Socket Programming for Client-Server Communication

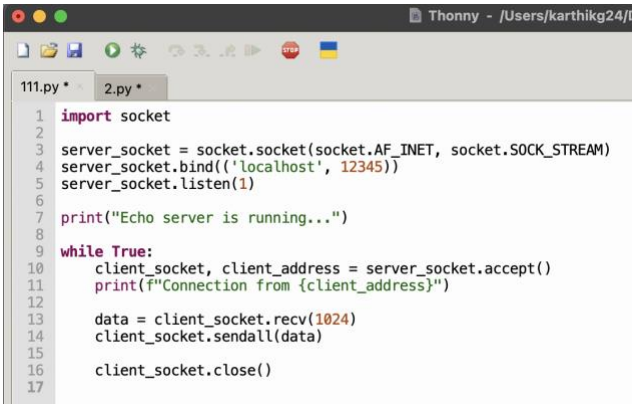
Objective:
The objective of this lab assignment is to familiarize with socket programming concepts practically. I will implement client-server communication using sockets in Python.

Setting Up:

- Set up a development environment with a programming language of your choice (e.g., Python, C/C++, Java).

Task 1: Simple Echo Server

- Implement a simple echo server that listens for incoming connections on a specified port.
- The server should receive data from the client, and then send back the same data to the client.
-



Description:
- Setting up the server socket: A socket object is created with AF_INET (IPv4 addressing) and SOCK_STREAM (TCP protocol).
- Binding to an address and port: The server is explicitly associated with a network interface (IP address) and port number to accept connections.
- Listening for connections: The server begins listening for requests using the listen() method, allowing multiple clients to connect sequentially.
- Accepting connections: When a client initiates a request, the server accepts it with the accept() method, which returns a new socket for communication with that client.
- Receiving and echoing data: Using the recv() method, the server retrieves the data sent by the client and immediately sends it back with the sendall() method.
- Closing the connection: Once the communication is complete, the server closes the client-specific connection to free resources.

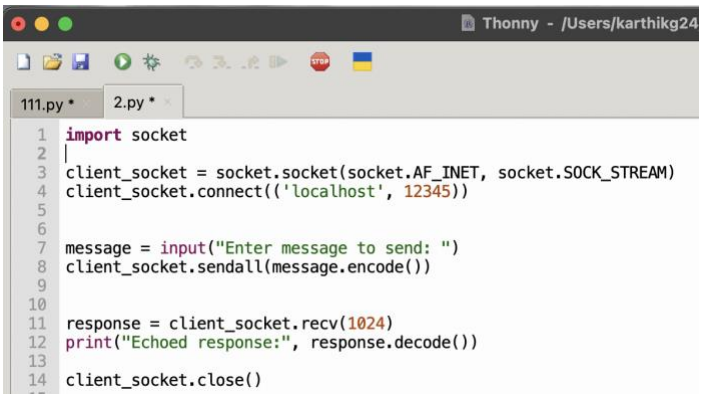
Observations:
-The server operates continuously, waiting for clients to connect and then echoing back their data.

-This implementation demonstrates a fundamental request-response pattern in networking.
- However, the server lacks robust **error handling** (e.g., unexpected disconnections, invalid input), which makes it vulnerable to ungraceful client behavior.

Code Snippets:
1. `server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)` - This line creates a socket object for the server to listen for incoming connections.
2. `server_socket.bind((HOST, PORT))` - This line binds the socket to the specified host and port number.
3. `server_socket.listen()` - This line starts listening for incoming connections on the specified port.
4. `client_socket, client_address = server_socket.accept()` - This line accepts a connection from a client and returns a new socket and client address.
5. `data = client_socket.recv(1024)` - This line receives data from the client.
6. `client_socket.sendall(data)` - This line sends the received data back to the client.

Task 2: Echo Client

- I will develop an echo client that connects to the echo server.
- The client should take user input, send it to the server, receive the echoed response, and display it to the user.



Description:
- Creating a client socket: Similar to the server, a socket object is created with IPv4 addressing and TCP protocol.
- Establishing a connection: The client connects to the server's IP and port using the connect() method.
- Sending user input: The client prompts the user for a message, encodes it into bytes, and sends it through the socket with sendall().
- Receiving server response: Using the recv() method, the client waits for the server's reply and decodes the message for display.
- Displaying output: The echoed response is printed to the console, confirming that the round-trip communication was successful.

Observations:
- The client successfully transmits data to the server and receives the echoed response.
- The design is interactive, relying on user input for communication.
- Similar to the server, it does not include input validation or handling cases such as server downtime, which are critical in real-world applications.

Code Snippets:
1. `client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)` - This line creates a socket object for the client to establish a connection with the server.
2. `client_socket.connect((HOST, PORT))` - This line connects the client socket to the server socket using the specified host and port number.
3. `client_socket.sendall(message.encode())` - This line sends user input (after encoding it) to the server.
4. `data = client_socket.recv(1024)` - This line receives the echoed response from the server.
5. `print("Received:", response.decode())` - This line prints the echoed response received from the server to the user.