TCP sockets in Python. The server listens for incoming connections on a specific port, and the client connects to the server to exchange messages.**

**### Instructions**

**#### Step 1: Create the Server Script**

**1. Create a new Python file named `server.py` and add the following code:**

**```python**

**import socket**

**# Create a socket object**

**server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)**

**# Define the port on which you want to connect**

**port = 50000**

**# Bind to the port**

**server\_socket.bind(("", port))**

**# Queue up to 5 requests**

**server\_socket.listen(5)**

**print("Server started. Waiting for connections...")**

**# Accept a connection**

**client\_socket, address = server\_socket.accept()**

**print("Got a connection from", address)**

**while True:**

**# Receive message from client**

**message = client\_socket.recv(1024).decode()**

**if not message:**

**break**

**print("Received message:", message)**

**# Send response back to client**

**response = input("Server: ")**

**client\_socket.send(response.encode())**

**# Close the connection**

**client\_socket.close()**

**```**

**#### Step 2: Create the Client Script**

**1. Create a new Python file named `client.py` and add the following code:**

**```python**

**import socket**

**# Create a socket object**

**client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)**

**# Define the port on which you want to connect**

**port = 50000**

**# Connect to the server**

**client\_socket.connect(("localhost", port))**

**print("Connected to server. Type 'exit' to quit.")**

**while True:**

**# Send message to server**

**message = input("Client: ")**

**if message.lower() == "exit":**

**break**

**client\_socket.send(message.encode())**

**# Receive response from server**

**response = client\_socket.recv(1024).decode()**

**print("Server:", response)**

**# Close the connection**

**client\_socket.close()**

**```**

**#### Step 3: Run the Server Script**

**1. Open a terminal or command prompt and navigate to the directory where you saved the `server.py` file.**

**2. Run the server script by typing:**

**```bash**

**python server.py**

**```**

**3. The server will start listening for incoming connections on port 50000.**

**#### Step 4: Run the Client Script**

**1. Open a new terminal or command prompt and navigate to the directory where you saved the `client.py` file.**

**2. Run the client script by typing:**

**```bash**

**python client.py**

**```**

**3. You should see a message indicating that the client is connected to the server. You can start sending messages to the server and receive responses.**

**### Additional Information**

**- To stop the server, you can use `Ctrl + C` in the terminal where the server is running.**

**- The client can be run multiple times from different terminals to simulate multiple clients connecting to the server.**

**### Next Steps**

**- Explore advanced topics like \*\*multi-threading\*\* and \*\*asynchronous communication\*\* for handling multiple clients.**

**- Add \*\*error handling\*\* and \*\*security features\*\* to make the client-server application more robust.**

**- Apply this knowledge to real-world \*\*projects and applications\*\* for better network communication.**

**---**