INSTITUTO INFNET DE TECNOLOGIA
EN – ESCOLA DE NEGÓCIOS
SDI - GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO
BLOCO DE ARQUITETURA DE COMPUTADORES, SISTEMAS OPERACIONAIS E REDES

MATHEUS MARTINS DA SILVA

ASSESSMENT
DESENVOLVIMENTO PYTHON PARA SISTEMAS OPERACIONAIS E REDES

Rio de janeiro
Setembro de 2021

MATHEUS MARTINS DA SILVA

ASSESSMENT DA DISCIPLINA DESENVOLVIMENTO PYTHON PARA SISTEMAS OPERACIONAIS E REDES

Assessment referente a disciplina Desenvolvimento Python para Sistemas Operacionais e Redes do bloco em arquitetura de computadores, sistemas operacionais e redes da graduação em Sistemas de Informação apresentado ao Instituto INFNET como requisito para a obtenção de grau na Atividade proposta.

Desenvolvimento Python para Sistemas Operacionais e Redes

Orientador: Thaís do Nascimento Viana

Rio de Janeiro

Setembro de 2021

Sumário

Assessment

Repositório: https://github.com/MatheusMartins1/AT_DR4_MATHEUS_MARTINS

1. Escreva um programa em Python que:

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

1.2. imprima o nome do processo e seu PID;

1.3. imprima também o percentual de uso de CPU e de uso de memória.

```
D:\Users\Matheus\Documents\workspacePython\AT_DR4_MATHEUS_MARTINS\venv\Scripts\python.exe "C:\Program Files\JetBrains\PyCharm 2020.1.1\plugins\python\helpers\pydev\pydevconsole.py" --mode=client --port=1070

import sys; print('Python %s on %s' % (sys.version, sys.platform))
sys.path.extend(['D:\\Users\\Matheus\\Documents\\workspacePython\\AT_DR4_MATHEUS_MARTINS', 'D:/Users/Matheus/Documents/workspacePython/AT_DR4_MATHEUS_MARTINS'])

PyDev console: starting.

Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec  7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)] on win32
>>> import psutil
    from funcoes import formata_tamanho

    def q1():

        processos = {}

        processos_ativos = psutil.pids()
        nomes_processos = []
        for proc in processos_ativos:
            try:
                processo = psutil.Process(proc)
                nome_processo = processo.name()

                if nome_processo not in nomes_processos:
                    processos[proc] = {}
                    processos[proc]["pid"] = processo.pid
                    processos[proc]["nome"] = nome_processo
                    processos[proc]["status"] = processo.status()
                    processos[proc]["memoria"] = processo.memory_info()[1]
                    processos[proc]["memoria_uso"] = processo.memory_percent()
                    processos[proc]["threads"] = processo.num_threads()
                    processos[proc]["cpu_percent"] = processo.cpu_percent(interval=1) / psutil.cpu_count()

                    nomes_processos.append(nome_processo)
                else:
                    processos[proc]["memoria"] += processo.memory_info()[1]
                    processos[proc]["memoria_uso"] += processo.memory_percent()
                    processos[proc]["threads"] += processo.num_threads()
                    processos[proc]["cpu_percent"] += processo.cpu_percent()

            except psutil.NoSuchProcess as e:
                print(f"Processo de PID {proc} não está em execução")

            except Exception:
                pass

        processos = [i[1] for i in sorted(processos.items(), key=lambda x: x[1]['memoria'], reverse=True)] #ordenando por memoria

        processos_retorno = {}
        for p in processos:
            index = processos.index(p)
            if index < 30: #Mantendo somente os 30 primeiros
                processos_retorno[index] = p
                processos_retorno[index]["memoria"] = formata_tamanho(p["memoria"])
                processos_retorno[index]["memoria_uso"] = round(p["memoria_uso"],2)
                # processos_retorno[index]["cpu_percent"] = round(p["cpu_percent"], 2)

        return processos_retorno

    q1_processos = q1()

    for p in q1_processos:
        print(q1_processos[p])

Processo de PID 12928 não está em execução
Processo de PID 14948 não está em execução
{'pid': 13868, 'nome': 'pycharm64.exe', 'status': 'running', 'memoria': '1.52 GB', 'memoria_uso': 9.65, 'threads': 111, 'cpu_percent': 3.9}
{'pid': 17256, 'nome': 'kited.exe', 'status': 'running', 'memoria': '1.41 GB', 'memoria_uso': 8.81, 'threads': 23, 'cpu_percent': 0.25}
{'pid': 4212, 'nome': 'vsserv.exe', 'status': 'running', 'memoria': '658.63 MB', 'memoria_uso': 2.83, 'threads': 150, 'cpu_percent': 0.0}
{'pid': 9472, 'nome': 'java.exe', 'status': 'running', 'memoria': '349.63 MB', 'memoria_uso': 0.14, 'threads': 20, 'cpu_percent': 0.0}
{'pid': 16856, 'nome': 'Microsoft.Python.LanguageServer.exe', 'status': 'running', 'memoria': '240.8 MB', 'memoria_uso': 0.86, 'threads': 13, 'cpu_percent': 0.0}
{'pid': 4324, 'nome': 'core.exe', 'status': 'running', 'memoria': '239.32 MB', 'memoria_uso': 1.48, 'threads': 141, 'cpu_percent': 0.0}
{'pid': 4612, 'nome': 'mongod.exe', 'status': 'running', 'memoria': '214.98 MB', 'memoria_uso': 0.07, 'threads': 33, 'cpu_percent': 0.0}
{'pid': 10860, 'nome': 'SearchApp.exe', 'status': 'stopped', 'memoria': '192.86 MB', 'memoria_uso': 0.42, 'threads': 52, 'cpu_percent': 0.0}
{'pid': 3308, 'nome': 'Code.exe', 'status': 'running', 'memoria': '160.86 MB', 'memoria_uso': 1.43, 'threads': 13, 'cpu_percent': 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.

```
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec  7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)] on win32
>>> # 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,subprocess
...
... def q2():
...     nome_arquivo = input("Insira o nome do arquivo: ")
...     caminho_diretorio = os.getcwd()
...     arquivo = os.path.join(caminho_diretorio,"arquivos",nome_arquivo)
...
...     if os.path.isfile(arquivo):
...         print(f"Arquivo {arquivo} encontrado. \nabrindo...")
...
...         if nome_arquivo.split('.')[1] == "txt":
...             subprocess.run([r'C:\Program Files\Notepad++\notepad++.exe',arquivo])
...
...         else:
...             print("O arquivo não é um txt será aberto em modo binário no notepad")
...             subprocess.run([r'notepad.exe',arquivo])
...     else:
...         print(f"Arquivo {arquivo} não encontrado.")
...
... q2()
...
Insira o nome do arquivo: arquivo.txt
Arquivo D:\Users\Matheus\Documents\workspacePython\AT_DR4_MATHEUS_MARTINS\arquivos\arquivo.txt não encontrado.
>>> q2()
Insira o nome do arquivo: arquivo_q4.txt
Arquivo D:\Users\Matheus\Documents\workspacePython\AT_DR4_MATHEUS_MARTINS\arquivos\arquivo_q4.txt encontrado.
abrindo...
>>> q2()
Insira o nome do arquivo: noite.jpg
Arquivo D:\Users\Matheus\Documents\workspacePython\AT_DR4_MATHEUS_MARTINS\arquivos\noite.jpg encontrado.
abrindo...
O arquivo não é um txt será aberto em modo binário no notepad
```

3. Escreva um programa em Python que:

3.1. 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.

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

3.3. gere um arquivo texto com os valores desta estrutura ordenados.

```
D:\Users\Matheus\Documents\workspacePython\AT_DR4_MATHEUS_MARTINS\venv\Scripts\python.exe "C:\Program Files\JetBrains\PyCharm 2020.1.1\plugins\python\helpers\pydev\pydevconsole.py" --mode=client --port=6667

import sys; print('Python %s on %s' % (sys.version, sys.platform))
sys.path.extend(['D:\\Users\\Matheus\\Documents\\workspacePython\\AT_DR4_MATHEUS_MARTINS', 'D:/Users/Matheus/Documents/workspacePython/AT_DR4_MATHEUS_MARTINS'])

PyDev console: starting.

Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec  7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)] on win32
>>> import os
... import json
... import time
... from funcoes import formata_tamanho,valida_diretorio
...
... def salva_arquivos(diretorio):
...
...     json_object = json.dumps(diretorio, indent=4)
...
...     with open("arquivos/lista_arquivos.json", "w+",encoding='utf-8') as arquivo_json:
...         arquivo_json.write(json_object)
...
... def busca_arquivo(arquivo,caminho_diretorio,tamanho):
...     try:
...         for item in os.listdir(caminho_diretorio):
...             if item == arquivo:
...                 if os.path.isfile(os.path.join(caminho_diretorio, item)):
...                     tamanho = os.stat(os.path.join(caminho_diretorio, item)).st_size
...                 elif os.path.isdir(os.path.join(caminho_diretorio, item)):
...                     tamanho += busca_arquivo(item,os.path.join(caminho_diretorio, item), tamanho)
...     except FileNotFoundError:
...         print("arquivo não encontrado", caminho_diretorio,arquivo)
...         pass
...     except NotADirectoryError:
...         tamanho += os.path.getsize(os.path.join(caminho_diretorio,item))
...     except PermissionError:
...         print("PermissionError")
...         return 0
...
...     return tamanho
```

```
... def q3(caminho):
...     arquivos = {}
...
...     lista_arquivos = os.listdir(caminho)
...
...     for i in lista_arquivos:
...         if not os.path.isdir(os.path.join(caminho, i)):
...             try:
...                 idx = lista_arquivos.index(i)
...                 arquivos[idx] = {}
...                 nome = os.path.splitext(i)
...                 arquivos[idx]["nome"] = nome[0]
...                 arquivos[idx]["tipo"] = nome[1][1:]
...                 arquivos[idx]["tamanho"] = busca_arquivo(i,caminho,0)
...                 arquivos[idx]["dt_criacao"] = time.strftime("%d/%m/%Y %H:%M:%S", time.localtime(os.stat(os.path.join(caminho,i)).st_atime))
...                 arquivos[idx]["dt_modificacao"] = time.strftime("%d/%m/%Y %H:%M:%S", time.localtime(os.stat(os.path.join(caminho,i)).st_mtime))
...             except Exception as e:
...                 print("Erro:",e)
...
...
...     arquivos = [i[1] for i in sorted(arquivos.items(), key=lambda x: x[1]['tamanho'], reverse=True)] #ordenando por tamanho
...
...     for a in arquivos:
...         index = arquivos.index(a)
...         arquivos[index]['tamanho'] = formata_tamanho(a["tamanho"])
...
...     return arquivos
... def exec_q3():
...     caminho_dir = input("Insira diretório: ")
...     caminho_dir = valida_diretorio(caminho_dir)
...     diretorios = q3(caminho_dir)
...     salva_arquivos(diretorios)
...     for d in diretorios:
...         print(d)
...
>>> exec_q3()
Insira diretório: ..
Diretório ' ' não encontrado, realizando busca em D:\Users\Matheus\Documents\workspacePython\AT_DR4_MATHEUS_MARTINS
Lista de arquivos salvo em 'arquivos/lista_arquivos.json'
{'nome': 'AT', 'tipo': 'py', 'tamanho': '4.07 KB', 'dt_criacao': '22/09/2021 12:34:51', 'dt_modificacao': '08/09/2021 20:40:34'}
{'nome': 'questao8', 'tipo': 'py', 'tamanho': '2.61 KB', 'dt_criacao': '22/09/2021 12:34:51', 'dt_modificacao': '22/09/2021 08:25:21'}
{'nome': 'questao3', 'tipo': 'py', 'tamanho': '2.38 KB', 'dt_criacao': '22/09/2021 12:39:09', 'dt_modificacao': '22/09/2021 12:39:09'}
{'nome': 'questao1', 'tipo': 'py', 'tamanho': '2.15 KB', 'dt_criacao': '22/09/2021 12:34:51', 'dt_modificacao': '18/09/2021 11:12:20'}
{'nome': 'questao5', 'tipo': 'py', 'tamanho': '1.76 KB', 'dt_criacao': '22/09/2021 12:34:51', 'dt_modificacao': '18/09/2021 10:38:40'}
{'nome': 'main', 'tipo': 'py', 'tamanho': '1.08 KB', 'dt_criacao': '22/09/2021 12:34:51', 'dt_modificacao': '18/09/2021 10:08:51'}
{'nome': 'questao9', 'tipo': 'py', 'tamanho': '851.00 B', 'dt_criacao': '22/09/2021 12:34:51', 'dt_modificacao': '22/09/2021 08:36:34'}
{'nome': 'questao2', 'tipo': 'py', 'tamanho': '850.00 B', 'dt_criacao': '22/09/2021 12:34:51', 'dt_modificacao': '17/09/2021 21:45:04'}
{'nome': 'funcoes', 'tipo': 'py', 'tamanho': '834.00 B', 'dt_criacao': '22/09/2021 12:34:51', 'dt_modificacao': '18/09/2021 11:19:00'}
{'nome': 'questao4', 'tipo': 'py', 'tamanho': '566.00 B', 'dt_criacao': '22/09/2021 12:34:51', 'dt_modificacao': '18/09/2021 10:32:43'}
{'nome': 'CONSTANTES', 'tipo': 'py', 'tamanho': '55.00 B', 'dt_criacao': '22/09/2021 12:34:51', 'dt_modificacao': '18/09/2021 10:43:38'}
{'nome': '.gitignore', 'tipo': '', 'tamanho': '46.00 B', 'dt_criacao': '22/09/2021 12:20:45', 'dt_modificacao': '22/09/2021 08:26:17'}
```

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

> Bom dia
> Você pode falar agora?

Resultado na tela:

?aroga ralaf edop êcoV

aid moB

```
sys.path.extend(['D:\\Users\\Matheus\\Documents\\workspacePython\\AT_DR4_MATHEUS_MARTINS', 'D:/Users/Matheus/Documents/workspacePython/AT_DR4_MATHEUS_MARTINS'])

PyDev console: starting.

Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec  7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)] on win32
>>> # Escreva um programa em Python que leia um arquivo texto e apresente na tela o seu conteúdo reverso. Exemplo:
...
... with open('arquivos/arquivo_q4.txt','r',encoding='utf-8') as arquivo:
...     texto = arquivo.readlines()
...
... bn = '\n'
... invertido = [i[::-1].replace(bn, '') for i in texto]
... invertido.reverse()
...
... print(bn.join(invertido))
...
?aroga ralaf edop êcoV
aid moB

>>>
```

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.

```
D:\Users\Matheus\Documents\workspacePython\AT_DR4_MATHEUS_MARTINS\venv\Scripts\python.exe "C:\Program Files\JetBrains\PyCharm 2020.1.1\plugins\python\helpers\pydev\pydevconsole.py" --mode=client --port=13296

import sys; print('Python %s on %s' % (sys.version, sys.platform))
sys.path.extend(['D:\\Users\\Matheus\\Documents\\workspacePython\\AT_DR4_MATHEUS_MARTINS', 'D:/Users/Matheus/Documents/workspacePython/AT_DR4_MATHEUS_MARTINS'])

PyDev console: starting.

Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec  7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)] on win32
>>> # 5.    Escreva um programa em Python que leia dois arquivos_q5, a.txt e b.txt, como a seguir: Seu programa deve somar elemento por elemento de cada arquivo e imprimir o resultado na tela. Isto é, o primeiro elemento de
    arquivos_q5 = {}

    def q5():
        tamanho = 0
        soma = 0
        for arq in ("b","a"):
            arquivo_dir = f'arquivos/{arq}.txt'
            with open(arquivo_dir, 'r', encoding='utf-8') as arq_byte:
                texto = arq_byte.read().split(" ")

            tamanho_texto = len(texto)
            arquivos_q5[arq] = {'tamanho': tamanho_texto,
                                'elementos': texto}

            tamanho = tamanho if tamanho_texto < tamanho else tamanho_texto

        total = []
        for el in range(tamanho):
            for letra in arquivos_q5:
                try:
                    valor = int(arquivos_q5[letra]['elementos'][el])
                    soma += valor
                    print(f"arquivo:{letra} | contador:{el} | valor: {valor}")
                except IndexError:
                    print(f"arquivo:{letra} | contador:{el} | valor: {valor} | Somando a zero: {soma}")

            total.append(soma)
            soma = 0

        return total


    print("\nTotal somado:",q5())
    print(f"Arquivo a: {arquivos_q5['a']['elementos']}")
    print(f"Arquivo b: {arquivos_q5['b']['elementos']}")

arquivo:b | contador:0 | valor: 19
arquivo:a | contador:0 | valor: 1
arquivo:b | contador:1 | valor: 56
arquivo:a | contador:1 | valor: 15
arquivo:b | contador:2 | valor: -43
arquivo:a | contador:2 | valor: -42
arquivo:b | contador:3 | valor: 23
arquivo:a | contador:3 | valor: 33
arquivo:b | contador:4 | valor: -7
arquivo:a | contador:4 | valor: -7
arquivo:b | contador:5 | valor: -11
arquivo:a | contador:5 | valor: -2
arquivo:b | contador:6 | valor: 33
arquivo:a | contador:6 | valor: 39
arquivo:b | contador:7 | valor: 21
arquivo:a | contador:7 | valor: 8
arquivo:b | contador:8 | valor: 61
arquivo:a | contador:8 | valor: 61 | Somando a zero: 61
arquivo:b | contador:9 | valor: 9
arquivo:a | contador:9 | valor: 9 | Somando a zero: 9

Total somado: [20, 71, -85, 56, -14, -13, 72, 29, 61, 9]
Arquivo a: ['1', '15', '-42', '33', '-7', '-2', '39', '8']
Arquivo b: ['19', '56', '-43', '23', '-7', '-11', '33', '21', '61', '9']
```

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

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

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

Servidor:

```
D:\Users\Matheus\Documents\workspacePython\AT_DR4_MATHEUS_MARTINS\venv\Scripts\python.exe "C:\Program Files\JetBrains\PyCharm 2020.1.1\plugins\python\helpers\pydev\pydevconsole.py" --mode=client --port=1148

import sys; print('Python %s on %s' % (sys.version, sys.platform))
sys.path.extend(['D:\\Users\\Matheus\\Documents\\workspacePython\\AT_DR4_MATHEUS_MARTINS', 'D:/Users/Matheus/Documents/workspacePython/AT_DR4_MATHEUS_MARTINS'])

Python Console
    import pickle
    from CONSTANTES import HOST,PORTA
    from questao3 import q3

    def qó_server():
        # Cria o socket
        socket_servidor = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        socket_servidor.bind((HOST, PORTA))
        socket_servidor.listen()

        while True:
            try:
                print("Servidor de nome", HOST, "esperando conexão na porta", PORTA)
                resposta = False
                (socket_cliente, addr) = socket_servidor.accept()
                print("\nConectado a:", socket_cliente, str(addr))

                msg = socket_cliente.recv(1024)
                msg = msg.decode('utf-8')

                resposta = q3(msg)

                if resposta:
                    # Prepara a lista para o envio
                    bytes_resp = pickle.dumps(resposta)

                    # Envia os dados
                    socket_cliente.send(bytes_resp)
                    print("enviado")

                while msg == "fim":
                    # socket_cliente.send(resp.encode('utf-8'))
                    socket_cliente.close()

            except Exception as e:
                print(e)


    qó_server()

Servidor de nome 127.0.0.1 esperando conexão na porta 9990

Conectado a: <socket.socket fd=964, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM, proto=0, laddr=('127.0.0.1', 9990), raddr=('127.0.0.1', 1289)> ('127.0.0.1', 1289)
enviado
Servidor de nome 127.0.0.1 esperando conexão na porta 9990
```

Cliente:

```
D:\Users\Matheus\Documents\workspacePython\AT_DR4_MATHEUS_MARTINS\venv\Scripts\python.exe "C:\Program Files\JetBrains\PyCharm 2020.1.1\plugins\python\helpers\pydev\pydevconsole.py" --mode=client --port=1237

import sys; print('Python %s on %s' % (sys.version, sys.platform))
sys.path.extend(['D:\\Users\\Matheus\\Documents\\workspacePython\\AT_DR4_MATHEUS_MARTINS', 'D:/Users/Matheus/Documents/workspacePython/AT_DR4_MATHEUS_MARTINS'])

PyDev console: starting.

Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec  7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)] on win32
>>> import json
    import socket
    import pickle
    from CONSTANTES import HOST,PORTA
    from questao3 import valida_diretorio

    def qó_cliente(diretorio):
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        resposta = ""
        caminho = valida_diretorio(diretorio)
        try:
            print(f"\nSocket conectando ao servidor em {HOST}:{PORTA}")
            s.connect((HOST, PORTA))

            s.send(caminho.encode('utf-8'))

            bytes = s.recv(4096)

            resposta = pickle.loads(bytes, fix_imports=True, encoding="utf-8")

        except Exception as erro:
            print("erro:",str(erro))

        print("fim conexão socket\n")
        s.close()

        return resposta
```

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

7.1. 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.

7.2. 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.

Servidor:

```
D:\Users\Matheus\Documents\workspacePython\AT_DR4_MATHEUS_MARTINS\venv\Scripts\python.exe "C:\Program Files\JetBrains\PyCharm 2020.1.1\plugins\python\helpers\pydev\pydevconsole.py" --mode=client --port=28869

import sys; print('Python %s on %s' % (sys.version, sys.platform))
sys.path.extend(['D:\\Users\\Matheus\\Documents\\workspacePython\\AT_DR4_MATHEUS_MARTINS', 'D:/Users/Matheus/Documents/workspacePython/AT_DR4_MATHEUS_MARTINS'])

PyDev console: starting.

Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec  7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)] on win32
>>> import socket
    import pickle
    import psutil,time
    from CONSTANTES import *
    from funcoes import formata_tamanho

    PORTA = PORTA + 1

    def retorna_memoria():
        memoria = psutil.virtual_memory()
        percent_uso = round(((memoria.used / memoria.total) * 100), 2)
        memoria_json = {
            "Total": formata_tamanho(memoria.total)
            , "Em uso": formata_tamanho(memoria.used)
            , "percent_uso": memoria.percent
            , "Livre": formata_tamanho(memoria.free)
            , "percent_livre": round(100 - percent_uso, 2)
        }
        return memoria_json


    udp = socket.socket (socket.AF_INET, socket.SOCK_DGRAM)
    udp.bind((HOST, PORTA))

    while True:
        print(f"Esperando receber conexão {HOST} na porta {PORTA}")

        (msg, cliente) = udp.recvfrom(1024)
        msg = msg.decode('utf-8')

        if msg == MSG_INICIO:
            time.sleep(5)
            resposta = retorna_memoria()
            bytes_resp = pickle.dumps(resposta)

            udp.sendto(bytes_resp,cliente)

    udp.close()

Esperando receber conexão 127.0.0.1 na porta 9991
Esperando receber conexão 127.0.0.1 na porta 9991
Esperando receber conexão 127.0.0.1 na porta 9991
```

Cliente:

```
D:\Users\Matheus\Documents\workspacePython\AT_DR4_MATHEUS_MARTINS\venv\Scripts\python.exe "C:\Program Files\JetBrains\PyCharm 2020.1.1\plugins\python\helpers\pydev\pydevconsole.py" --mode=client --port=28919

import sys; print('Python %s on %s' % (sys.version, sys.platform))
sys.path.extend(['D:\\Users\\Matheus\\Documents\\workspacePython\\AT_DR4_MATHEUS_MARTINS', 'D:/Users/Matheus/Documents/workspacePython/AT_DR4_MATHEUS_MARTINS'])

PyDev console: starting.

Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec  7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)] on win32
>>> import socket
    import pickle
    import time
    import json
    from CONSTANTES import *

    PORTA = PORTA + 1

    def q7_cliente():
        udp = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        udp.settimeout(5)
        dest = (HOST, PORTA)

        print(f"Realizando conexão {HOST} na porta {PORTA}")
        udp.sendto(MSG_INICIO.encode("utf-8"),dest)
        recebido = False

        print(f"Enviando requisição 1/6 - Tempo:{time.strftime('%H:%M:%S', time.localtime())}")
        try:
            (resposta, servidor) = udp.recvfrom(4096)
            recebido = True
        except socket.timeout:
            udp.sendto(MSG_INICIO.encode("utf-8"), dest)
            print(f"Enviando requisição 2/6 - Tempo:{time.strftime('%H:%M:%S', time.localtime())}")

        udp.close()

        if recebido:
            disco_json = pickle.loads(resposta, fix_imports=True, encoding="utf-8")
            disco_json['hora'] = time.strftime('%d/%m/%y %H:%M:%S',time.localtime())

            print(json.dumps(disco_json, indent=4, sort_keys=True), '\n')
            return disco_json

        else:
            print("Não foi possível recuperar os dados")
            return None


    q7_cliente()

Realizando conexão 127.0.0.1 na porta 9991
Enviando requisição 1/6 - Tempo:19:37:45
Enviando requisição 2/6 - Tempo:19:37:50
Dados retornados
{
    "Em uso": "10.37 GB",
    "Livre": "5.42 GB",
    "Total": "15.79 GB",
    "hora": "22/09/21 19:37:55",
    "percent_livre": 34.32,
    "percent_uso": 65.7
}
```

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.

```
D:\Users\Matheus\Documents\workspacePython\AT_DR4_MATHEUS_MARTINS\venv\Scripts\python.exe "C:\Program Files\JetBrains\PyCharm 2020.1.1\plugins\python\helpers\pydev\pydevconsole.py" --mode=client --port=30829

import sys; print('Python %s on %s' % (sys.version, sys.platform))
sys.path.extend(['D:\\Users\\Matheus\\Documents\\workspacePython\\AT_DR4_MATHEUS_MARTINS', 'D:/Users/Matheus/Documents/workspacePython/AT_DR4_MATHEUS_MARTINS'])

PyDev console: starting.

Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec  7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)] on win32
>>> # 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:
    import time
    from random import randint,random

    tamanho = 10
    vetor_b_tread = []

    def gera_vetor(tamanho):
        vertorA = []
        for i in range(1,tamanho):
            vertorA.append(randint(i, i*100))

        return vertorA


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

a) sequencialmente (sem concorrência);

```
    def q8_a(tamanho):
        # a.sequencialmente (sem concorrência)
        t_inicio = float(time.time())

        vetores_A = gera_vetor(tamanho)
        vetores_B = []
        for n in vetores_A:
            vetores_B.append(fatorial(n))

        t_fim = float(time.time())
        return t_fim - t_inicio
```

b) usando o módulo threading com 4 threads;

```
    def aux_b(vetor_a,inicio,fim,vetor_b_tread):
        try:
            for n in range(inicio,fim):
                vetor_b_tread[n] = fatorial(vetor_a[n])
        except:
            print(n, len(vetor_b_tread))

    def q8_b(qntdT,tamanho):
        # b.usando o módulo threading com 4 threads
        import threading

        treads = []
        t_inicio = float(time.time())

        vetor_a = gera_vetor(tamanho)
        vetor_b_tread = vetor_a.copy()

        for i in range(qntdT):
            ini = i * int(tamanho / qntdT)
            fim = (i + 1) * int(tamanho / qntdT)
            t = threading.Thread(target=aux_b,args=(vetor_a,ini,fim,vetor_b_tread))
            t.start()
            treads.append(t)

        for t_ativa in treads:
            t_ativa.join()

        t_fim = float(time.time())
        del treads
        return t_fim - t_inicio
```

c) usando o módulo multiprocessing com 4 processos.

```python
def q8_target(fila_entrada,fila_saida):
    fatiada = fila_entrada.get()
    fila_saida.put([fatorial(n) for n in fatiada])

def q8_c(qntdP,tamanho):
    # c.usando o módulo multiprocessing com 4 processos
    import multiprocessing
    t_inicio = float(time.time())

    fila_entrada = multiprocessing.Queue()
    fila_saida = multiprocessing.Queue()

    vetor_a = gera_vetor(tamanho)

    lista_processos = []
    for i in range(qntdP):
        lista_fatiada = vetor_a[i * int(tamanho / qntdP):(i + 1) * int(tamanho / qntdP)]
        fila_entrada.put(lista_fatiada)
        processo = multiprocessing.Process(target=q8_target,args=(fila_entrada,fila_saida))
        processo.start()  # inicia processo 0
        lista_processos.append(processo)

    lista_nova = []
    for i in range(qntdP):
        lista_nova.extend(fila_saida.get())

    t_fim = float(time.time())
    return t_fim - t_inicio


if __name__ == '__main__':
    print(f'Exercicio a - tempo: {q8_a(tamanho)}')
    print(f'Exercicio b - tempo: {q8_b(4,tamanho)}')
    print(f'Exercicio c - tempo: {q8_c(4,tamanho,)}')
```

Resultado:

```
D:\Users\Matheus\Documents\workspacePython\AT_DR4_MATHEUS_MARTINS\venv\Scripts\python.exe D:/Users/Matheus/Documents/workspacePython/AT_DR4_MATHEUS_MARTINS/questao8.py
Exercicio a - tempo: 0.0
Exercicio b - tempo: 0.000997304916381836
Exercicio c - tempo: 0.2533223628997803

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).

```python
import json
from questao8 import q8_a,q8_b,q8_c,vetor_b_tread

tamanho = 10

tamanhos = {
    1:{"tamanho":1000000},
    2:{"tamanho":5000000},
    3:{"tamanho":10000000},
}

# tamanhos = {
#     1:{"tamanho":100},
#     2:{"tamanho":500},
#     3:{"tamanho":1000},
# }

if __name__ == '__main__':

    for i in tamanhos:
        global vetor_b_tread
        vetor_b_tread = []
        tamanhos[i]['Sequencial'] = q8_a(tamanhos[i]["tamanho"])
        tamanhos[i]['Threading'] = q8_b(4,tamanhos[i]["tamanho"])
        tamanhos[i]['Multiprocessing'] = q8_c(4,tamanhos[i]["tamanho"])

    print(json.dumps(tamanhos, indent=4, sort_keys=True), '\n')
```

Resultado utilizando tamanhos 100, 500 e 1000:

```
D:\Users\Matheus\Documents\workspacePython\AT_DR4_MATHEUS_MARTINS\venv\Scripts\python.exe D:/Users/Matheus/Documents/workspacePython/AT_DR4_MATHEUS_MARTINS/questao9.py
99 99
499 499
999 999
{
    "1": {
        "Multiprocessing": 0.2911961078643799,
        "Sequencial": 0.20245361328125,
        "Threading": 0.2284811180114746,
        "tamanho": 100
    },
    "2": {
        "Multiprocessing": 16.463762998580933,
        "Sequencial": 30.173003673553467,
        "Threading": 29.46696448326111,
        "tamanho": 500
    },
    "3": {
        "Multiprocessing": 167.1916184425354,
        "Sequencial": 243.7824375629425,
        "Threading": 255.09235668182373,
        "tamanho": 1000
    }
}


Process finished with exit code 0
```