**Server.py**

Develop a simple TCP/IP client-server application in Python to handle reading and writing attributes associated with a person's name.

import socket

import json

**socket:** Provides low-level networking interface.

**json:** Used for encoding and decoding JSON data.

def read\_data(file\_path):

    try:

        with open(file\_path, 'r') as file:

            data = json.load(file)

        return data

    except FileNotFoundError:

        return {}

**read\_data:** Reads data from a JSON file specified by file\_path.

def write\_data(file\_path, data):

    with open(file\_path, 'w') as file:

        json.dump(data, file)

        file.write('\n')

**write\_data**: Writes data to a JSON file specified by file\_path, each entry on a new line.

def handle\_command(command, data, client\_socket):

    action, attribute, person\_name = command.split(" ", 2)

    if action == "Read":

        response = data.get(person\_name, {}).get(attribute, "Attribute not found.")

        client\_socket.send(response.encode())

    elif action == "Write":

        if person\_name not in data:

            data[person\_name] = {}

        data[person\_name][attribute] = input(f"Enter value for {attribute}: ")

        write\_data(FILE\_PATH, data +'\n')

        client\_socket.send("Write successful.".encode())

    else:

        client\_socket.send("Invalid command.".encode())

**handle\_command:** Processes incoming commands. Reads or writes data based on the action specified.

def start\_server():

    server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

    server\_socket.bind((HOST, PORT))

    server\_socket.listen()

    print(f"Server listening on {HOST}:{PORT}")

    while True:

        client\_socket, address = server\_socket.accept()

        print(f"Connection from {address}")

        data = read\_data(FILE\_PATH)

        command = client\_socket.recv(BUFFER\_SIZE).decode()

        handle\_command(command, data, client\_socket)

        client\_socket.close()

**start\_server:** Sets up and starts the server. Listens for incoming connections and handles commands.

if \_\_name\_\_ == "\_\_main\_\_":

    HOST = '127.0.0.1'

    PORT = 12345

    BUFFER\_SIZE = 1024

    FILE\_PATH = 'data.json'

    start\_server()

**The main block initializes server settings and starts the server loop, providing a clear entry point for execution.**

**Client.py**

import socket

def send\_command(command):

    with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as client\_socket:

        client\_socket.connect((SERVER\_HOST, SERVER\_PORT))

        client\_socket.send(command.encode())

        response = client\_socket.recv(BUFFER\_SIZE).decode()

        print(f"Server response: {response}")

if \_\_name\_\_ == "\_\_main\_\_":

    SERVER\_HOST = '127.0.0.1'

    SERVER\_PORT = 12345

    BUFFER\_SIZE = 1024

    while True:

        command = input("Enter command (e.g., Read/Write \"Attribute\" \"Person Name\"): ")

        send\_command(command)

**send\_command** is a function that takes a command (a string) as its argument.

socket.AF\_INET specifies the address family for IPv4. socket.SOCK\_STREAM indicates that this will be a TCP socket.

**response** = client\_socket.recv(BUFFER\_SIZE).decode() receives the server's response. BUFFER\_SIZE determines the maximum amount of data to be received at once. The response is then decoded from bytes to a string.