



ESTI - Escola Superior da Tecnologia da Informação
EDC - Graduação em Engenharia de Computação
Desenvolvimento Python para Redes e Sistemas Operacionais
Assessment

Aluno: Eloy Francisco Barbosa
Professor: Cassius Figueiredo
Data: 11/02/2019

Sumário

1..... 3

2..... 4

3..... 5

4..... 7

5..... 8

6..... 9

7..... 13

8..... 16

9..... 21

1.

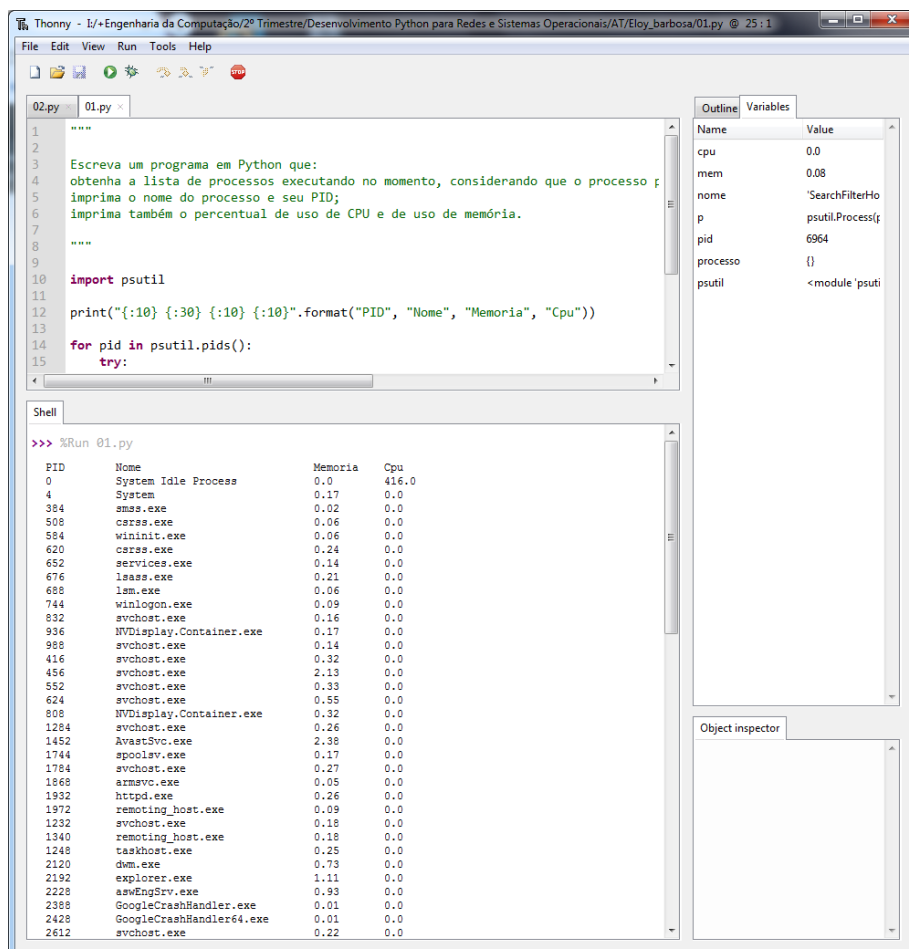
Escreva um programa em Python que:

- obtenha a lista de processos executando no momento, considerando que o processo pode deixar de existir enquanto seu programa manipula suas informações;
- imprima o nome do processo e seu PID;
- imprima também o percentual de uso de CPU e de uso de memória.

```
import psutil

print("{:10} {:30} {:10} {:10}".format("PID", "Nome", "Memoria",
"Cpu"))

for pid in psutil.pids():
    try:
        processo = {}
        p = psutil.Process(pid)
        pid = pid
        nome = p.name()
        mem = round(p.memory_percent(), 2)
        cpu = p.cpu_percent(interval=0.01)
        print("{:<10} {:<30} {:<10} {:<10}".format(pid, nome,
mem, cpu))
    except:
        pass
```



The screenshot shows a Python IDE window titled 'Thonny - I:\Engenharia da Computação\2º Trimestre\Desenvolvimento Python para Redes e Sistemas Operacionais\AT\Elcy_barbosa\01.py @ 25:1'. The editor displays a Python script that imports the 'psutil' module and iterates over all running processes. For each process, it prints the PID, name, memory usage percentage, and CPU usage percentage. The script includes error handling to catch any exceptions that might occur if a process terminates during execution.

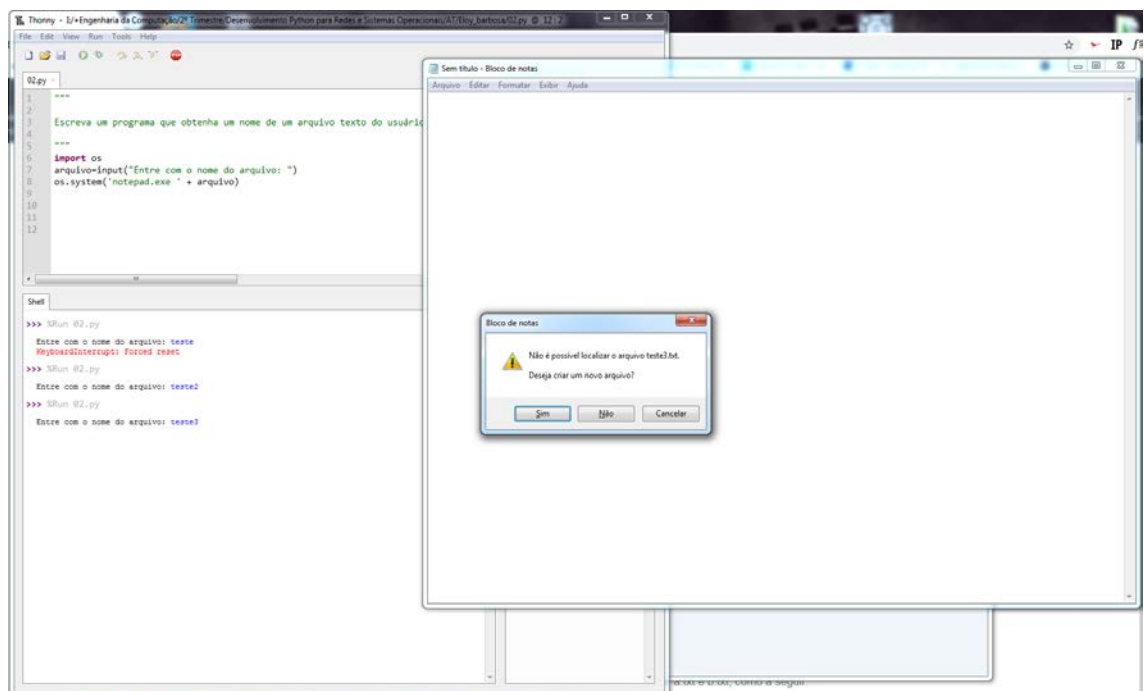
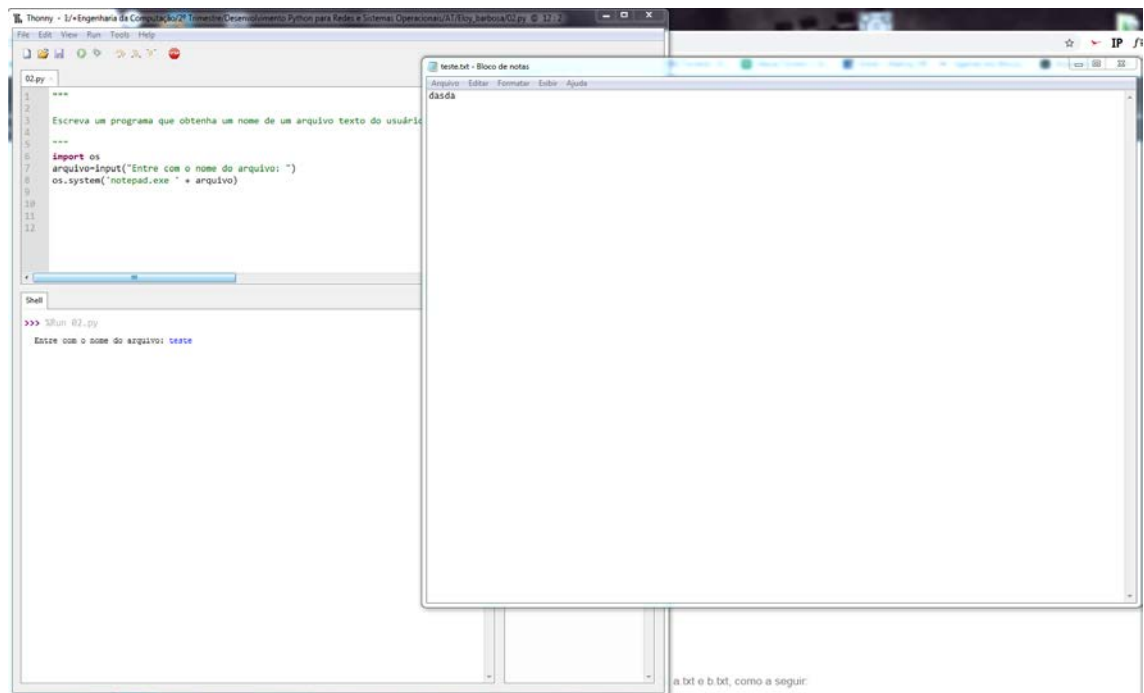
The output in the Shell window shows the following data:

PID	Nome	Memoria	Cpu
0	System Idle Process	0.0	416.0
4	System	0.17	0.0
384	smss.exe	0.02	0.0
508	csrss.exe	0.06	0.0
584	wininit.exe	0.06	0.0
620	csrss.exe	0.24	0.0
652	services.exe	0.14	0.0
676	lsass.exe	0.21	0.0
688	lsass.exe	0.06	0.0
744	winlogon.exe	0.09	0.0
832	svchost.exe	0.16	0.0
936	NVDisplay.Container.exe	0.17	0.0
968	svchost.exe	0.14	0.0
416	svchost.exe	0.32	0.0
456	svchost.exe	2.13	0.0
552	svchost.exe	0.33	0.0
624	svchost.exe	0.55	0.0
808	NVDisplay.Container.exe	0.32	0.0
1284	svchost.exe	0.26	0.0
1452	AvastSvc.exe	2.38	0.0
1744	spoolsv.exe	0.17	0.0
1784	svchost.exe	0.27	0.0
1868	atmsvc.exe	0.05	0.0
1932	httpd.exe	0.26	0.0
1972	remoting_host.exe	0.09	0.0
1232	svchost.exe	0.18	0.0
1340	remoting_host.exe	0.18	0.0
1248	taskhost.exe	0.25	0.0
2120	dwm.exe	0.73	0.0
2192	explorer.exe	1.11	0.0
2228	aswEngSvc.exe	0.93	0.0
2388	GoogleCrashHandler.exe	0.01	0.0
2428	GoogleCrashHandler64.exe	0.01	0.0
2612	svchost.exe	0.22	0.0

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 os
arquivo=input("Entre com o nome do arquivo: ")
os.system('notepad.exe ' + 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 time

arquivos = []

caminho = input('Informe o caminho de um diretório: ')
print()

try:
    for nome in (os.listdir(caminho)):
        arquivo = [ nome, int(os.stat(caminho + nome).st_size)]
        arquivos.append(arquivo)
    arquivos.sort(key=lambda o: o[1], reverse=True)
    print('A relação dos arquivos desse diretório em ordem
    decrescente pelo tamanho em bytes é a seguinte: \n\n')
    print('{:<30}'.format('Nome'), end='')
    print('{:<30}'.format('Tamanho em bytes'), end='\n\n')
    for x in range (len(arquivos)):
        print('{:<30}'.format(arquivos[x][0]), end='')
        print('{:<30}'.format(arquivos[x][1]), end='\n')

except (FileNotFoundError):
    print('O caminho informado não existe!')

now = ('Questão 03 - ' + time.strftime('%y%d%m%H%m%S') + ".txt")

txt = open(now, "w")
txt.write('{:<30}'.format('Nome'))
txt.write('{:<30}'.format('Tamanho em bytes'))
txt.write('\n')

for x in range (len(arquivos)):
    txt.write('{:<30}'.format(arquivos[x][0]))
    txt.write('{:<30}'.format(arquivos[x][1]))
    txt.write('\n')

txt.close()

print('\nFoi gerado o seguinte arquivo contendo essas
informações: {} '.format(now))
```

Thonny - I:/+Engenharia da Computação/2º Trimestre/Desenvolvimento Python para Redes e Sistemas Operacionais/AT/Eloy_barbosa/03.py @ 50:1

File Edit View Run Tools Help

03.py x

```

34 print( u caminho informado nao existe! )
35
36 now = ('Questão 03 - ' + time.strftime('%y%d%H%M%S') + ".txt")
37
38 txt = open(now, "w")
39 txt.write('{:<30}'.format('Nome'))
40 txt.write('{:<30}'.format('Tamanho em bytes'))
41 txt.write('\n')
42
43 for x in range (len(arquivos)):
44     txt.write('{:<30}'.format(arquivos[x][0]))
45     txt.write('{:<30}'.format(arquivos[x][1]))
46     txt.write('\n')
47
48 txt.close()

```

Outline Variables

Name	Value
arquivo	['xampp', 12288]
arquivos	[['pagefile.sys',
caminho	'c:'
nome	'xampp'
now	'Questão 03 - 1'
os	<module 'os' fr
time	<module 'time
txt	<_io.TextIOWra
x	24

Shell

Informe o caminho de um diretório: c:

A relação dos arquivos desse diretório em ordem decrescente pelo tamanho em bytes é a seguinte:

Nome	Tamanho em bytes
pagefile.sys	8589000704
hiberfil.sys	6441750528
sedre.log.1	512026
sedre.log	102561
License.rtf	29842
Windows	16384
Arquivos de Programas	12288
Program Files	12288
Program Files (x86)	12288
xampp	12288
ProgramData	8192
+Engenharia da Computação	4096
Documents and Settings	4096
Users	4096
\$AV_ASW	0
\$Recycle.Bin	0
Autodesk	0
Backup Moto E	0
e4d33fc90d50908364ac8188cc	0
LGMobileUpgrade	0
Media	0
MSOCache	0
PerfLogs	0
Recovery	0
System Volume Information	0

Foi gerado o seguinte arquivo contendo essas informações: Questão 03 - 190404050433.txt

>>>

Object inspector

4.

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

arquivo.txt

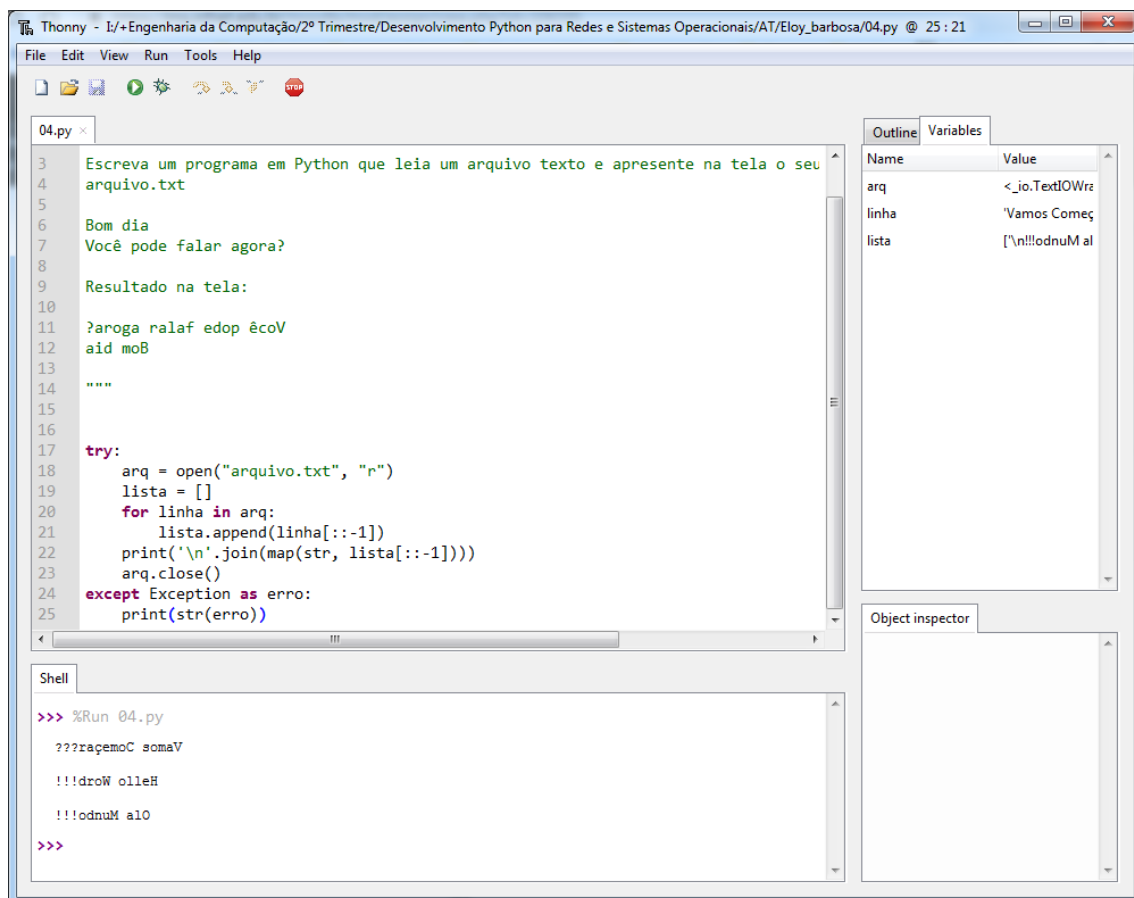
Bom dia
Você pode falar agora?

Resultado na tela:

?aroga ralaf edop êcoV

aid moB

```
try:
    arq = open("arquivo.txt", "r")
    lista = []
    for linha in arq:
        lista.append(linha[::-1])
    print('\n'.join(map(str, lista[::-1])))
    arq.close()
except Exception as erro:
    print(str(erro))
```



5.

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

a.txt

1 15 -42 33 -7 -2 39 8

b.txt

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 sobraem do maior devem ser somados a zero.

```
try:
    a = open("a.txt", "r")
    b = open("b.txt", "r")
    lista_a = []
    lista_b = []
    resultado = []
    for x in a:
        lista_a = x.split(" ")
    for y in b:
        lista_b = y.split(" ")

    print('a.txt : ', ' '.join(map(str, lista_a)))
    print('b.txt : ', ' '.join(map(str, lista_b)))

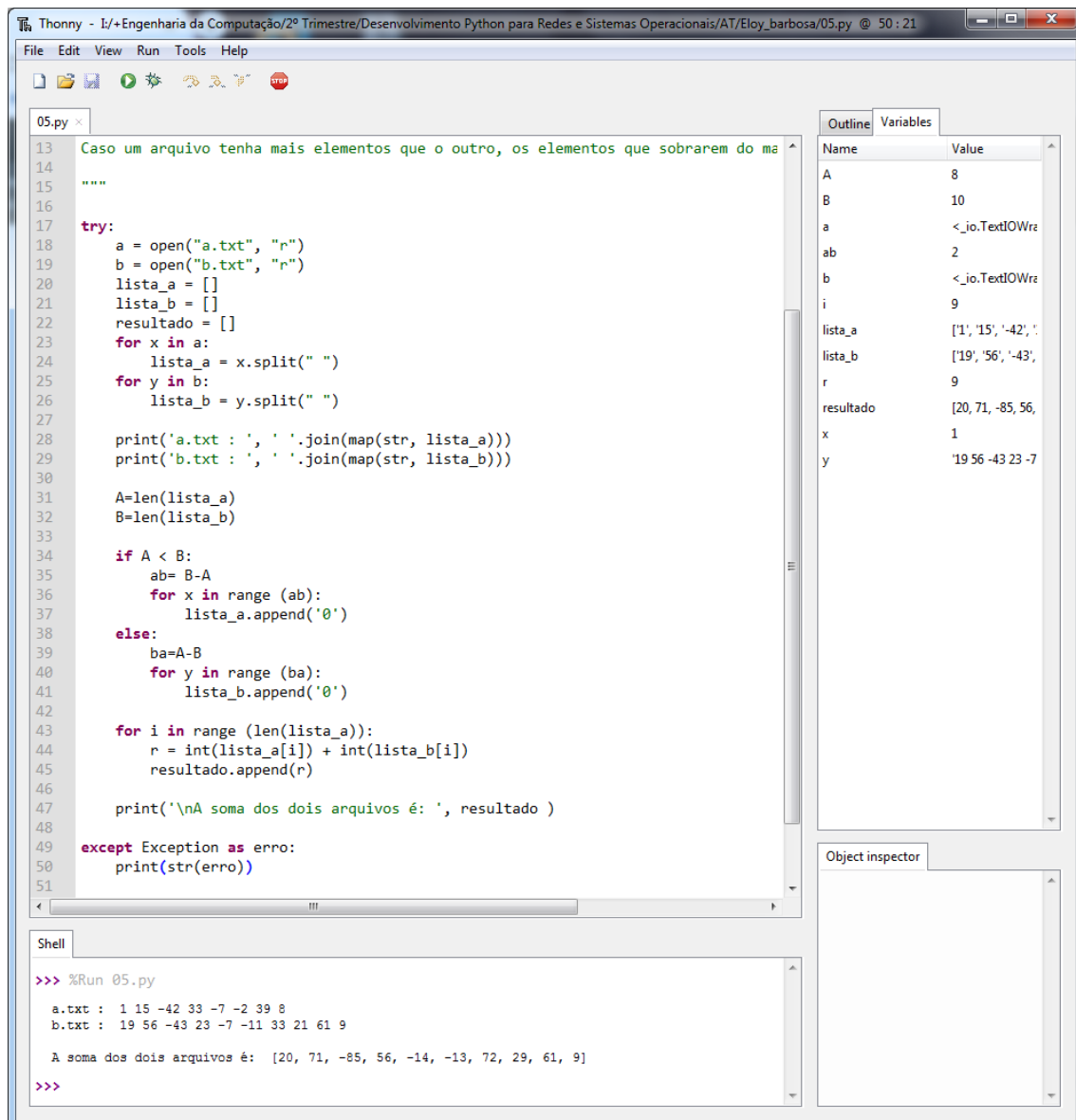
    A=len(lista_a)
    B=len(lista_b)

    if A < B:
        ab= B-A
        for x in range (ab):
            lista_a.append('0')
    else:
        ba=A-B
        for y in range (ba):
            lista_b.append('0')

    for i in range (len(lista_a)):
        r = int(lista_a[i]) + int(lista_b[i])
        resultado.append(r)

    print('\nA soma dos dois arquivos é: ', resultado )

except Exception as erro:
    print(str(erro))
```

6.

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

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

```
socket_cliente = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
arquivo = input('Informe o diretório: ')
print()
```

```

socket_cliente.connect((socket.gethostname(), 6666))
socket_cliente.send(arquivo.encode('utf-8'))

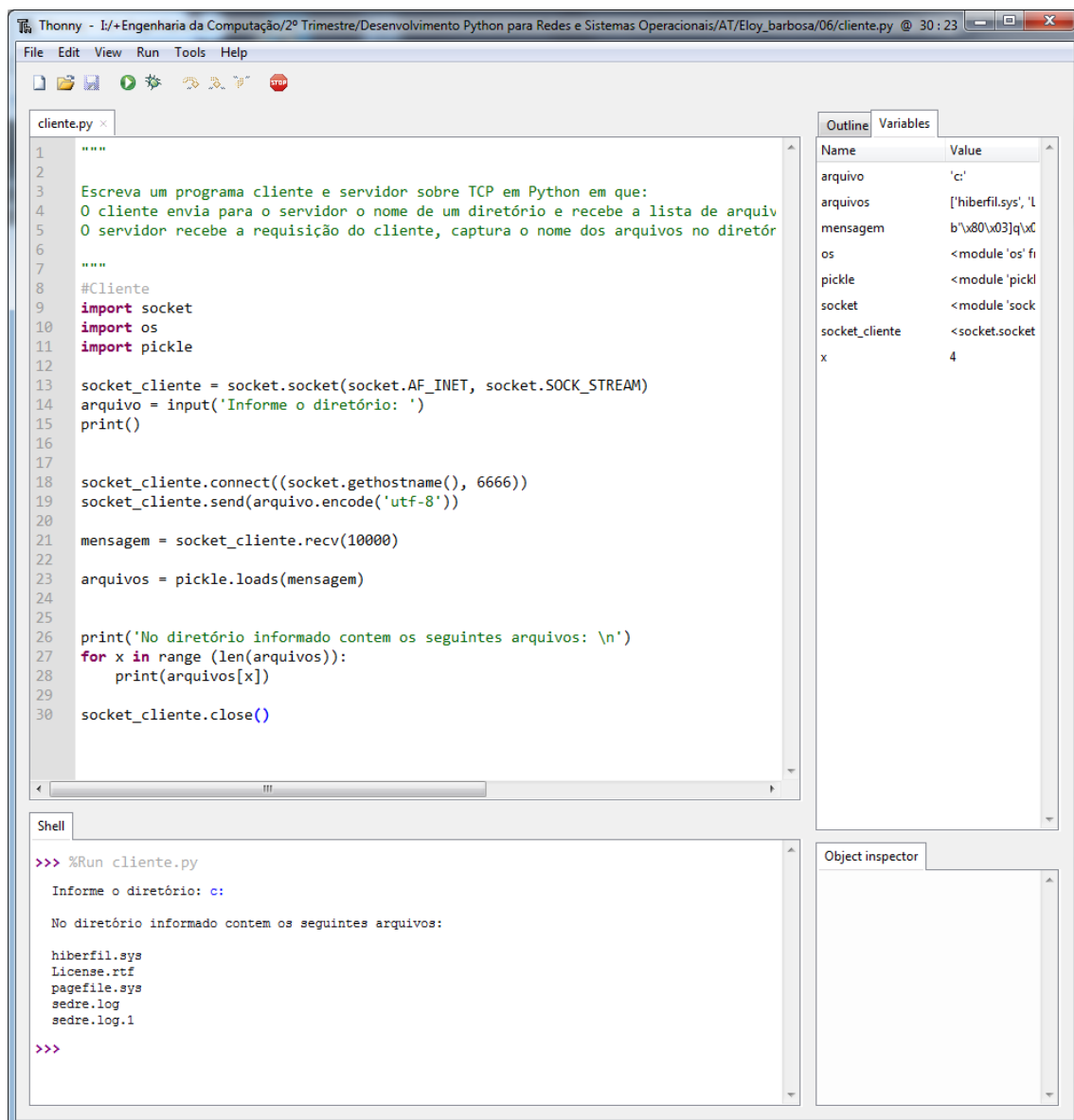
mensagem = socket_cliente.recv(10000)

arquivos = pickle.loads(mensagem)

print('No diretório informado contem os seguintes arquivos: \n')
for x in range (len(arquivos)):
    print(arquivos[x])

socket_cliente.close()

```



```

#Servidor
import socket
import os
import pickle

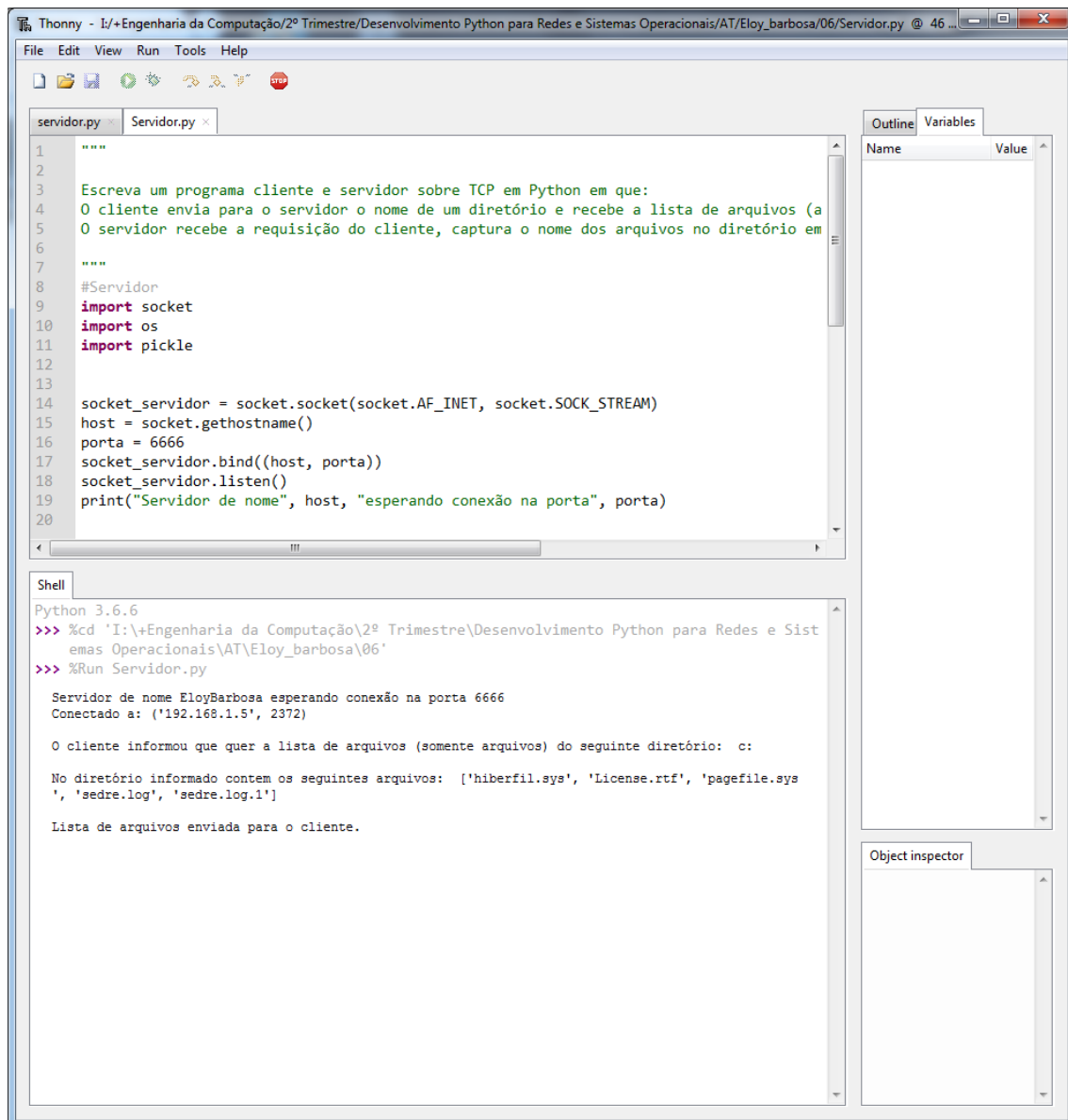
socket_servidor = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
host = socket.gethostname()
porta = 6666
socket_servidor.bind((host, porta))
socket_servidor.listen()
print("Servidor de nome", host, "esperando conexão na porta",
porta)

while True:
    (socket_cliente,addr) = socket_servidor.accept()
    print("Conectado a:", str(addr))
    print()
    arquivo = socket_cliente.recv(1024)
    print ('O cliente informou que quer a lista de arquivos
(somente arquivos) do seguinte diretório: ',
arquivo.decode('utf-8'))
    print()
    files=[]
    if os.path.exists(arquivo):
        arquivos = os.listdir(arquivo)
        for i in range (len(arquivos)):
            if os.path.isfile(arquivo + arquivos[i]) == True:
                files.append(arquivos[i].decode('utf-8'))
        print('No diretório informado contem os seguintes
arquivos: ', files)
        print()

        resposta = pickle.dumps(files)
        socket_cliente.send(resposta)
        print('Lista de arquivos enviada para o cliente.')
    else:
        pass

socket_cliente.close()

```



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.
- 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.

```
#Cliente
import socket
import time

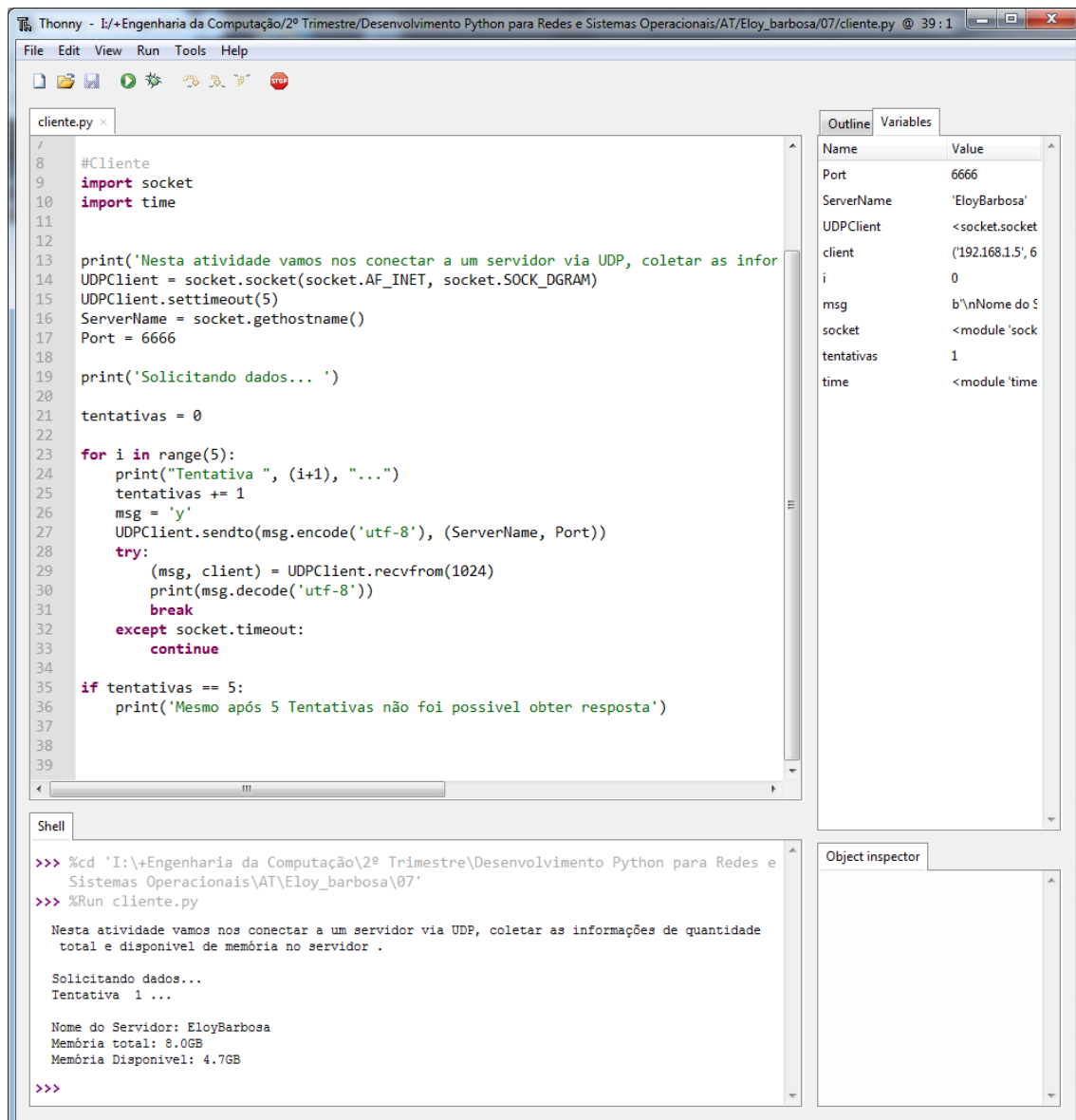
print('Nesta atividade vamos nos conectar a um servidor via UDP,
coletar as informações de quantidade total e disponível de
memória no servidor .\n')
UDPClient = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
UDPClient.settimeout(5)
ServerName = socket.gethostname()
Port = 6666

print('Solicitando dados... ')

tentativas = 0

for i in range(5):
    print("Tentativa ", (i+1), "...")
    tentativas += 1
    msg = 'y'
    UDPClient.sendto(msg.encode('utf-8'), (ServerName, Port))
    try:
        (msg, client) = UDPClient.recvfrom(1024)
        print(msg.decode('utf-8'))
        break
    except socket.timeout:
        continue

if tentativas == 5:
    print('Mesmo após 5 Tentativas não foi possível obter
resposta')
```



```

#Servidor
import socket
import psutil

```

```

UDPServer = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
ServerName = socket.gethostname()
Port = 6666
UDPServer.bind((ServerName, Port))

```

```

memoria_total = round(psutil.virtual_memory().total / (1024 *
1024 * 1024), 1)
memoria_disponivel = round(psutil.virtual_memory().free / (1024
* 1024 * 1024), 1)

```

```

try:
    print('Aguardando requisição na porta', Port, '...')
    (msg, client) = UDPServer.recvfrom(1024)

```

```

        print('Requisição aceita')
        if msg.decode() == 'y':
            UDPServer.sendto('\nNome do Servidor: {} \nMemória
total: {}GB \nMemória Disponível: {}GB'.format(ServerName,
memoria_total, memoria_disponivel).encode('utf-8'), client)
            print('Informações enviadas com sucesso')

        else:
            UDPServer.sendto('Erro: Digite "y".'.encode('utf-8'),
client)
            print('Informações não foram enviadas.')

except Exception as error:
    print(error)

```

Thonny - I:\Engenharia da Computação\2º Trimestre\Desenvolvimento Python para Redes e Sistemas Operacionais\AT\Eloy_barbosa\07\servidor.py @ 31 ...

File Edit View Run Tools Help

servidor.py x

```

6
7
8 """
9 #Servidor
10 import socket
11 import psutil
12
13 UDPServer = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
14 ServerName = socket.gethostname()
15 Port = 6666
16 UDPServer.bind((ServerName, Port))
17
18 memoria_total = round(psutil.virtual_memory().total / (1024 * 1024 * 1024), 1)
19 memoria_disponivel = round(psutil.virtual_memory().free / (1024 * 1024 * 1024), 1)
20
21
22
23 try:
24     print('Aguardando requisição na porta', Port, '...')
25     (msg, client) = UDPServer.recvfrom(1024)
26     print('Requisição aceita')
27     if msg.decode() == 'y':
28         UDPServer.sendto('\nNome do Servidor: {} \nMemória total: {}GB \nMemória Dispo
29         print('Informações enviadas com sucesso')
30
31     else:
32         UDPServer.sendto('Erro: Digite "y".'.encode('utf-8'), client)
33         print('Informações não foram enviadas.')
34
35 except Exception as error:
36     print(error)
37
38

```

Outline Variables

Name	Value
Port	6666
ServerName	'EloyB'
UDPServer	<sock
client	('192.1
memoria_disponivel	4.7
memoria_total	8.0
msg	b'y'
psutil	<moc
socket	<moc

Object inspector

Shell

```

>>> %cd 'I:\Engenharia da Computação\2º Trimestre\Desenvolvimento Python para Redes e Sist
emas Operacionais\AT\Eloy_barbosa\07'
>>> %Run servidor.py

Aguardando requisição na porta 6666 ...
Requisição aceita
Informações enviadas com sucesso

>>>

```

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);
- b. usando o módulo threading com 4 threads;
- c. usando o módulo multiprocessing com 4 processos.

a.

```
import time  
import random
```

```
resultado = []  
def fatorial(n):  
    fat = n  
    for i in range(n-1,1,-1):  
        fat = fat * i  
    #return fat  
    resultado.append(fat)  
  
t_inicio = float(time.time())  
  
listaFatorial = []  
  
def criaLista(n):  
    for i in range(n):  
        k = random.randint(1,10)  
        listaFatorial.append(k)  
  
criaLista(1000000)
```

```
for i in listaFatorial:  
    fatorial(i)
```

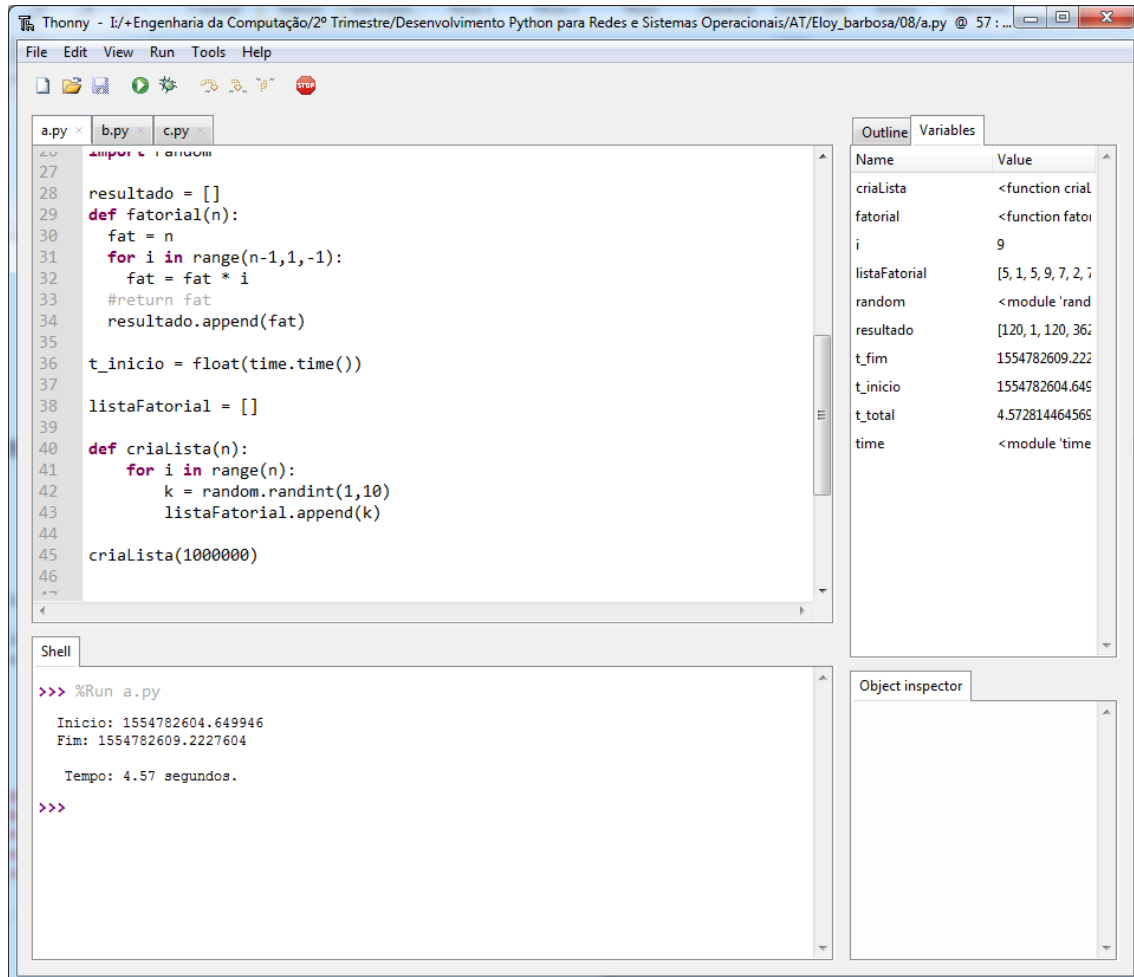
```
t_fim = float(time.time())
```



```

print('Inicio:',t_inicio)
print('Fim:',t_fim)
t_total = t_fim - t_inicio
print('\n', 'Tempo:',round(t_total,2), 'segundos.')

```



b.

```

import time
import random
import threading

```

```

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

```

```

def listaFatorial(lista, inicio, fim):
    for i in range(inicio, fim):
        bLista.append(fatorial(lista[i]))

```

```

t_inicio = float(time.time())
N = 1000000

```

```

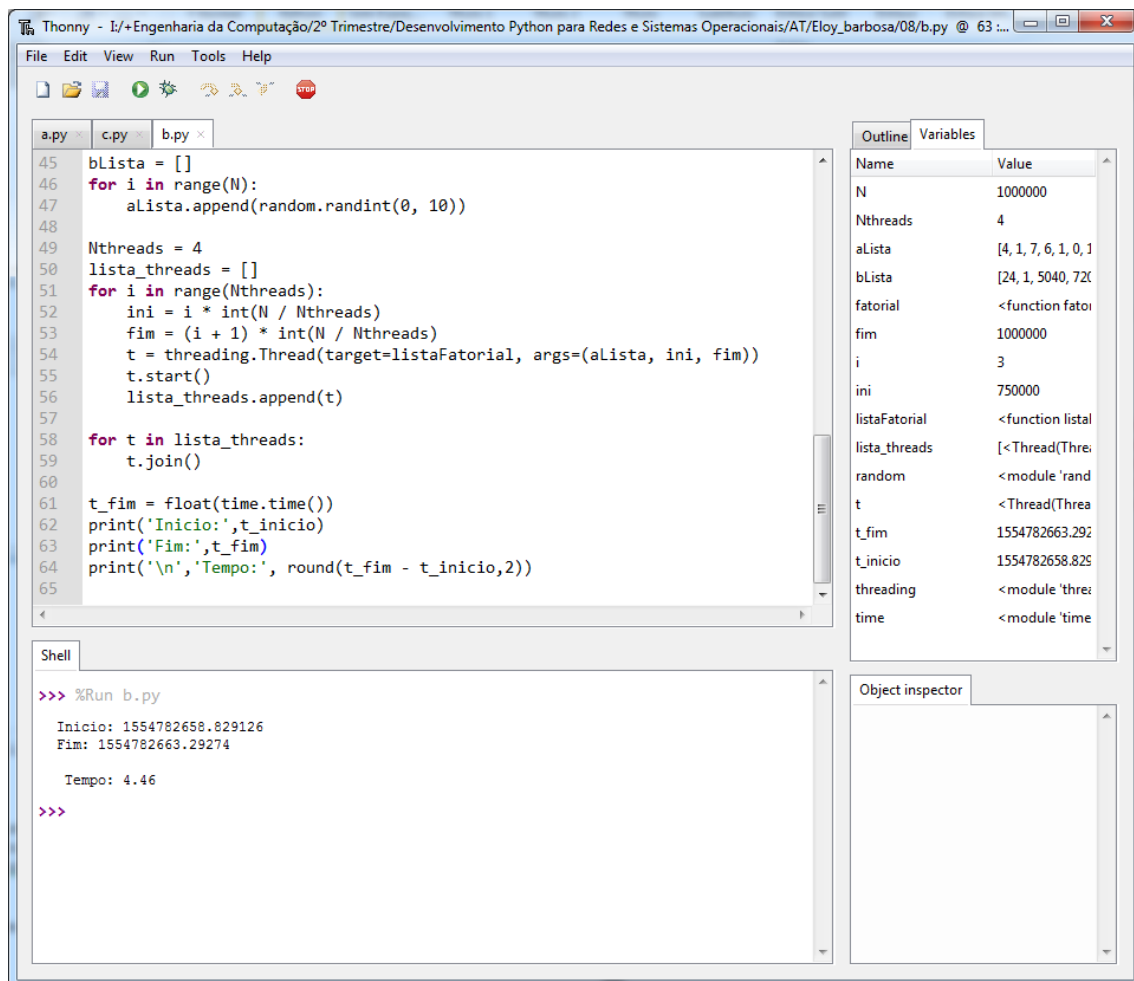
aLista = []
bLista = []
for i in range(N):
    aLista.append(random.randint(0, 10))

Nthreads = 4
lista_threads = []
for i in range(Nthreads):
    ini = i * int(N / Nthreads)
    fim = (i + 1) * int(N / Nthreads)
    t = threading.Thread(target=listaFatorial, args=(aLista,
ini, fim))
    t.start()
    lista_threads.append(t)

for t in lista_threads:
    t.join()

t_fim = float(time.time())
print('Inicio:', t_inicio)
print('Fim:', t_fim)
print('\n', 'Tempo:', round(t_fim - t_inicio, 2))

```



```

c.
import multiprocessing
import random
import time

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

def Main():
    N = 1000000

    t_inicio = float(time.time())

    aLista = []
    bLista = []

    for i in range(N):
        aLista.append(random.randint(0, 10))

    NProc = 4

    q_entrada = multiprocessing.Queue()
    q_saida = multiprocessing.Queue()

    lista_proc = []
    for i in range(NProc):
        ini = i * int(N / NProc)
        fim = (i + 1) * int(N / NProc)
        q_entrada.put(aLista[ini:fim])
        p = multiprocessing.Process(target=ListFat,
args=(q_entrada, q_saida))
        p.start()
        lista_proc.append(p)

    for i in range(0, NProc):
        bLista = q_saida.get(timeout=10)

    for p in lista_proc:
        p.join()

    t_fim = float(time.time())

    print('Inicio:',t_inicio)
    print('Fim:',t_fim)
    print('\n', 'Tempo :', round(t_fim - t_inicio, 2))
    input('Digite enter para sair...')

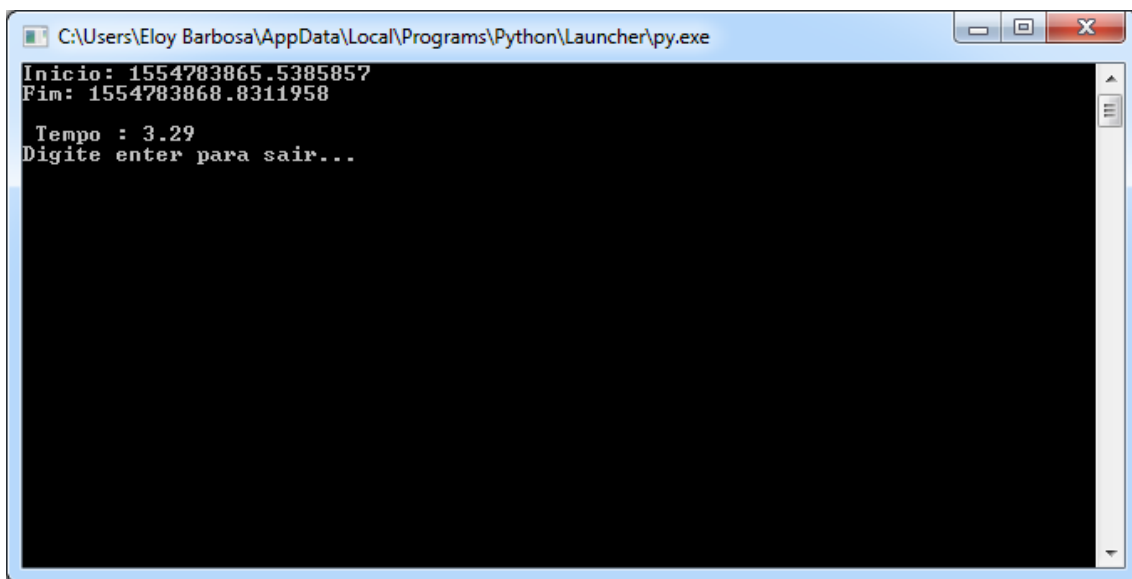
def fatorial(n):

```

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

def ListFat(q1, q2):
    l1 = q1.get()
    l2 = []
    for i in l1:
        l2.append(fatorial(l1[i]))
    q2.put(l2)

if __name__ == '__main__':
    Main()
```

A screenshot of a Windows command prompt window titled "C:\Users\Eloy Barbosa\AppData\Local\Programs\Python\Launcher\py.exe". The window has a black background with white text. The text displayed is: "Inicio: 1554783865.5385857", "Fim: 1554783868.8311958", "Tempo : 3.29", and "Digite enter para sair...". The window includes standard Windows window controls (minimize, maximize, close) in the top right corner.

```
C:\Users\Eloy Barbosa\AppData\Local\Programs\Python\Launcher\py.exe
Inicio: 1554783865.5385857
Fim: 1554783868.8311958
Tempo : 3.29
Digite enter para sair...
```

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).

Tempo de execução:

Sequencial: 4.57

Threading: 4.46

Multiprocessing: 3.29

Diante dos testes feitos, ficou notável que o calculo do Fatorial de 1000000 foi mais rápido com o método de Multiprocessing.