# Lab Manual: FTP Client-Server Program in Python

## Objective

The objective of this lab is to implement a basic FTP-like file upload functionality using Python's socket programming. This program will consist of a server that listens for incoming connections and a client that uploads files to the server.

## Prerequisites

- Python installed on your system (version 3.x recommended).
- Basic understanding of Python programming and file handling.
- Familiarity with socket programming concepts.

## Files

1. `ftp_server.py` - The server code that handles incoming file uploads.
2. `ftp_client.py` - The client code that allows users to upload files to the server.

## Implementation

### 1. Server Code (`ftp_server.py`)

```
import socket
import os

def start_ftp_server(upload_directory):
    # Ensure the upload directory exists; create it if it doesn't
    if not os.path.exists(upload_directory):
        os.makedirs(upload_directory)

    # Create a TCP/IP socket
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    # Bind the socket to the address and port
    server_address = ('localhost', 65432)
    server_socket.bind(server_address)

    # Listen for incoming connections
    server_socket.listen(1)
    print("FTP server is listening on port 65432...")

    while True:
        # Accept a connection from a client
        connection, client_address = server_socket.accept()
```

```
        try:
            print(f"Connection from {client_address}")

            # Receive the filename from the client
            filename = connection.recv(1024).decode()
            # Create the full path to save the uploaded file
            file_path = os.path.join(upload_directory, filename)
            print(f"Receiving file: {filename}")

            # Open the file in write-binary mode
            with open(file_path, 'wb') as f:
                # Receive the file data from the client
                bytes_received = connection.recv(1024)
                while bytes_received:
                    f.write(bytes_received)  # Write the received data to the
file
                    bytes_received = connection.recv(1024)  # Receive more
data

            print(f"File {filename} uploaded successfully to
{upload_directory}.")
        finally:
            # Close the connection
            connection.close()
            print(f"Connection closed with {client_address}")

if __name__ == '__main__':
    upload_directory = 'uploaded_files'  # Specify the directory to save
uploaded files
    start_ftp_server(upload_directory)  # Start the FTP server
```

## 2. Client Code (`ftp_client.py`)

```
import socket

def start_ftp_client():
    # Create a TCP/IP socket
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    # Connect to the server at the specified address and port
    server_address = ('localhost', 65432)
    client_socket.connect(server_address)

    try:
        while True:
            # Prompt user to enter the filename to upload
            filename = input("Enter the name of the file to upload (or 'exit'
to quit): ")
            if filename.lower() == 'exit':  # Check if the user wants to exit
                break

            try:
                # Open the file in read-binary mode
                with open(filename, 'rb') as f:
                    client_socket.send(filename.encode())  # Send the
filename first
```

```
                    # Send the file content in chunks
                    bytes_read = f.read(1024)
                    while bytes_read:
                        client_socket.send(bytes_read)  # Send each chunk of
data
                        bytes_read = f.read(1024)  # Read the next chunk
                print(f"Uploaded {filename} to the server.")
            except FileNotFoundError:
                # Handle the case where the file is not found
                print(f"File {filename} not found. Please try again.")

    finally:
        # Close the client connection
        client_socket.close()
        print("Client connection closed.")

if __name__ == '__main__':
    start_ftp_client()  # Start the FTP client
```

# How to Run the Programs

### Step 1: Run the Server

1. Save the server code in a file named `ftp_server.py`.
2. Open a terminal or command prompt.
3. Navigate to the directory where `ftp_server.py` is saved.
4. Run the server using the command:

   ```
   python ftp_server.py
   ```

5. The server will start listening for incoming connections and save uploaded files in the `uploaded_files` directory.

### Step 2: Run the Client

1. Save the client code in a file named `ftp_client.py`.
2. Open another terminal or command prompt.
3. Navigate to the directory where `ftp_client.py` is saved.
4. Run the client using the command:

   ```
   python ftp_client.py
   ```

5. Enter the name of the file you want to upload. The file must exist in the client's current directory.
6. Type `exit` to close the client connection.

# Testing the Program

1. Create a text file (e.g., `test.txt`) in the same directory as your client code.

2. Start the server first, then the client.
3. Use the client to upload `test.txt` to the server.
4. Check the `uploaded_files` directory created by the server to see if the file is present.