



---

## APOSTILA

---

# 120 EXERCÍCIOS RESOLVIDOS EM PYTHON



Alexandre Sauer

@prof.alesauer

- Especialista em Python  
- Mestre em Sistemas da Informação

# COMO USAR O MATERIAL

Esta apostila foi criada para ser mais do que apenas um repositório de exercícios resolvidos em Python; ela é uma ferramenta prática e poderosa para impulsionar suas habilidades de programação. Independentemente do seu nível de experiência, este material foi cuidadosamente projetado para guiá-lo através de conceitos-chave e práticas essenciais em Python, enquanto você trabalha em exercícios que simulam desafios reais.

## Dicas para Aproveitar ao Máximo o Material:

1. Leia o Enunciado com Atenção: o exercício começa com um enunciado detalhado que descreve o problema a ser resolvido. Antes de correr para a solução, leia o enunciado com calma, garantindo que você entenda completamente o que é solicitado. Muitas vezes, os detalhes são fundamentais para encontrar a solução mais eficiente.
2. Tente Resolver por Conta Própria: Mesmo que as soluções estejam disponíveis, é crucial que você tente resolver os exercícios por conta própria primeiro. Este processo de tentativa e erro é essencial para o aprendizado, permitindo que você descubra onde estão suas dificuldades e como superá-las.
3. Compare Sua Solução com a Oferecida: Depois de tentar resolver o exercício, compare sua solução com a que está no ebook. Veja quais abordagens foram semelhantes e onde você poderia melhorar. Isso não é apenas sobre certo ou errado; é sobre explorar diferentes maneiras de abordar o mesmo problema.



# COMO USAR O MATERIAL

4. Entenda Cada Linha de Código: Não basta apenas copiar e colar o código resolvido. Para realmente aprender, passe por cada linha de código, entendendo o que cada comando faz e como ele contribui para a solução do problema. Isso solidificará seus conhecimentos e melhorará suas habilidades de programação.

5. Experimente Modificar as Soluções: Pegue a solução oferecida e faça modificações. Tente otimizá-la, adicione funcionalidades ou explore outras formas de resolver o mesmo problema. Isso ajudará a desenvolver sua criatividade e flexibilidade como programador.

6. Pratique Regularmente: A chave para dominar qualquer habilidade é a prática constante. Não se limite a apenas resolver os exercícios uma vez; volte a eles depois de alguns dias ou semanas para reforçar o que você aprendeu e para ver o quanto você evoluiu.

7. Busque Entender o Contexto dos Exercícios: Cada exercício foi desenhado com um propósito, seja ele reforçar um conceito específico ou desafiá-lo a pensar de forma diferente. Procure entender o porquê de cada exercício e como ele se conecta com o uso de Python em situações do mundo real.

## Caminho para o Domínio

Este ebook é uma excelente ferramenta de estudo, mas seu aprendizado vai além dessas páginas. Use os conceitos aprendidos aqui como base para explorar projetos mais complexos e desafiadores. A jornada no mundo da programação é contínua, e cada exercício resolvido é um passo a mais em direção ao domínio do Python.



# SOBRE O PROFESSOR

Imagine-se entrando em uma sala de aula repleta de futuros profissionais de TI, todos ansiosos para absorver o conhecimento que os preparará para os desafios do mercado. Essa é a minha paixão: ensinar e transformar vidas através da educação em tecnologia.

Sou Alexandre Sauer, Mestre em Sistemas de Informação, pós-graduado em Engenharia de Telecomunicações e especialista em Engenharia de Sistemas.

Minha jornada acadêmica começou com um bacharelado em Sistemas de Informação e uma graduação em Tecnologia em Informática, complementada por um curso técnico em Eletrônica.

Minha carreira profissional é marcada por uma vasta experiência na área de Ciência da Computação, com ênfase em sistemas de computação, redes de computadores, desenvolvimento web e para dispositivos móveis.

Atualmente, atuo como Analista de Sistemas e como Professor. Em 2024, decidi expandir os horizontes criando o curso digital Simplifica Python, que tem como objetivo principal te levar do Zero ao Expert em Programação, usando a linguagem Python.



## Programa 1

**Enunciado:** Escreva um programa em Python para imprimir "Hello Python".

```
1. print("Olá Python")
2.
```

## Programa 2

**Enunciado:** Escreva um programa em Python para realizar operações aritméticas de adição e divisão.

```
1. # Adição
2. num1 = float(input("Digite o primeiro número para a adição: "))
3. num2 = float(input("Digite o segundo número para a adição: "))
4. soma_resultado = num1 + num2
5. print(f"Soma: {num1} + {num2} = {soma_resultado}")
6.
7. # Divisão
8. num3 = float(input("Digite o dividendo para a divisão: "))
9. num4 = float(input("Digite o divisor para a divisão: "))
10. if num4 == 0:
11.     print("Erro: Divisão por zero não é permitida.")
12. else:
13.     div_resultado = num3 / num4
14.     print(f"Divisão: {num3} / {num4} = {div_resultado}")
15.
```

## Programa 3

**Enunciado:** Escreva um programa em Python para encontrar a área de um triângulo.

```
1. # Entrada da base e altura do usuário
2. base = float(input("Digite o comprimento da base do triângulo: "))
3. altura = float(input("Digite a altura do triângulo: "))
4.
5. # Calcular a área do triângulo
6. area = 0.5 * base * altura
7.
8. # Exibir o resultado
9. print(f"A área do triângulo é: {area}")
10.
```

## Programa 4

**Enunciado:** Escreva um programa em Python para trocar duas variáveis.

```
1. # Entrada de duas variáveis
2. a = input("Digite o valor da primeira variável (a): ")
3. b = input("Digite o valor da segunda variável (b): ")
4.
5. # Exibir os valores originais
6. print(f"Valores originais: a = {a} b = {b}")
7.
8. # Trocar os valores usando uma variável temporária
9. temp = a
10. a = b
11. b = temp
12.
13. # Exibir os valores trocados
14. print(f"Valores trocados: a = {a} b = {b}")
15.
```

## Programa 5

**Enunciado:** Escreva um programa em Python para gerar um número aleatório.

```
1. import random
2. print(f"Número aleatório: {random.randint(1, 100)}")
3.
```

## Programa 6

**Enunciado:** Escreva um programa em Python para converter quilômetros em milhas.

```
1. kilometros = float(input("Digite a distância em quilômetros: "))
2. # Fator de conversão: 1 quilômetro = 0.621371 milhas
3. fator_conversao = 0.621371
4. milhas = kilometros * fator_conversao
5. print(f"{kilometros} quilômetros é igual a {milhas} milhas")
6.
```

## Programa 7

**Enunciado:** Escreva um programa em Python para converter Celsius em Fahrenheit.

```
1. celsius = float(input("Digite a temperatura em Celsius: "))
2. # Fórmula de conversão: Fahrenheit = (Celsius * 9/5) + 32
3. fahrenheit = (celsius * 9/5) + 32
4. print(f"{celsius} graus Celsius é igual a {fahrenheit} graus Fahrenheit")
5.
```

## Programa 8

**Enunciado:** Escreva um programa em Python para exibir um calendário.

```
1. import calendar
2. ano = int(input("Digite o ano: "))
3. mes = int(input("Digite o mês: "))
4. calendario = calendar.month(ano, mes)
5. print(calendario)
```

## Programa 9

**Enunciado:** Escreva um programa em Python para resolver uma equação quadrática.

```
1. import math
2. # Entrada dos coeficientes
3. a = float(input("Digite o coeficiente a: "))
4. b = float(input("Digite o coeficiente b: "))
5. c = float(input("Digite o coeficiente c: "))
6.
7. # Calcular o discriminante
8. discriminante = b**2 - 4*a*c
9. # Verificar se o discriminante é positivo, negativo ou zero
10. if discriminante > 0:
11.     # Duas raízes reais e distintas
12.     raiz1 = (-b + math.sqrt(discriminante)) / (2*a)
13.     raiz2 = (-b - math.sqrt(discriminante)) / (2*a)
14.     print(f"Raiz 1: {raiz1}")
15.     print(f"Raiz 2: {raiz2}")
16. elif discriminante == 0:
17.     # Uma raiz real (repetida)
18.     raiz = -b / (2*a)
19.     print(f"Raiz: {raiz}")
20. else:
21.     # Raízes complexas
22.     parte_real = -b / (2*a)
23.     parte_imaginaria = math.sqrt(abs(discriminante)) / (2*a)
24.     print(f"Raiz 1: {parte_real} + {parte_imaginaria}i")
25.     print(f"Raiz 2: {parte_real} - {parte_imaginaria}i")
26.
```

## Programa 10

**Enunciado:** Escreva um programa em Python para trocar duas variáveis sem uma variável temporária.

```
1. a = 5
2. b = 10
3.
4. # Trocar sem uma variável temporária
5. a, b = b, a
6.
7. print("Após a troca:")
8. print("a =", a)
9. print("b =", b)
10.
```

## Programa 11

**Enunciado:** Escreva um programa em Python para verificar se um número é positivo, negativo ou zero.

```
1. num = float(input("Digite um número: "))
2. if num > 0:
3.     print("Número positivo")
4. elif num == 0:
5.     print("Zero")
6. else:
7.     print("Número negativo")
8.
```

## Programa 12

**Enunciado:** Escreva um programa em Python para verificar se um número é par ou ímpar.

```
1. num = int(input("Digite um número: "))
2. if num % 2 == 0:
3.     print("Este é um número par")
4. else:
5.     print("Este é um número ímpar")
6.
```

## Programa 13

**Enunciado:** Escreva um programa em Python para verificar se um ano é bissexto.

```

1. ano = int(input("Digite um ano: "))
2. # Ano dividido por 100 significa ano do século (terminando com 00)
3. # Ano do século dividido por 400 é ano bissexto
4. if (ano % 400 == 0) and (ano % 100 == 0):
5.     print("{} é um ano bissexto".format(ano))
6. # Não dividido por 100 significa que não é ano do século
7. # Ano dividido por 4 é ano bissexto
8. elif (ano % 4 == 0) and (ano % 100 != 0):
9.     print("{} é um ano bissexto".format(ano))
10. # Se não for dividido por ambos 400 (ano do século) e 4 (não ano do século)
11. # O ano não é bissexto
12. else:
13.     print("{} não é um ano bissexto".format(ano))
14.

```

## Programa 14

**Enunciado:** Escreva um programa em Python para verificar se um número é primo.

```

1. num = int(input("Digite um número: "))
2. # Definir uma variável de sinalização
3. flag = False
4. if num == 1:
5.     print(f"{num} não é um número primo")
6. elif num > 1:
7.     # Verificar fatores
8.     for i in range(2, num):
9.         if (num % i) == 0:
10.             flag = True # Se o fator for encontrado, definir a sinalização como True
11.             # Sair do loop
12.             break
13. # Verificar se a sinalização é True
14. if flag:
15.     print(f"{num} não é um número primo")
16. else:
17.     print(f"{num} é um número primo")
18.

```

## Programa 15

**Enunciado:** Escreva um programa em Python para imprimir todos os números primos em um intervalo de 1-10.

```
1. # Programa em Python para exibir todos os números primos dentro de um intervalo
2. limite_inferior = 1
3. limite_superior = 10
4. print("Números primos entre", limite_inferior, "e", limite_superior, "são:")
5. for num in range(limite_inferior, limite_superior + 1):
6.     # Todos os números primos são maiores que 1
7.     if num > 1:
8.         for i in range(2, num):
9.             if (num % i) == 0:
10.                 break
11.             else:
12.                 print(num)
13.
```

## Programa 16

**Enunciado:** Escreva um programa em Python para encontrar o fatorial de um número.

```
1. num = int(input("Digite um número: "))
2. fatorial = 1
3.
4. if num < 0:
5.     print("Fatorial não existe para números negativos")
6. elif num == 0:
7.     print("O fatorial de 0 é 1")
8. else:
9.     for i in range(1, num + 1):
10.         fatorial = fatorial * i
11.     print(f"O fatorial de {num} é {fatorial}")
12.
```

## Programa 17

**Enunciado:** Escreva um programa em Python para exibir a tabela de multiplicação.

```
1. num = int(input("Exibir a tabela de multiplicação de: "))
2.
3. for i in range(1, 11):
4.     print(f"{num} X {i} = {num * i}")
5.
```

## Programa 18

**Enunciado:** Escreva um programa em Python para imprimir a sequência de Fibonacci.

```
1. nterms = int(input("Quantos termos? "))
2. # Primeiros dois termos
3. n1, n2 = 0, 1
4. count = 0
5. # Verificar se o número de termos é válido
6. if nterms <= 0:
7.     print("Por favor, insira um número inteiro positivo")
8. elif nterms == 1:
9.     print(f"Sequência de Fibonacci até {nterms} termo:")
10.    print(n1)
11. else:
12.     print("Sequência de Fibonacci:")
13.     while count < nterms:
14.         print(n1)
15.         nth = n1 + n2
16.         # Atualizar valores
17.         n1 = n2
18.         n2 = nth
19.         count += 1
20.
```

## Programa 19

**Enunciado:** Escreva um programa em Python para verificar se um número é um Número de Armstrong.

```

1. num = int(input("Digite um número: "))
2.
3. # Calcular o número de dígitos em num
4. num_str = str(num)
5. num_digitos = len(num_str)
6.
7. # Inicializar variáveis
8. soma_potencias = 0
9. num_temp = num
10.
11. # Calcular a soma dos dígitos elevados à potência de num_digitos
12. while num_temp > 0:
13.     digito = num_temp % 10
14.     soma_potencias += digito ** num_digitos
15.     num_temp //= 10
16.
17. # Verificar se é um número de Armstrong
18. if soma_potencias == num:
19.     print(f"{num} é um número de Armstrong.")
20. else:
21.     print(f"{num} não é um número de Armstrong.")
22.

```

## Programa 20

**Enunciado:** Escreva um programa em Python para encontrar os Números de Armstrong em um intervalo.

```

1. # Entrada do intervalo pelo usuário
2. limite_inferior = int(input("Digite o limite inferior do intervalo: "))
3. limite_superior = int(input("Digite o limite superior do intervalo: "))
4.
5. # Iterar pelos números no intervalo
6. for num in range(limite_inferior, limite_superior + 1):
7.     # Encontrar o número de dígitos em num
8.     ordem = len(str(num))
9.     num_temp = num
10.    soma = 0
11.
12.    while num_temp > 0:
13.        digito = num_temp % 10
14.        soma += digito ** ordem
15.        num_temp //= 10
16.
17.    # Verificar se num é um número de Armstrong
18.    if num == soma:
19.        print(num)
20.

```

## Programa 21

**Enunciado:** Escreva um programa em Python para encontrar a soma dos números naturais.

```
1. limite = int(input("Digite o limite: "))
2. # Inicializar a soma
3. soma = 0
4. # Usar um loop for para calcular a soma dos números naturais
5. for i in range(1, limite + 1):
6.     soma += i
7. # Imprimir a soma
8. print("A soma dos números naturais até", limite, "é:", soma)
9.
```

## Programa 22

**Enunciado:** Escreva um programa em Python para encontrar o MMC.

```
1. def calcular_mmc(x, y):
2.     if x > y:
3.         maior = x
4.     else:
5.         maior = y
6.     while(True):
7.         if((maior % x == 0) and (maior % y == 0)):
8.             mmc = maior
9.             break
10.            maior += 1
11.    return mmc
12. num1 = int(input('Digite o primeiro número: '))
13. num2 = int(input('Digite o segundo número: '))
14. print("O MMC é", calcular_mmc(num1, num2))
15.
```

## Programa 23

**Enunciado:** Escreva um programa em Python para encontrar o MDC.

```

1. def calcular_mdc(x, y):
2.     if x > y:
3.         menor = y
4.     else:
5.         menor = x
6.
7.     for i in range(1, menor + 1):
8.         if((x % i == 0) and (y % i == 0)):
9.             mdc = i
10.
11.    return mdc
12.
13. num1 = int(input('Digite o primeiro número: '))
14. num2 = int(input('Digite o segundo número: '))
15. print("O MDC é", calcular_mdc(num1, num2))
16.

```

## Programa 24

**Enunciado:** Escreva um programa em Python para converter decimal para binário, octal e hexadecimal.

```

1. num_dec = int(input('Digite um número decimal: '))
2. print("O valor decimal de", num_dec, "é:")
3. print(bin(num_dec), "em binário.")
4. print(oct(num_dec), "em octal.")
5. print(hex(num_dec), "em hexadecimal.")
6.

```

## Programa 25

**Enunciado:** Escreva um programa em Python para encontrar o valor ASCII de um caractere.

```

1. char = str(input("Digite um caractere: "))
2. print("O valor ASCII de '" + char + "' é", ord(char))
3.

```

## Programa 26

**Enunciado:** Escreva um programa em Python para criar uma calculadora simples com 4 operações matemáticas básicas.

```

1. # Esta função adiciona dois números
2. def adicionar(x, y):
3.     return x + y
4.
5. # Esta função subtrai dois números
6. def subtrair(x, y):
7.     return x - y
8.
9. # Esta função multiplica dois números
10. def multiplicar(x, y):
11.     return x * y
12.
13. # Esta função divide dois números
14. def dividir(x, y):
15.     return x / y
16.
17. print("Selecione a operação.")
18. print("1. Adicionar")
19. print("2. Subtrair")
20. print("3. Multiplicar")
21. print("4. Dividir")
22.
23. while True:
24.     # Receber a entrada do usuário
25.     escolha = input("Digite a escolha (1/2/3/4): ")
26.
27.     # Verificar se a escolha é uma das quatro opções
28.     if escolha in ('1', '2', '3', '4'):
29.         try:
30.             num1 = float(input("Digite o primeiro número: "))
31.             num2 = float(input("Digite o segundo número: "))
32.         except ValueError:
33.             print("Entrada inválida. Por favor, insira um número.")
34.             continue
35.
36.         if escolha == '1':
37.             print(num1, "+", num2, "=", adicionar(num1, num2))
38.
39.         elif escolha == '2':
40.             print(num1, "-", num2, "=", subtrair(num1, num2))
41.
42.         elif escolha == '3':
43.             print(num1, "*", num2, "=", multiplicar(num1, num2))
44.
45.         elif escolha == '4':
46.             print(num1, "/", num2, "=", dividir(num1, num2))
47.
48.         # Verificar se o usuário quer outra calculação
49.         proxima_calculacao = input("Quer fazer outra calculação? (sim/não): ")
50.         if proxima_calculacao.lower() != "sim":
51.             break
52.         else:
53.             print("Entrada inválida")
54.

```

## Programa 27

**Enunciado:** Escreva um programa em Python para exibir a sequência de Fibonacci usando recursão.

```

1. # Programa em Python para exibir a sequência de Fibonacci
2. def recur_fibo(n):
3.     if n <= 1:
4.         return n
5.     else:
6.         return(recur_fibo(n-1) + recur_fibo(n-2))
7. nterms = int(input("Digite o número de termos (maior que 0): "))
8.
9. # Verificar se o número de termos é válido
10. if nterms <= 0:
11.     print("Por favor, insira um número inteiro positivo")
12. else:
13.     print("Sequência de Fibonacci:")
14.     for i in range(nterms):
15.         print(recur_fibo(i))
16.

```

## Programa 28

**Enunciado:** Escreva um programa em Python para encontrar o fatorial de um número usando recursão.

```

1. # Fatorial de um número usando recursão
2. def recur_fatorial(n):
3.     if n == 1:
4.         return n
5.     else:
6.         return n * recur_fatorial(n - 1)
7.
8. num = int(input("Digite o número: "))
9.
10. # Verificar se o número é negativo
11. if num < 0:
12.     print("Desculpe, fatorial não existe para números negativos")
13. elif num == 0:
14.     print("O fatorial de 0 é 1")
15. else:
16.     print("O fatorial de", num, "é", recur_fatorial(num))
17.

```

## Programa 29

**Enunciado:** Escreva um programa em Python para calcular seu Índice de Massa Corporal (IMC).

```
1. def calcular_imc(altura, peso):
2.     return round((peso / altura ** 2), 2)
3.
4. altura = float(input("Digite sua altura em metros: "))
5. peso = float(input("Digite seu peso em kg: "))
6.
7. print("Bem-vindo ao calculador de IMC.")
8. imc = calcular_imc(altura, peso)
9. print("Seu IMC é:", imc)
10.
11. if imc <= 18.5:
12.     print("Você está abaixo do peso.")
13. elif 18.5 < imc <= 24.9:
14.     print("Seu peso é normal.")
15. elif 25 < imc <= 29.9:
16.     print("Você está com sobrepeso.")
17. else:
18.     print("Você está obeso.")
19.
```

## Programa 30

**Enunciado:** Escreva um programa em Python para calcular o logaritmo natural de qualquer número.

```
1. import math
2. num = float(input("Digite um número: "))
3. if num <= 0:
4.     print("Por favor, insira um número positivo.")
5. else:
6.     # Calcular o logaritmo natural (base e) do número
7.     resultado = math.log(num)
8.     print(f"O logaritmo natural de {num} é: {resultado}")
9.
```

## Programa 31

**Enunciado:** Escreva um programa em Python para a soma dos cubos dos primeiros n números naturais.

```

1. def soma_dos_cubos_dos_numeros_naturais(n):
2.     if n <= 0:
3.         return 0
4.     else:
5.         total = sum([i**3 for i in range(1, n + 1)])
6.     return total
7.
8. # Entrada do número de números naturais
9. n = int(input("Digite o valor de n: "))
10.
11. if n <= 0:
12.     print("Por favor, insira um número inteiro positivo.")
13. else:
14.     resultado = soma_dos_cubos_dos_numeros_naturais(n)
15.     print(f"A soma dos cubos dos primeiros {n} números naturais é: {resultado}")
16.

```

## Programa 32

**Enunciado:** Escreva um programa em Python para encontrar a soma de um array.

```

1. # Encontrar a soma do array usando sum()
2. array = [1, 2, 3]
3. soma = sum(array)
4. print('A soma do array é', soma)
5.
6. # Função para encontrar a soma dos elementos em um array
7. def soma_do_array(array):
8.     total = 0
9.     for elemento in array:
10.         total += elemento
11.     return total
12.
13. # Exemplo de uso:
14. array = [1, 2, 3]
15. resultado = soma_do_array(array)
16. print("A soma do array é:", resultado)
17.

```

## Programa 33

**Enunciado:** Escreva um programa em Python para encontrar o maior elemento em um array.

```

1. def encontrar_maior_elemento(array):
2.     if not array:
3.         return "O array está vazio"
4.
5.     maior_elemento = array[0]
6.     for elemento em array:
7.         if elemento > maior_elemento:
8.             maior_elemento = elemento
9.
10.    return maior_elemento
11.
12. # Exemplo de uso:
13. meu_array = [10, 20, 30, 99]
14. resultado = encontrar_maior_elemento(meu_array)
15. print(f"O maior elemento no array é: {resultado}")
16.

```

## Programa 34

**Enunciado:** Escreva um programa em Python para rotacionar um array.

```

1. def rotacionar_array(array, d):
2.     n = len(array)
3.
4.     # Verificar se 'd' é válido, deve estar dentro do intervalo do array
5.     se d < 0 ou d >= n:
6.         return "Valor de rotação inválido"
7.
8.     # Criar um novo array para armazenar os elementos rotacionados.
9.     array_rotacionado = [0] * n
10.
11.    # Realizar a rotação.
12.    para eu em gama (n):
13.        array_rotacionado [i] = array [(i + d) % n]
14.
15.    return array_rotacionado
16.
17. # Array de entrada
18. array = [1, 2, 3, 4, 5]
19.
20. # Número de posições para rotacionar
21. d = 2
22.
23. # Chame a função rotacionar_array
24. resultado = rotacionar_array(array, d)
25.
26. # Imprima o array rotacionado
27. print("Array original:", array)
28. print("Array rotacionado:", resultado)
29.

```

## Programa 35

**Enunciado:** Escreva um programa em Python para dividir o array e adicionar a primeira parte ao final.

```

1. def dividir_e_adicionar(array, k):
2.     if k <= 0 ou k >= len(array):
3.         return array
4.
5.     # Dividir o array em duas partes
6.     primeira_parte = array[:k]
7.     segunda_parte = array[k:]
8.
9.     # Adicionar a primeira parte ao final da segunda parte
10.    resultado = segunda_parte + primeira_parte
11.
12.    return resultado
13.
14. # Testar a função
15. array = [1, 2, 3, 4, 5]
16. k = 3
17. resultado = dividir_e_adicionar(array, k)
18. print("Array original:", array)
19. print("Array após dividir e adicionar:", resultado)
20.

```

## Programa 36

**Enunciado:** Escreva um programa em Python para verificar se um array é monótono. Um array monótono é aquele que é totalmente não crescente ou não decrescente.

```

1. def eh_monotono(array):
2.     crescente = decrescente = True
3.
4.     for i em gama (1, len(array)):
5.         if array[i] > array[i - 1]:
6.             decrescente = False
7.         elif array[i] < array[i - 1]:
8.             crescente = False
9.
10.    return crescente ou decrescente
11.
12. # Testar a função
13. array1 = [1, 2, 2, 3] # Monótono (não decrescente)
14. array2 = [3, 2, 1] # Monótono (não crescente)
15. array3 = [1, 3, 2, 4] # Não monótono
16.
17. print("array1 é monótono:", eh_monotono(array1))
18. print("array2 é monótono:", eh_monotono(array2))
19. print("array3 é monótono:", eh_monotono(array3))
20.

```

## Programa 37

**Enunciado:** Escreva um programa em Python para adicionar duas matrizes.

```
1. # Função para adicionar duas matrizes
2. def adicionar_matrizes(mat1, mat2):
3.     # Verificar se as matrizes têm as mesmas dimensões
4.     if len(mat1) != len(mat2) ou len(mat1[0]) != len(mat2[0]):
5.         return "As matrizes devem ter as mesmas dimensões para adição"
6.
7.     # Inicializar uma matriz de resultado vazia com as mesmas dimensões
8.     resultado = []
9.     for i em gama (len(mat1)):
10.         linha = []
11.         for j em gama (len(mat1[0])):
12.             linha.append(mat1[i][j] + mat2[i][j])
13.         resultado.append(linha)
14.
15.     return resultado
16.
17. # Matrizes de entrada
18. matriz1 = [
19.     [1, 2, 3],
20.     [4, 5, 6],
21.     [7, 8, 9]
22. ]
23.
24. matriz2 = [
25.     [9, 8, 7],
26.     [6, 5, 4],
27.     [3, 2, 1]
28. ]
29.
30. # Chame a função adicionar_matrizes
31. resultado_matriz = adicionar_matrizes(matriz1, matriz2)
32.
33. # Exibir o resultado
34. if isinstance(resultado_matriz, str):
35.     print(resultado_matriz)
36. else:
37.     print("Soma das matrizes:")
38.     para linha em resultado_matriz:
39.         print(linha)
40.
```

## Programa 38

**Enunciado:** Escreva um programa em Python para multiplicar duas matrizes.

```
1. # Função para multiplicar duas matrizes
2. def multiplicar_matrizes(mat1, mat2):
3.     # Determinar as dimensões das matrizes de entrada
4.     linhas1 = len(mat1)
5.     colunas1 = len(mat1[0])
6.     linhas2 = len(mat2)
7.     colunas2 = len(mat2[0])
8.
9.     # Verificar se a multiplicação é possível
10.    if colunas1 != linhas2:
11.        return "Multiplicação de matrizes não é possível. Número de colunas na
primeira matriz deve ser igual ao número de linhas na segunda matriz."
12.
13.    # Inicializar a matriz de resultado com zeros
14.    resultado = [[0 para _ em gama (colunas2)] para _ em gama (linhas1)]
15.
16.    # Realizar a multiplicação das matrizes
17.    para i em gama (linhas1):
18.        para j em gama (colunas2):
19.            para k em gama (colunas1):
20.                resultado[i][j] += mat1[i][k] * mat2[k][j]
21.
22.    return resultado
23.
24. # Matrizes de exemplo
25. matriz1 = [
26.     [1, 2, 3],
27.     [4, 5, 6]
28. ]
29.
30. matriz2 = [
31.     [7, 8],
32.     [9, 10],
33.     [11, 12]
34. ]
35.
36. # Multiplicar as matrizes
37. resultado_matriz = multiplicar_matrizes(matriz1, matriz2)
38.
39. # Exibir o resultado
40. if isinstance(resultado_matriz, str):
41.     print(resultado_matriz)
42. else:
43.     print("Resultado da multiplicação das matrizes:")
44.     para linha em resultado_matriz:
45.         print(linha)
46.
```

## Programa 39

**Enunciado:** Escreva um programa em Python para verificar se um número é um Número de Disarium. Um número Disarium é um número que é igual à soma de seus dígitos, cada um elevado à potência de sua respectiva posição.

Por exemplo, 89 é um número Disarium porque  $8^1 + 9^2 = 8 + 81 = 89$ .

```
1. def eh_numero_disarium(numero):
2.     # Converter o número em string para iterar sobre seus dígitos
3.     num_str = str(numero)
4.
5.     # Calcular a soma dos dígitos elevados às suas respectivas posições
6.     soma_digitos = sum(int(i) ** (index + 1) para index, i em enumerate(num_str))
7.
8.     # Verificar se a soma é igual ao número original
9.     return soma_digitos == numero
10.
11. # Receber um número do usuário
12. try:
13.     num = int(input("Digite um número: "))
14.
15.     # Verificar se é um número Disarium
16.     se eh_numero_disarium(num):
17.         print(f"{num} é um número Disarium.")
18.     senão:
19.         print(f"{num} não é um número Disarium.")
20. exceto ValueError:
21.     print("Entrada inválida. Por favor, insira um número válido.")
22.
```

## Programa 40

**Enunciado:** Escreva um programa em Python para encontrar os N maiores elementos de uma lista.

```
1. def encontrar_n_maiores_elementos(lista, n):
2.     # Ordenar a lista em ordem decrescente
3.     lista_ordenada = sorted(lista, reverse=True)
4.
5.     # Obter os primeiros N elementos
6.     maiores_elementos = lista_ordenada[:n]
7.
8.     return maiores_elementos
9.
10. # Lista de números de exemplo
11. numeros = [30, 10, 45, 5, 20, 50, 15, 3, 345, 54, 67, 87, 98, 100, 34]
12.
13. # Número de maiores elementos a encontrar
14. N = int(input("N = "))
15.
16. # Encontrar os N maiores elementos da lista
17. resultado = encontrar_n_maiores_elementos(numeros, N)
18.
19. # Imprimir os N maiores elementos
20. print(f"Os {N} maiores elementos na lista são:", resultado)
21.
```

## Programa 41

**Enunciado:** Escreva um programa em Python para imprimir números pares em uma lista.

```
1. # Lista de números de exemplo
2. numeros = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
3.
4. # Usar uma compreensão de lista para filtrar números pares
5. numeros_pares = [num para num em numeros se num % 2 == 0]
6.
7. # Imprimir os números pares
8. print("Números pares na lista:", numeros_pares)
9.
```

## Programa 42

**Enunciado:** Escreva um programa em Python para imprimir números ímpares em uma lista.

```
1. # Lista de números de exemplo
2. numeros = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
3.
4. # Usar uma compreensão de lista para filtrar números pares
5. numeros_pares = [num para num em numeros se num % 2 == 0]
6.
7. # Imprimir os números pares
8. print("Números pares na lista:", numeros_pares)
9.
```

## Programa 43

**Enunciado:** Escreva um programa em Python para remover listas vazias de uma lista de listas.

```
1. # Lista de exemplo contendo listas
2. lista_de_listas = [[1, 2, 3], [], [4, 5], [], [6, 7, 8], []]
3.
4. # Usar uma compreensão de lista para remover listas vazias
5. lista_filtrada = [lista for lista in lista_de_listas if lista]
6.
7. # Imprimir a lista filtrada
8. print("Lista após remover listas vazias:", lista_filtrada)
9.
```

## Programa 44

**Enunciado:** Escreva um programa em Python para clonar ou copiar uma lista.

```
1. # 1. Usando o operador de fatia
2. lista_original = [1, 2, 3, 4, 5]
3. lista_clonada = lista_original[:]
4. print(lista_clonada)
5.
6. # 2. Usando o construtor list()
7. lista_original = [1, 2, 3, 4, 5]
8. lista_clonada = list(lista_original)
9. print(lista_clonada)
10.
11. # 3. Usando compreensão de lista
12. lista_original = [1, 2, 3, 4, 5]
13. lista_clonada = [item for item in lista_original]
14. print(lista_clonada)
15.
```

## Programa 45

**Enunciado:** Escreva um programa em Python para contar as ocorrências de um elemento em uma lista.

```
1. def contar_ocorrencias(lista, elemento):
2.     count = lista.count(elemento)
3.     return count
4. # Exemplo de uso:
5. minha_lista = [1, 2, 3, 4, 2, 5, 2, 3, 4, 6, 5]
6. elemento_para_contar = 2
7. ocorrencias = contar_ocorrencias(minha_lista, elemento_para_contar)
8. print(f"O elemento {elemento_para_contar} aparece {ocorrencias} vezes na lista.")
9.
```

## Programa 46

**Enunciado:** Escreva um programa em Python para dividir e juntar uma string.

```

1. # Dividir uma string em uma lista de palavras
2. entrada_str = "Programa em Python para dividir e juntar uma string"
3. lista_palavras = entrada_str.split() # Por padrão, divide em espaços em branco
4.
5. # Juntar a lista de palavras em uma string
6. separador = " " # Especificar o separador entre as palavras
7. saida_str = separador.join(lista_palavras)
8.
9. # Imprimir os resultados
10. print("String original:", entrada_str)
11. print("Lista de palavras divididas:", lista_palavras)
12. print("String juntada:", saida_str)
13.

```

## Programa 47

**Enunciado:** Escreva um programa em Python para encontrar palavras incomuns de duas strings.

```

1. def palavras_incomuns(str1, str2):
2.     # Dividir as strings em palavras e converter em conjuntos
3.     palavras1 = set(str1.split())
4.     palavras2 = set(str2.split())
5.
6.     # Encontrar palavras incomuns fazendo a diferença simétrica dos conjuntos
7.     palavras_incomuns_set = palavras1.symmetric_difference(palavras2)
8.
9.     # Converter o conjunto de palavras incomuns de volta para uma lista
10.    lista_palavras_incomuns = list(palavras_incomuns_set)
11.
12.    return lista_palavras_incomuns
13.
14. # Entradas de duas strings
15. string1 = "Esta é a primeira string"
16. string2 = "Esta é a segunda string"
17.
18. # Encontrar palavras incomuns entre as duas strings
19. incomuns = palavras_incomuns(string1, string2)
20.
21. # Imprimir as palavras incomuns
22. print("Palavras incomuns:", incomuns)
23.

```

## Programa 48

**Enunciado:** Escreva um programa em Python para mesclar dois dicionários.

```
1. # Dicionários de exemplo
2. dicionario1 = {'a': 1, 'b': 2}
3. dicionario2 = {'c': 3, 'd': 4}
4.
5. # Mesclar dicionários usando o operador ** em Python 3.5+
6. dicionario_mesclado = {**dicionario1, **dicionario2}
7. # Imprimir o dicionário mesclado
8. print("Dicionário mesclado:", dicionario_mesclado)
9.
```

## Programa 49

**Enunciado:** Escreva um programa em Python para converter um dicionário em uma lista de tuplas.

```
1. # Dicionário de exemplo
2. meu_dicionario = {'a': 1, 'b': 2, 'c': 3}
3.
4. # Converter dicionário em lista de tuplas
5. lista_tuplas = list(meu_dicionario.items())
6.
7. # Imprimir a lista de tuplas
8. print("Lista de tuplas:", lista_tuplas)
9.
```

## Programa 50

**Enunciado:** Escreva um programa em Python para converter uma lista de tuplas em um dicionário.

```

1. # Lista de tuplas de exemplo
2. lista_tuplas = [('a', 1), ('b', 2), ('c', 3)]
3. # Converter lista de tuplas em dicionário
4. meu_dicionario = dict(lista_tuplas)
5. # Imprimir o dicionário
6. print("Dicionário:", meu_dicionario)
7.

```

## Programa 51

**Enunciado:** Escreva um programa em Python para verificar se um dicionário está vazio.

```

1. # Dicionário de exemplo
2. meu_dicionario = {}
3. # Verificar se o dicionário está vazio
4. if not meu_dicionario:
5.     print("O dicionário está vazio.")
6. else:
7.     print("O dicionário não está vazio.")
8.

```

## Programa 52

**Enunciado:** Escreva um programa em Python para combinar duas listas em um dicionário.

```

1. # Listas de exemplo
2. chaves = ['a', 'b', 'c']
3. valores = [1, 2, 3]
4.
5. # Combinar listas em um dicionário usando a função zip
6. dicionario = dict(zip(chaves, valores))
7.
8. # Imprimir o dicionário
9. print("Dicionário combinado:", dicionario)
10.

```

## Programa 53

**Enunciado:** Escreva um programa em Python para contar a frequência de elementos em uma lista.

```
1. from collections import Counter
2. # Lista de exemplo
3. lista = [1, 2, 2, 3, 3, 3, 4, 4, 4, 4]
4. # Contar a frequência de elementos na lista usando Counter
5. frequencia = Counter(lista)
6. # Imprimir a frequência de elementos
7. print("Frequência de elementos na lista:", frequencia)
8.
```

## Programa 54

**Enunciado:** Escreva um programa em Python para converter uma string em um dicionário.

```
1. def string_para_dicionario(string):
2.     # Inicializar um dicionário vazio
3.     dicionario = {}
4.
5.     # Dividir a string em pares chave-valor
6.     pares = string.split(", ")
7.
8.     # Iterar pelos pares chave-valor e adicioná-los ao dicionário
9.     for par in pares:
10.         chave, valor = par.split(": ")
11.         dicionario[chave] = valor
12.
13.     return dicionario
14.
15. # String de exemplo
16. entrada_string = "a: 1, b: 2, c: 3"
17.
18. # Converter a string em um dicionário
19. resultado = string_para_dicionario(entrada_string)
20.
21. # Imprimir o dicionário
22. print("Dicionário:", resultado)
23.
```

## Programa 55

**Enunciado:** Escreva um programa em Python para combinar dois dicionários adicionando valores para chaves comuns.

```

1. from collections import Counter
2. # Dicionários de exemplo
3. dicionario1 = {'a': 1, 'b': 2, 'c': 3}
4. dicionario2 = {'b': 3, 'c': 4, 'd': 5}
5. # Usar Counter para combinar dicionários
6. resultado = Counter(dicionario1) + Counter(dicionario2)
7. # Converter o resultado de volta para um dicionário
8. dicionario_combinado = dict(resultado)
9. # Imprimir o dicionário combinado
10. print("Dicionário combinado:", dicionario_combinado)
11.

```

## Programa 56

**Enunciado:** Escreva um programa em Python para encontrar a chave com o valor máximo em um dicionário.

```

1. # Dicionário de exemplo
2. meu_dicionario = {'a': 10, 'b': 20, 'c': 30, 'd': 25}
3.
4. # Encontrar a chave com o valor máximo
5. chave_max = max(meu_dicionario, key=meu_dicionario.get)
6.
7. # Imprimir a chave com o valor máximo
8. print("Chave com o valor máximo:", chave_max)
9.

```

## Programa 57

**Enunciado:** Escreva um programa em Python para encontrar a chave com o valor mínimo em um dicionário.

```

1. # Dicionário de exemplo
2. meu_dicionario = {'a': 10, 'b': 20, 'c': 5, 'd': 25}
3.
4. # Encontrar a chave com o valor mínimo
5. chave_min = min(meu_dicionario, key=meu_dicionario.get)
6.
7. # Imprimir a chave com o valor mínimo
8. print("Chave com o valor mínimo:", chave_min)
9.

```

## Programa 58

**Enunciado:** Escreva um programa em Python para remover itens de um dicionário enquanto o percorre.

```

1. # Dicionário de exemplo
2. meu_dicionario = {'a': 1, 'b': 2, 'c': 3, 'd': 4}
3.
4. # Criar uma lista de chaves a serem removidas
5. chaves_para_remover = [chave for chave, valor in meu_dicionario.items() se valor % 2
== 0]
6.
7. # Remover as chaves do dicionário
8. para chave em chaves_para_remover:
9.     del meu_dicionario[chave]
10.
11. # Imprimir o dicionário após a remoção
12. print("Dicionário após a remoção:", meu_dicionario)
13.

```

## Programa 59

**Enunciado:** Escreva um programa em Python para ordenar um dicionário por chave.

```

1. # Dicionário de exemplo
2. meu_dicionario = {'b': 2, 'a': 1, 'd': 4, 'c': 3}
3.
4. # Ordenar o dicionário por chave usando dict() e sorted()
5. dicionario_ordenado = dict(sorted(meu_dicionario.items()))
6.
7. # Imprimir o dicionário ordenado
8. print("Dicionário ordenado por chave:", dicionario_ordenado)
9.

```

## Programa 60

**Enunciado:** Escreva um programa em Python para ordenar um dicionário por valor.

```

1. # Dicionário de exemplo
2. meu_dicionario = {'a': 2, 'b': 1, 'd': 4, 'c': 3}
3. # Ordenar o dicionário por valor usando dict() e sorted()
4. dicionario_ordenado = dict(sorted(meu_dicionario.items(), key=lambda item: item[1]))
5. # Imprimir o dicionário ordenado
6. print("Dicionário ordenado por valor:", dicionario_ordenado)
7.

```

## Programa 61

**Enunciado:** Escreva um programa em Python para concatenar dicionários para criar um novo.

```
1. # Dicionários de exemplo
2. dicionario1 = {'a': 1, 'b': 2}
3. dicionario2 = {'c': 3, 'd': 4}
4. dicionario3 = {'e': 5, 'f': 6}
5.
6. # Concatenar dicionários usando o operador **
7. dicionario_concatenado = {**dicionario1, **dicionario2, **dicionario3}
8.
9. # Imprimir o dicionário concatenado
10. print("Dicionário concatenado:", dicionario_concatenado)
11.
```

## Programa 62

**Enunciado:** Escreva um programa em Python para verificar se uma chave existe em um dicionário.

```
1. # Dicionário de exemplo
2. meu_dicionario = {'a': 1, 'b': 2, 'c': 3}
3.
4. # Chave a ser verificada
5. chave = 'b'
6.
7. # Verificar se a chave existe no dicionário
8. if chave in meu_dicionario:
9.     print(f"A chave '{chave}' existe no dicionário.")
10. else:
11.     print(f"A chave '{chave}' não existe no dicionário.")
12.
```

## Programa 63

**Enunciado:** Escreva um programa em Python para imprimir um dicionário de dicionários.

```

1. # Dicionário de dicionários de exemplo
2. meu_dicionario = {
3.     'dicionario1': {'a': 1, 'b': 2},
4.     'dicionario2': {'c': 3, 'd': 4},
5.     'dicionario3': {'e': 5, 'f': 6}
6. }
7.
8. # Imprimir o dicionário de dicionários
9. for chave_externa, dicionario_interno in meu_dicionario.items():
10.    print(f"{chave_externa}: {dicionario_interno}")
11.

```

## Programa 64

**Enunciado:** Escreva um programa em Python para calcular a média de valores em um dicionário.

```

1. # Dicionário de exemplo
2. meu_dicionario = {'a': 10, 'b': 20, 'c': 30}
3.
4. # Calcular a média dos valores
5. media = sum(meu_dicionario.values()) / len(meu_dicionario)
6.
7. # Imprimir a média
8. print("A média dos valores no dicionário é:", media)
9.

```

## Programa 65

**Enunciado:** Escreva um programa em Python para encontrar a interseção de dois dicionários.

```

1. # Dicionários de exemplo
2. dicionario1 = {'a': 1, 'b': 2, 'c': 3}
3. dicionario2 = {'b': 2, 'c': 4, 'd': 5}
4.
5. # Encontrar a interseção de dois dicionários
6. intersecao = {chave: dicionario1[chave] for chave in dicionario1 if chave in
dicionario2 and dicionario1[chave] == dicionario2[chave]}
7.
8. # Imprimir a interseção
9. print("Interseção dos dicionários:", intersecao)
10.

```

## Programa 66

**Enunciado:** Escreva um programa em Python para criar um dicionário a partir de duas listas.

```

1. # Listas de exemplo
2. chaves = ['a', 'b', 'c']
3. valores = [1, 2, 3]
4.
5. # Criar um dicionário a partir de duas listas usando zip
6. dicionario = dict(zip(chaves, valores))
7.
8. # Imprimir o dicionário
9. print("Dicionário criado a partir de duas listas:", dicionario)
10.

```

## Programa 67

**Enunciado:** Escreva um programa em Python para remover chaves múltiplas de um dicionário.

```

1. # Dicionário de exemplo
2. meu_dicionario = {'a': 1, 'b': 2, 'c': 3, 'd': 4}
3. # Chaves a serem removidas
4. chaves_para_remover = ['b', 'd']
5. # Remover as chaves do dicionário
6. for chave in chaves_para_remover:
7.     if chave in meu_dicionario:
8.         del meu_dicionario[chave]
9. # Imprimir o dicionário após a remoção
10. print("Dicionário após remover chaves múltiplas:", meu_dicionario)
11.

```

## Programa 68

**Enunciado:** Escreva um programa em Python para criar um dicionário a partir de duas listas.

```

1. # Listas de exemplo
2. chaves = ['a', 'b', 'c']
3. valores = [1, 2, 3]
4.
5. # Criar um dicionário a partir de duas listas usando zip
6. dicionario = dict(zip(chaves, valores))
7.
8. # Imprimir o dicionário
9. print("Dicionário criado a partir de duas listas:", dicionario)
10.

```

## Programa 69

**Enunciado:** Escreva um programa em Python para classificar um dicionário por chave.

```

1. # Dicionário de exemplo
2. meu_dicionario = {'b': 2, 'a': 1, 'd': 4, 'c': 3}
3.
4. # Classificar o dicionário por chave usando dict() e sorted()
5. dicionario_classificado = dict(sorted(meu_dicionario.items()))
6.
7. # Imprimir o dicionário classificado
8. print("Dicionário classificado por chave:", dicionario_classificado)
9.

```

## Programa 70

**Enunciado:** Escreva um programa em Python para classificar um dicionário por valor.

```

1. # Dicionário de exemplo
2. meu_dicionario = {'a': 2, 'b': 1, 'd': 4, 'c': 3}
3.
4. # Classificar o dicionário por valor usando dict() e sorted()
5. dicionario_classificado = dict(sorted(meu_dicionario.items(), key=lambda item:
item[1]))
6.
7. # Imprimir o dicionário classificado
8. print("Dicionário classificado por valor:", dicionario_classificado)
9.

```

## Programa 71

**Enunciado:** Escreva um programa em Python para encontrar o tamanho de um dicionário.

```

1. # Dicionário de exemplo
2. meu_dicionario = {'a': 1, 'b': 2, 'c': 3}
3.
4. # Encontrar o tamanho do dicionário
5. tamanho = len(meu_dicionario)
6.
7. # Imprimir o tamanho do dicionário
8. print("O tamanho do dicionário é:", tamanho)
9.

```

## Programa 72

**Enunciado:** Escreva um programa em Python para inverter um dicionário.

```

1. # Dicionário de exemplo
2. meu_dicionario = {'a': 1, 'b': 2, 'c': 3}
3.
4. # Inverter o dicionário usando compreensão de dicionário
5. dicionario_invertido = {valor: chave for chave, valor in meu_dicionario.items()}
6.
7. # Imprimir o dicionário invertido
8. print("Dicionário invertido:", dicionario_invertido)
9.

```

## Programa 73

**Enunciado:** Escreva um programa em Python para mesclar dois dicionários.

```

1. # Dicionários de exemplo
2. dionario1 = {'a': 1, 'b': 2}
3. dionario2 = {'c': 3, 'd': 4}
4.
5. # Mesclar dicionários usando o operador **
6. dionario_mesclado = {**dionario1, **dionario2}
7.
8. # Imprimir o dicionário mesclado
9. print("Dicionário mesclado:", dionario_mesclado)
10.

```

## Programa 74

**Enunciado:** Escreva um programa em Python para remover duplicatas de uma lista.

```

1. # Lista de exemplo com duplicatas
2. lista_com_duplicatas = [1, 2, 2, 3, 4, 4, 5]
3.
4. # Remover duplicatas convertendo a lista em um conjunto e de volta para uma lista
5. lista_sem_duplicatas = list(set(lista_com_duplicatas))
6.
7. # Imprimir a lista sem duplicatas
8. print("Lista sem duplicatas:", lista_sem_duplicatas)
9.

```

## Programa 75

**Enunciado:** Escreva um programa em Python para verificar se duas listas têm pelo menos um elemento comum.

```

1. def verificar_elemento_comum(lista1, lista2):
2.     # Usar interseção de conjuntos para verificar elementos comuns
3.     return any(elemento in lista2 for elemento in lista1)
4.
5. # Listas de exemplo
6. lista1 = [1, 2, 3]
7. lista2 = [4, 5, 6, 2]
8.
9. # Verificar se há pelo menos um elemento comum
10. if verificar_elemento_comum(lista1, lista2):
11.     print("As listas têm pelo menos um elemento comum.")
12. else:
13.     print("As listas não têm nenhum elemento comum.")
14.

```

## Programa 76

**Enunciado:** Escreva um programa em Python para calcular a soma dos números em uma lista.

```

1. # Lista de exemplo
2. lista = [1, 2, 3, 4, 5]
3.
4. # Calcular a soma dos números na lista
5. soma = sum(lista)
6.
7. # Imprimir a soma
8. print("A soma dos números na lista é:", soma)
9.

```

## Programa 77

**Enunciado:** Escreva um programa em Python para encontrar a média dos números em uma lista.

```

1. # Lista de exemplo
2. lista = [1, 2, 3, 4, 5]
3. # Calcular a média dos números na lista
4. media = sum(lista) / len(lista)
5.
6. # Imprimir a média
7. print("A média dos números na lista é:", media)
8.

```

## Programa 78

**Enunciado:** Escreva um programa em Python para encontrar o desvio padrão dos números em uma lista.

```

1. import math
2. def calcular_desvio_padrao(lista):
3.     # Calcular a média
4.     media = sum(lista) / len(lista)
5.
6.     # Calcular a soma dos quadrados das diferenças da média
7.     soma_diferencias_quadradas = sum((x - media) ** 2 for x in lista)
8.
9.     # Calcular o desvio padrão
10.    desvio_padrao = math.sqrt(soma_diferencias_quadradas / len(lista))
11.
12.    return desvio_padrao
13.
14. # Lista de exemplo
15. lista = [1, 2, 3, 4, 5]
16.
17. # Calcular o desvio padrão
18. desvio_padrao = calcular_desvio_padrao(lista)
19.
20. # Imprimir o desvio padrão
21. print("O desvio padrão dos números na lista é:", desvio_padrao)
22.

```

## Programa 79

**Enunciado:** Escreva um programa em Python para calcular a variância dos números em uma lista.

```

1. def calcular_variancia(lista):
2.     # Calcular a média
3.     media = sum(lista) / len(lista)
4.
5.     # Calcular a soma dos quadrados das diferenças da média
6.     soma_diferencias_quadradas = sum((x - media) ** 2 for x in lista)
7.
8.     # Calcular a variância
9.     variancia = soma_diferencias_quadradas / len(lista)
10.
11.    return variancia
12.
13. # Lista de exemplo
14. lista = [1, 2, 3, 4, 5]
15.
16. # Calcular a variância
17. variancia = calcular_variancia(lista)
18.
19. # Imprimir a variância
20. print("A variância dos números na lista é:", variancia)
21.

```

## Programa 80

**Enunciado:** Escreva um programa em Python para calcular a mediana dos números em uma lista.

```
1. def calcular_mediana(lista):
2.     # Ordenar a lista
3.     lista_ordenada = sorted(lista)
4.
5.     # Calcular o comprimento da lista
6.     n = len(lista_ordenada)
7.
8.     # Calcular a mediana
9.     if n % 2 == 0:
10.         # Se o comprimento for par, a mediana é a média dos dois elementos do meio
11.         mediana = (lista_ordenada[n//2 - 1] + lista_ordenada[n//2]) / 2
12.     else:
13.         # Se o comprimento for ímpar, a mediana é o elemento do meio
14.         mediana = lista_ordenada[n//2]
15.
16.     return mediana
17.
18. # Lista de exemplo
19. lista = [1, 2, 3, 4, 5]
20.
21. # Calcular a mediana
22. mediana = calcular_mediana(lista)
23.
24. # Imprimir a mediana
25. print("A mediana dos números na lista é:", mediana)
26.
```

## Programa 81

**Enunciado:** Escreva um programa em Python para calcular a moda dos números em uma lista.

```
1. from collections import Counter
2.
3. def calcular_moda(lista):
4.     # Contar a frequência de cada elemento na lista
5.     contador = Counter(lista)
6.
7.     # Encontrar a frequência máxima
8.     frequencia_maxima = max(contador.values())
9.
10.    # Encontrar todos os elementos que têm a frequência máxima
11.    modas = [chave for chave, valor in contador.items() se valor == frequencia_maxima]
12.
13.    return modas
14.
15. # Lista de exemplo
16. lista = [1, 2, 2, 3, 4, 4, 4, 5, 5]
17.
18. # Calcular a moda
19. modas = calcular_moda(lista)
20.
21. # Imprimir a moda
22. print("A moda dos números na lista é:", modas)
23.
```

## Programa 82

**Enunciado:** Escreva um programa em Python para remover valores duplicados de uma lista.

```
1. # Lista de exemplo com valores duplicados
2. lista_com_duplicatas = [1, 2, 2, 3, 4, 4, 5]
3.
4. # Remover duplicatas convertendo a lista em um conjunto e de volta para uma lista
5. lista_sem_duplicatas = list(set(lista_com_duplicatas))
6.
7. # Imprimir a lista sem duplicatas
8. print("Lista sem duplicatas:", lista_sem_duplicatas)
9.
```

## Programa 83

**Enunciado:** Escreva um programa em Python para encontrar elementos comuns em duas listas.

```
1. # Listas de exemplo
2. lista1 = [1, 2, 3, 4, 5]
3. lista2 = [4, 5, 6, 7, 8]
4.
5. # Encontrar elementos comuns usando interseção de conjuntos
6. elementos_comuns = list(set(lista1) & set(lista2))
7.
8. # Imprimir os elementos comuns
9. print("Elementos comuns nas listas:", elementos_comuns)
10.
```

## Programa 84

**Enunciado:** Escreva um programa em Python para encontrar a diferença de elementos entre duas listas.

```
1. # Listas de exemplo
2. lista1 = [1, 2, 3, 4, 5]
3. lista2 = [4, 5, 6, 7, 8]
4.
5. # Encontrar a diferença de elementos usando diferença de conjuntos
6. diferenca = list(set(lista1) - set(lista2))
7.
8. # Imprimir a diferença de elementos
9. print("Diferença de elementos entre as listas:", diferenca)
10.
```

## Programa 85

**Enunciado:** Escreva um programa em Python para encontrar a união de dois conjuntos.

```
1. # Conjuntos de exemplo
2. conjunto1 = {1, 2, 3}
3. conjunto2 = {3, 4, 5}
4.
5. # Encontrar a união dos conjuntos
6. uniao = conjunto1.union(conjunto2)
7.
8. # Imprimir a união dos conjuntos
9. print("União dos conjuntos:", uniao)
10.
```

## Programa 86

**Enunciado:** Escreva um programa em Python para encontrar a interseção de dois conjuntos.

```
1. # Conjuntos de exemplo
2. conjunto1 = {1, 2, 3}
3. conjunto2 = {3, 4, 5}
4.
5. # Encontrar a interseção dos conjuntos
6. intersecao = conjunto1.intersection(conjunto2)
7.
8. # Imprimir a interseção dos conjuntos
9. print("Interseção dos conjuntos:", intersecao)
10.
```

## Programa 87

**Enunciado:** Escreva um programa em Python para encontrar a diferença de dois conjuntos.

```
1. # Conjuntos de exemplo
2. conjunto1 = {1, 2, 3}
3. conjunto2 = {3, 4, 5}
4.
5. # Encontrar a diferença dos conjuntos
6. diferenca = conjunto1.difference(conjunto2)
7.
8. # Imprimir a diferença dos conjuntos
9. print("Diferença dos conjuntos:", diferenca)
10.
```

## Programa 88

**Enunciado:** Escreva um programa em Python para verificar se um conjunto é subconjunto de outro.

```
1. # Conjuntos de exemplo
2. conjunto1 = {1, 2, 3}
3. conjunto2 = {3, 4, 5}
4.
5. # Encontrar a diferença dos conjuntos
6. diferenca = conjunto1.difference(conjunto2)
7.
8. # Imprimir a diferença dos conjuntos
9. print("Diferença dos conjuntos:", diferenca)
10.
```

## Programa 89

**Enunciado:** Escreva um programa em Python para verificar se dois conjuntos são disjuntos.

```
1. # Conjuntos de exemplo
2. conjunto1 = {1, 2, 3}
3. conjunto2 = {4, 5, 6}
4.
5. # Verificar se os conjuntos são disjuntos
6. sao_disjuntos = conjunto1.isdisjoint(conjunto2)
7.
8. # Imprimir o resultado
9. print(f"Os conjuntos {conjunto1} e {conjunto2} são disjuntos: {sao_disjuntos}")
10.
```

## Programa 90

**Enunciado:** Escreva um programa em Python para encontrar o máximo e o mínimo em um conjunto.

```
1. # Conjunto de exemplo
2. conjunto = {1, 2, 3, 4, 5}
3.
4. # Encontrar o valor máximo e mínimo no conjunto
5. maximo = max(conjunto)
6. minimo = min(conjunto)
7.
8. # Imprimir os valores máximo e mínimo
9. print("Valor máximo no conjunto:", maximo)
10. print("Valor mínimo no conjunto:", minimo)
11.
```

## Programa 91

**Enunciado:** Escreva um programa em Python para encontrar a soma de todos os elementos em um conjunto.

```
1. # Conjunto de exemplo
2. conjunto = {1, 2, 3, 4, 5}
3.
4. # Calcular a soma de todos os elementos no conjunto
5. soma = sum(conjunto)
6.
7. # Imprimir a soma
8. print("A soma dos elementos no conjunto é:", soma)
9.
```

## Programa 92

**Enunciado:** Escreva um programa em Python para encontrar a média de todos os elementos em um conjunto.

```

1. # Conjunto de exemplo
2. conjunto = {1, 2, 3, 4, 5}
3.
4. # Calcular a média de todos os elementos no conjunto
5. media = sum(conjunto) / len(conjunto)
6.
7. # Imprimir a média
8. print("A média dos elementos no conjunto é:", media)
9.

```

## Programa 93

**Enunciado:** Escreva um programa em Python para verificar se um conjunto é um superconjunto de outro.

```

1. # Conjuntos de exemplo
2. conjunto1 = {1, 2, 3, 4, 5}
3. conjunto2 = {1, 2}
4.
5. # Verificar se conjunto1 é superconjunto de conjunto2
6. eh_superconjunto = conjunto1.issuperset(conjunto2)
7.
8. # Imprimir o resultado
9. print(f"O conjunto {conjunto1} é superconjunto de {conjunto2}: {eh_superconjunto}")
10.

```

## Programa 94

**Enunciado:** Escreva um programa em Python para calcular a diferença simétrica de dois conjuntos.

```

1. # Conjuntos de exemplo
2. conjunto1 = {1, 2, 3}
3. conjunto2 = {3, 4, 5}
4.
5. # Calcular a diferença simétrica dos conjuntos
6. diferenca_simetrica = conjunto1.symmetric_difference(conjunto2)
7.
8. # Imprimir a diferença simétrica
9. print("Diferença simétrica dos conjuntos:", diferenca_simetrica)
10.

```

## Programa 95

**Enunciado:** Escreva um programa em Python para verificar se um conjunto é vazio.

```
1. # Conjunto de exemplo
2. conjunto = set()
3.
4. # Verificar se o conjunto é vazio
5. eh_vazio = len(conjunto) == 0
6.
7. # Imprimir o resultado
8. print("O conjunto está vazio:", eh_vazio)
9.
```

## Programa 96

**Enunciado:** Escreva um programa em Python para copiar um conjunto para outro.

```
1. # Conjunto de exemplo
2. conjunto_original = {1, 2, 3, 4, 5}
3.
4. # Copiar o conjunto usando o método copy()
5. conjunto_copia = conjunto_original.copy()
6.
7. # Imprimir o conjunto copiado
8. print("Conjunto copiado:", conjunto_copia)
9.
```

## Programa 97

**Enunciado:** Escreva um programa em Python para limpar todos os elementos de um conjunto.

```
1. # Conjunto de exemplo
2. conjunto = {1, 2, 3, 4, 5}
3.
4. # Limpar todos os elementos do conjunto
5. conjunto.clear()
6.
7. # Imprimir o conjunto após a limpeza
8. print("Conjunto após a limpeza:", conjunto)
9.
```

## Programa 98

**Enunciado:** Escreva um programa em Python para encontrar elementos exclusivos em duas listas.

```

1. # Listas de exemplo
2. lista1 = [1, 2, 3, 4, 5]
3. lista2 = [4, 5, 6, 7, 8]
4.
5. # Encontrar elementos exclusivos usando diferença de conjuntos
6. exclusivos_lista1 = list(set(lista1) - set(lista2))
7. exclusivos_lista2 = list(set(lista2) - set(lista1))
8.
9. # Imprimir os elementos exclusivos
10. print("Elementos exclusivos na lista1:", exclusivos_lista1)
11. print("Elementos exclusivos na lista2:", exclusivos_lista2)
12.

```

## Programa 99

**Enunciado:** Escreva um programa em Python para encontrar o segundo maior número em uma lista.

```

1. # Lista de exemplo
2. lista = [1, 2, 3, 4, 5]
3.
4. # Encontrar o segundo maior número na lista
5. segundo_maior = sorted(lista)[-2]
6.
7. # Imprimir o segundo maior número
8. print("O segundo maior número na lista é:", segundo_maior)
9.

```

## Programa 100

**Enunciado:** Escreva um programa em Python para encontrar o segundo maior número em uma lista.

```

1. # Lista de exemplo
2. lista = [1, 2, 3, 4, 5]
3.
4. # Encontrar o segundo maior número na lista
5. segundo_maior = sorted(lista)[-2]
6.
7. # Imprimir o segundo maior número
8. print("O segundo maior número na lista é:", segundo_maior)
9.

```

## Programa 101

**Enunciado:** Escreva um programa em Python para remover elementos comuns de duas listas.

```

1. # Listas de exemplo
2. lista1 = [1, 2, 3, 4, 5]
3. lista2 = [4, 5, 6, 7, 8]
4.
5. # Encontrar elementos comuns usando interseção de conjuntos
6. comuns = list(set(lista1) & set(lista2))
7.
8. # Remover elementos comuns de ambas as listas
9. lista1 = [item for item in lista1 se item não estiver em comuns]
10. lista2 = [item for item in lista2 se item não estiver em comuns]
11.
12. # Imprimir as listas resultantes
13. print("Lista 1 após remover elementos comuns:", lista1)
14. print("Lista 2 após remover elementos comuns:", lista2)
15.

```

## Programa 102

**Enunciado:** Escreva um programa em Python para encontrar o segundo maior número em um conjunto.

```

1. # Conjunto de exemplo
2. conjunto = {1, 2, 3, 4, 5}
3.
4. # Encontrar o segundo maior número no conjunto
5. segundo_maior = sorted(conjunto)[-2]
6.
7. # Imprimir o segundo maior número
8. print("O segundo maior número no conjunto é:", segundo_maior)
9.

```

## Programa 103

**Enunciado:** Escreva um programa em Python para encontrar o segundo menor número em um conjunto.

```

1. # Conjunto de exemplo
2. conjunto = {1, 2, 3, 4, 5}
3.
4. # Encontrar o segundo menor número no conjunto
5. segundo_menor = sorted(conjunto)[1]
6.
7. # Imprimir o segundo menor número
8. print("O segundo menor número no conjunto é:", segundo_menor)
9.

```

## Programa 104

**Enunciado:** Escreva um programa em Python para encontrar o valor máximo e mínimo de um dicionário.

```

1. # Dicionário de exemplo
2. meu_dicionario = {'a': 10, 'b': 20, 'c': 30, 'd': 5}
3.
4. # Encontrar o valor máximo e mínimo no dicionário
5. valor_maximo = max(meu_dicionario.values())
6. valor_minimo = min(meu_dicionario.values())
7.
8. # Imprimir os valores máximo e mínimo
9. print("Valor máximo no dicionário:", valor_maximo)
10. print("Valor mínimo no dicionário:", valor_minimo)
11.

```

## Programa 105

**Enunciado:** Escreva um programa em Python para contar o número de ocorrências de um caractere em uma string.

```

1. # String de exemplo
2. string = "programação em Python"
3.
4. # Caractere a ser contado
5. caractere = 'a'
6.
7. # Contar o número de ocorrências do caractere na string
8. ocorrencias = string.count(caractere)
9.
10. # Imprimir o número de ocorrências
11. print(f"O caractere '{caractere}' aparece {ocorrencias} vezes na string.")
12.

```

## Programa 106

**Enunciado:** Escreva um programa em Python para verificar se uma lista está vazia.

```

1. # Lista de exemplo
2. lista = []
3. # Verificar se a lista está vazia
4. esta_vazia = len(lista) == 0
5. # Imprimir o resultado
6. print("A lista está vazia:", esta_vazia)
7.

```

## Programa 107

**Enunciado:** Escreva um programa em Python para verificar se um dicionário está vazio.

```
1. # Dicionário de exemplo
2. dicionario = {}
3.
4. # Verificar se o dicionário está vazio
5. esta_vazio = len(dicionario) == 0
6.
7. # Imprimir o resultado
8. print("O dicionário está vazio:", esta_vazio)
9.
```

## Programa 108

**Enunciado:** Escreva um programa em Python para verificar se um conjunto está vazio.

```
1. # Conjunto de exemplo
2. conjunto = set()
3.
4. # Verificar se o conjunto está vazio
5. esta_vazio = len(conjunto) == 0
6.
7. # Imprimir o resultado
8. print("O conjunto está vazio:", esta_vazio)
9.
```

## Programa 109

**Enunciado:** Escreva um programa em Python para verificar se uma string está vazia.

```
1. # String de exemplo
2. string = ""
3.
4. # Verificar se a string está vazia
5. esta_vazia = len(string) == 0
6.
7. # Imprimir o resultado
8. print("A string está vazia:", esta_vazia)
```

## Programa 110

**Enunciado:** Escreva um programa em Python para remover elementos duplicados de uma lista mantendo a ordem.

```
1. # Lista de exemplo com duplicatas
2. lista_com_duplicatas = [1, 2, 2, 3, 4, 4, 5]
3.
4. # Remover duplicatas mantendo a ordem
5. lista_sem_duplicatas = []
6. [lista_sem_duplicatas.append(item) for item in lista_com_duplicatas se item não
estiver em lista_sem_duplicatas]
7.
8. # Imprimir a lista sem duplicatas
9. print("Lista sem duplicatas (mantendo a ordem):", lista_sem_duplicatas)
10.
```

## Programa 111

**Enunciado:** Escreva um programa em Python para encontrar a soma de uma sequência aritmética.

```
1. def soma_sequencia_aritmetica(a, d, n):
2.     # Fórmula da soma da sequência aritmética
3.     soma = (n / 2) * (2 * a + (n - 1) * d)
4.     return soma
5.
6. # Termo inicial (a), diferença comum (d) e número de termos (n)
7. a = 1
8. d = 1
9. n = 10
10.
11. # Calcular a soma da sequência aritmética
12. soma = soma_sequencia_aritmetica(a, d, n)
13.
14. # Imprimir a soma
15. print(f"A soma dos primeiros {n} termos da sequência aritmética é:", soma)
16.
```

## Programa 112

**Enunciado:** Escreva um programa em Python para encontrar a soma de uma sequência geométrica.

```

1. def soma_sequencia_geometrica(a, r, n):
2.     # Fórmula da soma da sequência geométrica
3.     if r == 1:
4.         soma = a * n
5.     else:
6.         soma = a * (1 - r**n) / (1 - r)
7.     return soma
8.
9. # Termo inicial (a), razão comum (r) e número de termos (n)
10. a = 1
11. r = 2
12. n = 10
13.
14. # Calcular a soma da sequência geométrica
15. soma = soma_sequencia_geometrica(a, r, n)
16.
17. # Imprimir a soma
18. print(f"A soma dos primeiros {n} termos da sequência geométrica é:", soma)
19.

```

## Programa 113

**Enunciado:** Escreva um programa em Python para encontrar o máximo divisor comum (MDC) de dois números usando recursão.

```

1. def mdc_recursivo(a, b):
2.     # Base: se b for 0, retorne a
3.     if b == 0:
4.         return a
5.     # Caso contrário, retorne mdc(b, a % b)
6.     return mdc_recursivo(b, a % b)
7.
8. # Dois números de exemplo
9. a = 48
10. b = 18
11.
12. # Calcular o MDC usando recursão
13. mdc = mdc_recursivo(a, b)
14.
15. # Imprimir o MDC
16. print(f"O máximo divisor comum de {a} e {b} é:", mdc)
17.

```

## Programa 114

**Enunciado:** Escreva um programa em Python para encontrar o mínimo múltiplo comum (MMC) de dois números.

```

1. def mdc(a, b):
2.     while b:
3.         a, b = b, a % b
4.     return a
5.
6. def mmc(a, b):
7.     return abs(a * b) // mdc(a, b)
8.
9. # Dois números de exemplo
10. a = 12
11. b = 15
12.
13. # Calcular o MMC
14. resultado_mmc = mmc(a, b)
15.
16. # Imprimir o MMC
17. print(f"O mínimo múltiplo comum de {a} e {b} é:", resultado_mmc)
18.

```

## Programa 115

**Enunciado:** Escreva um programa em Python para verificar se um número é primo usando recursão.

```

1. def eh_primo_recursivo(n, divisor=None):
2.     if divisor is None:
3.         divisor = n - 1
4.     if divisor == 1:
5.         return True
6.     if n % divisor == 0:
7.         return False
8.     return eh_primo_recursivo(n, divisor - 1)
9.
10. # Número de exemplo
11. numero = 29
12.
13. # Verificar se o número é primo usando recursão
14. primo = eh_primo_recursivo(numero)
15.
16. # Imprimir o resultado
17. print(f"O número {numero} é primo:", primo)
18.

```

## Programa 116

**Enunciado:** Escreva um programa em Python para calcular a potência de um número usando recursão.

```

1. def potencia_recursiva(base, expoente):
2.     # Base: se o expoente for 0, retorne 1
3.     if expoente == 0:
4.         return 1
5.     # Caso contrário, multiplique a base pela potência da base elevada ao expoente - 1
6.     return base * potencia_recursiva(base, expoente - 1)
7.
8. # Base e expoente de exemplo
9. base = 2
10. expoente = 3
11.
12. # Calcular a potência usando recursão
13. resultado = potencia_recursiva(base, expoente)
14.
15. # Imprimir o resultado
16. print(f"{base} elevado a {expoente} é:", resultado)
17.

```

## Programa 117

**Enunciado:** Escreva um programa em Python para calcular a raiz quadrada de um número usando o método de Newton.

```

1. def raiz_quadrada_newton(num, estimativa=1.0):
2.     if abs(estimativa**2 - num) < 0.00001:
3.         return estimativa
4.     else:
5.         return raiz_quadrada_newton(num, (estimativa + num / estimativa) / 2)
6.
7. # Número de exemplo
8. numero = 25
9.
10. # Calcular a raiz quadrada usando o método de Newton
11. raiz = raiz_quadrada_newton(numero)
12.
13. # Imprimir o resultado
14. print(f"A raiz quadrada de {numero} é aproximadamente:", raiz)
15.

```

## Programa 118

**Enunciado:** Escreva um programa em Python para converter um número decimal em binário usando recursão.

```

1. def decimal_para_binario(n):
2.     if n == 0:
3.         return "0"
4.     elif n == 1:
5.         return "1"
6.     else:
7.         return decimal_para_binario(n // 2) + str(n % 2)
8.
9. # Número decimal de exemplo
10. numero = 10
11.
12. # Converter o número decimal em binário usando recursão
13. binario = decimal_para_binario(numero)
14.
15. # Imprimir o resultado
16. print(f"O número decimal {numero} em binário é:", binario)
17.

```

## Programa 119

**Enunciado:** Escreva um programa em Python para encontrar a soma de números naturais até  $n$  usando recursão.

```

1. def soma_numeros_naturais(n):
2.     if n == 1:
3.         return 1
4.     else:
5.         return n + soma_numeros_naturais(n - 1)
6.
7. # Número de exemplo
8. n = 10
9.
10. # Calcular a soma de números naturais até n usando recursão
11. soma = soma_numeros_naturais(n)
12.
13. # Imprimir o resultado
14. print(f"A soma dos primeiros {n} números naturais é:", soma)
15.

```

## Programa 120

**Enunciado:** Escreva um programa em Python para encontrar a soma de números naturais até n usando recursão.

```
1. def soma_numeros_naturais(n):
2.     if n == 1:
3.         return 1
4.     else:
5.         return n + soma_numeros_naturais(n - 1)
6.
7. # Número de exemplo
8. n = 10
9.
10. # Calcular a soma de números naturais até n usando recursão
11. soma = soma_numeros_naturais(n)
12.
13. # Imprimir o resultado
14. print(f"A soma dos primeiros {n} números naturais é:", soma)
15.
```