TCP CHAT ROOM

Introduction

Python is a good language for implementing computer networking . It allows   us to create solid applications very easily. Now we will be implementing TCP chat room in Python which would help us analyze relationship between server and client.We can add custom chat rooms etc.

Client-server Architecture

For our application we will use client server architecture.This means that we will be having multiple clients and one server that would control everything and provides data to the clients.

However we will be writing two Python codes ie one will be for server and the other for the clients.We will have to run the server first , so that there is a chat ,which the clients can connect too.The clients themselves will be communicating with each other through central server.

Implementing the Server

Now we start implementing the server first . For that we need to import two libraries ie socket and threading . Socket will be used for network connection and threading will be used for performing multiple tasks at one time.

```
import socket

import threading
```

The Next task is to define our connection data and initialize our socket.We will need an Ipaddress for our host and free port number for our server.
For this we will use the localhost address which is 127.0.0.1 and the port 55555.The port is actually irrelevant but we have to make sure that the port is free and not reserved. If we are running the script on an actual server , specify the IP address of the server as a host.

# Connection Data

```
host = '127.0.0.1'
port = 55555

# Starting Server
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind((host, port))
server.listen()

# Lists For Clients and Their Nicknames
clients = []
nicknames = []
```

when we define our socket, we need to pass two parameters .These define the types of socket we want to use.The first one (AF_INET) defines that we are using an internet socket connection rather than unix socket. The Second one tells us the type of protocol we want to use. SOCK_STREAM indicates us that we are using TCP not UDP.

After defining the socket , we bind it to our host and the specified port by passing a tuple that contains both values . We then put our server into listening mode so that it waits for the client to connect.At the end we create two empty list which will be used to store connected clients and their nicknames later onn.

```
# Sending Messages To All Connected Clients
def broadcast(message):
    for client in clients:
        client.send(message)
```

Here we define a little function that is going to help us broadasting the messages and makes the code more readable.It just send message to the connected clients and client’s list.We will use this function in other functions.Now the below function will be use to handle the messages from the clients

```
# Handling Messages From Clients
def handle(client):
    while True:
        try:
            # Broadcasting Messages
            message = client.recv(1024)
            broadcast(message)
        except:
            # Removing And Closing Clients
            index = clients.index(client)
            clients.remove(client)
            client.close()
            nickname = nicknames[index]
            broadcast('{} left!'.format(nickname).encode('ascii'))
            nicknames.remove(nickname)
            break
```

Now the while loop won’t stop unless there is an exception of something that went wrong.The function accepts client as a parameter . Everytime a client connects to our server we run the function for it and it starts an endless loop.

It just recieves the message from the client and boradcast it to all the connected clients.So when one clients recieves a message every client can view it.Now if for some reason there is an error with the connection to this client , we remove it and its nickname ,close the connection and broadcast that this client has left the chat.After that we break the loop and this thread comes to an end .We are almost done with the server except one function below.

```
# Receiving / Listening Function
def receive():
    while True:
        # Accept Connection
        client, address = server.accept()
        print("Connected with {}".format(str(address)))

        # Request And Store Nickname
        client.send('NICK'.encode('ascii'))
        nickname = client.recv(1024).decode('ascii')
        nicknames.append(nickname)
        clients.append(client)

        # Print And Broadcast Nickname
        print("Nickname is {}".format(nickname))
        broadcast("{} joined!".format(nickname).encode('ascii'))
        client.send('Connected to server!'.encode('ascii'))

        # Start Handling Thread For Client
        thread = threading.Thread(target=handle, args=(client,))
        thread.start()
```

When we are ready to run our server , we will execute this receive function.It also starts an endless while loop which constantly acceprs new connections from clienrs. Once a client is connected it sends the String “NICK to it , which will tell the client that its nickname is requested . After that it waits for a response (which contains the nickname) and appends the client with the respective nickname to the lists . After that , we print and broadcast this information . Finally , we start a new thread that runs the previously implemented handling function for this particular event . Now we can just run this function and our server is done .

Notce that we are always encoding and decoding the messages here . The reason for this is that we can only send bytes and not string . So we always need to encode messages (for eg using ASCII), when we send them and decode them , when we receive them.

```
Recieve()
```

IMPLEMENTING THE CLIENT

A server is pretty useless without clients that connect to it . So now weare going to implement our client . For this, we will again need to import the same libraries . Now this is the second seperate script.

```
import socket
import threading
```

The first step of the client are to choose a nickname and to connect to our server. We will need to know the exact address and the port at which our server is running.

```
# Choosing Nickname
nickname = input("Choose your nickname: ")

# Connecting To Server
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(('127.0.0.1', 55555))
```

As you can see , we are using a different function here . Instead of binding the data and listening , we are connecting to an existing server.

Now , a client needs to have two threads that are running at the same time . The first one will constantly receive data from the server and the second one will send our own messages to the server . So we will need two functions here . Let’s sart with the receiving part.

```
#Listening to Server and Sending Nickname
def receive():
    while True:
        try:
            # Receive Message From Server
            # If 'NICK' Send Nickname
            message = client.recv(1024).decode('ascii')
            if message == 'NICK':
                client.send(nickname.encode('ascii'))
            else:
                print(message)
        except:
            # Close Connection When Error
            print("An error occured!")
            client.close()
            break
```

Again , we have an endless   while loop here . It constantly tries to receive messages and to print them onto the screen . If the message is ‘NICK’ however . It doesn’t print it but it sends its nickname to the server . In case there is some error , we close the connection and break the loop . Now we just need a function for sending messages and we are almost done.

```
# Sending Messages To Server
def write():
    while True:
        message = '{}: {}'.format(nickname, input(''))
        client.send(message.encode('ascii'))
```

The writing function is a quite short one . It also runs in an endless loop which is always waiting for an input from the user. Once it gets some ,it combines it with the nickname and sends it to the server . That's it . Now we need to start two threads and run these two functions.

```
# Starting Threads For Listening And Writing
receive_thread = threading.Thread(target=receive)
receive_thread.start()

write_thread = threading.Thread(target=write)
write_thread.start()
```

Now we are done . We have a fully functioning server and working clients that can connect to it and communicate with eachother.

RUNNING THE CHAT