**servidor**

import socket

import threading

lock = threading.Lock()

class ThreadedServer(object):

def \_\_init\_\_(self, host, port):

self.lista = []

self.host = host

self.port = port

self.sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

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

self.sock.bind((self.host, self.port))

def listen(self):

self.sock.listen(5)

while True:

# print(threading.active\_count())

client, address = self.sock.accept()

client.settimeout(1000)

if threading.active\_count()>1:

self.addToLista(threading.Thread(target = self.listenToClient,args = (client,address)));

else:

threading.Thread(target = self.listenToClient,args = (client,address)).start()

def addToLista(self,t):

self.lista.append(t);

def consomeLista(self):

print(len(self.lista))

if(len(self.lista) >0 ):

t = self.lista.pop();

t.start();

def listenToClient(self, client, address):

size = 4096

while True:

try:

data = client.recv(size)

for a in data.split(b'\r\n'):

if(b"number=" in a):

a = a.decode("utf-8")

a = a.replace("number=","")

a = int(a)

if data:

# Set the response to echo back the recieved data

# response = data

# response += "\r\nbody: AEHOE"

response = b'HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<html><body>'+str(a\*2).encode()+b'</body></html>\r\n'

# response = b'<html><body><h1>OLA PROFESSOR</h1></body></html>\r\n'

client.send(response)

client.close()

else:

raise error('Client disconnected')

print("OK")

break;

except ValueError:

print(ValueError)

client.close()

raise

return False

self.consomeLista();

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

while True:

port\_num = input("Port? ")

try:

port\_num = int(port\_num)

break

except ValueError:

pass

ThreadedServer('',port\_num).listen()

**cliente**

import requests

import threading

def CALL():

r = requests.post("http://localhost:158", data={'number': 12524})

print(r.status\_code, r.reason)

print(r.text)

def CACALL():

for \_ in range(100):

threading.Thread(target = CALL,args = ()).start()

for \_ in range(100):

threading.Thread(target = CACALL,args = ()).start()