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:

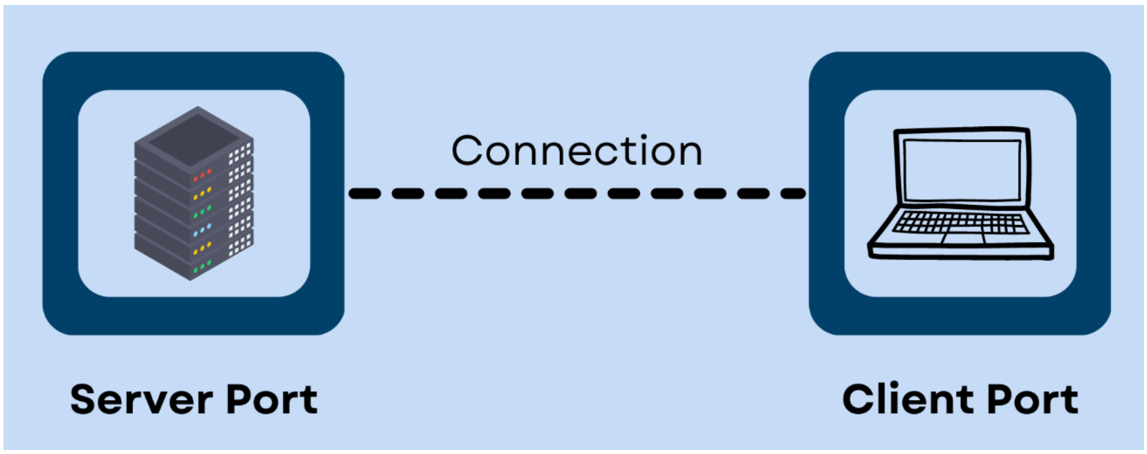
- Basic understanding of Python programming, networking concepts, and the fundamentals of client-server architecture.

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.

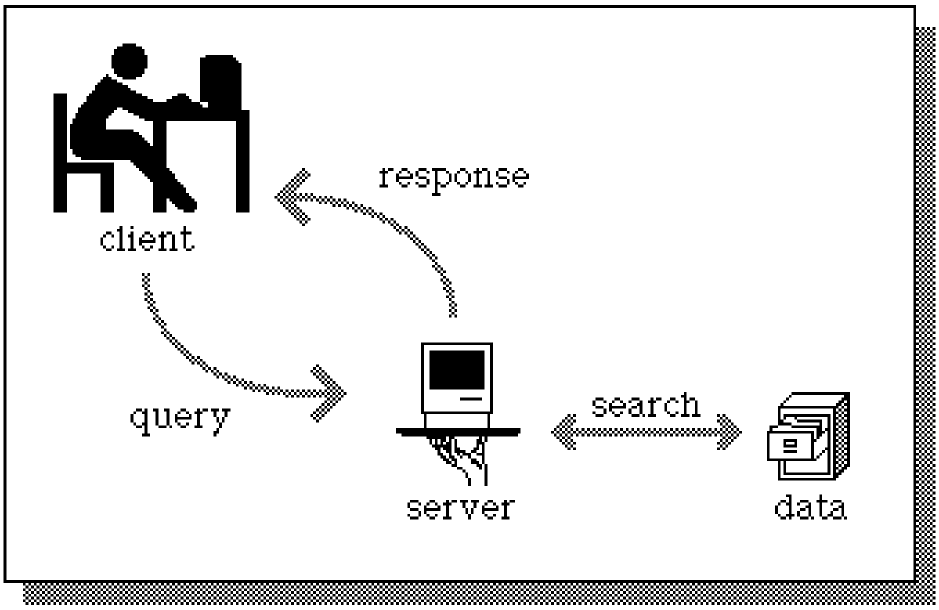


Understanding the intricacies of socket programming is essential for building various network applications that rely on efficient and secure data exchange. Sockets play a crucial role in establishing connections and facilitating the transfer of data between different devices over a network

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:

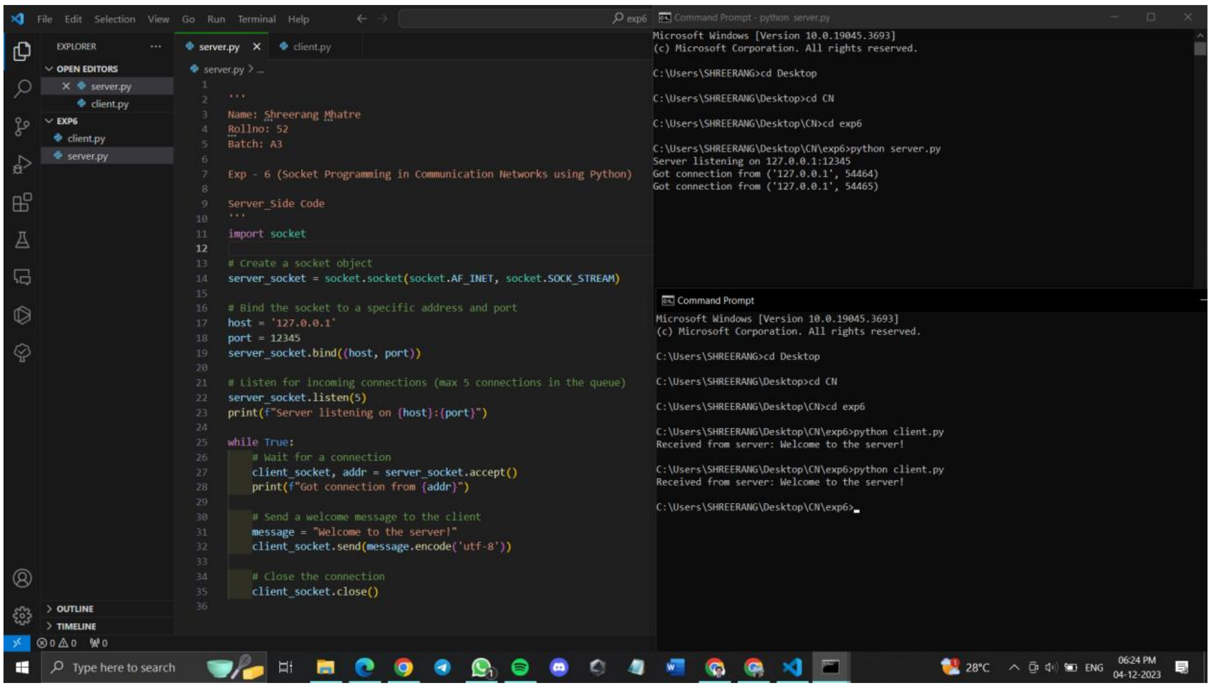
Executed client-server communication using Python's socket programming, showcasing data transmission and reception. Highlights the vital role of sockets in network communication.

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

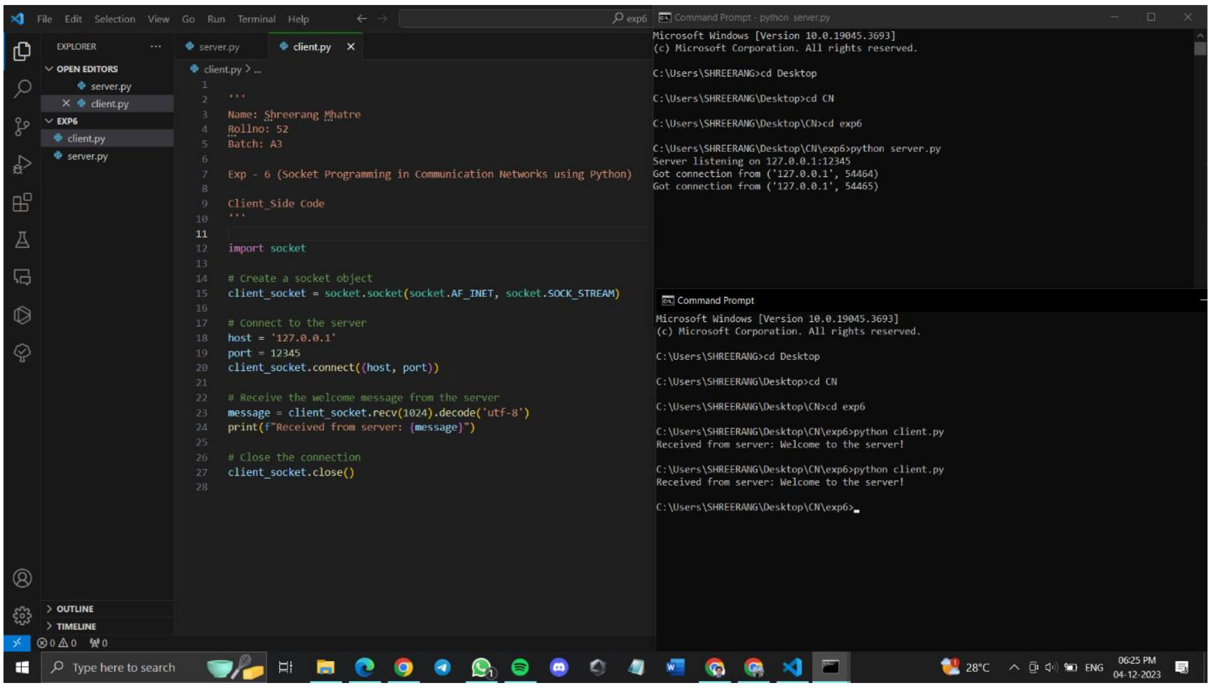


```python
'''
Name: Shreerang Mhatre
Rollno: 52
Batch: A3

Exp - 6 (Socket Programming in Communication Networks using Python)

Server_Side Code
'''
import socket

# Create a socket object
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Bind the socket to a specific address and port
host = '127.0.0.1'
port = 12345
server_socket.bind((host, port))

# Listen for incoming connections (max 5 connections in the queue)
server_socket.listen(5)
print(f"Server listening on {host}:{port}")

while True:
    # Wait for a connection
    client_socket, addr = server_socket.accept()
    print(f"Got connection from {addr}")

    # Send a welcome message to the client
    message = "Welcome to the server!"
    client_socket.send(message.encode('utf-8'))

    # Close the connection
    client_socket.close()
```

```
Microsoft Windows [Version 10.0.19045.3693]
(c) Microsoft Corporation. All rights reserved.

C:\Users\SHREERANG>cd Desktop

C:\Users\SHREERANG\Desktop>cd CN

C:\Users\SHREERANG\Desktop\CN>cd exp6

C:\Users\SHREERANG\Desktop\CN\exp6>python server.py
Server listening on 127.0.0.1:12345
Got connection from ('127.0.0.1', 54464)
Got connection from ('127.0.0.1', 54465)
```

```
Microsoft Windows [Version 10.0.19045.3693]
(c) Microsoft Corporation. All rights reserved.

C:\Users\SHREERANG>cd Desktop

C:\Users\SHREERANG\Desktop>cd CN

C:\Users\SHREERANG\Desktop\CN>cd exp6

C:\Users\SHREERANG\Desktop\CN\exp6>python client.py
Received from server: Welcome to the server!

C:\Users\SHREERANG\Desktop\CN\exp6>python client.py
Received from server: Welcome to the server!

C:\Users\SHREERANG\Desktop\CN\exp6>_
```

Client side

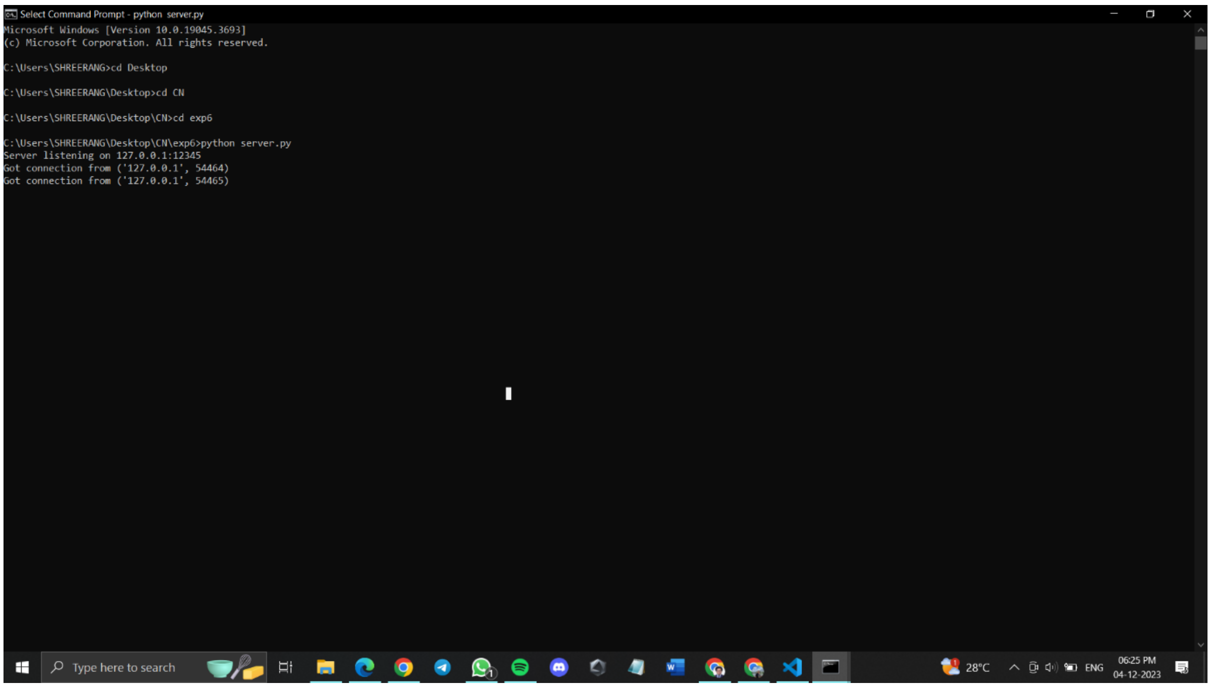


```python
'''
Name: Shreerang Mhatre
Rollno: 52
Batch: A3

Exp - 6 (Socket Programming in Communication Networks using Python)

Client_Side Code
'''

import socket

# Create a socket object
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Connect to the server
host = '127.0.0.1'
port = 12345
client_socket.connect((host, port))

# Receive the welcome message from the server
message = client_socket.recv(1024).decode('utf-8')
print(f"Received from server: {message}")

# Close the connection
client_socket.close()
```

```
Microsoft Windows [Version 10.0.19045.3693]
(c) Microsoft Corporation. All rights reserved.

C:\Users\SHREERANG>cd Desktop

C:\Users\SHREERANG\Desktop>cd CN

C:\Users\SHREERANG\Desktop\CN>cd exp6

C:\Users\SHREERANG\Desktop\CN\exp6>python server.py
Server listening on 127.0.0.1:12345
Got connection from ('127.0.0.1', 54464)
Got connection from ('127.0.0.1', 54465)
```

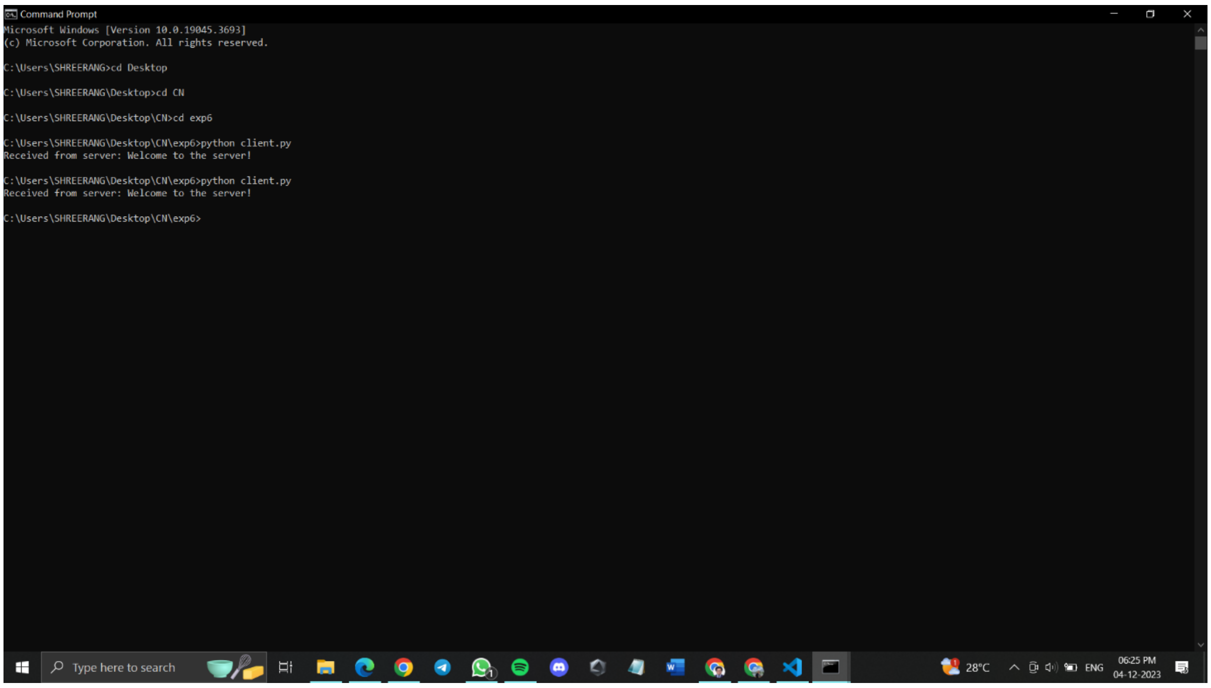
```
Microsoft Windows [Version 10.0.19045.3693]
(c) Microsoft Corporation. All rights reserved.

C:\Users\SHREERANG>cd Desktop

C:\Users\SHREERANG\Desktop>cd CN

C:\Users\SHREERANG\Desktop\CN>cd exp6

C:\Users\SHREERANG\Desktop\CN\exp6>python client.py
Received from server: Welcome to the server!

C:\Users\SHREERANG\Desktop\CN\exp6>python client.py
Received from server: Welcome to the server!

C:\Users\SHREERANG\Desktop\CN\exp6>_
```

Output of Server Side:



```
Microsoft Windows [Version 10.0.19045.3693]
(c) Microsoft Corporation. All rights reserved.

C:\Users\SHREERANG>cd Desktop

C:\Users\SHREERANG\Desktop>cd CN

C:\Users\SHREERANG\Desktop\CN>cd exp6

C:\Users\SHREERANG\Desktop\CN\exp6>python server.py
Server listening on 127.0.0.1:12345
Got connection from ('127.0.0.1', 54464)
Got connection from ('127.0.0.1', 54465)
```

Output of Client Side:



```
Microsoft Windows [Version 10.0.19045.3693]
(c) Microsoft Corporation. All rights reserved.

C:\Users\SHREERANG>cd Desktop

C:\Users\SHREERANG\Desktop>cd CN

C:\Users\SHREERANG\Desktop\CN>cd exp6

C:\Users\SHREERANG\Desktop\CN\exp6>python client.py
Received from server: Welcome to the server!

C:\Users\SHREERANG\Desktop\CN\exp6>python client.py
Received from server: Welcome to the server!

C:\Users\SHREERANG\Desktop\CN\exp6>
```

Exp 6

* Post Lab Questions:

Q1) what is the role of the socket library in Python when working with communication networks?

-> The socket library in Python is a powerful tool for working with communication networks. It provides a comprehensive set of tools for creating, managing, and customizing network connections, making it easier for developers to build. various network applications, including servers, clients and peer-to peer systems.

Socket programing is a way of connecting two nodes on a network to communicate with each other. one socket (node) listens on a particular port at an IP, while the other socket reaches out to the other to form a connection. The server forms the listner socket while the client reaches out to the server. They are the real backbones behind web browsing.

Q2) Explain the significance of the bind() function in the socket programming process.

→ The bind() function in socket programming is used to associate a socket with a specific IP address and port number on the local machine. It is an essential step in the process of creating a server application that listens for incoming client connections.

When a server socket is created, it is assigned a port number and an IP address. The bind() function is used to specify the IP address and port number that the server socket will listen on. Once the socket is bound to a specific IP address and port number, it can start listening for incoming client connections.

Q3) How does the accept() fonction contribute to establishing a connection between a client and a server?

-> The accept () function is used by a server to accept a connection request from a client. when a server socket is created, it is set to listen for incoming client connections. The accept() function is called by the server to accept a connection request from a client that is waiting to connect.

when a client sends a connection request to the server uses the fonction to accept the connection request and create a new socket for the client. This new socket is used to communicate with the client. The accept() function returns a new socket descriptor that is used to communicate with the client.