MARATONA VOUCHER 2024

Questões de baixa complexidade ---- Cada questão vale 30 pontos.

1) Faça um algoritmo para imprimir "Hello, World!" na tela.
2) Crie uma variável x com o valor 10 e imprimir o valor de x.
3) Calcular a soma de dois números inteiros a e b, onde a = 5 e b = 7.
4) Dado um número n, imprimir "Par" se n for par, ou "Ímpar" se n for ímpar.
5) Criar uma lista de números de 1 a 5 e imprimir o terceiro elemento da lista.
6) Imprimir os números de 1 a 10 usando um laço for.
7) Verificar se a palavra "Python" está contida na string "Eu estou aprendendo Python".
8) Criar um dicionário com chaves nome, idade e cidade, e imprimir o valor associado à chave idade.
9) Calcular a média de três notas n1, n2 e n3, e imprimir o resultado.
10) Solicitar ao usuário para digitar seu nome e depois imprimir "Olá, [nome]!"
11) Calcular o quadrado de um número n.
12) Verificar se um número x é maior que 10.
13) Concatenar duas strings str1 e str2.
14) Criar uma lista vazia e adicionar os números de 1 a 3.
15) Remover o último elemento de uma lista numeros.
16) Multiplicar todos os elementos de uma lista [2, 4, 6] por 2.
17) Verificar se um número y está na lista [1, 2, 3, 4, 5].
18) Criar uma função que retorna o dobro de um número.
19) Criar uma função que recebe um nome como parâmetro e imprime "Olá, [nome]!"
20) Converter a string "123" em um número inteiro.
21) Criar uma tupla com os elementos a, b, c e imprimir o segundo elemento.
22) Solicitar ao usuário para digitar dois números e imprimir a soma deles.
23) Imprimir os números pares de 1 a 10.
24) Verificar se a lista [1, 2, 3] está vazia.
25) Criar uma string e converter todos os caracteres para maiúsculas.
26) Criar uma lista com os quadrados dos números de 1 a 5.
27) Criar um dicionário que mapeia números para seus quadrados, de 1 a 3.
28) Solicitar ao usuário para digitar um número e verificar se ele é positivo.
29) Criar uma função que verifica se um número é par.
30) Calcular o fatorial de um número n.

Média Complexidade **---- Cada questão vale 100 pontos.**

31) Implemente uma função que recebe uma lista de números e retorna uma nova lista contendo apenas os números pares.
Exemplo:
print(filtrar_pares([1, 2, 3, 4, 5, 6]))
Saída: [2, 4, 6]

32) Escreva uma função que verifica se uma string é um palíndromo (lê-se da mesma forma de trás para frente).
Exemplo:
print(eh_palindromo("A base do teto desaba"))
Saída: True

33) Crie uma função que recebe uma lista de números e retorna a soma dos números que estão em índices pares.
Exemplo:
print(soma_indices_pares([1, 2, 3, 4, 5, 6]))
Saída: 9

34) Escreva uma função que recebe duas listas e retorna uma nova lista que contém apenas os elementos comuns entre elas, sem repetições.
Exemplo:
print(intersecao_listas([1, 2, 3, 4], [3, 4, 5, 6]))
Saída: [3, 4]

35) Implemente uma função que calcula o número de vogais em uma string.
Exemplo:
print(contar_vogais("Programação"))
Saída: 5

36) Crie uma função que receba uma lista de números e retorne o segundo maior número.
Exemplo:
print(segundo_maior([10, 20, 4, 45, 99, 99]))
Saída: 45

37) Escreva uma função que verifica se uma sequência de números está ordenada de forma crescente.
Exemplo:
print(esta_ordenada([1, 2, 3, 4, 5]))
Saída: True
print(esta_ordenada([5, 3, 2, 1]))

Saída: False

38) Implemente uma função que inverte as palavras de uma string mantendo a ordem das palavras.
Exemplo:
print(inverter_palavras("Olá mundo"))
Saída: "álO odnum"

39) Crie uma função que receba um número inteiro e retorne a soma dos seus dígitos.
Exemplo:
print(soma_digitos(12345))
Saída: 15

40) Escreva uma função que gere todos os números primos até um dado número n.
def gerar_primos(n):
Exemplo:
print(gerar_primos(20))
Saída: [2, 3, 5, 7, 11, 13, 17, 19]

41) Implemente uma função que conte quantas vezes cada caractere aparece em uma string.
Exemplo:
print(contar_caracteres("abacaxi"))
Saída: {'a': 3, 'b': 1, 'c': 1, 'x': 1, 'i': 1}

42) Crie uma função que receba uma lista de strings e retorne a mais longa.
Exemplo:
print(string_mais_longa(["casa", "carro", "avião"]))
Saída: "carro"

43) Escreva uma função que receba uma matriz (lista de listas) e retorne a sua transposta.
Exemplo:
```
matriz = [
   [1, 2, 3],
   [4, 5, 6]
]
print(transposta(matriz))
Saída: [[1, 4], [2, 5], [3, 6]]
```

44) Implemente uma função que recebe uma string e remove todos os caracteres duplicados, mantendo a ordem original.

Exemplo:
print(remover_duplicados("banana"))
Saída: "ban"

45) Crie uma função que converta um número romano para um número inteiro.
Exemplo:
print(romano_para_inteiro("XIV"))
Saída: 14

46) Escreva uma função que retorne a mediana de uma lista de números.
Exemplo:
print(mediana([1, 2, 3, 4, 5]))
Saída: 3
print(mediana([1, 2, 3, 4]))
Saída: 2.5

47) Implemente uma função que receba uma lista de números e retorne a soma acumulada (lista onde cada elemento é a soma dos elementos anteriores e o atual).
Exemplo:
print(soma_acumulada([1, 2, 3, 4]))
Saída: [1, 3, 6, 10]

48) Crie uma função que calcula o MMC (mínimo múltiplo comum) entre dois números.
Exemplo:
print(mmc(12, 18))
Saída: 36

49) Crie uma função que determine se dois números são amigos. Dois números são considerados amigos se a soma dos divisores próprios de um número é igual ao outro número e vice-versa.
Exemplo:
print(sao_amigos(220, 284))
Saída: True
print(sao_amigos(1184, 1210))
Saída: True
print(sao_amigos(30, 42))
Saída: False

Alta Complexidade **---- Cada questão vale 300 pontos.**

50) Implemente uma função que calcula o produto de dois números inteiros sem utilizar o operador de multiplicação (*). Utilize apenas adições e subtrações.
Exemplo:
print(multiplicacao(6, 7))
Saída: 42
print(multiplicacao(-6, 7))
Saída: -42
print(multiplicacao(6, -7))
Saída: -42
print(multiplicacao(-6, -7))
Saída: 42

51) Escreva uma função que calcule a soma dos números ímpares de 1 até um número n dado.
Exemplo:
print(soma_impares(10))
Saída: 25

52) Crie uma função que verifique se uma string é uma anagrama de outra string.
Exemplo:
print(eh_anagrama("amor", "roma"))
Saída: True
print(eh_anagrama("python", "java"))
Saída: False

53) Implemente uma função que determine se um número é um número perfeito. Um número perfeito é um número que é igual à soma de seus divisores próprios (excluindo ele mesmo).
Exemplo:
print(eh_numero_perfeito(6))
Saída: True
print(eh_numero_perfeito(28))
Saída: True
print(eh_numero_perfeito(12))
Saída: False

54) Crie uma função que gere o fatorial de um número inteiro n usando apenas um loop for.
Exemplo:
print(fatorial(5))
Saída: 120

print(fatorial(0))
Saída: 1

55) Escreva uma função que encontre o menor número em uma lista de números.
Exemplo:
print(menor_numero([3, 5, 7, 2, 8]))
Saída: 2
print(menor_numero([10, 20, 30]))
Saída: 10

56) Implemente uma função que encontre a maior sequência de caracteres consecutivos iguais em uma string.
Exemplo:
print(maior_sequencia("aabbccdddde"))
Saída: "dddd"
print(maior_sequencia("aaabbbcccc"))
Saída: "cccc"

57) Crie uma função que verifique se uma lista está ordenada em ordem crescente.
Exemplo:
print(esta_ordenada([1, 2, 3, 4, 5]))
Saída: True
print(esta_ordenada([1, 3, 2, 4]))
Saída: False

58) Escreva uma função que encontre todos os números primos até um número n sem utilizar a biblioteca math.
Exemplo:
print(primos_ate_n(20))
Saída: [2, 3, 5, 7, 11, 13, 17, 19]

59) Implemente uma função que calcule a soma dos quadrados dos números em uma lista.
Exemplo:
print(soma_quadrados([1, 2, 3]))
Saída: 14
print(soma_quadrados([4, 5, 6]))
Saída: 77

60) Crie uma função que receba uma lista e uma posição, e retorne o elemento na posição indicada, mas com todos os elementos após essa posição removidos.
Exemplo:
print(cortar_lista([1, 2, 3, 4, 5], 2))
Saída: [1, 2, 3]

print(cortar_lista([10, 20, 30, 40], 1))
Saída: [10, 20]

Boa sorte!!!