INSTITUTO INFNET

ESCOLA SUPERIOR DE TECNOLOGIA

GRADUAÇÃO EM ANÁLISE E DESENVOLVIMENTO

PROGRAMAÇÃO COM PYTHON

TESTE DE PERFORMANCE – TP2

ALUNO: GABRIEL GOMES DE SOUZA

PROFESSOR(A): DÁCIO MOREIRA DE SOUZA

E-MAIL: gabriel.gsouza@al.infnet.edu.br

Sumário

1. Conteúdo

Questão 01

```
# Contador de Números Positivos
# Problema: Escreva um programa que utilize um loop for para percorrer uma lista de números gerados aleatoriamente e conte quantos são positivos.

import random

def geradorLista(inicial, final):
    """
    Gera uma lista de números inteiros aleatórios dentro de um intervalo.

    Args:
        inicial (int): O limite inferior do intervalo.
        final (int): O limite superior do intervalo.

    Returns:
        list: Uma lista de 5 números inteiros aleatórios dentro do intervalo especificado.
    """
    lista = []
    for i in range(5):  # Gera 5 números aleatórios
        lista.append(random.randint(inicial, final))
    return lista

def contadorPositivos(lista):
    """
    Conta o número de elementos positivos em uma lista.

    Args:
        lista (list): Uma lista de números inteiros.

    Returns:
```

```
        int: O número de elementos positivos na lista.
    """
    numPositivos = 0
    for i in lista:
        if i > 0:
            numPositivos += 1
    return numPositivos

# Gerar uma lista de números aleatórios entre -50 e 50
lista_gerada = geradorLista(-50, 50)
print("Lista gerada:", lista_gerada)

# Contar o número de elementos positivos na lista gerada
num_positivos = contadorPositivos(lista_gerada)
print("A lista tem", num_positivos, "números positivos.")
```

Questão 02

```
# Acumulador de Soma
# Problema: Desenvolva um programa que use um loop while
para somar todos os números até 100 e imprimir o resultado.

def soma(numero):
    """
    Esta função utiliza um loop while para somar todos os
números até o número fornecido como argumento e retorna o
resultado.

    Args:
    numero (int): O número até o qual se deseja somar.

    Returns:
    int: O resultado da soma dos números de 1 até o número
fornecido.
```

```
    """
    contador = 0  # Inicia o contador em 0
    somatorio = 0  # Inicia o somatório em 0

    # Loop while para somar os números até o número
fornecido
    while contador != numero:  # Enquanto o contador não
alcançar o número fornecido
        contador += 1  # Incrementa o contador em 1 a cada
iteração
        somatorio += contador  # Adiciona o valor do
contador ao somatório

    return somatorio

# Imprime o resultado
print("A soma dos números de 1 até 100 é", soma(100))
```

Questão 03

```
# Concatenação e Organizador de Listas
# Problema: Crie um programa que concatene duas listas
fornecidas pelo usuário e organize a lista em ordem
crescente.

def concatenarOrganizar(lista1, lista2):
    """
    Concatena duas listas fornecidas e organiza a lista
resultante em ordem crescente.

    Args:
        lista1 (list): A primeira lista a ser concatenada.
        lista2 (list): A segunda lista a ser concatenada.

    Returns:
        list: A lista resultante da concatenação e
```

```
organização das duas listas em ordem crescente.
    """
    # Concatena as duas listas
    lista1.extend(lista2)
    # Ordena a lista resultante em ordem crescente
    lista1.sort()
    return lista1

# Listas fornecidas pelo usuário
lista1 = []
lista2 = []

# Loop para inserir elementos na lista1
for i in range(1, 5):
    lista1.append(int(input("Digite um número para lista 1:
")))

# Loop para inserir elementos na lista2
for i in range(1, 5):
    lista2.append(int(input("Digite um número para lista 2:
")))

# Chama a função para concatenar e organizar as listas
resultado = concatenarOrganizar(lista1, lista2)

# Imprime o resultado
print(resultado)
```

Questão 04

```
# Conversor de Temperatura
# Problema: Implemente uma função que converta a temperatura
de Celsius para Fahrenheit.

def celsius_para_fahrenheit(celsius):
    """
```

```
    Converte a temperatura de Celsius para Fahrenheit.

    Args:
        celsius (float): A temperatura em graus Celsius a
ser convertida.

    Returns:
        float: A temperatura equivalente em graus
Fahrenheit.
    """
    # Fórmula de conversão de Celsius para Fahrenheit
    fahrenheit = (celsius * 9/5) + 32
    return fahrenheit

# Solicita ao usuário para inserir a temperatura em Celsius
temperatura_celsius = float(input("Insira uma temperatura em
Celsius (C°): ").replace(",", "."))

# Chama a função para converter a temperatura
temperatura_fahrenheit =
celsius_para_fahrenheit(temperatura_celsius)

# Imprime o resultado
print("{}°C equivale a {}°F".format(temperatura_celsius,
temperatura_fahrenheit))
```

Questão 05

```
# Gerador de Lista de Quadrados
# Problema: Escreva um programa que utilize um loop for para
gerar uma lista dos quadrados dos números de 1 a 20.

def gerar_lista_quadrados():
    """
    Gera uma lista dos quadrados dos números de 1 a 20.
```

```
    Returns:
        list: Lista dos quadrados dos números de 1 a 20.
    """
    lista = []  # Inicializa uma lista vazia para armazenar
os quadrados
    for i in range(1, 21):  # Loop de 1 a 20
        lista.append(i ** 2)  # Adiciona o quadrado do
número atual à lista
    return lista  # Retorna a lista de quadrados


# Chama a função e imprime a lista gerada
print(gerar_lista_quadrados())
```

Questão 06

```
# Listagem de Números Primos
# Problema: Escreva um programa que verifique entre todos os
números de 1 a 100 quais são números primos e exiba uma
lista com todos.

def verificaPrimo(numero):
    """
    Verifica se um número é primo.

    Args:
        numero (int): O número a ser verificado.

    Returns:
        bool: True se o número for primo, False caso
contrário.
    """
    if numero <= 1:  # Se o número for menor ou igual a 1,
não é primo
        return False
```

```
    for i in range(2, numero):  # Loop para verificar
divisibilidade por números diferentes de 1 e o próprio
número
        if numero % i == 0:  # Se for divisível por algum
número diferente de 1 e ele mesmo, não é primo
            return False

    return True  # Se não for divisível por nenhum número
diferente de 1 e ele mesmo, é primo

numeros_primos = []
for num in range(1, 101):
    if verificaPrimo(num):  # Se o número for primo,
adiciona à lista de números primos
        numeros_primos.append(num)

print("Números primos de 1 a 100:")
print(numeros_primos)
```

Questão 07

```
# Função com Parâmetros Padrões
# Problema: Implemente uma função que desenhe uma linha
padrão na tela sem que sejam passados argumentos para a
função, porém aceitando como parâmetros definidos por
posição um caractere para construir a linha e o comprimento
da linha.

def desenhar_linha(caractere='-', comprimento=25):
    """
    Desenha uma linha na tela usando um caractere
especificado e um comprimento.

    Args:
    - caractere (str): O caractere usado para desenhar a
linha. O padrão é '-'.
```

```
    - comprimento (int): O comprimento da linha a ser
desenhada. O padrão é 10.
    """
    # Desenha a linha na tela
    print(caractere * comprimento)

# Chamada da função sem argumentos (usando os padrões)
desenhar_linha()

# Chamada da função com argumentos específicos
desenhar_linha('*', 20)
```

Questão 08

```
# Função de Potência Customizada
# Problema: Escreva uma função que calcule a potência de um
número com um expoente padrão de 2, permitindo ao usuário
alterar esse expoente com parâmetros por Keyword e retorne o
resultado da operação.

def potencia(base, expoente=2):
    """
    Calcula a potência de um número com um expoente padrão
de 2.

    Args:
        base (float): O número base.
        expoente (float, optional): O expoente para o qual a
base será elevada.
            Padrão é 2.

    Returns:
        float: O resultado da operação de potência.

    """
    return base ** expoente
```

```
# Testando a função
base = int(input("Digite um número para a base da potencia:
"))
expoente = int(input("Digite um número para o expoente da
potencia: "))
resultado = potencia(base, expoente)
print(f"{base} elevado a {expoente} é {resultado}.")

# Testando a função com o expoente padrão
resultado_padrao = potencia(base)
print(f"{base} elevado ao expoente padrão é
{resultado_padrao}.")
```

Questão 09

```
# Função de Fatorial
# Problema: Escreva uma função que calcule o fatorial de um
número passado como argumento e retorne o resultado.

def fatorial(n):
    """
    Calcula o fatorial de um número inteiro não negativo.

    Args:
        n (int): O número para o qual o fatorial será
calculado.

    Returns:
        int: O fatorial de n.
    """
    if n < 0:
        raise ValueError("O fatorial não está definido para
números negativos.")
    if n == 0:
        return 1
```

```
    else:
        return n * fatorial(n - 1)

# Testando a função
numero = int(input("Digite um número para calculo de
fatorial: "))
resultado = fatorial(numero)

# Formatando o resultado com separadores de milhar
resultado_formatado = "{:,}".format(resultado).replace(",",
".")
print(f"O fatorial de {numero} é {resultado_formatado}.")
```

Questão 10

```
# Conversor de Base Numérica
# Problema: Desenvolva um programa que converta um número da
base decimal para binária usando um loop while.

#numero = int(input("Digite um número para conversão: "))
decimal = 39
test = 0
def decimal_para_binario(decimal):
    """
    Converte um número decimal para binário.

    Args:
    - decimal (int): O número decimal a ser convertido.

    Returns:
    - str: O número binário equivalente como uma string.
    """
    if decimal == 0:
        return '0'

    binario = ''
```

```
    while decimal > 0:
        resto = decimal % 2
        binario = str(resto) + binario
        decimal //= 2
        #print("Número:", decimal, "\tResto:", resto)
    return binario

# Solicita ao usuário o número decimal
numero_decimal = int(input("Digite um número decimal: "))

# Chama a função para converter o número e exibe o resultado
numero_binario = decimal_para_binario(numero_decimal)
print("O número binário equivalente é:", numero_binario)
```

Questão 11

```
# Ordenador de Lista por Tamanho de Palavra
# Problema: Implemente um programa que organize uma lista de palavras em ordem crescente de tamanho.

lista = ["12345", "1234", "1234567", "12345", "12345678", "123456789", "123"]
lista_ordenada = sorted(lista, key=len)

def filtrar_por_tamanho(lista):
    """
    Filtra uma lista de strings por tamanho, organizando-as em ordem crescente de comprimento.

    Args:
        lista (list): Uma lista de strings a serem filtradas e organizadas.

    Returns:
        tuple: Uma tupla contendo a lista original e a lista ordenada por tamanho das strings.
```

```
    """
    lista_ordenada = sorted(lista, key=len)
    return lista, lista_ordenada

# Testando a função e imprimindo os resultados
lista_original, lista_ordenada = filtrar_por_tamanho(lista)
print("Lista original:", lista_original)
print("Lista organizada por tamanho de string:",
lista_ordenada)
```

Questão 12

```
# Filtragem de Lista por Condição
# Problema: Desenvolva um programa que filtre uma lista de
números, removendo aqueles que não satisfazem uma condição
específica (por exemplo, ser par).

lista = [12345, 1234, 1234567, 12345, 12345678, 123456789,
123]

def filtrar_numeros_pares(lista):
    """
    Filtra uma lista de números, removendo aqueles que não
são pares.

    Args:
        lista (list): Uma lista de números inteiros.

    Returns:
        list: Uma nova lista contendo apenas os números
pares da lista original.
    """
    lista_filtrada = [num for num in lista if num % 2 == 0]
    return lista_filtrada

# Exemplo de uso da função
```

```
lista_filtrada = filtrar_numeros_pares(lista)
print("Lista original:", lista)
print("Lista filtrada (apenas números pares):",
lista_filtrada)
```

Questão 13

```
# Função Geradora de Histograma
# Problema: Escreva uma função que receba uma lista de
números e imprima um histograma na tela (usando asteriscos
para representar a frequência dos números).

import random

def gerarLista(tamanho, valor_minimo, valor_maximo):
    """
    Gera uma lista de números aleatórios.

    Args:
        tamanho (int): O tamanho da lista a ser gerada.
        valor_minimo (int): O valor mínimo que pode ser
gerado na lista.
        valor_maximo (int): O valor máximo que pode ser
gerado na lista.

    Returns:
        list: Uma lista de números inteiros aleatórios com o
tamanho especificado e dentro do intervalo especificado.
    """
    return [random.randint(valor_minimo, valor_maximo) for i
in range(tamanho)]

def histograma(lista):
    """
    Imprime um histograma na tela usando asteriscos para
representar a frequência dos números na lista.
```

```
    Args:
        lista (list): Uma lista de números inteiros.

    Returns:
        None
    """
    largura = max(lista)  # Determina a largura máxima do
histograma
    for i in range(largura, 0, -1):
        linha = ""
        for n in lista:
            if n >= i:
                linha += "\033[1;41m * \033[m "
            else:
                linha += "    "  # Adiciona espaços em
branco para manter o alinhamento
        print(linha)
    print(end=" ")
    for n in range(0, len(lista)):
        if lista[n] < 10:
            print(lista[n], end="   ")
        else:
            print(lista[n], end="  ")

# Exemplo de uso:
tamanho_da_lista = 10
valor_minimo = 1
valor_maximo = 10
lista = gerarLista(tamanho_da_lista, valor_minimo,
valor_maximo)
histograma(lista)
```

Questão 14

```
# Simulador de Lançamento de Dados
```

```python
# Problema: Crie um programa que simule o lançamento de um dado n vezes, armazenando os resultados em uma lista, depois calcule a média dos resultados e calcule a distribuição de frequência dos elementos da lista.

import random

def simular_lancamento_dado(n_lancamentos):
    """
    Simula o lançamento de um dado n vezes.

    Args:
        n_lancamentos (int): O número de vezes que o dado será lançado.

    Returns:
        tuple: Uma tupla contendo a lista dos resultados dos lançamentos e a média dos resultados.
    """
    lista_resultados = []

    for _ in range(n_lancamentos):
        resultado = random.randint(1, 6)
        lista_resultados.append(resultado)

    media = sum(lista_resultados) / n_lancamentos

    return lista_resultados, media

def calcular_distribuicao_frequencia(lista):
    """
    Calcula a distribuição de frequência dos elementos em uma lista.

    Args:
```

```
        lista (list): A lista contendo os elementos.

    Returns:
        dict: Um dicionário onde as chaves são os elementos
únicos da lista e os valores são as frequências desses
elementos.
    """
    distribuicao_frequencia = {}

    for elemento in lista:
        if elemento in distribuicao_frequencia:
            distribuicao_frequencia[elemento] += 1
        else:
            distribuicao_frequencia[elemento] = 1

    return distribuicao_frequencia

# Simula lançamento de dado 5 vezes
n_lancamentos = 5
resultados, media = simular_lancamento_dado(n_lancamentos)

print("Resultados dos lançamentos:", resultados)
print("Média dos resultados:", media)

distribuicao_frequencia =
calcular_distribuicao_frequencia(resultados)
print("Distribuição de frequência:",
distribuicao_frequencia)
```

Questão 15

```
# Função Geradora de Senhas Aleatórias
# Problema: Escreva uma função que gere uma senha aleatória
contendo letras maiúsculas, minúsculas, números e símbolos.

import random
```

```python
def gerarSenha(numCaracteres=8):
    """
    Gera uma senha aleatória contendo letras maiúsculas, minúsculas, números e símbolos.

    Args:
    - numCaracteres (int): O número de caracteres que a senha deve conter. O padrão é 8.

    Returns:
    - senha (str): A senha gerada.
    """

    # Conjuntos de caracteres possíveis para a senha
    letras_maiusculas = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
    letras_minusculas = 'abcdefghijklmnopqrstuvwxyz'
    numeros = '0123456789'
    simbolos = '!@#$%&*()_+-=[]{}|;:,.<>?'

    # Concatenando todos os conjuntos de caracteres em uma única string
    caracteres = letras_maiusculas + letras_minusculas + numeros + simbolos

    # Gerando a senha aleatória
    senha = ''.join(random.choice(caracteres) for _ in range(numCaracteres))

    return senha

# Exemplo de uso
senha = gerarSenha(12)  # Gerar senha com 12 caracteres
print(senha)
```

Questão 16

```
# Validador de CPF
# Problema: Crie um programa que valide um número de CPF fornecido pelo usuário (considerando apenas a validação dos dígitos verificadores).

def validar_cpf(cpf):
    """
    Valida um número de CPF fornecido pelo usuário (considerando apenas a validação dos dígitos verificadores).

    Args:
    - cpf (str): O CPF a ser validado. Deve ser uma string contendo apenas os dígitos do CPF.

    Returns:
    - bool: Retorna True se o CPF for válido (os dígitos verificadores estiverem corretos), False caso contrário.
    """

    cpf = ''.join(filter(str.isdigit, cpf))  # Remover caracteres não numéricos

    if len(cpf) != 11 or cpf == cpf[0] * 11:  # Verificar se o CPF tem 11 dígitos e não é uma sequência repetida
        return False

    soma1 = sum(int(cpf[i]) * (10 - i) for i in range(9))  # Calculando o primeiro dígito verificador
    digito1 = (soma1 * 10) % 11
    if digito1 == 10:
        digito1 = 0

    soma2 = sum(int(cpf[i]) * (11 - i) for i in range(10))
```

```
# Calculando o segundo dígito verificador
    digito2 = (soma2 * 10) % 11
    if digito2 == 10:
        digito2 = 0

    return digito1 == int(cpf[9]) and digito2 ==
int(cpf[10])  # Verificando se os dígitos calculados
coincidem com os dígitos fornecidos

# Exemplo de uso
cpf = input("Digite o CPF (somente números): ")
if validar_cpf(cpf):
    print("CPF válido.")
else:
    print("CPF inválido.")
```

Questão 17

```
# Simulador de Caixa Eletrônico
# Problema: Desenvolva um programa que simule a operação de
um caixa eletrônico, permitindo ao usuário sacar uma quantia
especificada e retornando as notas necessárias para o
montante.

def caixa_eletronico(valor_saque):
    """
    Simula a operação de um caixa eletrônico, permitindo ao
usuário sacar uma quantia especificada e retornar as notas
necessárias para o montante, utilizando as notas
disponíveis: 100, 50, 20, 10, 5 e 2 reais.
    Em seguida, imprime na tela as notas necessárias para o
saque ou uma mensagem indicando que não é possível sacar o
valor solicitado, caso não seja possível combinar as notas
disponíveis para alcançar o valor desejado.

    Args:
```

```
    - valor_saque: int
        O valor que o usuário deseja sacar do caixa
eletrônico.

    Returns:
    Esta função não retorna nenhum valor. Ela imprime na
tela as notas necessárias para o saque, com base nas notas
disponíveis no caixa eletrônico.
    """
    notas_disponiveis = [100, 50, 20, 10, 5, 2]
    notas_para_saque = {}

    for nota in notas_disponiveis:
        quantidade_notas = valor_saque // nota
        if quantidade_notas > 0:
            notas_para_saque[nota] = quantidade_notas
            valor_saque %= nota

    if valor_saque != 0:
        print("Não é possível sacar o valor solicitado.")
    else:
        print("Notas para o saque:")
        for nota, quantidade in notas_para_saque.items():
            print(f"{quantidade} nota(s) de R${nota},00")

valor = int(input("Digite o valor que deseja sacar: "))
caixa_eletronico(valor)
```