Chatroom manual

pour creer une program qui nous , on dois avoir deux codes: le code de serveur et le code du client

- le code du serveur: il nous permit de initialiser une serveur pour diffuser(broadcast) les messages.
    - dans cette example, le nome de fichier qui contient cette code est appele: **chatroom.py**
- le code du client: chaque client doit executer cette code pour pouvoir d'envoiyer et de recevoire une message

1- chatroom.py

tout d'abord, on commence par importer les modules qu'on va utiliser:

```
import threading
import socket

host = '127.0.0.1' // ip address du localhost(loopback ip address)
port = 55555

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

server.bind((host, port)) // on specifie l'addrese du serveur

server.listen() // lancer le serveur
```

l'ip address est comme une addresse pour l'ordinateur(inclut aussi presque tous les appareils: IOT)
port est comme une specefique addresse dans l'ip address(par exemple: l'addresse d'une maison dans une addresse specefique), donc il'es utilise pour une specifique processus.

- ip address: est une addresse pour l'appareil.
- port: addrese d'une service/processus dans une appareil

socket est une endpoint(point de termination) où les données seront envoyées et reçues.
au dessus, pour creer

- ip address + port: identifie une socket.

donc, au-dessus on a specifie deux parametres pour creer une serveur

- **AF_INET:** specifier la famille d'addresse, ici on a IPV4 mais il y'a une autre famille qu'il s'appel IPV6(nouveau)
- **SOCK_STREAM:** specifier le type du socket, ici on a TCP(plus fiable) au lieu de UDP(plus rapide)

```
clients = []
nicknames = []

def broadcastMessage(message):
    for client in clients:
        client.send(message)
```

- la fonction du braodcastMessage envois la message a tous les client qui sont connecte a le serveur

```
def handleClient(client):
    while True:
        try:
            message = client.recv(1024)
            broadcastMessage(message)
        except:
            index = clients.index(client)
            clients.remove(client)
            nickname = nicknames[index]
            print(f"{nickname} left the chat !".encode(ascii)) # why encoding in ascii?
            nicknames.remove(nickname)
            break
            # so we're constantly trying to get messages from the client. this will give you
```

dans cette fonction, on lance une boucle infinie:

- si tout va bien: le client recevois une message, 1024 est le maximum des donnes que le client peut recevoir a la meme fois, il implique 1024 octets(buffer size)
- si il'y a une une erreur(c'est a dire: si le client n'existe plus): on supprime le client dans la liste des clients et supprime le nom du client. on ecrit aussi que a les clients que le client specifie a partie

fonction encode: pour convertir le messages en bytes, on decode dans le code du client

```
def receive():
    while True:
        client, address = server.accept()
        ## in this case we're always gonna have one address because we send from the same co
        print(f'Connected with {str(address)}')
        client.send("NICK".encode('ascii')) ## send a message to the client
        nickname = client.recv(1024).decode('ascii')
        nicknames.append(nickname)
        clients.append(client)

        broadcastMessage(f"{nickname} has joined the chat room".encode("ascii"))
        client.send("Connected to the server".encode("ascii"))

        thread = threading.Thread(target=handleClient, args=(client,))
        thread.start()
```

cette fonction lance aussi une boucle infine, s'il y'a une connectoin la fonction accept recoit le client et l'addresse(on fait ici unpacking)
qu'on on envoit "NICK": ca signifie que on demande de nickname du client

```
 thread = threading.Thread(target=handleClient, args=(client,))
        thread.start()
```

ici on lance une thread pour chaque client, pour les gerer a la meme fois.

3- **Client.py**

```
import threading, socket


nickname = input("choose a nickname: ")

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(('127.0.0.1', 55555))
```

ici on fait la meme chose que precedant

```
def receive():
    while True:
        try:
            message = client.recv(1024).decode('ascii')
            if(message == 'NICK'):
                client.send(nickname.encode("ascii"))
            else:
```

```
                print(message)
        except:
            print("An error occurred")
            client.close()
            break
```

on lance aussi une boucle infinie pour que le client recevoit une message, si cette message est "NICK" alors ca signifie que le serveur nous demande de envoiyer le nickname
si il y'a une erreur alors on ferme le client

```
def write():
    while True:
        message = f"{nickname}: {input("")}"
        client.send(message.encode("ascii"))
```

cette fonction est pour l'ecrire d'une message

```
receive_thread = threading.Thread(target=receive)
write_thread = threading.Thread(target=write)

receive_thread.start()
write_thread.start()
```

alors on lance ces deux fonctions dans deux threads different, pour recevoir et envoyenment a la meme fois