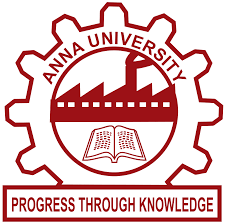
****

***CHAT 360***

**Simple Server and Client Chat using Python**

***Submitted by:***

*1921121* ***-*** *DHIWAKAR S R*

*1921124 - DIVIYARASU S*

**50 IT 503 – PROGRAMMING USING PYTHON**

**B.Tech (Information Technology)**

K.S. RANGASAMY COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

TIRUCHENGODE – 637 215

**NOV 2021**

### Server and Client Chat:

### *In Python language, socket (or network socket) is a module used to communicate between two computers. It provides two types of interface to access the network, namely low-level (platform dependent connections — Example: Telnet) and high-level (application dependent connections — Example: HTTP, FTP, SMTP, etc.). This is a simple tutorial to establish the low-level socket connection between server and client to communicate messages using the TCP/IP protocol.*

*We have used two scripts****server.py****to serve data by sending an invitation to the client, and****client.py****to receive data on acceptance of the invitation. After accepting the invitation, both server and client share messages mutually.*

***INTRODUCTION:***

*The****Chat Application****is very common today*

*offered either via a****web application****or****mobile application****. Learning to write a****Chat Application****is goodfor understanding many****network communication****concepts and can be useful to build other****network applications****.*

***Chat Application****provides communication between two parties i.e.****sender****and****receiver****. The****sender****is someone who initiates and send a message to other known as****receiver****; receiver at other end receives the message. The role of****sender****and****receiver****is not fixed and keep exchanging during communication, so in simple words, at a point, someone who sends the message is a****sender****and who receive the message is called* ***Receiver*** *In****networking****terms,****sender****and****receiver****are denoted as****source****and****destination****respectively.*

Communication can be of many types depending upon the method of communication and the number of parties involved. Some of the scenarios are :

1. **Simplex or one-way communication:** Only one party is able to send the message and other parties can only receive.
2. **Duplex or two-way communication:** Both parties can send and receive messages.

**Duplex communication** is a common way of communication and can be **one-to-one** (simple chat) or **many-to-many** (chat room).

*So, to create a****Python******Chat Application****, one has to write a****server****program and****client****program/s (****sender****and****receiver****). Suppose, two parties Alice and Bob want to chat with each other and ask you to develop a chat application then being a developer you have to write a****server******program****and a****client******program****(different instance of the same program will be used by both Alice and Bob or evenmore users)*.

***SERVER SCRIPT:***

**Server program** has all the logic to control and regulate the **Chat**, so most of the chat logic is implemented with a **server** program. So first step of communication is to identify the users, how server do this? In network communication, users are identified by a **socket** which is nothing but a combination of **IP address** and **port address**. So, for human understanding, Alice and Bob will be chatting but for a **network**, it is two **sockets** process which is sending and receiving bytes.  Steps involved in this process is as follows:

1. Create socket
2. Communicate the socket address
3. Keep waiting for an incoming connection request/s
4. Connect to client
5. Receive the message
6. Decode the destination user and select the socket
7. Send a message to the intended client
8. Keep repeating step 5 & 6 as per users wish
9. Exit i.e. end the communication by terminating the connection

An server script performs the sequence of functions such as socket(), bind(), listen(), and accept() (repeats for more than one client) to communicate with the client. The description of each functions used in the server script are given bellow:

* **socket()** – creates a socket using the address family, socket type and protocol
* **bind()** – binds the socket to the given address (host name, and port number *pair*)
* **listen()** – enables a server to accept connections from the client(s)
* **accept()** – waits and accepts connection request from the client(s)
* **gethostname()** – retrieves host name of the machine
* **gethostbyname()** – translates a host name to IPv4 format address
* **recv()**– receives message sent through TCP
* **decode()**– decodes the message using the codec
* **send()**– sends message sent through TCP

***CLIENT:***

**Client** script is run by the user, so the same **client** code will be run by a different user but each will have a separate **socket** so they will have their unique **communication channel**. Client script uses to be thin because it has very less work i.e. it only connect with the **server** and send and receive messages. The steps involved in **client script** are:

* Create a unique client socket per instance/user.
* Connect to the server with given socket address (IP and port).
* Send and receive messages.
* Repeat step 3 as per configuration.
* Close the connection.

An client script performs the sequence of functions such as socket(), and connect() to communicate with the server. The description of each functions used in the server script are given bellow:

* **socket()** – creates a socket using the address family, socket type and protocol.
* **connect()** – connects to a server socket at address.

***IMPLEMENTATION PROCEDURE:***

* *Run the****server.py****script in Python application.*
* *Note down the local IP address and pass it to the clients (invitation). Alternatively, you can also get the local IP address through online using [L-IP](http://www.biogem.org/tool/l-ip/" \t "https://www.biob.in/2018/04/_blank) tool.*
* *Run the****client.py****script in Python application using the local IP address sent by the server (accept the invitation).*
* *Sent and/or receive messages from the server/client mutually.*

**Note:**

In this lan Chat project, we are only able to connect with each other when both the **SERVER** and **CLIENT** are get connected with the **same network**.

***SOURCE CODE:***

***#Server.py***

import time, socket, sys

print("\nWelcome to Chat 360\n")

print("Initialising....\n")

time.sleep(1)

s = socket.socket()

host = socket.gethostname()

ip = socket.gethostbyname(host)

port = 1234

s.bind((host, port))

print(host, "(", ip, ")\n")

name = input(str("Enter your name: "))

s.listen(1)

print("\nWaiting for incoming connections...\n")

conn, addr = s.accept()

print("Received connection from ", addr[0], "(", addr[1], ")\n")

s\_name = conn.recv(1024)

s\_name = s\_name.decode()

print(s\_name, "has connected to the chat\nEnter [e] to exit chat\n")

conn.send(name.encode())

while True:

message = input(str("Me : "))

if message == "[e]":

message = "Left chat!"

conn.send(message.encode())

print("\n")

break

conn.send(message.encode())

message = conn.recv(1024)

message = message.decode()

print(s\_name, ":", message)

***# Client.py***

import time, socket, sys

print("\nWelcome to Chat 360\n")

print("Initialising....\n")

time.sleep(1)

s = socket.socket()

shost = socket.gethostname()

ip = socket.gethostbyname(shost)

print(shost, "(", ip, ")\n")

host = input(str("Enter server address: "))

name = input(str("\nEnter your name: "))

port = 1234

print("\nTrying to connect to ", host, "(", port, ")\n")

time.sleep(1)

s.connect((host, port))

print("Connected...\n")

s.send(name.encode())

s\_name = s.recv(1024)

s\_name = s\_name.decode()

print(s\_name, "has joined the chat\nEnter [e] to exit chat\n")

while True:

message = s.recv(1024)

message = message.decode()

print(s\_name, ":", message)

message = input(str("Me : "))

if message == "[e]":

message = "Left chat!"

s.send(message.encode())

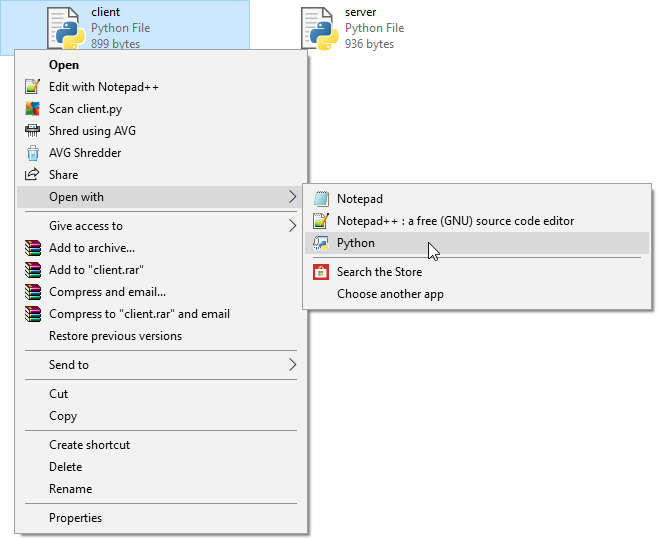
print("\n")

break

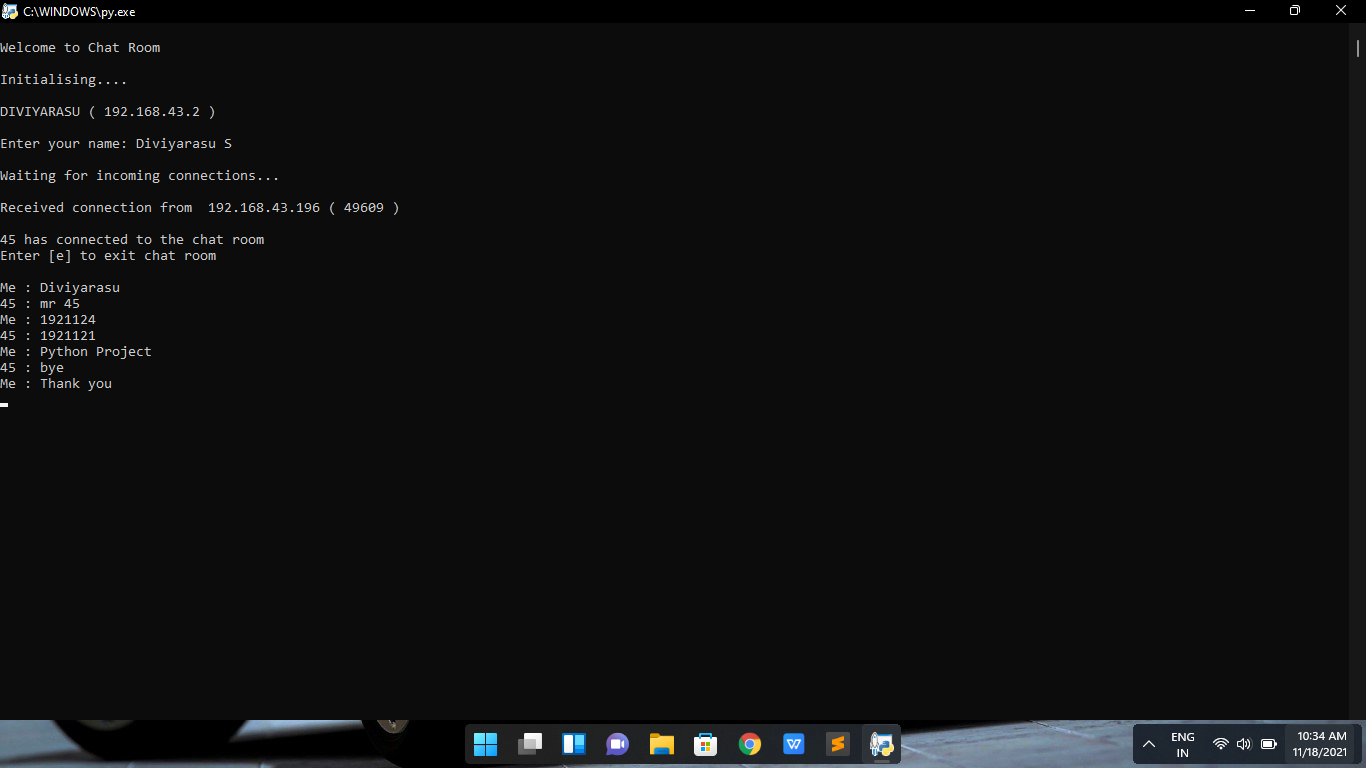
s.send(message.encode())

***SCREENSHOTS:***

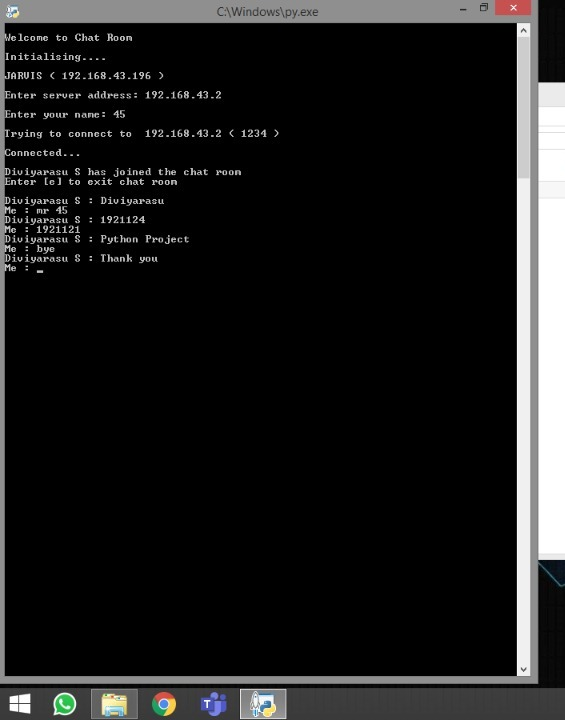
1. Run the ***server.py*** script in Python application.



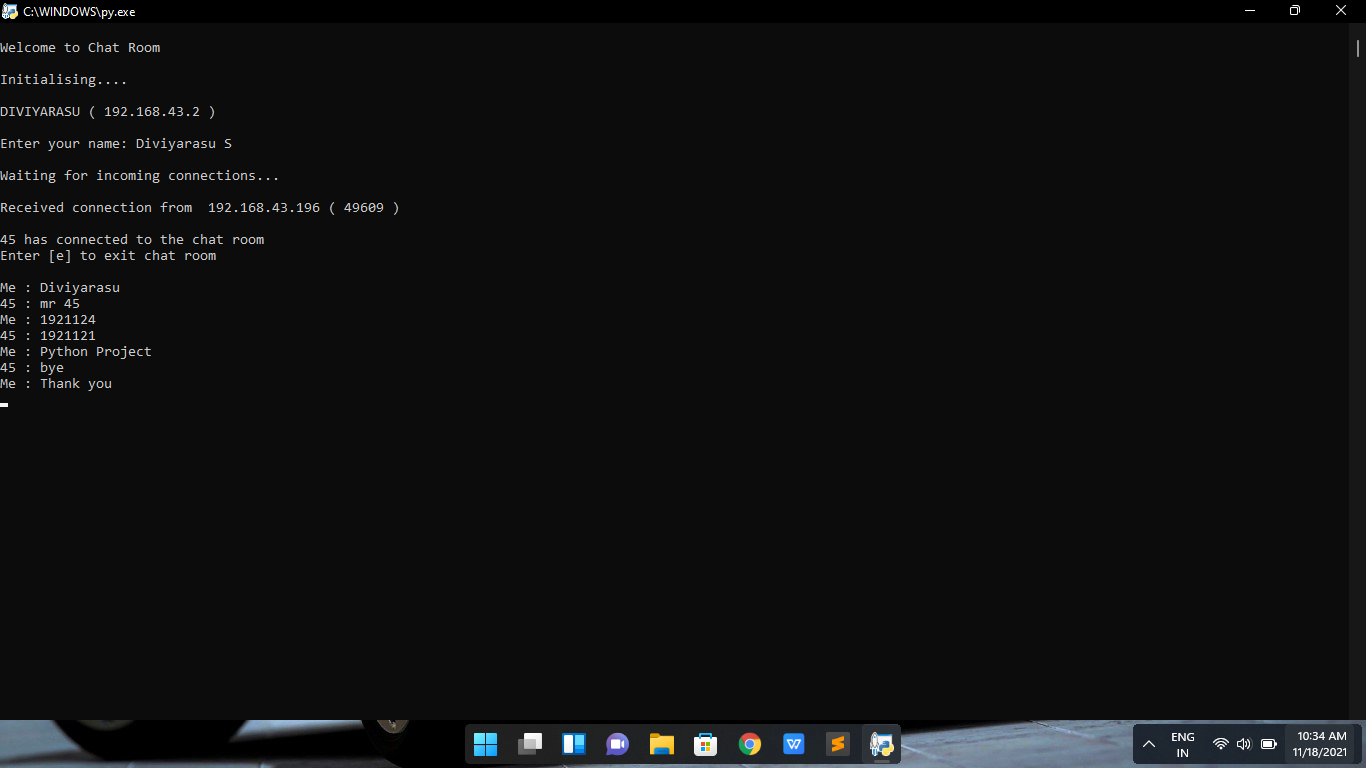
2.Note down the local IP address and pass it to the clients (invitation). Alternatively, you can also get the local IP address through online using [L-IP](http://www.biogem.org/tool/l-ip/" \t "https://www.biob.in/2018/04/_blank) tool.



4.Run the ***client.py*** script in Python application using the local IP address sent by the server (accept the invitation).



1. Send and/or receive messages from the server/client mutually.



**CONCLUSION:**

*Writing Python programs that use IPC with sockets is rather simple.The example given above can certainly be extended to handle something more complex .*