Python ChatRoom Based on TCP AI

### 

Title: AI TCP ChatRoom In Python

# Chapter 1 - Introductory

## 1.1 Introduction:

The chatroom is excellent for communication over the network and the TCP connection ensures reliability.TCP Chatroom is built via the computer networking programming language Python3 which is the latest one. This TCP chatroom enables us to have one server that hosts the chat while multiple clients are connected to it and then communicate with each other via the main server.

This is a full-working chatroom, and you can easily send and receive messages also over the internet and locally by using the server’s IP. The port needs to be opened for the public connection.

Another advantage of this chatroom is that a rule of kicking and banning is activated as the admin site which is implemented using AI techniques so that the errors can be handled in such a way that becomes useful and trigger specific operation.

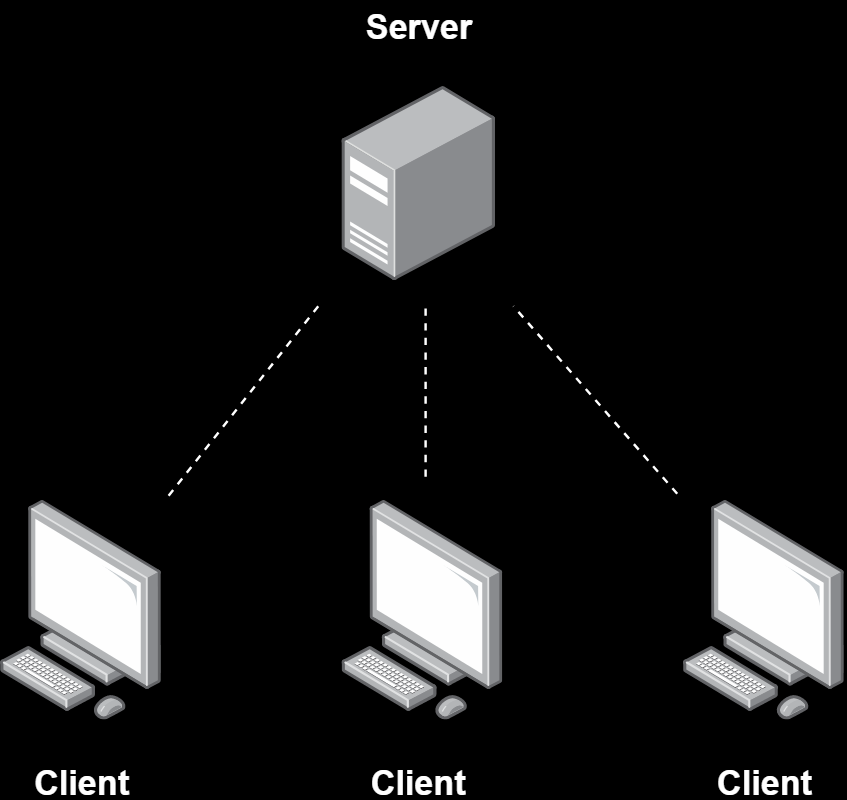
## 1.2 Abstract:

We’ve built a chatroom in python based on TCP protocols. The TCP protocol is the transmission control protocol so that the communication between clients to clients and clients to servers is reliable. The message transmitted over TCP sends the report back to the server if any error occurred why it is more preferable to UDP.

While using python3 we used multi-threading and socket language to implement it using two libraries socket and threading and whenever a client connects to the network a new thread is generated to handle it. So on disconnection of the client, an error-handling procedure is the cause of the helping source to tell the server that there is an error triggered at the client side which is ultimately taken as client disconnection.

We have established a server-client network, where the messages are centralized and each client talks to others via the centralized server. The non-reserved port is assigned to the server.

### **1.2.1 Server-Client Network:**



#### **1.2.1.1 Server-side**:

On the server-side, the server is established using an INET socket and TCP connection. The IP is used as the server’s IP and the port is the non-reserved one. The server always receiving the client’s messages while multiple threads are running for multiple clients. It accepts the client connection.

The message is broadcasted to all clients via the server.

#### **1.2.1.2 Client-Side:**

On the client-side, the running server’s IP and port are needed to connect to it. After a successful connection, it then needs to write the message to broadcast via the server. There are two threads on the client-side, one for receiving and others for writing messages. Broadcasted messages are shown on each client.

The server must be running for the Client. On the server side, 3 different modules manage, broadcast the messages, and receives the messages from the client.

#### **1.2.1.3 Extra Powers and Features:**

Extra powers and features are the functionality of a user to kick and ban a specific user based on the implementation of kicking and banning algo. implemented by using the AI techniques to handle the clients in such a way that the error can be used to add the client to the ban list or kick. If the client has been kicked then he can rejoin. But if he was banned, he wasn’t able to come back and a bann option occurs until he gets unbanned.

## 1.3 Pros of Our TCP Chatroom:

1. Based on Latest Python 3 which is more reliable which supports the latest technology.
2. The total software size is less than an MB so even a computer with less than a 500 MB ram can run it.
3. LAN compatible.
4. WAN compatible ( By specifying open ports and host address).
5. Multi-threading is compatible.
6. ASCII encryption for messages.
7. TCP is based on reliable communication.
8. Structural programming.
9. Upgradable.

## 1.4 Technology:

* Python 3
* Pycharm
* Windows 10

## 1.5 Requirements:

* Python 3
* 500MB Ram, 10MB storage
* Multi-threading CPU ( Recommended)
* Python enabled machine

# Chapter 2 Methodology:

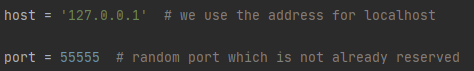
The procedure for Implementing the server:

## 2.1 Server Implementation:

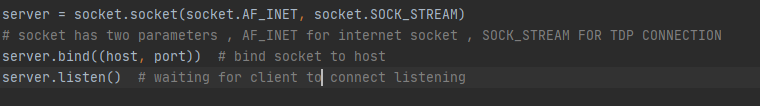
1. For Network connection and performing operations, we have imported two library files, socket, and threading.



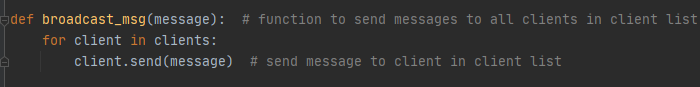
1. The host address is generated, in this, we use the localhost address but for the server. The server address has to be used and the port which is not reserved is to be used.



1. The server socket is defined using AF\_INET for internet socket and SOCK\_STREAM for TCP protocols. Then the listen() module starts the server to listen.



1. Two lists are generated to store clients and their nicknames. Then the module Broadcast is used for broadcasting the messages to all clients are created.

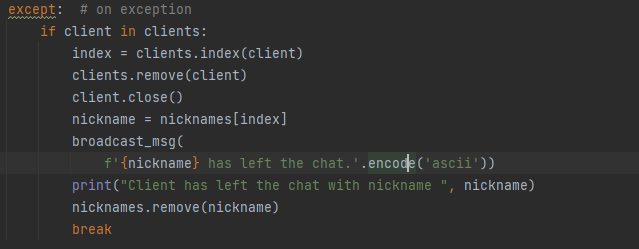


1. The module manage gets client as an argument is used for managing the client’s request and broadcast the messages to all clients. if an exception occurs the client is removed and a disconnection interrupt is broadcasted to the rest clients the loop is then terminated to close the thread.



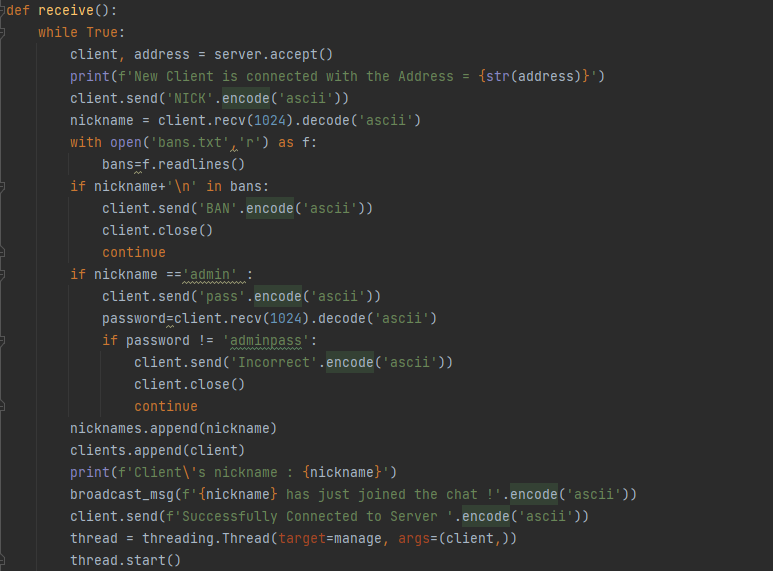
If the message is kick or ban from the user then if the user is admin then specific command is executed. and on the execution of the ban command banned nickname is added to the bans.txt.

**Exception Handling:**



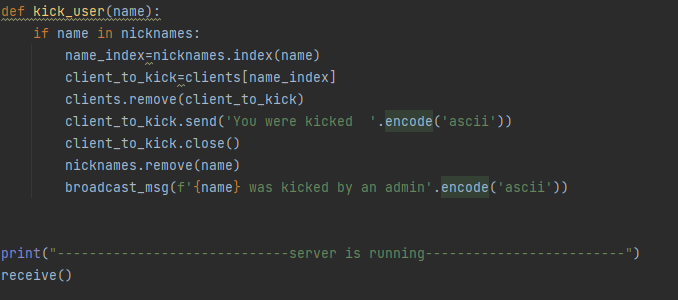
If the exception is encountered mean, the client is no longer connected then it is removed from the client’s list and the message then broadcasted.

1. The module Recieve is executed when the server is started. It accepts the client’s connection and asks for the nickname. It is responsible for managing and executing all other modules and starting a new thread for the new client. It constantly executes while the server is up.



The client is being checked in the txt file, if he is banned then he is not allowed to enter the chatroom. if it is admin then the password is asked which on in correction, the client is removed due to encryption and wrong password to save privacy.

1. Function kick\_user



Function kick\_user gets the name parameter and then kicks the specific person the message broadcasted to all users.

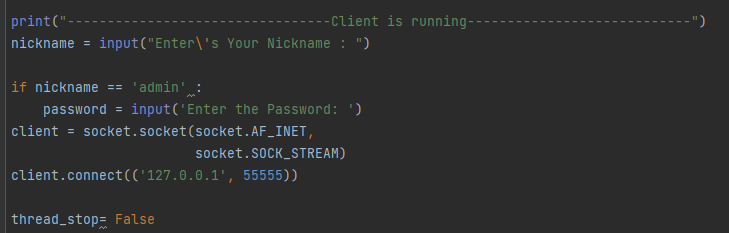
## 2.2 Client Implementation:

Procedure to Implement the Client Script:

1. Same as Server, the two libraries are imported, Socket and Thread.

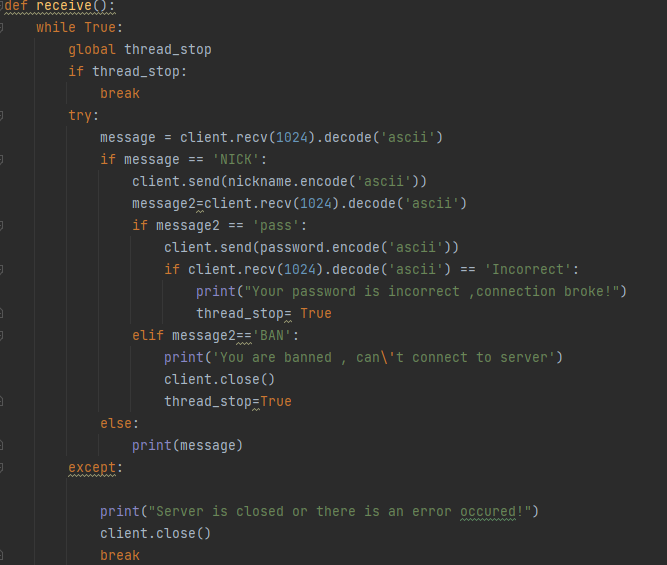


1. The client is asked to enter the nickname which is then used as its identifier in the broadcasting messages and to the server.



Nickname is checked on the client site if he is the admin, the password is pre-saved on the client site so it can be sent to the server. A thread\_stop variable is made to be used later.

The client’s socket is defined over the internet socket and TCP connection, Then the client is connected to the server via the server’s IP and port.

1. The receive module is generated to carry out the messages from the server. The messages encoded in ASCII are decoded into again its original type. If the message is NICK then the user has to give the nickname for identification purposes. Using the print the broadcasted messages are printed over the client’s screen which is from another client via server. If an exception occurred, the error will be shown. 

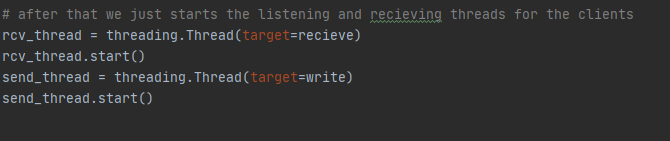
The above program also shows that, if the server asks for a pass then a pass must be provided by the client. An incorrect password means the client can’t log in.

1. The module write is defined for the user to write messages to the server, then broadcast to the clients. Messages are encoded in ASCII so they can be sent in bytes.

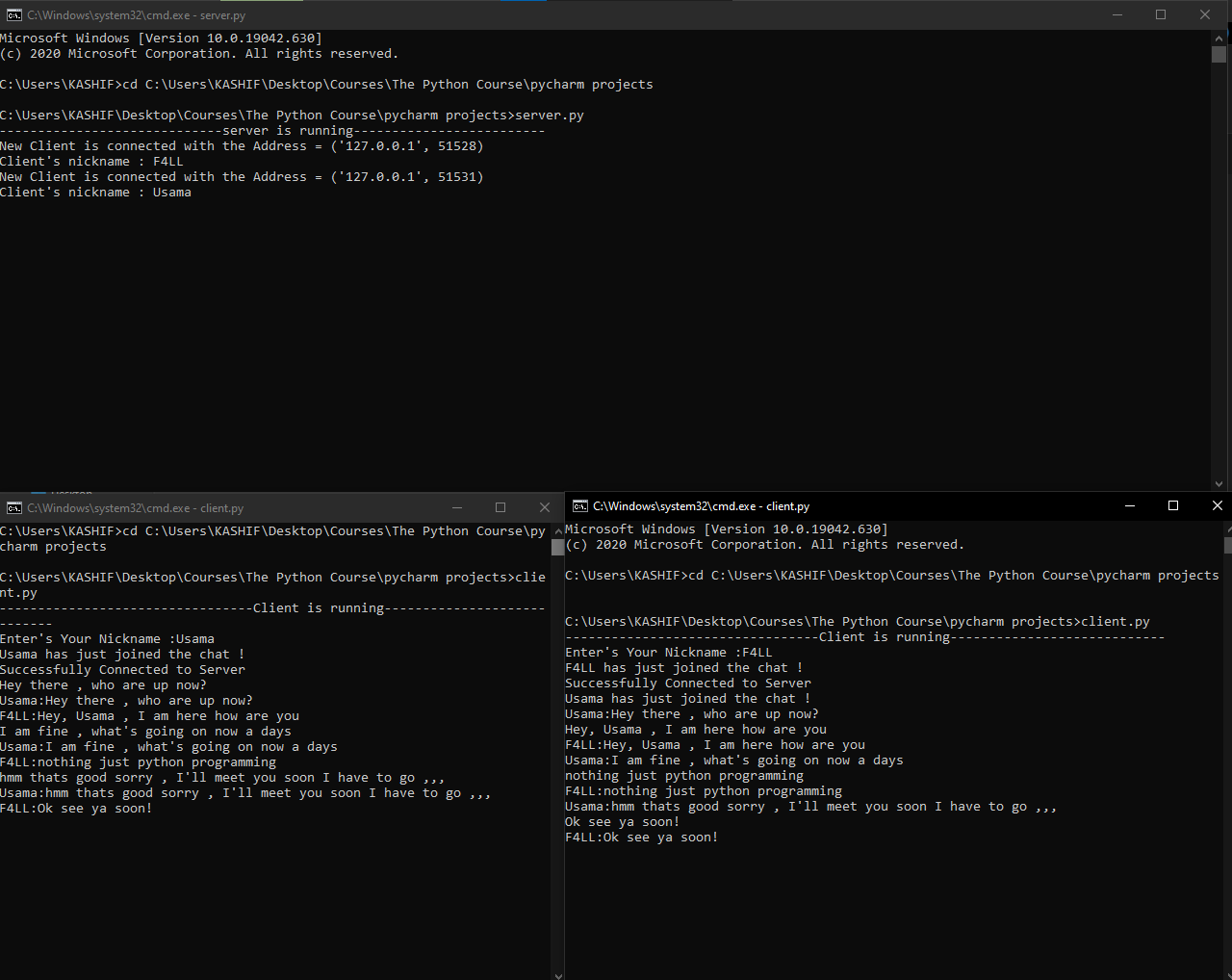


Message is written by the user, if he is a admin then he can perform commands start with $ symbol , like kick and ban , as name ,:1 and space 1 1+1=2 numbers so the $ must be at nickname+2 where $kick+space=6 and $ban+space=5.

1. Two Threads are generated for write and receive messages, which is started by using start().



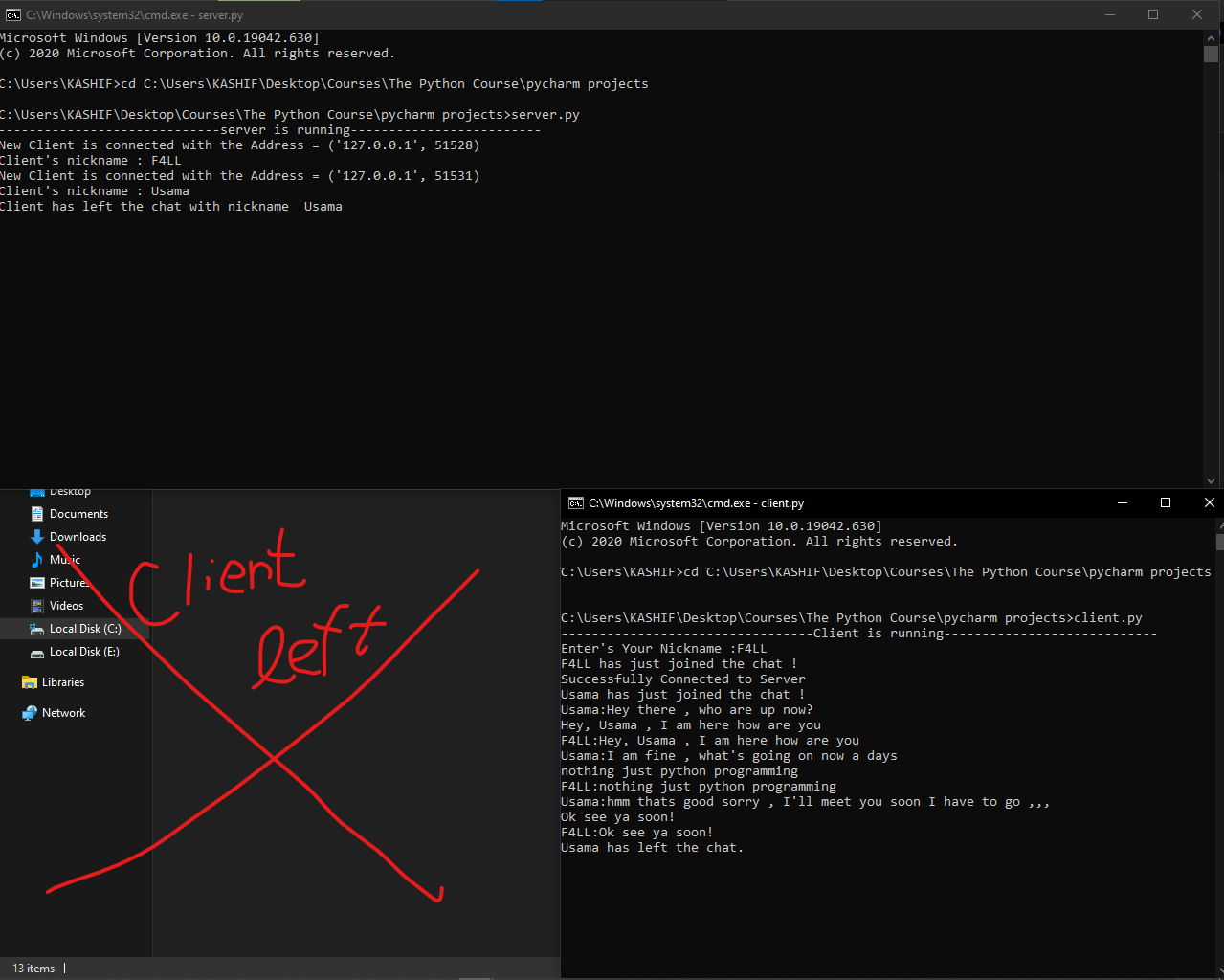
## 2.2 Implementation:



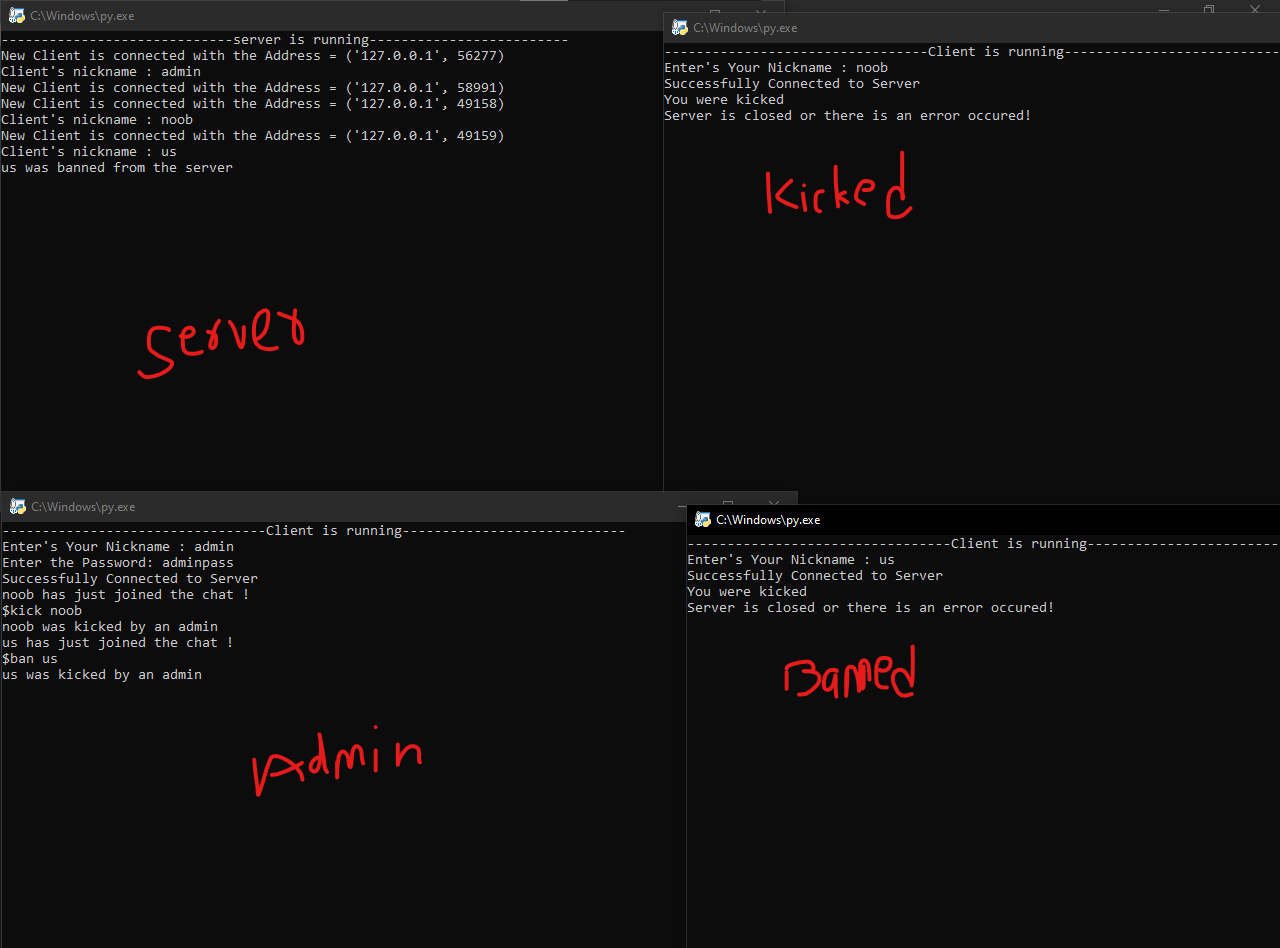
Here we can see the TCP chatroom is running with multiple clients (let’s consider two clients ), The client is named nickname Usama, and the other client is named nickname as F4LL when Usama wants to communicate to the F4ll, he writes the message which broadcasted via the server and seen on the client’s side.

While on the server’s side the client’s IP and port and his connected nickname which he wrote are seen. If any client leaves the chatroom, it displays the specific client left.

So, it enables multiple clients to communicate with each other easily by connecting to the server.



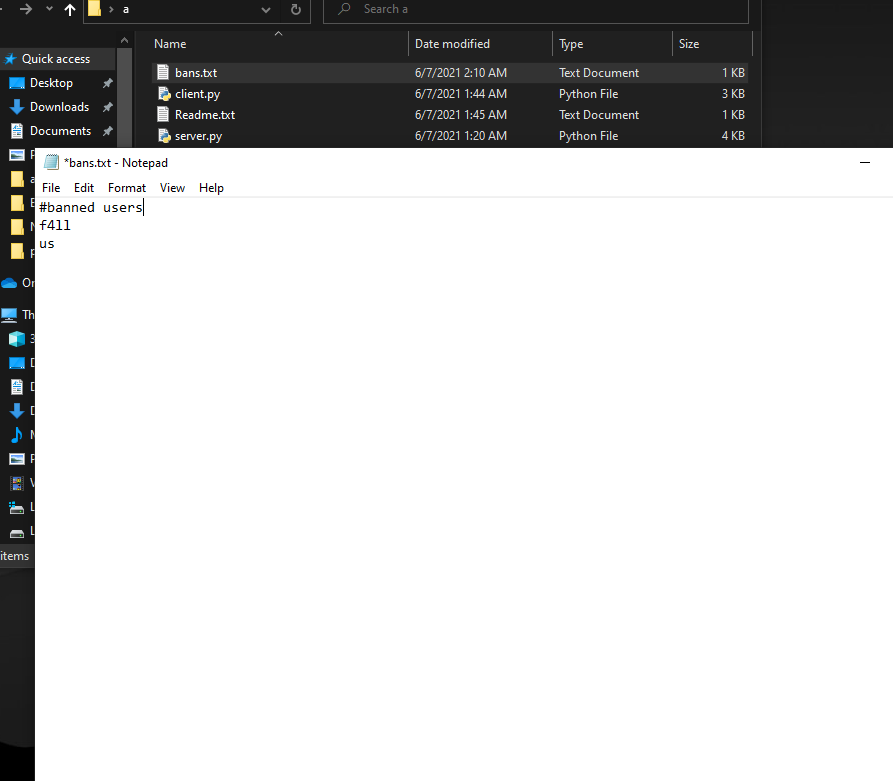
## 2.3 Ban and Kick Feature:



In the above example, admin banned us and kicked noob , noob can reconnect but the us can’t. admin has the privilege. Admin is asked a password to enter but not the others.

**Ban.txt:**

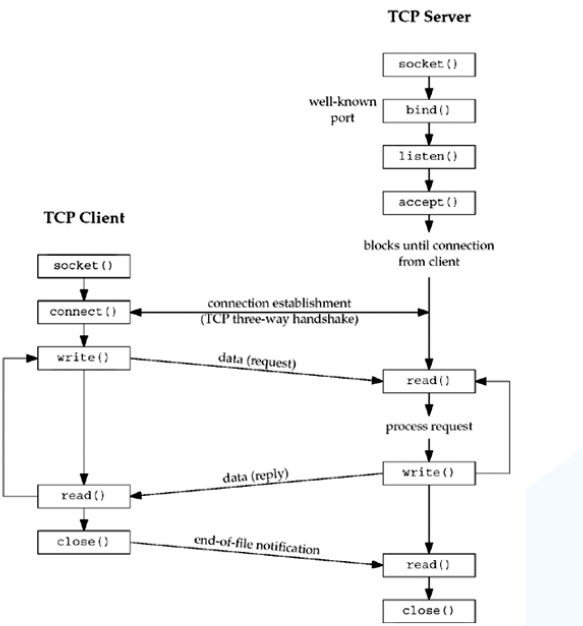
It contains all the banned users. (For now, i haven’t added any encryption to this txt file as it is on the server pc and can’t be accessed by client)

****

## For The Public Connection:

To implement the public connection, the port will need to be opened for the public. The port can be checked online if it is opened for the specific IP or not while some ports are reserved. The public IP can then be used to connect to the server.

## TCP Chart:



# Conclusion:

At the very end, we have our awesome and lightweight TCP chat room running on full-efficient. This chatroom is efficient in terms of size and term of speed. The clients can be connected from different places by providing the server’s public IP and opening the port. In the example, we use the local hosts but can be implemented publicly.

This awesome TCP chat room ultimately enables you to communicate with your project members, school friends, office colleagues easily just by adjusting a host server.

# Citation:

* <https://en.wikipedia.org/wiki/Transmission_Control_Protocol>
* <https://en.wikipedia.org/wiki/Python_(programming_language)>
* <http://docs.python.org/3/howto/sockets.html>
* <https://docs.python.org/3/library/threading.html>
* The Python Bible Volume 1: Python Programming For Beginners (Basics, Introduction) Kindle Edition