**CHAT ROOM**

For understanding the following program a basic knowledge over the following is required:

1. Socket
2. Client & server model

It’s better to start understanding this program using the analogies rather than divining deep with the definitions which is already flooding over the internet

Before getting started take a look on the Socket programming example at: <https://www.journaldev.com/15906/python-socket-programming-server-client>

**Sockets**:

It’s just an end point between two communication nodes (say; nodes are individual PC or IPs) what it does is establish an environment suitable for communication.

The primary socket API functions and methods in this module are:

* socket()
* bind()
* listen()
* accept()
* connect()
* connect\_ex()
* send()
* recv()
* close()

**Algorithm**:

1. Initialize the socket onto the host(given IP and Port Address)
2. Start listening on the port specified and wait for any request
3. If there is any request from the client(through the defined port) accept the request and establish the connection, simultaneously Look for other request actively from the other clients if any
4. Start to receive messages from the connected clients and also broadcast the received messages to all other connected clients along with our message if any(enter in the command line)
5. All the connections can be ended if any of connected client or the host enter “end”
6. Thus creating an Chat room using socket programming in python

**Note:**

The clients are nothing but the node(individual Machine) while the Server is the Host(individual Machine)

For the security purposes it is desired to use the loopback IP-address (127.0.0.1) using which the communication can be established only within the Host-network

Let’s analyze breaking the code in pieces

**Libraries:**

import socket  
import sys  
import \_thread

the “socket” library contains all the required methods for the socket programming,The primary socket API functions and methods in this module are:

* socket()
* bind()
* listen()
* accept()
* connect()
* connect\_ex()
* send()
* recv()
* close()

while the “sys” library contains the methods for accessing the OS variables or operations in our program we used it for inputting the IP-address through the command line arguments (directly from the Shell), since there should be many active operations to be keep going the thread library is used in simple to perform “Multithreading”

IP\_address = str(sys.argv[1]) # Standard loop-back interface address (127.0.0.1-local-host)  
Port\_ID = int(sys.argv[2])

As said earlier the above lines tend to intake the user input while execution through command line, the “sys.argv[0]” by default holds the file name

**Server:**

server = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
server.bind((IP\_address, Port\_ID))  
server.listen(Max\_clients)

the first line creates an object of the socket of the characteristics “socket.AF\_INET, socket.SOCK\_STREAM” where the socket.AF\_INET defines the type of IP-version as “IPV4”,while there is another version called “IPV6” which is developed due to increased number of networks for now just keep in mind that they define the frame format for the data communication. While the socket.SOCK\_STREAM defines the type of IP-protocol to use here it defines TCP which is more reliable and error free, while another type of protocol can also be used which is Termed as UDP which is not reliable yet very fast but there may be transmission loss

the server.bind associates with the defined ip-address and port

and the server.listen(max\_clients) will be snooping over the binded port for any request(from client) “max\_clients” variable says the .listen() to snoop request for total “max\_clients” value

if its 10 then 10 clients can enter the chat room if 100 then 100 clients can enter.

def connection\_establish():  
 count = 0  
 while count < Max\_clients:  
 print("Waiting for Client to Connect.")  
  
 new\_client\_connect, IP\_address\_of\_client = server.accept()  
 list\_of\_client\_IP.append(IP\_address\_of\_client)  
 list\_of\_client\_socket.append(new\_client\_connect)  
 print("New client of IP-address: " + repr(list\_of\_client\_IP[-1]) + "is connected")  
 count = len(list\_of\_client\_socket)  
 print("List of Connected devices is: " + repr(list\_of\_client\_IP))  
 \_thread.start\_new\_thread(receive\_message(IP\_address\_of\_client), ())  
connection\_establish()

if there is any request from the clients the server has to respond right so the “server.accept()” comes into the pitch since we have to accept many clients into our chat room we need to put it in a while loop where the condition is present to accept only the desired number of clients to be in the room which is max\_clients

the server.accept returns a socket object of the client and the ip&port ID of the client

repr() is more or less similar to the str() which is returns the passed arguments within “ ‘ ‘ ” and more precise output

the \_thread.start\_new\_thread() takes the function() and the arguments of that function in its arguments and execute that function as a thread or a separate process so that the code does not get looped within that function ie: the connection\_establish() function has a while loop which goes forever(if max\_clients are not joined into the room) in that case the rest of the code in our program will not be executed hence these threads are used in order to keep the whole function running as a separate process

at last there is a function call for the “connection\_establish()”

the function called through that thread is receive\_messages(IP\_addres\_of\_clients) which is given below

def receive\_message(ip):  
 while True:  
 for clients in list\_of\_client\_socket:  
 message = clients.recv(1024).decode()  
 broadcast(message, ip, clients)  
 print(str(ip) + ": " + repr(message))

there a infinite loop where the messages from all the connected clients are received and stored into the message variable

**.recv()** method will return the received message from the client object

the arguments passed to the receive\_message() ie; ip is nothing but the ip address of the client who sends the message it is then broadcasted to every client except for the one who sent it

this can be understood by elaborating the broadcast() function

def broadcast(received\_message, clients, client\_socket):  
 for client in list\_of\_client\_socket:  
 if client != client\_socket:  
 received\_message = str(clients) + ": " + received\_message  
 client.send(received\_message.encode())

the broadcast() function takes 3 arguments which is the

1. message\_recieved
2. clients – Ip address
3. clients\_socket object

the for loop goes through all the client\_socket object and sends the received\_message to all the clients along with the prefixed IP address using client.send() except for the client from whom the received\_message is obtained

def send\_message():  
 while True:  
 my\_message = input("=>")  
 if my\_message == "show connected device":  
 print("List of Connected devices is: " + str(list\_of\_client\_IP))  
 else:  
 broadcast(my\_message, IP\_address, None)  
 if my\_message == "end":  
 for all\_clients in list\_of\_client\_socket:  
 all\_clients.close()  
 server.close()  
 exit(0)  
  
  
\_thread.start\_new\_thread(send\_message, ())

finally for us to send the message to all other clients this send\_message() function is defined, what it does is it prompts the user to input something which you want to chat and then check if you have entered “show connected device” if you have entered the same then it will print the list of connected IPs else it will broadcast your message to all the clients if you have entered end then all the connection are closed and the program is terminated

if you don’t want all the connections to be terminated only by the host not by all other clients then rename the my\_message in the above for inputting your message to something else and also don’t forget to change the arguments passed to the broadcast() function and the if condition

Note:

If you are confused about repr() you can use str() instead

.encode() method will just encode your string into an default encoding format like UTF-8

**Client:**

The client lies simple and similar to server

import socket

import sys  
import \_thread  
my\_message = "HI"  
IP\_address = str(sys.argv[1]) # Host IP-addresss  
Port\_ID = int(sys.argv[2]) # HOST port ID  
  
client = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
client.connect((IP\_address, Port\_ID))  
  
  
def send\_message(threadname, my\_message):  
 while my\_message != "end":  
 my\_message = input("=>")  
  
 client.send(my\_message.encode())  
 if my\_message == "end":  
 client.close()  
 exit(0)  
  
  
\_thread.start\_new\_thread(send\_message, ("client\_send\_thread", my\_message))  
while True:  
 data = client.recv(1024).decode()  
 print(str(data))  
  
client.close()

it checks for my\_message equal to end if not prompts to enter something and then sends it to the server if entered “end” then the client terminates this is run infinite times through multithreading and the program enters a infinite while loop where the message from the server is being snooped continuously and printed over the command line

**Server\_Socket\_complete Code:**

import socket  
import sys  
import \_thread  
from threading import Thread  
  
# Simple initialization of variable(in python they should be termed as objects)  
  
Max\_clients = 10  
message = "Welcome to chat room!!"  
list\_of\_client\_IP = []  
list\_of\_client\_socket = []  
  
# inputting the the IP-address and port id for the HOST  
  
IP\_address = str(sys.argv[1]) # Standard loop-back interface address (127.0.0.1-local-host)  
Port\_ID = int(sys.argv[2]) # Port to listen on (non-privileged ports are > 1023)  
print("Server successfully Established on IP-address: " + IP\_address + "and Port: " + str(Port\_ID))  
  
# 3 main steps to establish a socket  
# server is the object of socket created  
# bind associates all IPaddress and the port  
# .listen() keeps snooping through the port for any requests  
server = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
server.bind((IP\_address, Port\_ID))  
server.listen(Max\_clients)  
  
# this function tends to broadcast the messages to every client except back to the sender itself  
  
def broadcast(received\_message, clients, client\_socket):  
 for client in list\_of\_client\_socket:  
 if client != client\_socket:  
 received\_message = str(clients) + ": " + received\_message  
 client.send(received\_message.encode())  
  
# this fuunction recieves the messages from all the connected clients   
  
def receive\_message(ip):  
 while True:  
 for clients in list\_of\_client\_socket:  
 message = clients.recv(1024).decode()  
 broadcast(message, ip, clients)  
 print(str(ip) + ": " + str(message))  
  
# this funtion tend to send the message by the the host to every client connected  
  
def send\_message():  
 while True:  
 my\_message\_s = input("=>")  
 if my\_message\_s == "show connected device":  
 print("List of Connected devices is: " + str(list\_of\_client\_IP))  
 else:  
 broadcast(my\_message\_s, IP\_address, None)  
 if my\_message\_s == "end":  
 for all\_clients in list\_of\_client\_socket:  
 all\_clients.close()  
 server.close()  
 exit(0)  
  
  
\_thread.start\_new\_thread(send\_message, ()) # multithreadding  
  
# accept any of the client request from the defined port and pass the object to the recieve\_message funtion   
# for every new client joining the server  
def connection\_establish():  
 count = 0  
 while count < Max\_clients:  
 print("Waiting for Client to Connect.")  
  
 new\_client\_connect, IP\_address\_of\_client = server.accept()  
 list\_of\_client\_IP.append(IP\_address\_of\_client)  
 list\_of\_client\_socket.append(new\_client\_connect)  
 print("New client of IP-address: " + repr(list\_of\_client\_IP[-1]) + "is connected")  
 count = len(list\_of\_client\_socket)  
 print("List of Connected devices is: " + repr(list\_of\_client\_IP))  
 \_thread.start\_new\_thread(receive\_message(IP\_address\_of\_client), ())  
connection\_establish()

**Client\_socket code:**

#!/usr/bin/env python3  
import sys  
import socket  
import \_thread  
my\_message = "HI"  
IP\_address = str(sys.argv[1]) # Host IP-addresss  
Port\_ID = int(sys.argv[2]) # HOST port ID  
  
client = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
client.connect((IP\_address, Port\_ID))  
  
  
def send\_message(threadname, my\_message):  
 while my\_message != "end":  
 my\_message = input("=>")  
  
 client.send(my\_message.encode())  
 if my\_message == "end":  
 client.close()  
 exit(0)  
  
  
\_thread.start\_new\_thread(send\_message, ("client\_send\_thread", my\_message))  
while True:  
 data = client.recv(1024).decode()  
 print(str(data))  
  
client.close()

**How to Execute:**

1. Navigate to the folder where Server\_py.py is present
2. Open command prompt or terminal over there and enter “python Server\_py.py 127.0.0.1 6000” note: port number can be any value greater than 1024 and the ip-address given here is a loopback ip which can be used for communicating within the host network for communication through Ethernet use public ip-address but on your own risk where port forwarding is performed
3. Now the server should start running now navigate to the folder where Client\_py.py is present and open command prompt or terminal and enter “python Client\_py.py 127.0.0.1 6000” here you should enter the ip address and the port with which you have created the server\_socket if not mentioned in the server then use “socket.gethostname()” method for clear view go through <https://www.geeksforgeeks.org/display-hostname-ip-address-python/>
4. now you should see a connection being established hooray you can start sending messages
5. further from another node you can follow step 3 to connect to the server

**Thank you**