Veeransh Shah
Reg. No. : 221070063
TY CS BTech

# EXPERIMENT 4

**Aim :** Study of Socket Programming and Client–Server model.

**Theory :**

- **What is Socket ?**

A socket serves as an endpoint for communication between two nodes or machines within a network, uniquely identified by an IP address and a port number. Client-side programming encompasses all the programming activities performed on the client side of this communication link. In client-server communication, the fundamental process begins with the client waiting for the server to start. Once the server is active, the client initiates a request through the connection established between their respective sockets. Subsequently, the client awaits a response from the server.

- **Establish a Socket Connection**

For communication between two machines, sockets must be present at both endpoints. The client needs to know the IP address of the server machine to establish a connection. A socket can be created with a single line of code:

```
Socket clientSocket = new Socket("127.0.0.1", 4000);
```

Here, clientSocket represents the socket at the client side, 4000 is the TCP port number, and 127.0.0.1 is the IP address of the server machine.

- **Communication**

Once sockets are successfully created, communication between them is facilitated using data streams. In networking, there are two types of data streams: the Input Stream and the Output Stream. The server's input stream is connected to the client's output stream, while the client's input stream is connected to the server's output stream.

- **Closing the Connection**

Finally, once the purpose is fulfilled the programmer needs to close the connection established earlier

- **Server Programming**

Server runs on a machine that is present over a network and bound to a particular port number.

- **Establish a Socket Connection**

Using the Socket class in java, one can create a socket with a single line of code:
ServerSocket serverSocket = new ServerSocket(4000);
Where the argument 4000 is our port number and the server waits for a connection request from the client side at the server socket. Once a connection request is received then the below line of code gets executed.
Socket clientSocket = serverSocket.accept();
This means that if the protocol is maintained and followed properly, then the request would be accepted by the server.

- **Close the Connection**

Finally, once the purpose is fulfilled the client sends a request to the server to terminate the connection between the client socket and server socket.

## Code & Output :

### server.py

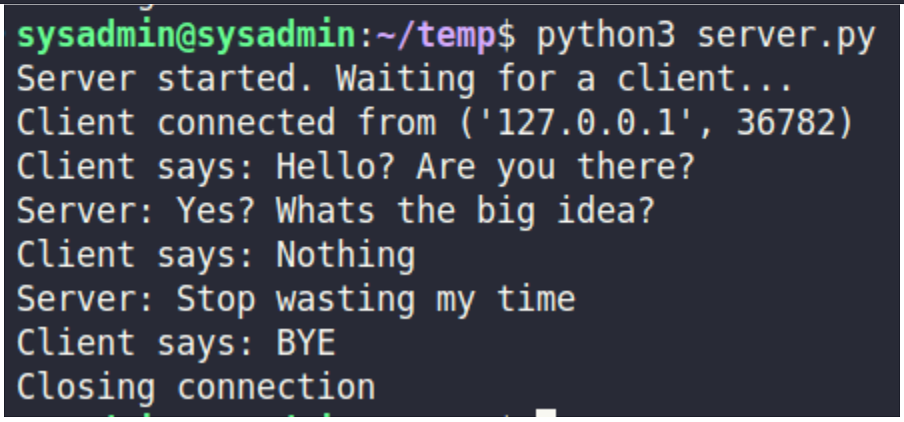
```python
import socket

class Server:
   def __init__(self, host='127.0.0.1', port=5000):
       self.server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
       self.server_socket.bind((host, port))
       self.server_socket.listen(1)
       print("Server started. Waiting for a client...")
       self.client_socket, address = self.server_socket.accept()
       print(f"Client connected from {address}")
       self.communicate()

   def communicate(self):
       while True:
           data = self.client_socket.recv(1024).decode('utf-8')
           if data == "BYE":
               print("Closing connection")
               break
           print(f"Client says: {data}")
           response = input("Server: ")
           self.client_socket.send(response.encode('utf-8'))

       self.client_socket.close()
       self.server_socket.close()

if __name__ == "__main__":
   Server()
```



```
sysadmin@sysadmin:~/temp$ python3 server.py
Server started. Waiting for a client...
Client connected from ('127.0.0.1', 36782)
Client says: Hello? Are you there?
Server: Yes? Whats the big idea?
Client says: Nothing
Server: Stop wasting my time
Client says: BYE
Closing connection
```

**client.py**

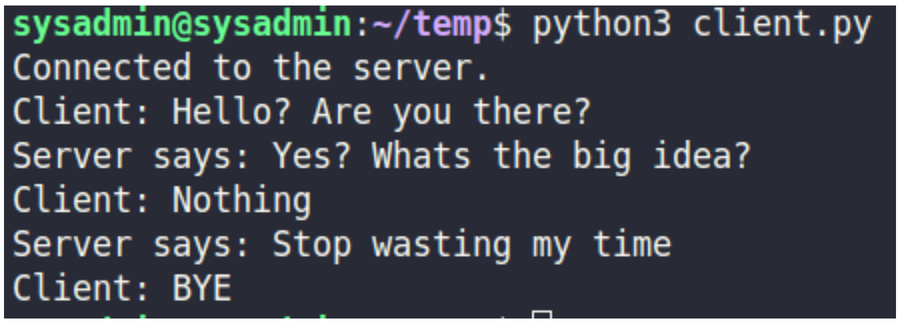
```python
import socket

class Client:
   def __init__(self, host='127.0.0.1', port=5000):
       self.client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
       self.client_socket.connect((host, port))
       print("Connected to the server.")
       self.communicate()

   def communicate(self):
       while True:
           message = input("Client: ")
           self.client_socket.send(message.encode('utf-8'))
           if message == "BYE":
               break
           response = self.client_socket.recv(1024).decode('utf-8')
           print(f"Server says: {response}")

       self.client_socket.close()

if __name__ == "__main__":
   Client()
```



```
sysadmin@sysadmin:~/temp$ python3 client.py
Connected to the server.
Client: Hello? Are you there?
Server says: Yes? Whats the big idea?
Client: Nothing
Server says: Stop wasting my time
Client: BYE
```

**Conclusion :**
In this experiment, we understood the need for sockets and how to implement them in a Python program. We also implemented two programs using sockets that help establish a connection between a client and a server.