Candidato: Gabriel Lopes

Vaga: Estágio em Desenvolvimento de Software de Soluções Digitais para Agricultura de Precisão

**Questão** **1.** *Escreva uma função que receba uma string e imprima a quantidade de vogais (a, e, i, o, u) presentes na string. A contagem deve ser feita independentemente de letras maiúsculas ou minúsculas.  
Exemplo: Olá, mundo!*

*Resposta: 4*

***Codigo:***

def contar\_vogais\_usuario():  
 #Define uma string contendo todas as vogais, incluindo vogais acentuadas  
 vogais = 'aáàâãeéêiíoóôõuú'  
  
 #Inicializa o total de vogais para acumular a soma de todas as frases  
 total\_vogais = 0  
  
 #Laço infinito que vai pedir repetidamente frases ao usuário  
 while True:  
  
 #Solicita ao usuário que digite uma frase e converte para minúsculas  
 frase = input("Digite uma palavra ou uma frase: ").lower()  
  
 #Se a frase for uma string vazia, o laço para  
 if frase == '':  
 break  
  
 #Inicializa o contador de vogais para a frase atual  
 contador = 0  
  
 #Itera sobre cada caractere da frase digitada  
 for caractere in frase:  
  
 #Se o caractere for uma vogal (incluindo as acentuadas), incrementa o contador  
 if caractere in vogais:  
 contador += 1  
  
 #Soma o numero de vogais da frase atual ao total de todas as frases  
 total\_vogais += contador  
  
 #Exibe o numero de vogais encontradas na frase atual  
 print(f'Frase: "{frase}" tem {contador} vogais.')  
  
 #Após o laço ser cencerrado, exibe o total de vogais encontradas em todas as frases digitadas  
 print(f'Total de vogais em todas as frases: {total\_vogais}')  
  
#Chama a função para iniciar a contagem de vogais com interação do usuário  
contar\_vogais\_usuario()

***Raciocínio do código.***

A string vogais = ' aáàâãeéêiíoóôõuú ' define os caracteres que serão contados. A variável total\_vogais é inicializada como 0 para armazenar a soma de vogais em todas as frases inseridas pelo usuário.

O loop while True permite que o usuário insira várias frases. O loop continua até o usuário digitar uma string vazia (frase == ''), momento em que o loop é encerrado com break.

A função input() captura a frase do usuário, convertendo-a para letras minúsculas com lower() para garantir que as vogais maiúsculas sejam tratadas corretamente.

Um loop for percorre cada caractere da frase e verifica se está na string vogais. Para cada vogal encontrada, o contador da frase (contador) é incrementado.

Após contar as vogais da frase atual, o valor é adicionado à variável total\_vogais, que mantém o total acumulado de todas as frases.

**Questão 2**. *Escreva uma função que imprima o maior número par e o maior número ímpar de uma lista de números inteiros.*

*Exemplo: [1, 2, 3, 4, 5]*

*Resposta: O maior par é 4 e o maior ímpar é 5*

***Código***

def maior\_par\_impar():  
 #Recebe a entrada do usuário com números separados por espaços  
 #O input é convertido para uma lista de inteiros usando map(int, ...)  
 numeros = list(map(int, input("Digite os números separados por espaços: ").split()))  
  
 #Cria uma lista de números pares a partir da lista de números  
 pares = [num for num in numeros if num % 2 == 0]  
  
 #Cria uma lista de números ímpares a partir da lista de números  
 impares = [num for num in numeros if num % 2 != 0]  
  
 #Se houver números pares, encontra o maior; caso contrário, retorna None  
 maior\_par = max(pares) if pares else None  
  
 #Se houver números ímpares, encontra o maior; caso contrário, retorna None  
 maior\_impar = max(impares) if impares else None  
  
 #Exibe o maior número par e o maior número ímpar (ou None se não houver)  
 print(f'O maior par é {maior\_par} e o maior ímpar é {maior\_impar}')  
  
#Chama a função para que o usuário possa inserir números e ver o resultado  
maior\_par\_impar()

***Raciocínio do código.***

A função input() captura essa string, e split() divide os números em uma lista de strings.

A função map(int, ...) converte cada string da lista em um número inteiro, e list() transforma isso em uma lista de inteiros.

Os números pares são filtrados com num % 2 == 0 e os ímpares com num % 2 != 0. O maior de cada lista é encontrado usando max(), com tratamento para o caso de listas vazias.

O programa imprime o maior número par e o maior ímpar. Se não houver números pares ou ímpares, será exibido None.

**Questão 3.***Escreva uma função que receba duas strings e imprima se uma é um anagrama da outra. Duas strings são anagramas se podem ser rearranjadas para formar uma à outra, ignorando espaços e diferenças de maiúsculas e minúsculas.*

*Exemplo: "amor", "roma"*

*Resposta: São anagramas*

*Exemplo: "hello", "world"*

*Resposta: Não são anagramas*

**Código**

def sao\_anagramas():  
 #Recebe a primeira string do usuário  
 string1 = input("Digite a primeira palavra ou frase: ")  
  
 #Recebe a segunda string do usuário  
 string2 = input("Digite a segunda palavra ou frase: ")  
  
 #Remove os espaços e converte as strings para letras minúsculas  
 string1 = string1.replace(" ", "").lower()  
 string2 = string2.replace(" ", "").lower()  
  
 #Verifica se as duas strings têm o mesmo tamanho após o tratamento  
 #Se o tamanho for diferente, não podem ser anagramas  
 if len(string1) != len(string2):  
 print("Não são anagramas")  
 return #Encerra a função se o tamanho for diferente  
  
 #Ordena as letras de ambas as strings e compara  
 #Se as strings ordenadas forem iguais, são anagramas  
 if sorted(string1) == sorted(string2):  
 print("São anagramas")  
 else:  
 #Se as strings ordenadas forem diferentes, não são anagramas  
 print("Não são anagramas")  
  
#Chama a função para o usuário testar se duas strings são anagramas  
sao\_anagramas()

***Raciocínio do código.***

Entrada do usuário: São coletadas duas strings através da função input().

Espaços são removidos com replace(" ", "") e ambas as strings são convertidas para minúsculas usando lower(), para garantir que a comparação não seja afetada por diferenças de formatação.

Se as strings normalizadas não tiverem o mesmo comprimento, o programa já conclui que não são anagramas.

Comparação ordenada: As strings são convertidas em listas de caracteres e ordenadas com sorted(). Se as listas ordenadas forem idênticas, significa que as strings possuem os mesmos caracteres na mesma quantidade, e, portanto, são anagramas.