Cyber Security

Lab-2

Gohel Mruganshi Jatinbhai
B20CS014

1. Your server program "server1" will be a single process server that can handle only one client at a time. If a second client tries to chat with the server while one client's session is already in progress, the second client's socket operations should see an error.

In the below code, the server will create a socket object, bind it to an address and port, and start listening for incoming connections. and after the establishment of the connection, it will send a handshaking message to the client and then close the connection. On the other hand, the client creates a socket object, connects to the server, and it will send a string in the format “operand1 operation operand2 “ and operations will be +, *, -, and /.

client

A client is a user or machine which generates a request to the server in return it gets the response from the server side. To perform these tasks we have to know two things which are the **Port Number** and **IP Address of the Server**.

```
import socket
SERVER = "127.0.0.1"
PORT = 12345
client = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
client.connect((SERVER, PORT))
while True:
  inp = input("Enter the operation in the form opreand operator oprenad: ")
  if inp == "End":
    break
  client.send(inp.encode())
  answer = client.recv(1024)
  print("Answer is "+answer.decode())
client.close()
```

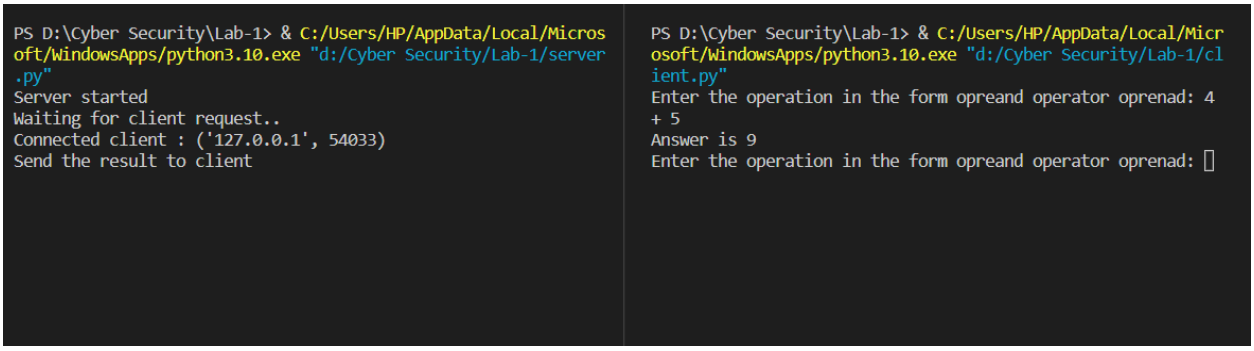
Server

The server-side is a machine on which all the operations are executed as requested by the client-side, we will perform addition, subtraction, multiplication, and division operation on the server. To do so the server received input from a client and performs the basic operation by itself. when the result is ready Python server then sends the output to the client machine.

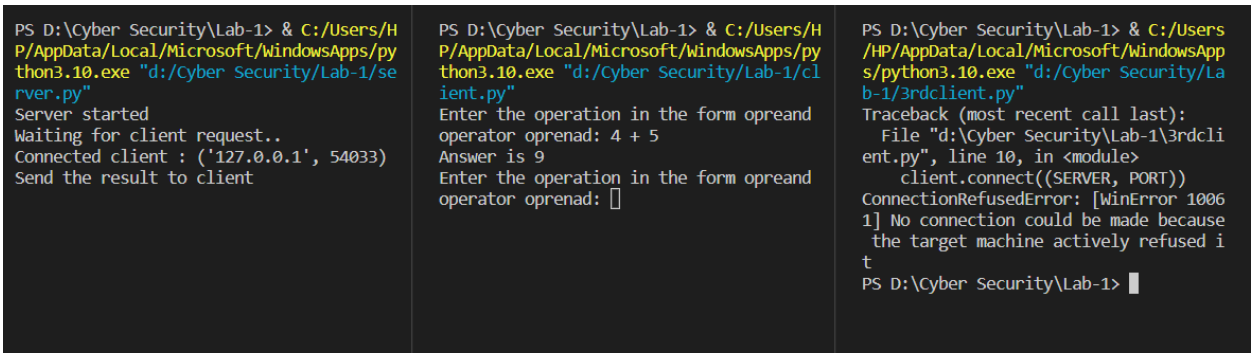
```
import socket
SERVER_ADDRESS = "localhost"
SERVER_PORT = 12345
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind((SERVER_ADDRESS, SERVER_PORT))
server_socket.listen(1)
print("Server started")
print("Waiting for client request..")
clientConnection, clientAddress = server_socket.accept()
print("Connected client :", clientAddress)
msg = ''
while True:
  data = clientConnection.recv(1024)
  msg = data.decode()
  ans=eval(msg)
  print("Send the result to client")

  output = str(ans)
  clientConnection.send(output.encode())
clientConnection.close()
```

Output

Case-1: Single client



Case-2: If a second client tries to chat with the server while one client's session is already in progress



2. Your server program "server2" will be a multi-process or multi-threaded server that will fork a process for every new client it receives. Multiple clients should be able to simultaneously chat with the server.

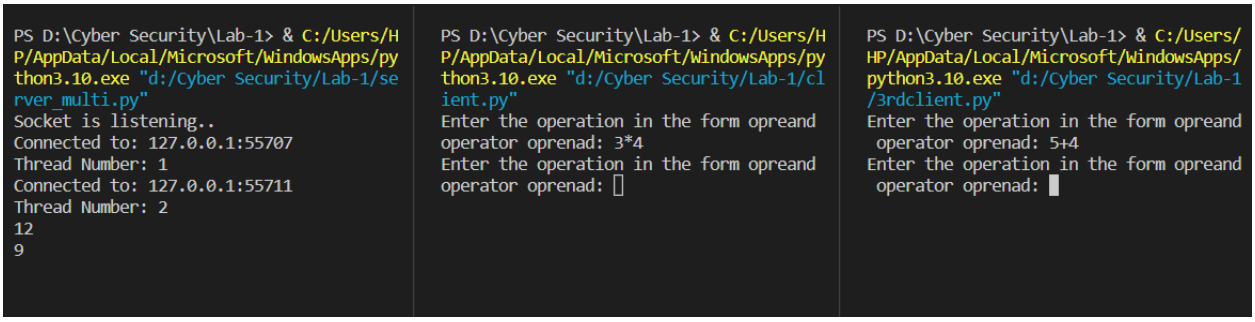
To create a multi-threaded server We have to create a brand new function and name it `multi_threaded_client()`; this connects every client from the various address provided by the server simultaneously. Within the multi_threaded_client function, the `connection.recv(1024)` method is separately getting the data from every client and returning the server response to a specific client. The server must run relentlessly, seldom our server's connections might get deteriorated due to some hidden errors. To make the server connection run constantly, we will take the help of a while loop. The server socket connection is combined with the accept method and conjugates the constant process in one go. It interprets the process and returns the connected clients' information such as the thread number and address given to it. As you can see, the **start_new_thread** function is handling the connection; it establishes a new thread for every client to perform server connection duty specifically.

Server

```
import socket
import os
from _thread import *
ServerSideSocket = socket.socket()
host = '127.0.0.1'
port = 2005
ThreadCount = 0
```

```
try:
    ServerSideSocket.bind((host, port))
except socket.error as e:
    print(str(e))
print('Socket is listening..')
ServerSideSocket.listen(5)
def multi_threaded_client(connection):
    connection.send(str.encode('Server is working:'))
    while True:
        data = connection.recv(1024)
        msg = data.decode()
        ans=eval(msg)
        output = str(ans)
        Client.send(output.encode())
    connection.close()
while True:
    Client, address = ServerSideSocket.accept()
    print('Connected to: ' + address[0] + ':' + str(address[1]))
    start_new_thread(multi_threaded_client, (Client, ))
    ThreadCount += 1
    print('Thread Number: ' + str(ThreadCount))
ServerSideSocket.close()
```

Output



3. (Bonus Question): Server that works with any type of number, any number of operands, and other arithmetic operations.

Now to full fill above conditions I have used the eval() library. Below is Server code.

Server

```
import socket
SERVER_ADDRESS = "localhost"
SERVER_PORT = 2005
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind((SERVER_ADDRESS, SERVER_PORT))
```

```
server_socket.listen()
print("Server started")
print("Waiting for client request..")
clientConnection, clientAddress = server_socket.accept()
print("Connected client :", clientAddress)
msg = ''
while True:
  data = clientConnection.recv(1024)
  msg = data.decode()
  ans=eval(msg)
  print("Send the result to client")
  output = str(ans)
  clientConnection.send(output.encode())
clientConnection.close()
```

Output