

José Manuel Chávez Reynoso 4528485

Sistemas para IoT

U1 A1: Cliente-Servidor en Python

Brief

We wrote two python scripts to create a client-server application.

The server and client communicate thru port 5000, in this case we're using loopback address as the communication is made within the same host.

The server must be running before client connects, so it starts hearing for incoming connections, then we run the client and we are able to establish connection from point to point.

Code

```
//server.py
import socket

def server():
    host = socket.gethostname()
    port = 5000

    server_socket = socket.socket()
    server_socket.bind((host, port))
    server_socket.listen(2)

    conn, address = server_socket.accept()
    print("Conexion: " + str(address))
    while True:
        data = conn.recv(1024).decode()
        if not data:
            break
        print("User: " + str(data))
        data = input(' -> ')
        conn.send(data.encode())

    conn.close()

if __name__ == '__main__':
    server()
```


//client.py

```
import socket

def client():
    host = socket.gethostname()
    port = 5000

    client_socket = socket.socket()
    client_socket.connect((host, port))

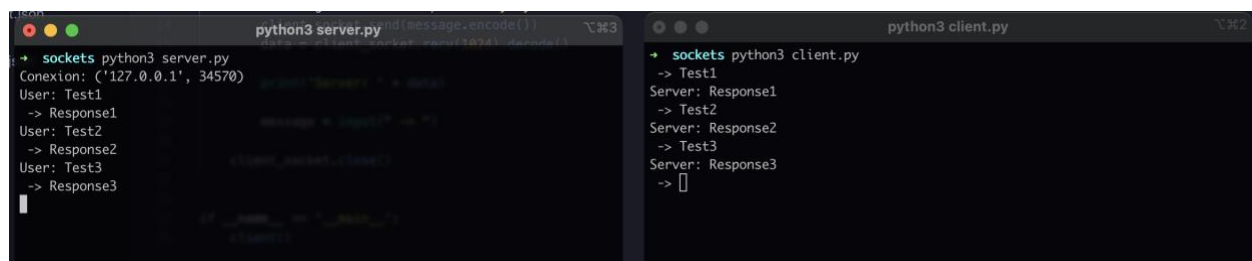
    message = input("-> ")

    while message.lower().strip() != 'byebye':
        client_socket.send(message.encode())
        data = client_socket.recv(1024).decode()
        print("Server: " + data)
        message = input("-> ")

    client_socket.close()

if __name__ == '__main__':
    client()
```

Evidence



The screenshot shows two terminal windows side-by-side. The left window, titled 'python3 server.py', shows the server's output: 'Conexion: ('127.0.0.1', 34570)', 'User: Test1', 'Server: Response1', 'User: Test2', 'Server: Response2', 'User: Test3', and 'Server: Response3'. The right window, titled 'python3 client.py', shows the client's input: '-> Test1', '-> Test2', '-> Test3', and an empty line '-> '.