#! python3

# Defining a server for sending and receiving messages

# through network, applying 2 child-thread

import socket, sys, threading,time

PORT = 64164

HOST = '192.168.0.64'#socket.gethostname()

#SHOULD BE 2 or 5

MAXUSER = int(input("User limit (2 or 5): "))

STARTLEVEL = 1

class ThreadClient(threading.Thread):

"""heritance of a thread-object to communicate with the client"""

def \_\_init\_\_(self, conn, thname):

threading.Thread.\_\_init\_\_(self, name=thname)

self.conn = conn

def run(self):

global glob\_ready, STARTLEVEL

#Communication with client

name= self.getName() # All threads have an ID

while True:

start=self.conn.recv(1).decode('UTF-8')

if start!=chr(0):

print("start:",start)

else:

msgClient=''

while True:

curr=self.conn.recv(1).decode('UTF-8')

if curr==chr(0):break

msgClient+=curr

try:

print(msgClient)

#msgClient=msgClient.decode('UTF-8')

except Exception as e:

print(e)

print(msgClient)

if msgClient=='#fin#' or msgClient=="":

break

if msgClient=='help.get':

x=True

locking.acquire()

for client in conn\_Cli:

if client!=name:

sendmsg(conn\_Cli[name],bytes("SERVER>>> "+client+' online.', 'utf-8'))

x=False

#time.sleep(0.5)

if x: sendmsg(conn\_Cli[name],bytes("SERVER>>> Currently you are the only one online", 'utf-8'))

sendmsg(conn\_Cli[name],bytes("#LEVEL#%s"%STARTLEVEL, 'utf-8'))

locking.release()

continue

if msgClient.startswith("!level"):

STARTLEVEL=int(msgClient.split()[1])

for client in conn\_Cli:

if client!=name:

message="%s> %s" % (name, msgClient)

sendmsg(conn\_Cli[client],bytes(message, 'utf-8'))

sendmsg(conn\_Cli[client],bytes("#LEVEL#%s"%STARTLEVEL, 'utf-8'))

continue

if msgClient.startswith("#pic#"):

glob\_ready=False

message= msgClient+"#pic#"+name

print(message)

locking.acquire()

for client in conn\_Cli:

if client!=name:

sendmsg(conn\_Cli[client],bytes(message, 'utf-8'))

img=b''

a=0

length=int(msgClient.split("#pic#")[1])

#sendmsg(conn\_Cli[name],bytes("#prog#"+str(len(str(length))), 'utf-8'))

time.sleep(0.15)

while a<length:

msgImg=self.conn.recv(4096)

img+=msgImg

a+=len(msgImg)

for client in conn\_Cli:

if client!=name:

conn\_Cli[client].send(msgImg)

#sendmsg(conn\_Cli[name],bytes("#prog#"+str(a),'utf-8'))

locking.release()

glob\_ready=True

continue

if msgClient.startswith("#GAME#"):

locking.acquire()

for client in conn\_Cli:

if client!=name:

sendmsg(conn\_Cli[client],bytes(msgClient,'utf-8'))

locking.release()

continue

message="%s> %s" % (name, msgClient)

print(message)

locking.acquire()

for client in conn\_Cli:

if client!=name:

sendmsg(conn\_Cli[client],bytes(message, 'utf-8'))

locking.release()

# Closing the connection

self.conn.close() # Server side connection close

locking.acquire()

for client in conn\_Cli:

if client!=name:

sendmsg(conn\_Cli[client],bytes("#DELETE#%s"%name, 'utf-8'))

sendmsg(conn\_Cli[client],bytes("SERVER>>> %s Has disconnected."%name, 'utf-8'))

locking.release()

del conn\_Cli[name] # deleting reg from dictionary

print("Client, %s has disconnected."%name)

#End of thread

def sendmsg(conn,msg):

conn.send(bytes(chr(0), 'utf-8'))

conn.send(msg)

conn.send(bytes(chr(0), 'utf-8'))

# Initializing server - creating socket

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

try:

mySocket.bind((HOST, PORT))

except socket.error:

print("Didn't manage to bind the socket with this address..")

time.sleep(2)

sys.exit()

print("Server is ready, waiting...")

mySocket.listen()

# Managing connecting clients

conn\_Cli={}

locking=threading.Lock()

glob\_ready=True

while True:

try:

connec, addr = mySocket.accept()

# Received a connection, initializing new thread

name=connec.recv(1024).decode('UTF-8')

if name in conn\_Cli:

name=name+'({0})'.format(addr[1])

while len(conn\_Cli)==MAXUSER:

connec.send(bytes('no', 'utf-8'))

continue

th= ThreadClient(connec,name)

while not glob\_ready:

continue

th.start()

# Registering the connection

it= th.getName()

conn\_Cli[it]=connec

print("Client, %s has connected, IP address %s, port %s." \

% (it, addr[0], addr[1]))

locking.acquire()

connec.send(bytes(name, 'utf-8'))

sendmsg(connec,bytes("SERVER>>> Successful connection, welcome, %s"%it, 'utf-8'))

sendmsg(connec,bytes("#MAXUSER#%s"%MAXUSER, 'utf-8'))

for client in conn\_Cli:

if client!=name:

sendmsg(conn\_Cli[client], bytes("SERVER>>> Client %s connected to the server, IP address %s, port %s." \

% (it, addr[0], addr[1]), 'utf-8'))

sendmsg(conn\_Cli[client],bytes("#PLAYER#%s"%name, 'utf-8'))

sendmsg(connec,bytes("#PLAYER#%s"%client, 'utf-8'))

locking.release()

except Exception as e:

print(e)