ESTI – ESCOLA SUPERIOR DA TECNOLOGIA DA INFORMAÇÃO
GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO
BLOCO DE DESENVOLVIMENTO ANDROID

Materia
Desenvolvimento Python para Redes e Sistemas Operacionais

LUCAS LANGE BARROZO
PROFESSOR:CASSIUS FIGUEIREDO

RIO DE JANEIRO, 04 de abril de 2019

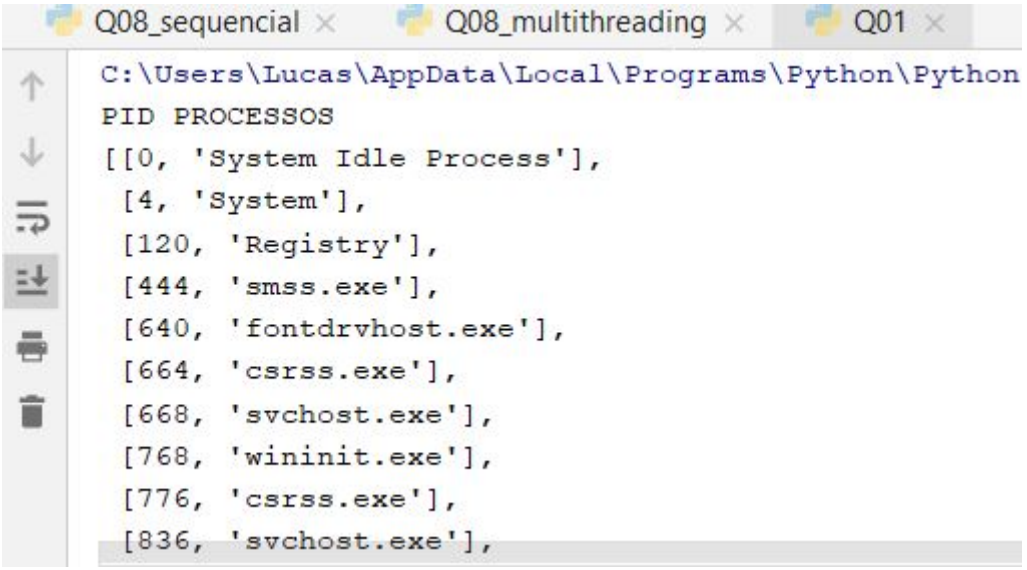
1)Escreva um programa em Python que:

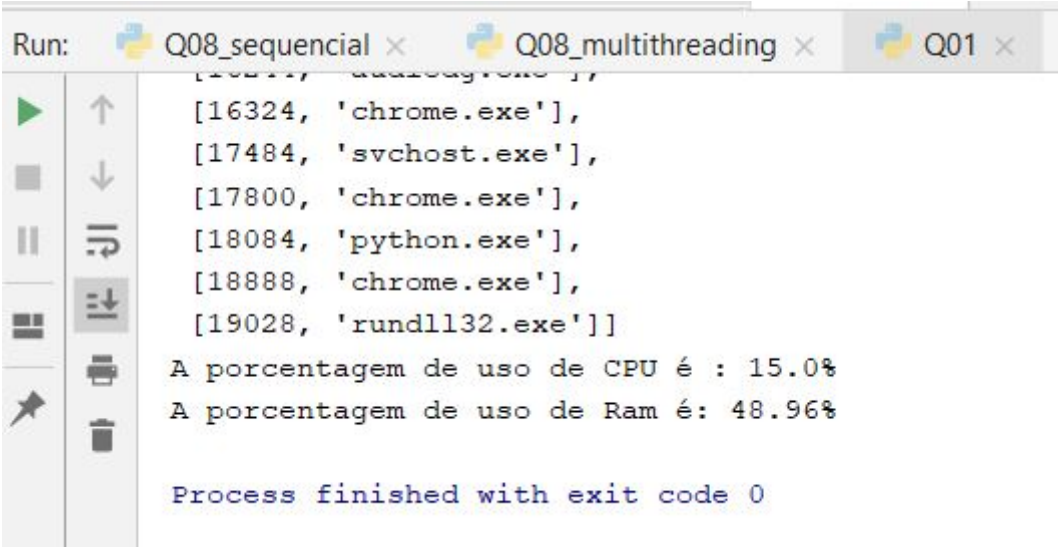
a)obtenha a lista de processos executando no momento, considerando que o processo pode deixar de existir enquanto seu programa manipula suas informações;

b)imprima o nome do processo e seu PID;

c)imprima também o percentual de uso de CPU e de uso de memória.

```
import psutil , pprint

lista_process =[]

try:

   for proc in psutil.process_iter():

       lista_process.append([proc.pid, proc.name()])

except Exception as e:

   print(f'Ao tentar capturar algum processo, o seguinte erro ocorreu: {e}')

print('LISTA PROCESSOS')

pprint.pprint(lista_process)

mem_virtual = psutil.virtual_memory()

mem_percent = mem_virtual.used / mem_virtual.total

print(f'A porcentagem de uso de CPU é : {psutil.cpu_percent()}%')

print(f'A porcentagem de uso de Ram é: {round(mem_percent*100,2)}% ')
```

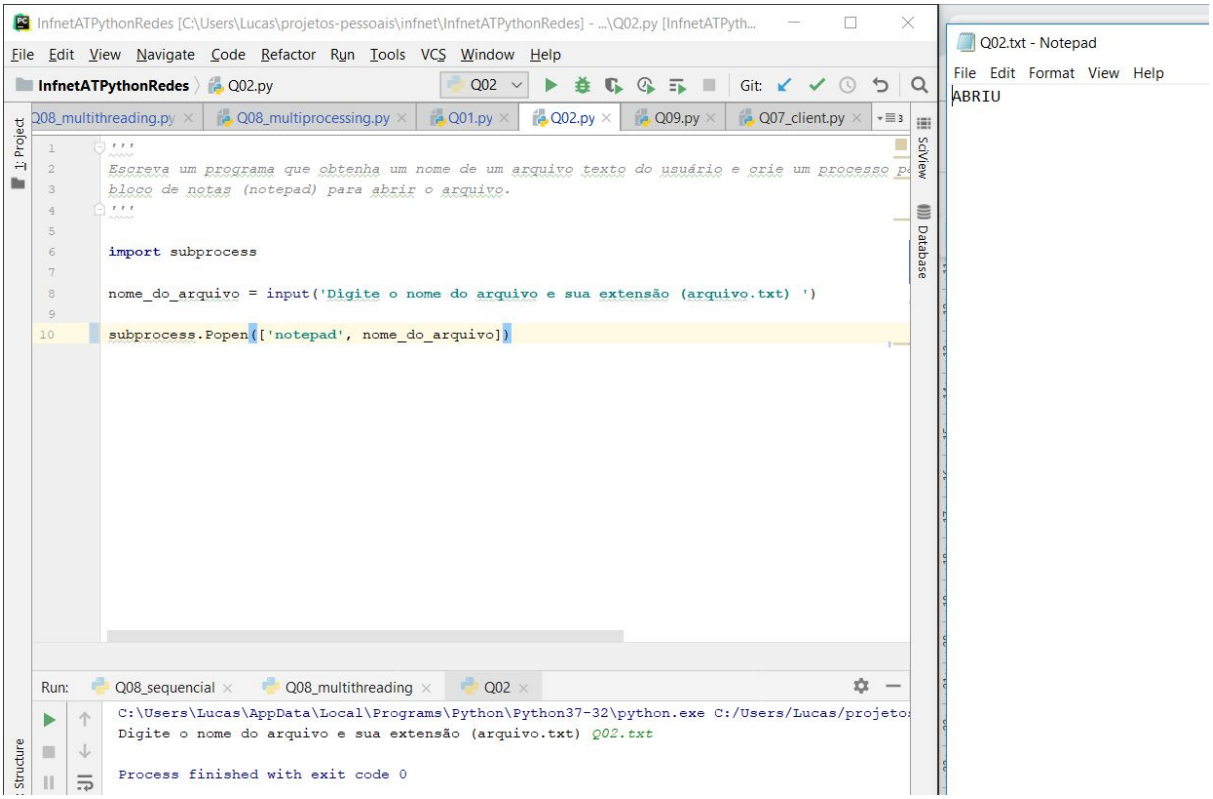




A Biblioteca pprint foi utilizada para melhor visualização

2)Escreva um programa que obtenha um nome de um arquivo texto do usuário e crie um processo para executar o programa do sistema Windows bloco de notas (notepad) para abrir o arquivo.

```
import subprocess

nome_do_arquivo = input('Digite o nome do arquivo e sua extensão (arquivo.txt) ')

subprocess.Popen(['notepad', nome_do_arquivo])
```



3)Escreva um programa em Python que:

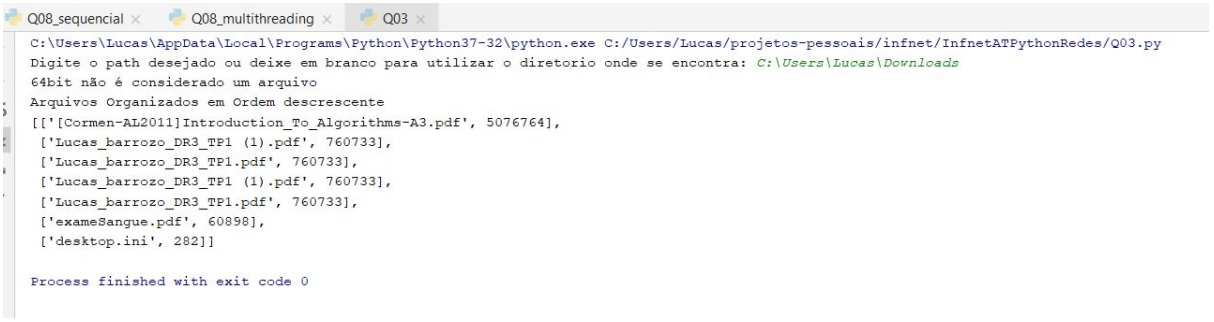
a)gere uma estrutura que armazena o nome dos arquivos em um determinado diretório e a quantidade de bytes que eles ocupam em disco. Obtenha o nome do diretório do usuário.

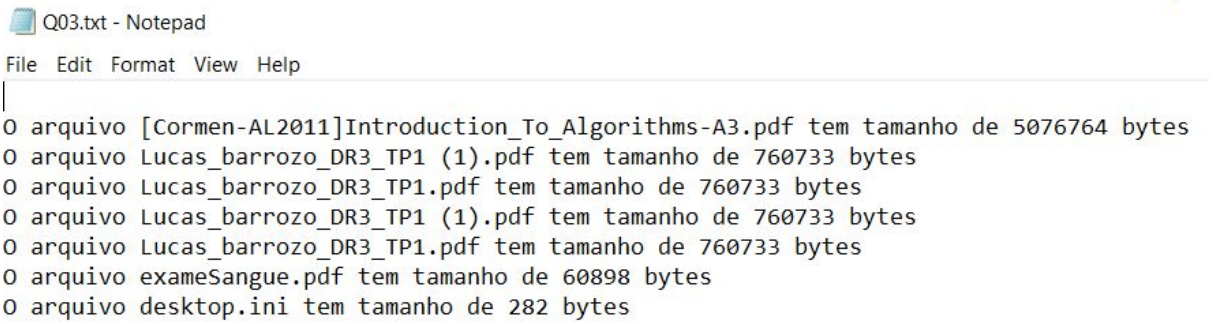
b)Ordene decrescentemente esta estrutura pelo valor da quantidade de bytes ocupada em disco (pode usar as funções sort ou sorted);

c)gere um arquivo texto com os valores desta estrutura ordenados.

```
import os
import pprint


def arquivos_diretorio(diretorio):
   arquivos = []
   total_ocupado = []
   if diretorio == '':
       for arquivo in os.listdir(os.getcwd()):
           try:
               if os.path.isfile(arquivo):
                   arquivos.append(arquivo)
                   total_ocupado.append(os.stat(arquivo).st_size)
               else:
                   print(arquivo, 'não é considerado um arquivo')
           except Exception as e:
               print(e)
   else:
       for arquivo in os.listdir(diretorio):
           try:
               if os.path.isfile(str(diretorio+'\\'+arquivo)):
                   arquivos.append(arquivo)
                   total_ocupado.append(os.stat(diretorio+'\\'+arquivo).st_size)
               else:
                   print(arquivo, 'não é considerado um arquivo')
```

```
            except Exception as e:
                print(e)
    dic_organizado = dict(zip(arquivos, total_ocupado))
    pprint.pprint(dic_organizado)
    total_ocupado_sorted = sorted(total_ocupado, reverse=True)
    pares_finais = []
    for tamanho in total_ocupado_sorted:
        for arquivo_dic, size in dic_organizado.items():
            if size == tamanho:
                pares_finais.append([arquivo_dic, size])
    print(pares_finais)
    texto = open('Q03.txt','w')
    for pares in pares_finais:
        texto.writelines(f'\nO arquivo {pares[0]} tem tamanho de {pares[1]} bytes')
    texto.close()
dir = input('Digite o path desejado ou deixe em branco para utilizar o diretorio onde se encontra:
')
print(dir)
if os.path.exists(dir) or dir == '':
    arquivos_diretorio(dir)
else:
    print('Diretorio não encontrado, por favor digite o caminho completo (Ex:
C:\\Users\\Lucas\\Downloads )')
```





4)Escreva um programa em Python que leia um arquivo texto e apresente na tela o seu conteúdo reverso. Exemplo:

arquivo.txt

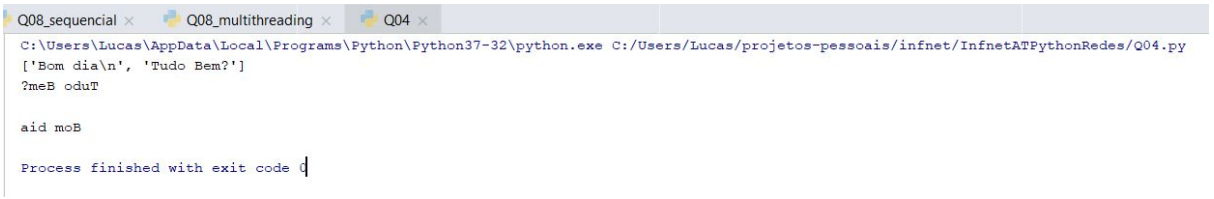
> **Bom dia**
>
> **Você pode falar agora?**

Resultado na tela:

?aroga ralaf edop êcoV

aid moB

```python
import os

if os.path.isfile('Q04.txt'):

    texto = open('Q04.txt', 'r')



    linhas = texto.readlines()

    print(linhas)

    for n in range(len(linhas)-1,-1,-1):



        if '\\n' in linhas[n]:

            linhas[n].replace('\\n', '')



        print(linhas[n][::-1])



else:

    print('Crie um arquivo chamado "Q04.txt" com algum texto dentro')
```



5)Escreva um programa em Python que leia dois arquivos, a.txt e b.txt, como a seguir:

a.txt		b.txt

1 15 -42 33 -7 -2 39 8		19 56 -43 23 -7 -11 33 21 61 9

Seu programa deve somar elemento por elemento de cada arquivo e imprimir o resultado na tela. Isto é, o primeiro elemento de a.txt deve ser somado ao primeiro elemento de b.txt, segundo elemento de a.txt deve ser somado ao segundo elemento de b.txt, e assim sucessivamente. Caso um arquivo tenha mais elementos que o outro, os elementos que sobrarem do maior devem ser somados a zero.

```
import os

if (os.path.isfile('a.txt')) and (os.path.isfile('b.txt')):

   def soma_arquivos(arquivo_a, arquivo_b):



       a = open(arquivo_a, 'r')

       b = open(arquivo_b, 'r')



       lista_a = list(map(int,a.read().split(' ')))

       print('lista a',lista_a)

       lista_b = list(map(int,b.read().split(' ')))

       print('lista b',lista_b)

       resultado =[]

       if len(lista_a) == len(lista_b):

           for number in range(0, len(lista_a)):

               resultado.append(lista_a[number] + lista_b[number])
```

```
        elif len(lista_a) > len(lista_b):
            for number in range(0, len(lista_b)):
                resultado.append(lista_a[number] + lista_b[number])

            for number in range(len(lista_b), len(lista_a)):
                resultado.append(lista_a[number]+0)

        elif len(lista_a) < len(lista_b):
            for number in range(0, len(lista_a)):
                resultado.append(lista_a[number] + lista_b[number])

            for number in range(len(lista_a) , len(lista_b)):
                resultado.append(lista_b[number] + 0)
        return print(f' A soma de elemento por elemento dos arquivos é {resultado}')
    a = 'a.txt'
    b = 'b.txt'
    soma_arquivos(a,b)
else:
    print('Verifique se os arquivos "a.txt" e "b.txt" existem e contem números')
```

```
C:\Users\Lucas\AppData\Local\Programs\Python\Python37-32\python.exe C:/Users/Lucas/projetos-pessoais/infnet/InfnetATPythonRedes/Q05.py
lista a [1, 15, -42, 33, -7, -2, 39, 8, 100, 13]
lista b [19, 56, -43, 23, -7, -11, 33, 21, 61, 9]
 A soma de elemento por elemento dos arquivos é [20, 71, -85, 56, -14, -13, 72, 29, 161, 22]

Process finished with exit code 0
```

6) Escreva um programa cliente e servidor sobre TCP em Python em que:

a)O cliente envia para o servidor o nome de um diretório e recebe a lista de arquivos (apenas arquivos) existente nele.

b)O servidor recebe a requisição do cliente, captura o nome dos arquivos no diretório em questão e envia a resposta ao cliente de volta.

Cliente:

```
import socket
import pickle
import os
# Cria o socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)


try:
   # Tenta se conectar ao servidor
   s.connect((socket.gethostname(), 9999))
   msg = os.getcwd()
   # Envia mensagem codificada em bytes ao servidor
```

```
    s.send(msg.encode('ascii'))



    msg2 = s.recv(10000)
    lista_arquivo= pickle.loads(msg2)


    print(f'Os arquivos encontrados em {msg} '
          f'\nsão : {lista_arquivo} ')
    # Fecha conexão com o servidor
    s.close()
except Exception as erro:
    print(str(erro))
```

Servidor:

```
import socket
import os
import pickle


# Cria o socket
socket_servidor = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# Obtém o nome da máquina
host = socket.gethostname()
porta = 9999
# Associa a porta
socket_servidor.bind((host, porta))
# Escutando...
```

```
socket_servidor.listen()

print("Servidor de nome", host, "esperando conexão na porta", porta)

while True:

 # Aceita alguma conexão

 (socket_cliente,addr) = socket_servidor.accept()

 print("Conectado a:", str(addr))

 path = socket_cliente.recv(1024)

 # Decodifica mensagem em ASCII

 print (path.decode('ascii'))

 arquivos =[]

 for arquivo in os.listdir(path):

   try:

     if os.path.isfile(arquivo):

       arquivos.append(arquivo)

     else:

       print(arquivo, 'não é considerado um arquivo')

   except Exception as e:

     print(e)

 lista_arquivo = pickle.dumps(arquivos)



 socket_cliente.send(lista_arquivo)

 socket_cliente.close()
```

Print Cliente:

```
C:\Users\Lucas\AppData\Local\Programs\Python\Python37-32\python.exe C:/Users/Lucas/projetos-pessoais/infnet/InfnetATPythonRedes/Q06_client.py
Os arquivos encontrados em C:\Users\Lucas\projetos-pessoais\infnet\InfnetATPythonRedes
são : [b'Q01.py', b'Q02.py', b'Q02.txt', b'Q03.py', b'Q03.txt', b'Q04.py', b'Q04.txt', b'Q05.py', b'Q05a.txt', b'Q05b.txt', b'Q06_client.py', b'Q06

Process finished with exit code 0
```

7)Escreva um programa cliente e servidor sobre UDP em Python que:

a)O cliente envia para o servidor o pedido de obtenção da quantidade total e disponível de memória no servidor e espera receber a resposta durante 5s. Caso passem os 5s, faça seu programa cliente tentar novamente mais 5 vezes (ainda esperando 5s a resposta) antes de desistir.

```
import socket, time, pickle

# Cria o socket

udp= socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

host = socket.gethostname()

port = 9991

dest = (host, port)

try:

   # Tenta se conectar ao servidor

   # udp.connect((socket.gethostname(), 9999))

   msg = input('Digite qualquer comando para receber informações de memória ou digite fim para
terminar ')
```

```
    if msg != 'fim':
        for i in range(5):
            # Envia mensagem vazia apenas para indicar a requisição
            udp.sendto(msg.encode('ascii'), dest)
            bytes = udp.recv(1024)
            # Converte os bytes para lista
            lista = pickle.loads(bytes)
            print(f'O total de memória é de: {lista[0]} bytes')
            print(f'A quantidade de memória usada é de: {lista[1]} bytes')
            time.sleep(5)
        msg = 'fim'
        udp.sendto(msg.encode('ascii'), dest)
    else:
        udp.sendto(msg.encode('ascii'), dest)
        udp.close()
except Exception as erro:
 print('OLHA O ERRO',str(erro))
# Fecha o socket
udp.close()
input("Pressione qualquer tecla para sair...")
```

```
C:\Users\Lucas\AppData\Local\Programs\Python\Python37-32\python.exe C:/Users/Lucas/projetos-pessoais/infnet/InfnetATPythonRedes/Q07_client.py
Digite qualquer comando para receber informações de memória ou digite fim para terminar as
O total de memória é de: 17057812480 bytes
A quantidade de memória usada é de: 6833303552 bytes
O total de memória é de: 17057812480 bytes
A quantidade de memória usada é de: 6832689152 bytes
O total de memória é de: 17057812480 bytes
A quantidade de memória usada é de: 6832578560 bytes
O total de memória é de: 17057812480 bytes
A quantidade de memória usada é de: 6833274880 bytes
O total de memória é de: 17057812480 bytes
A quantidade de memória usada é de: 6832332800 bytes
Pressione qualquer tecla para sair...

Process finished with exit code 0
```

b)O servidor repetidamente recebe a requisição do cliente, captura a informação da quantidade total e disponível de memória há no servidor e envia a resposta ao cliente de volta.

```
import socket

import psutil

import pickle



HOST = socket.gethostname() # Endereco IP do Servidor

PORT = 9991                 # Porta que o Servidor está esperando

udp = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

orig = (HOST, PORT)

udp.bind(orig)

print('Esperando receber na porta', PORT, '...')

while True:

   (msg, cliente) = udp.recvfrom(1024)

   if msg.decode('ascii') == 'fim':

       break

   print(f"Enviando dados de memória para {cliente}, {msg.decode('ascii')}")

   resposta = []

   mem = psutil.virtual_memory()

   mem_total = mem.total

   resposta.append(mem_total)

   mem_used = mem.used

   resposta.append(mem_used)

   bytes_resp = pickle.dumps(resposta)

   udp.sendto(bytes_resp, cliente)
```

```
udp.close()
```

```
C:\Users\Lucas\AppData\Local\Programs\Python\Python37-32\python.exe C:/Users/Lucas/projetos-pessoais/infnet/InfnetATPythonRedes/Q07_server.py
Esperando receber na porta 9991 ...
Enviando dados de memória para ('169.254.20.217', 50666), as
Enviando dados de memória para ('169.254.20.217', 50666), as
Enviando dados de memória para ('169.254.20.217', 50666), as
Enviando dados de memória para ('169.254.20.217', 50666), as
Enviando dados de memória para ('169.254.20.217', 50666), as

Process finished with exit code 0
```

8)Escreva 3 programas em Python que resolva o seguinte problema:

Dado um vetor A de tamanho N com apenas números inteiros positivos, calcule o fatorial de cada um deles e armazene o resultado em um vetor B.

Para calcular o fatorial, utilize a seguinte função:

```
def fatorial(n):
fat = n
for i in range(n-1,1,-1):
  fat = fat * i
return(fat)
```

Os modos de desenvolver seu programa devem ser:

a)sequencialmente (sem concorrência);

```
import random

import time
```

```python
def fatorial(n):
    fat = n
    for i in range(n-1,1,-1):
        fat = fat * i
    return(fat)
def fatorial_lista(funcao, vetor_a, vetor_b):
    for n in vetor_a:
        vetor_b.append(funcao(n))


a =[]
n = 5000
b = []
for i in range(0, n ):
    a.append(random.randint(2,10))


print('Vetor a',len(a))
tempo_inicial = time.time()


fatorial_lista(fatorial,a,b)
print('Vetor b',len(b))
tempo_final = time.time()
print(f'O tempo total foi de sequencial: {tempo_final - tempo_inicial}')
```

```
Vetor a 5000
Vetor b 5000
O tempo total foi de sequencial: 0.004976511001586914

Process finished with exit code 0
```

b)usando o módulo threading com 4 threads;

```
import random

import time

import threading


def fatorial(n):

   fat = n

   for i in range(n-1,1,-1):

       fat = fat * i

   return(fat)


def fatorial_lista(funcao, vetor_a, vetor_b):

   for n in vetor_a:

       vetor_b.append(funcao(n))


a =[]

n = 5000

b = []

for i in range(0, n ):

   a.append(random.randint(2,10))


tamanho = len(a)

print('Vetor a',tamanho)

tempo_inicial = time.time()


t0 = threading.Thread(target=fatorial_lista, args=(fatorial,a[0:int(tamanho/4)],b))
```

```
t0.start()

t1 = threading.Thread(target=fatorial_lista, args=(fatorial,a[int(tamanho/4): int(tamanho/2)],b))

t1.start()

t2 = threading.Thread(target=fatorial_lista, args=(fatorial,a[int(tamanho/2) :
int(tamanho*(3/4))],b))

t2.start()

t3 = threading.Thread(target=fatorial_lista, args=(fatorial,a[int(tamanho*(3/4)) :
int(tamanho)],b))

t3.start()



t0.join()

t1.join()

t2.join()

t3.join()



print('vetor b', len(b))

tempo_final = time.time()

print(f'O tempo total de multithreading foi de: {tempo_final - tempo_inicial}')
```

```
Vetor a 5000
vetor b 5000
O tempo total de multithreading foi de: 0.005985736846923828

Process finished with exit code 0
```

c)usando o módulo multiprocessing com 4 processos.

```
import random

import time

import multiprocessing
```

```
def fatorial(n):
    fat = n
    for i in range(n-1,1,-1):
        fat = fat * i
    return(fat)


def fatorial_lista(funcao, vetor_a,q):


    vetor_b =[]
    for n in vetor_a:
        vetor_b.append(funcao(n))


    q.put(vetor_b)


    # print(queue.get())
    # print(len(queue.get()))


if __name__ == "__main__":


    a =[]
    n = 10000000
    b = []
    q = multiprocessing.Queue()
    for i in range(0, n ):
        a.append(random.randint(2,10))


    tamanho = len(a)
    print('Vetor a',tamanho)
```

```
    tempo_inicial = time.time()



    p0 = multiprocessing.Process(target=fatorial_lista, args=(fatorial,a[0:int(tamanho/4)],q))

    p0.start()

    p1 = multiprocessing.Process(target=fatorial_lista, args=(fatorial,a[int(tamanho/4):
int(tamanho/2)],q))

    p1.start()

    p2 = multiprocessing.Process(target=fatorial_lista, args=(fatorial,a[int(tamanho/2) :
int(tamanho*(3/4))],q))

    p2.start()

    p3 = multiprocessing.Process(target=fatorial_lista, args=(fatorial,a[int(tamanho*(3/4)) :
int(tamanho)],q))

    p3.start()



    while len(a) != len(b):

        while q.empty() is False:

            b += q.get()



    p0.join()

    p1.join()

    p2.join()

    p3.join()

    print('vetor b', len(b))

    tempo_final = time.time()

    print(f'O tempo total de multiprocessing foi de: {tempo_final - tempo_inicial}')
```

```
Vetor a 5000
vetor b 5000
O tempo total de multiprocessing foi de: 0.3394744396209717

Process finished with exit code 0
```

9)Teste todos os 3 programas da questão 8, capture os tempos de execução deles e compare-os, explicando os resultados de tempos. Varie o valor de N em 1.000.000, 5000.000, 10.000.000 (ou escolha números maiores ou melhores de acordo com a velocidade de processamento do computador utilizado para testes).

Obs.: Para testar, crie um vetor com apenas um número relativamente grande (10, por exemplo) ou use a função random para gerar um vetor com números aleatórios. Cuidado ao usar números muito grandes, pois o fatorial pode resultar em um valor que o computador não consiga representar por falta de precisão.

R: Todas as listas são geradas com função aleatória de 2 - 10. A contagem do tempo se dá apenas depois do vetor A ser gerado, isto é , apenas o calculo do vetor b (fatorial de cada elemento do vetor a) tem o tempo contado.

N = 1 000 000

Sequencial:

```
Vetor a 1000000
Vetor b 1000000
O tempo total de sequencial foi de: 0.7639439105987549

Process finished with exit code 0
```

Multithreading:

```
Vetor a 1000000
vetor b 1000000
O tempo total de multithreading foi de: 0.7848987579345703

Process finished with exit code 0
```

Multiprocessing:

```
Vetor a 1000000
vetor b 1000000
O tempo total de multiprocessing foi de: 0.6512179374694824

Process finished with exit code 0
```

N = 5 000 000

Sequencial:

```
Vetor a 5000000
Vetor b 5000000
O tempo total de sequencial foi de: 3.694183349609375

Process finished with exit code 0
```

Multithreading:

```
Vetor a 5000000
vetor b 5000000
O tempo total de multithreading foi de: 3.918818712234497

Process finished with exit code 0
```

Multiprocessing:

```
Vetor a 5000000
vetor b 5000000
O tempo total de multiprocessing foi de: 2.0784385204315186

Process finished with exit code 0
```

N = 10 000 000

Sequencial:

```
Vetor a 10000000
Vetor b 10000000
O tempo total de sequencial foi de: 7.693428993225098

Process finished with exit code 0
```

Multithreading:

```
Vetor a 10000000
vetor b 10000000
O tempo total de multithreading foi de: 7.855942964553833

Process finished with exit code 0
```

Multiprocessing:

```
Vetor a 10000000
vetor b 10000000
O tempo total de multiprocessing foi de: 3.968414068222046

Process finished with exit code 0
```

Se comparado com os resultados da questão 8, isto é vetor a com 5000 elementos, podemos ver uma mudança de muito grande de resultados. Com um vetor de 5000 elementos o multiprocessing foi o mais lento, sendo 100 vez mais lento que o multithreading e o sequencial. O sequencial foi marginalmente mais rápido do que o multithreading.

Entretanto, ao realizar a partir de 1 milhão, podemos notar que o multiprocessing passou a ser mais rápido, indicando que a paralelização por processos diferentes fica mais eficiente com listas maiores. Multithreading foi o mais lento que o sequencial ficando atrás do sequencial e isso pode apontar um custo de sistema operacional na operação de multithread.

Uma explicação, encontrada no stackoverflow, pode ser um problema do GIL(Global Interpreter Lock) encontrado no cpython. O GIL é necessário por conta da gerência de memória do Cpython. O GIL não permite que diversas threads executem Python bytecodes de uma vez só.

Realizar cálculos em múltiplas threads pode cair a performance, possivelmente por esse problema de impossibilidade de diversos códigos serem realizados simultaneamente, adicionando um tempo extra do sistema operacional decidir qual é o próximo código que deve ser enviado ao processador.

Bibliografia:

https://stackoverflow.com/questions/10789042/python-multi-threading-slower-than-serial , acessado dia 03/04/2019