srv

class Etudiant :

"Définition d'un étudiant avec son nom, prénom, liste de notes et coeff"

def \_\_init\_\_(self, nom, prenom, notes): #un seul init par classe

self.nom = nom #ajoute valeur passée en paramètre

self.prenom = prenom

self.notes = notes

def ajouternote (self,note, coeff):

self.notes.append([note, coeff])

#print (self.notes)

def nbnotes (self):

nb = len(self.notes) # ou len(notes) possible avec etud1.nbnotes(etud1.notes)

print(nb)

def moyenne (self):

somme = 0

coeff = 0

for i in range (len(self.notes)):

somme = somme + int(self.notes[i][0])\*int(self.notes[i][1])

coeff = coeff + int(self.notes[i][1])

moy = somme / coeff

print (moy)

return moy

def afficher (self):

print("le nom de l'étudiant est : ", self.nom)

print("le prénom de l'étudiant est : ", self.prenom)

print("la liste de notes de l'étudiant est : ", self.notes)

#Programme principal

listetud=[]

etud1 = Etudiant("Dupont", "Bernard", []) #ajoute attribut de la classe à etud1

#[] permet de rentrer les notes dans un tabl eau avec une méthode (ici ajouternote) car 2 paramètres

#print(etud1) --> affiche l'emplacement dans la RAM de l'etud1 #permet d'afficher les attributs

etud1.ajouternote(15,6)

etud1.ajouternote(15,6)

#etud1.nbnotes(etud1.notes)

listetud.append(etud1) #etud1 non crée par le client

class Promotion :

def \_\_init\_\_(self, promo, liste):

self.promo = promo

self.liste = liste

def ajouteretudiant(self,etudiant): #on récupère les étudiants déjà crée

self.liste.append(etudiant)

def nbetudiant(self):

nb = len(self.liste)

print (nb)

def moyenne(self):

tot = 0

for i in range (len(self.liste)):

tot = tot + self.liste[i].moyenne() #pour chaque étudiant de la liste, fonction moyenne de la classe étudiant est utilisée

print (tot)

promo1 = Promotion(1,[])

promo1.ajouteretudiant(etud1) #pas besoin de de renseigner chaque nom, prénom...

print(promo1.liste[0].prenom) #dans la liste d'étudiant qui est un tableau, on veut un objet

promo1.nbetudiant()

promo1.moyenne() #affiche plusieurs moyenne car print dans moyenne étudiant

listepromo = []

listepromo.append(promo1)

###############

#SERVEUR#

# Serveur TCP Multi Thread

import socket

import os

from \_thread import \*

ServerSocket = None

host = '127.0.0.1' #même addresse et port srv et client

port = 9090

clients = []

nbclients = 0

numclient = None

def main():

global nbclients

ServerSocket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

ServerSocket.setsockopt(socket.SOL\_SOCKET, socket.SO\_REUSEADDR, 1)

try:

ServerSocket.bind((host, port))

except socket.error as e:

print(str(e))

finally:

print('Waiting for a Connection..')

ServerSocket.listen(50) #le srv peut écouter 50 clients max en même temps

while True:

client, address = ServerSocket.accept()

print('Connected to: ' + address[0] + ':' + str(address[1]))

client.send(str.encode(str(nbclients)))

clients.append(client)

print("Liste clients : ", clients)

start\_new\_thread(threaded\_client, (client, ))

nbclients+=1

print('Thread Number: ' + str(nbclients))

def commande(data, connection):

global nbclients

global clients

reponse = "Non" #initié ici, donc si renvoi de "non" sur le client = pas de valeurs dans réponse donc erreur (correspond pas au if)

repartition = data.decode('utf-8').split("|") #décode la requête du client (variable data) puis sépare les valeurs

####################################################

#créer un étudiant

if repartition[0] == "Etudiant":

notes = repartition[3].split(";") #séparation des notes

etudiant = Etudiant(repartition[1], repartition[2], []) #rajoute à la classe Etudiant

for i in range (len(notes)): #séparation des notes et coeff

depart=notes[i].split("/")

etudiant.ajouternote(depart[0],depart[1]) #ajout dans liste notes de la classe Etudiant

reponse = "\n\nÉtudiant ajouté avec succès !\n" #renvoi au client

etudiant.afficher() #affiche l'étudiant crée dans le terminal srv

listetud.append(etudiant) #ajoute etud dans la liste pour vérifier qu'il existe plus tard

#####################################################

#ajout de notes

elif repartition[0] == "Ajout":

notes = repartition[2].split(";") #séparation des notes

nom = repartition[1] #on récupère le nom pour le trouver dans listetud

etudiant\_existe = 0

for etudiant in listetud: #boucle de test, si nom existe pas, ne fait rien et le client reçcoit "non" car reponse est vide

if etudiant.nom == nom:

etudiant\_existe = 1

for i in range(len(notes)):

depart = notes[i].split("/")

etudiant.ajouternote(depart[0], depart[1])

reponse = "\n\nNotes ajoutées avec succès !\n"

etudiant.afficher()

if etudiant\_existe == 0 :

reponse = "\n\nL'étudiant n'existe pas, veuillez le créer.\n"

####################################################

#moyenne

elif repartition[0] == "Moyenne":

nom = repartition[1]

etudiant\_existe = 0

for etudiant in listetud:

if etudiant.nom == nom:

etudiant\_existe = 1

etudiant.moyenne()

etudiant.afficher()

reponse = f"\n\nLa moyenne de {nom} est {etudiant.moyenne()}\n"

if etudiant\_existe == 0 :

reponse = "\n\nL'étudiant n'existe pas.\n"

#############################################

#créer promo

elif repartition[0] == "Promo":

promo = Promotion(repartition[1],[])

reponse = "\n\nPromotion ajoutée avec succès !\n"

listepromo.append(promo)

###########################################

#ajouter un étudiant dans une promo

elif repartition[0] == "AjoutEtud":

etudiant = Etudiant(repartition[2], repartition[3], [])

notes = repartition[4].split(";")

for i in range (len(notes)):

depart=notes[i].split("/")

etudiant.ajouternote(depart[0],depart[1])

reponse = "\n\nÉtudiant ajouté avec succès !\n"

etudiant.afficher()

listetud.append(etudiant)

for promo in listepromo :

if repartition[1] == promo:

promo.ajouteretudiant(etudiant)

reponse = "\n\nL'étudiant est ajouté dans la promo.\n"

else:

reponse = "\n\nLa promo n'existe pas, veuillez la créer.\n"

connection.send(reponse.encode())

def threaded\_client(connection):

global nbclients

global clients

while True:

data = connection.recv(2048)

reply = '\n>>' + data.decode('utf-8') + '\n'

for client in clients:

client.sendall(str.encode(reply))

# Appel de la fonction de traitement de la commande

commande(data, connection)

if \_\_name\_\_== "\_\_main\_\_":

main()

client

import socket

import os

from \_thread import \*

ClientSocket = None

host = '127.0.0.1'

port = 9090

myNumber = 0

def main():

ClientSocket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

ClientSocket.setsockopt(socket.SOL\_SOCKET, socket.SO\_REUSEADDR, 1)

try:

ClientSocket.connect((host, port))

except socket.error as e:

print(str(e))

finally:

print("connected to Server !")

myNumber = int(ClientSocket.recv(1024))

print("myNumber client received : ", myNumber)

start\_new\_thread(threaded\_server, (ClientSocket, myNumber))

while True:

print("\nRequêtes possibles : \n1. Créer un étudiant \n2. Ajouter une note \n3. Calculer la moyenne d'un élève \n4. Créer une promo \n5. Ajouter étudiant dans une promo\n9. Quitter\n")

requete=input("Entrez la requête : ")

######################################################

#créaton étudiant

if requete== "1":

nom\_etud = input("Nom de l'étudiant : ")

prenom\_etud = input("Prénom de l'étudiant : ")

notes = input("Rentrez les notes et coeff (ex : 17/8;19/6): ")

# Créer une commande d'ajout d'étudiant

commande = f"Etudiant|{nom\_etud}|{prenom\_etud}|{notes}"

# Envoyer la commande au serveur

ClientSocket.send(commande.encode())

# Attendre la réponse du serveur (vous devrez implémenter cette partie)

reponse = ClientSocket.recv(1024)

print(reponse.decode())

##################################################

#ajout notes

elif requete== "2":

nom\_etud = input("Nom de l'étudiant : ")

notes = input("Rentrez les notes et coeff (ex : 17/8;19/6):")

# Créer une commande d'ajout d'étudiant

commande = f"Ajout|{nom\_etud}|{notes}"

# Envoyer la commande au serveur

ClientSocket.send(commande.encode())

# Attendre la réponse du serveur (vous devrez implémenter cette partie)

reponse = ClientSocket.recv(1024)

print(reponse.decode())

###############################################

#Moyenne

elif requete== "3":

nom\_etud = input("Nom de l'étudiant : ")

# Créer une commande d'ajout d'étudiant

commande = f"Moyenne|{nom\_etud}"

# Envoyer la commande au serveur

ClientSocket.send(commande.encode())

# Attendre la réponse du serveur (vous devrez implémenter cette partie)

reponse = ClientSocket.recv(1024)

print(reponse.decode())

###############################################

#créer promo

elif requete=="4":

promo = input("Nom de la promotion : ")

# Créer une commande d'ajout d'étudiant

commande = f"Promo|{promo}"

# Envoyer la commande au serveur

ClientSocket.send(commande.encode())

# Attendre la réponse du serveur (vous devrez implémenter cette partie)

reponse = ClientSocket.recv(1024)

print(reponse.decode())

############################################

#ajouter étudiant dans une promo

elif requete == "5":

promo = input ("Nom de la promotion : ")

nom\_etud = input("Nom de l'étudiant : ")

prenom\_etud = input("Prénom de l'étudiant : ")

notes = input("Rentrez les notes et coeff (ex : 17/8;19/6): ")

# Créer une commande d'ajout d'étudiant

commande = f"AjoutEtud|{promo}|{nom\_etud}|{prenom\_etud}|{notes}"

# Envoyer la commande au serveur

ClientSocket.send(commande.encode())

# Attendre la réponse du serveur (vous devrez implémenter cette partie)

reponse = ClientSocket.recv(1024)

print(reponse.decode())

elif requete=="9": #se déconnecte

print("\nAu revoir\n")

break

else :

print("\nRequête non valide. \n") #renvoi du choix de requête

def threaded\_server(connection, num):

while True:

response = connection.recv(1024)

print(response.decode('utf-8'))

if \_\_name\_\_== "\_\_main\_\_":

main()