**Semester: V (2023-24) Subject:** Communication Networks

**Name: Shreerang Mahtre Class: T.Y.B.Tech. El&CE**

**Roll No: 52 Batch: A3**

**Experiment No: 06**

**Name of the Experiment**: Socket Programming in Communication Networks using Python

**Performed on: 30/10/2023**

**Submitted on: 04/11/2023**



**Aim:** To demonstrate the implementation of client-server communication using socket programming in Python**.**

**Prerequisite:**

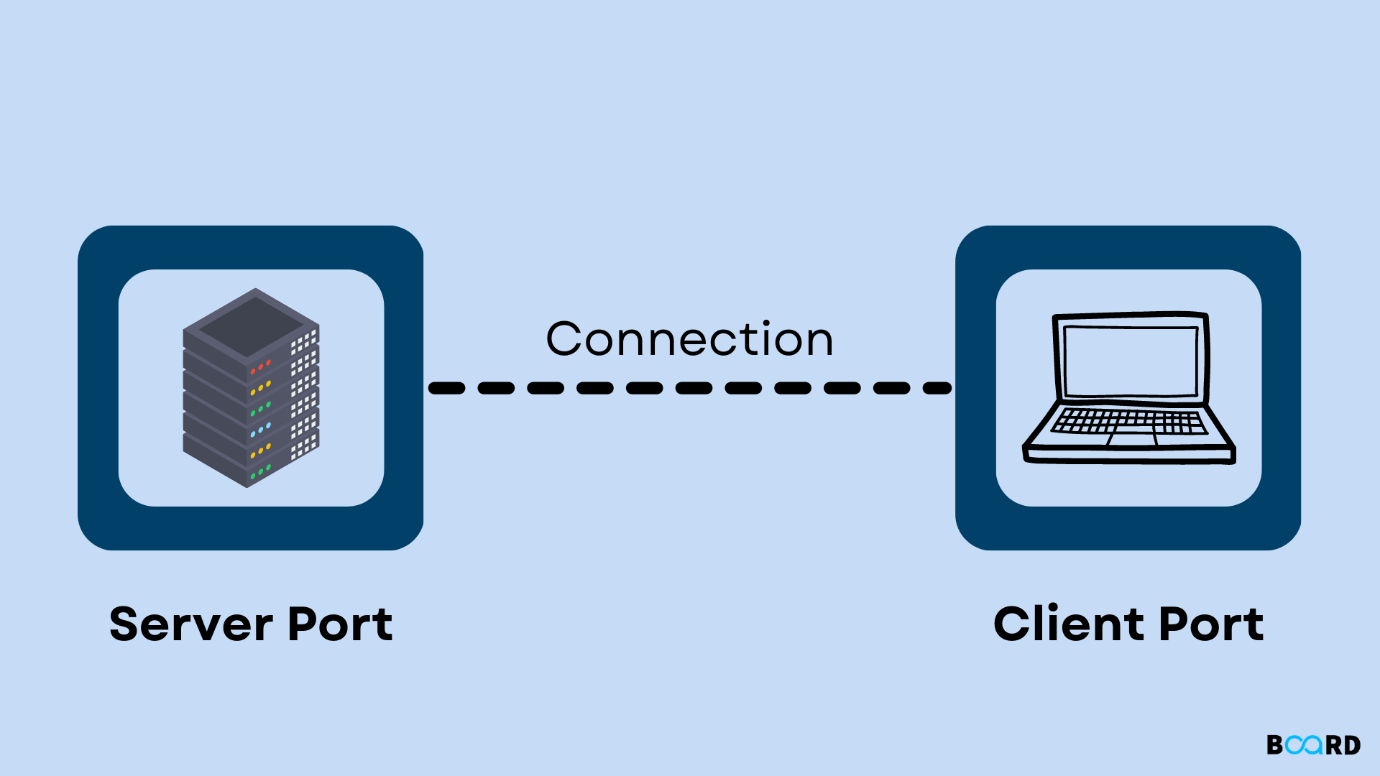
* Basic understanding of Python programming, networking concepts, and the fundamentals of client-server architecture.

**Objectives:**

1. Understand the concept of socket programming and its relevance in communication networks.
2. Implement a basic client-server model using Python's socket library.

**Theory:**

Socket programming is a key concept in communication networks that allows two nodes to establish a connection and exchange data. The most common type of socket applications are client-server applications, where one side acts as the server and waits for connections from clients.



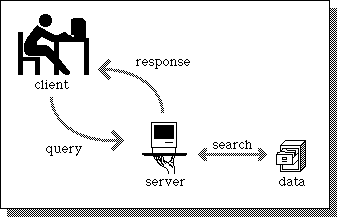
Understanding the intricacies of socket programming is essential for building various network applications that rely on efficient and secure data exchange. Sockets play a crucial role in establishing connections and facilitating the transfer of data between different devices over a network

Sockets are broadly categorized into two types:

1**. Stream Sockets (TCP):** They offer a reliable, connection-oriented, byte-stream service, ensuring data integrity and sequencing. TCP guarantees ordered delivery without loss or duplication, making it ideal for applications like file transfer and web services.

2. **Datagram Sockets (UDP):** They provide a connectionless, message-oriented service, sending datagrams without establishing a connection. UDP is suitable for applications requiring low latency and tolerant of data loss, such as real-time multimedia and online gaming.

Sockets serve as the endpoints for sending and receiving data across a network.



* In this practical, we utilize the socket library in Python, which provides an interface to create and manage sockets.
* By implementing socket programming, we can facilitate communication between a client and a server, enabling data transmission and reception over the network.

**Procedure:**

Step 1: Setting up the Server

1. Open your preferred Python integrated development environment (IDE) or text editor.

2. Begin by importing the socket library: `import socket`.

3. Create a socket object: `server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)`.

4. Bind the socket to a specific IP address and port: `server\_socket.bind(('localhost', 12345))`.

5. Listen for incoming connections: `server\_socket.listen(5)`.

6. Accept incoming client connections: `client\_socket, client\_address = server\_socket.accept()`.

Step 2: Setting up the Client

1. Open a new Python file in your IDE or text editor.

2. Import the socket library: `import socket`.

3. Create a socket object for the client: `client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)`.

4. Connect to the server using its IP address and port: `client\_socket.connect(('localhost', 12345))`.

Step 3: Data Transmission

1. Implement sending data from the client to the server: `client\_socket.sendall(b'Hello, server!')`.

2. Receive the data on the server side: `data = client\_socket.recv(1024)`.

3. Process the received data: `print("Received:", data.decode('utf-8'))`.

Step 4: Close the Connection

1. Close the client and server connections when the data transmission is complete:

- For the client: `client\_socket.close()`.

- For the server: `server\_socket.close()`.

**Conclusion:**

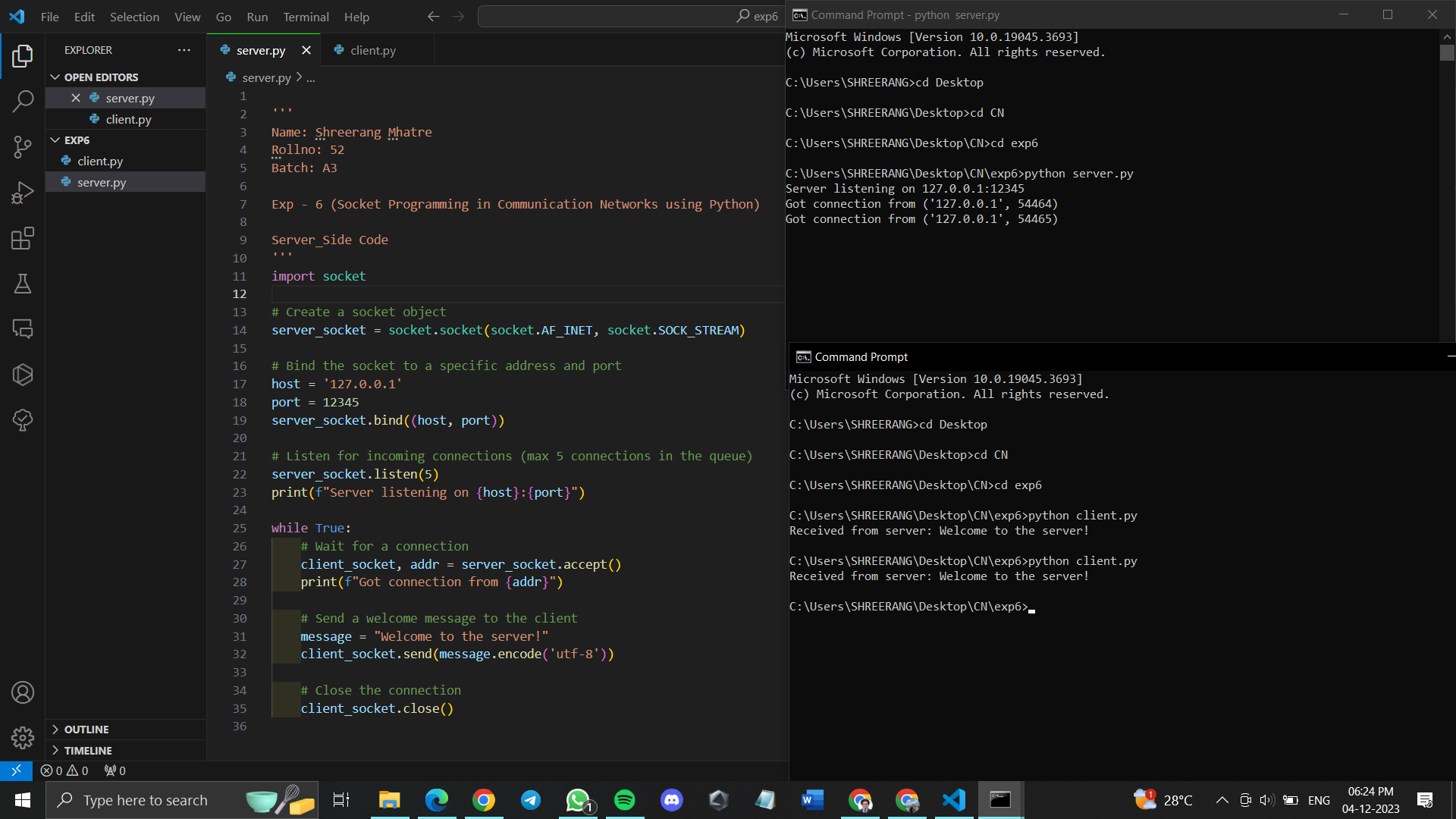
Executed client-server communication using Python's socket programming, showcasing data transmission and reception. Highlights the vital role of sockets in network communication.

**Post Lab Questions:**

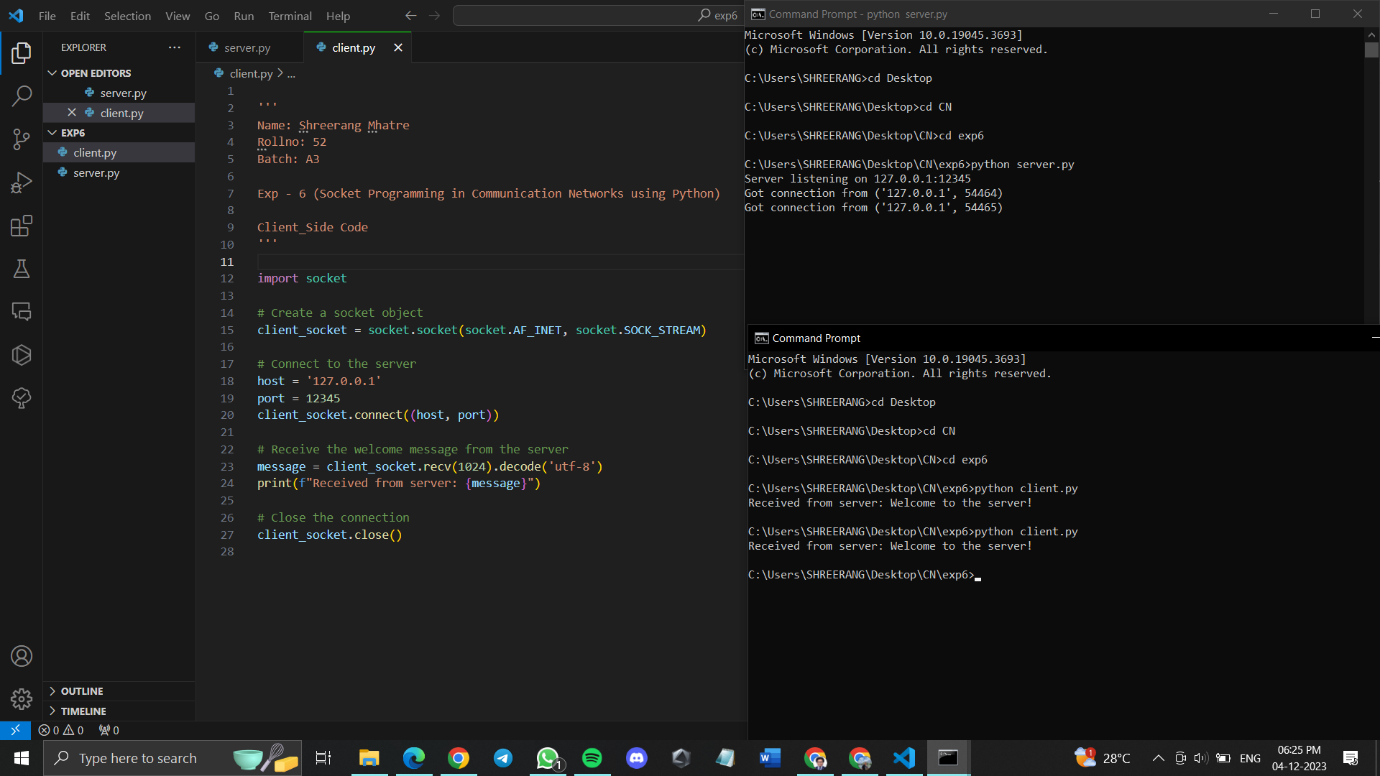
1. What is the role of the socket library in Python when working with communication networks?
2. Explain the significance of the bind() function in the socket programming process.
3. How does the accept() function contribute to establishing a connection between a client and a server?

**Execution of Socket Programming in Communication Networks using Python**

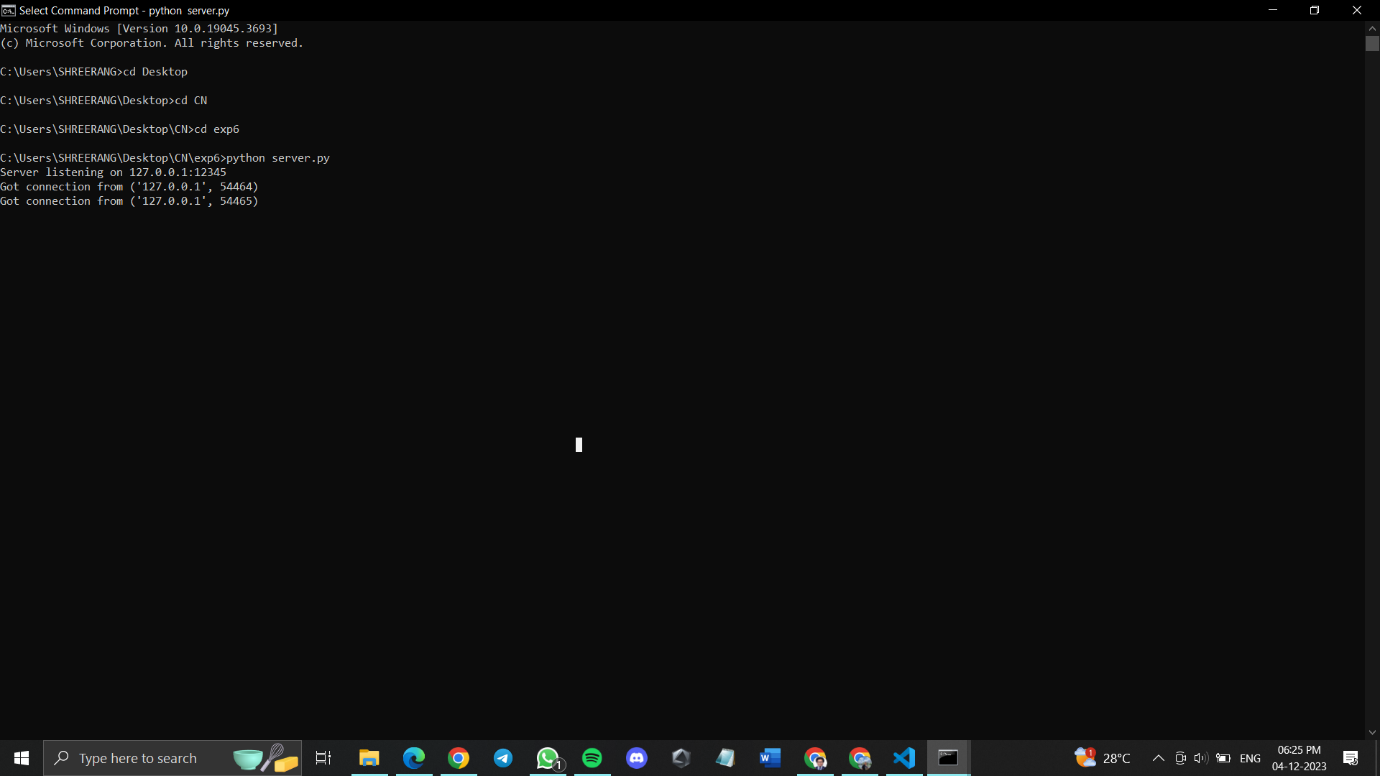
***Server Side***



***Client side***



***Output of Server Side:***



***Output of Client Side:***

