

Aula 17

Revisão

Para definir uma função em Python, você deve usar a palavra-chave "def", seguida pelo nome da função e, opcionalmente, uma lista de argumentos entre parênteses. A estrutura geral é a seguinte :

```
1 def mostra_mensagem():
2     print("Olá, mundo!")
3
4 mostra_mensagem()
```

Acima, temos uma função que não tem nenhum parâmetro. O que ele faz é apenas exibir uma mensagem sempre que for chamada.

```
1 def multiplica(n1, n2):
2     print(f'{n1} x {n2} = {n1 * n2}')
3
4 for i in range(1, 11):
5     for j in range(1, 11):
6         multiplica(i, j)
7     print('_____')
```

A função multiplicação tem dois parâmetros, que vai multiplicar eles e mostrar o valor. Depois, usando o loop for, envio os valores para dentro da função de modo que mostre a tabuada dos números 1 a 10.

```
1 import random
2
3 def verifica_par(numero):
4     if numero % 2 == 0:
5         return True
6     return False
7
8 for i in range(10):
9     numero = random.randint(1, 100)
10    if verifica_par(numero):
11        print(f'O número {numero} é par!')
12    else:
13        print(f'O número {numero} é ímpar!')
```

A função acima vai receber um valor (gerado aleatoriamente dentro do for), verificar se é par ou ímpar e retornar o resultado booleano, que posteriormente vai ser usado no teste do if para mostrar a mensagem se é par ou ímpar.

Tipo None

Em Python, None é um valor especial que representa a ausência de valor. É usado para indicar que uma variável não tem valor ou que uma função não retorna nada. É semelhante a null em outras linguagens de programação.

Quando uma variável é definida como None, ela não tem nenhum valor atribuído a ela. Isso é diferente de uma variável que não foi definida, que geraria um erro se você tentasse usá-la em seu código.

```
>>> valor = None
>>> print(valor)
None
>>>
```

Aqui, a variável x foi definida como None, então quando imprimimos o valor de x, o resultado é None.

None também é frequentemente usado em funções para indicar que a função não retorna nenhum valor.

```
>>> def funcao_generica():
...     # realiza alguma coisa
...     return None
...
>>> print(funcao_generica())
None
>>>
```

Aqui, a função função_generica() não retorna nenhum valor explicitamente usando a declaração return, mas como None é o valor padrão de retorno de uma função sem declaração return, então podemos omitir essa declaração.

```
>>> def funcao_generica():
...     # realiza alguma coisa
...     pass
...
>>> print(funcao_generica())
None
>>>
```

Em resumo, None é um valor especial em Python usado para indicar a ausência de valor. Ele é usado frequentemente para inicializar variáveis, indicar que uma função não retorna nada ou como um valor padrão de argumento de função.

Docstrings

Os "docstrings" (documentation strings) em Python são cadeias de caracteres usadas para documentar código. Eles podem ser incluídos em módulos, classes, funções e métodos para fornecer informações sobre o que eles fazem e como usá-los. Essas strings são acessíveis através do atributo doc e podem ser exibidas usando a função "help()".

Os "docstrings" geralmente começam e terminam com aspas triplas simples (""") ou duplas (""").

Os docstrings são uma forma de documentar o seu código de maneira clara e concisa, para que outros desenvolvedores e usuários possam entender o que ele faz e como usá-lo. Eles são usados para explicar o que uma classe, função ou método faz, quais parâmetros ele aceita, o que ele retorna e como usá-lo.

Por exemplo, para documentar uma função chamada "soma", você poderia escrever :

```
>>> def soma(a, b):
...     """Essa função vai receber dois valores e retornar a soma deles.
...     Parâmetros:
...         a (int): primeiro valor
...         b (int): segundo valor
...     Return:
...         resultado (int): resultado do cálculo
...     """
...     resultado = a + b
...     return resultado
...
```

Abaixo usamos a função built-in "help()" para ver a documentação da função, que também pode ser usado para os módulos, classes e métodos.

```
>>> print(soma(1,2))
3
>>> help(soma)
Help on function soma in module __main__:

soma(a, b)
    Essa função vai receber dois valores e retornar a soma deles.
    Parâmetros:
        a (int): primeiro valor
        b (int): segundo valor
    Return:
        resultado (int): resultado do cálculo

>>>
```

Também podemos acessar essa string de documentação usando o atributo "doc" da função.

```
>>> print(soma.__doc__)
Essa função vai receber dois valores e retornar a soma deles.
    Parâmetros:
        a (int): primeiro valor
        b (int): segundo valor
    Return:
        resultado (int): resultado do cálculo

>>>
```

Funções – parte Deux

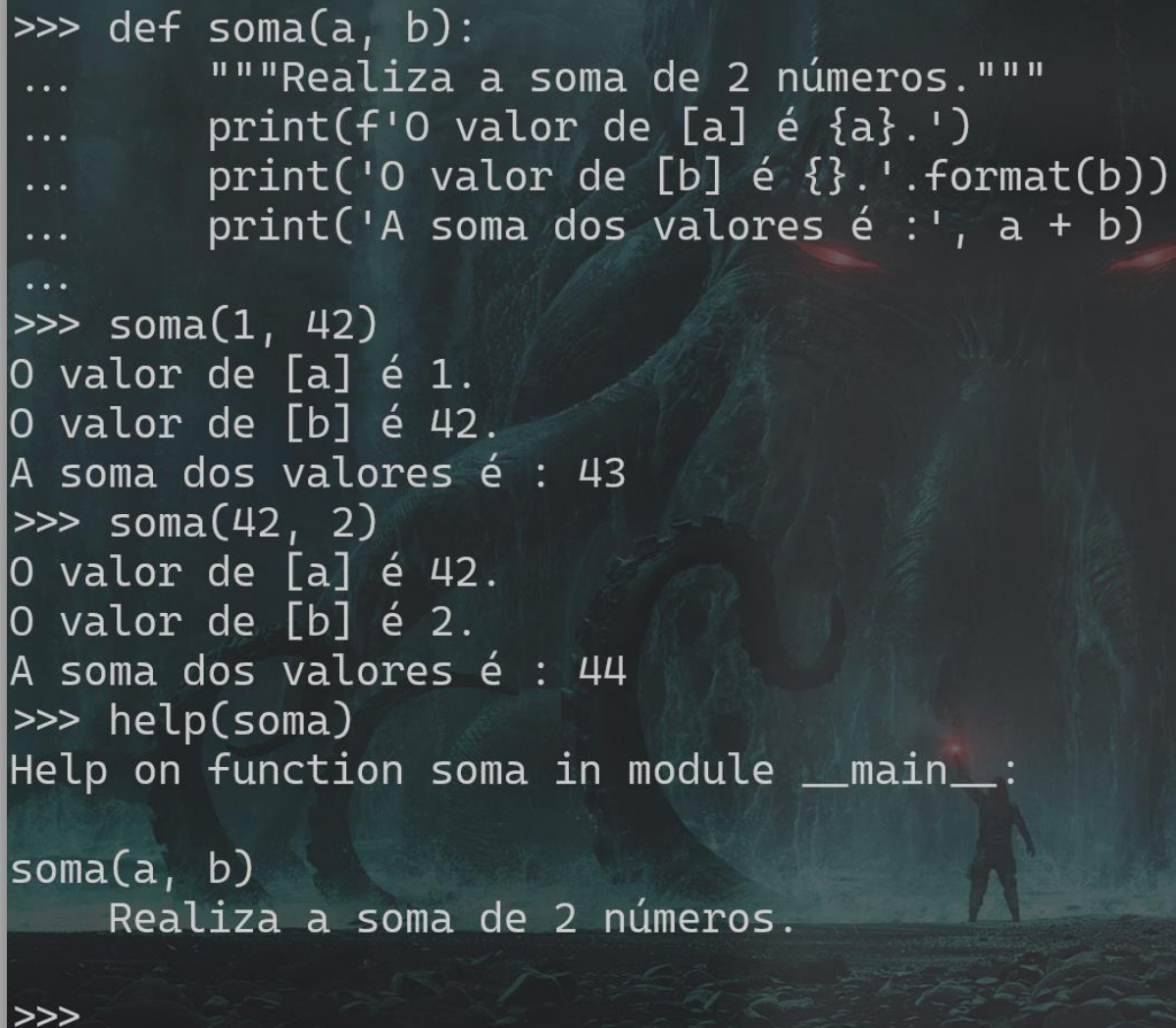
Argumentos nas Funções

Os argumentos em uma função em Python são valores ou variáveis que são passados para a função quando ela é chamada.

Existem alguns tipos de argumentos nas funções :

Argumentos Posicionais

Argumentos posicionais são aqueles que são passados para a função na ordem em que eles aparecem na definição da função. Por exemplo, na função abaixo, "a" e "b" são argumentos posicionais :

A terminal window showing Python code and its execution. The background of the terminal is a dark, atmospheric image of a dragon in a misty, rocky landscape. The text is white, providing a high contrast against the dark background.

```
>>> def soma(a, b):  
...     """Realiza a soma de 2 números."""  
...     print(f'O valor de [a] é {a}.')  
...     print('O valor de [b] é {}'.format(b))  
...     print('A soma dos valores é :', a + b)  
...  
>>> soma(1, 42)  
O valor de [a] é 1.  
O valor de [b] é 42.  
A soma dos valores é : 43  
>>> soma(42, 2)  
O valor de [a] é 42.  
O valor de [b] é 2.  
A soma dos valores é : 44  
>>> help(soma)  
Help on function soma in module __main__:  
  
soma(a, b)  
    Realiza a soma de 2 números.  
  
>>>
```

Repare que a posição dos valores passados para a função afeta os valores das variáveis locais a cada execução.

Argumentos Palavra-chave

Argumentos de palavra-chave são aqueles que são passados para a função usando o nome do argumento, seguido do sinal de igual e o valor. Por exemplo, na função abaixo, "base" e "altura" são argumentos de palavra-chave :

```
>>> def area_retangulo(base, altura):  
...     '''Calcula a área do retângulo.'''  
...     print(f'O valor de [base] é {base}')...     print('O valor de [altura] é {}'.format(altura))  
...     print('A área do retângulo é :', base * altura)  
...  
>>> area_retangulo(base=1, altura=42)  
O valor de [base] é 1  
O valor de [altura] é 42  
A área do retângulo é : 42  
>>>  
>>> area_retangulo(altura=42, base=1)  
O valor de [base] é 1  
O valor de [altura] é 42  
A área do retângulo é : 42  
>>>
```

Repare que agora, usando as palavras-chaves para cada chamada da função, podemos alterar a ordem, mas ainda manter os valores das variáveis locais.

Argumentos Opcionais

Além disso, as funções também podem ter argumentos opcionais, que são especificados na definição da função com o sinal de igual "=" seguido de um valor padrão. Isso significa que se esses argumentos não forem fornecidos quando a função for chamada, eles usarão o valor padrão especificado.

```
>>> def saudacao(nome, saudacao="Olá"):  
...     print(saudacao + ", " + nome)  
...  
>>> saudacao("João")  
Olá, João  
>>> saudacao("Maria", "Bom dia")  
Bom dia, Maria  
>>>
```

Na primeira chamada da função, apenas um valor foi passado para ela, que foi atribuído à variável "nome". Na segunda chamada, um segundo valor também foi passado, logo, ele sobrescreve o valor padrão definido de "saudacao" na função.

Também é comum usar None como um valor padrão de argumento de função.

```
>>> def a_funcao(valor=None):
...     if valor is None:
...         valor = 42
...     else:
...         valor = 12
...     return valor
...
>>> print(a_funcao())
42
>>> print(a_funcao(1))
12
>>>
```

Aqui, a função `a_funcao()` tem um parâmetro "valor" que é definido como None por padrão. Se a função for chamada sem um valor para "valor", ele assumirá o valor padrão de None. No corpo da função, atribuímos um novo valor dependendo do que foi recebido como argumento na função.

IMPORTANTE : os argumentos opcionais devem vir ao final da declaração dos argumentos na função. Se o primeiro argumento tiver um valor padrão, todos os subsequentes deverão ter valores padrão.

Abaixo, um exemplo de erro de sintaxe será gerado :

```
>>> def saudacao(nome=1, saudacao):
File "<stdin>", line 1
    def saudacao(nome=1, saudacao):
                        ^^^^^^^^^
SyntaxError: non-default argument follows default argument
>>>
```

Exercícios para Praticar

Exercícios sobre Tipo None

1. Crie uma função que não receba nenhum argumento, mas retorne None.
2. Escreva uma função que receba um número inteiro como argumento e retorne None se o número for par, depois mostre um texto se o return for None ou outro texto se não for.
3. Escreva uma função que receba uma lista de números e retorne None se a lista estiver vazia, depois mostre um texto se o return for None ou outro texto se não for.
4. Crie uma função que receba uma string como argumento e retorne None se a string estiver vazia, depois mostre um texto se o return for None ou outro texto se não for.
5. Escreva uma função que receba duas listas e retorne None se ambas as listas tiverem tamanhos diferentes, depois mostre um texto se o return for None ou outro texto se não for.
6. Crie uma função que receba um dicionário e retorne None se o dicionário estiver vazio, depois mostre um texto se o return for None ou outro texto se não for.
7. Escreva uma função que receba uma lista de números e retorne None se todos os números forem menores que 10, depois mostre um texto se o return for None ou outro texto se não for.
8. Crie uma função que receba uma lista de strings e retorne None se todas as strings tiverem menos de 5 caracteres, depois mostre um texto se o return for None ou outro texto se não for.
9. Escreva uma função que receba uma lista de dicionários e retorne None se todos os dicionários estiverem vazios, depois mostre um texto se o return for None ou outro texto se não for.
10. Crie uma função que receba um número e retorne None se o número for menor que 0, depois mostre um texto se o return for None ou outro texto se não for.
11. Escreva uma função que receba uma lista de números e retorne None se a soma dos números for maior que 100, depois mostre um texto se o return for None ou outro texto se não for.
12. Crie uma função que receba uma lista de strings e retorne None se alguma das strings não tiver a letra "a", depois mostre um texto se o return for None ou outro texto se não for.
13. Escreva uma função que receba uma lista de dicionários e retorne None se algum dos dicionários não tiver a chave "nome", depois mostre um texto se o return for None ou outro texto se não for.
14. Crie uma função que receba um número e retorne None se o número for igual a zero, depois mostre um texto se o return for None ou outro texto se não for.
15. Escreva uma função que receba uma lista de strings e retorne None se alguma das strings tiver mais de 10 caracteres, depois mostre um texto se o return for None ou outro texto se não for.
16. Crie uma função que receba uma lista de dicionários e retorne None se todos os dicionários tiverem a mesma chave, depois mostre um texto se o return for None ou outro texto se não for.
17. Escreva uma função que receba uma lista de números e retorne None se a média dos números for menor que 5, depois mostre um texto se o return for None ou outro texto se não for.
18. Crie uma função que receba uma lista de strings e retorne None se alguma das strings for um palíndromo, depois mostre um texto se o return for None ou outro texto se não for.
19. Escreva uma função que receba uma lista de dicionários e retorne None se algum dos dicionários não tiver a chave "idade", depois mostre um texto se o return for None ou outro texto se não for.
20. Crie uma função que receba um número e retorne None se o número não for inteiro, depois mostre um texto se o return for None ou outro texto se não for.

Exercícios sobre Docstrings

1. Crie uma função que receba dois argumentos e retorne a soma desses valores. Adicione uma docstring explicando o que a função faz.
2. Escreva uma função que receba uma lista e retorne o número de itens na lista. Adicione uma docstring descrevendo os argumentos e o que a função retorna.
3. Crie uma função que receba um número e retorne o quadrado desse número. Adicione uma docstring explicando como usar a função.
4. Escreva uma função que receba um nome e uma idade e retorne uma string com uma mensagem personalizada. Adicione uma docstring descrevendo a função e seus argumentos.
5. Crie uma função que receba uma lista de números e retorne o maior número da lista. Adicione uma docstring explicando como a função funciona e os argumentos que ela espera.

6. Escreva uma função que receba uma string e retorne a mesma string em letras maiúsculas. Adicione uma docstring descrevendo como a função deve ser usada.
7. Crie uma função que receba uma lista de números e retorne a média desses números. Adicione uma docstring explicando como usar a função e o que ela retorna.
8. Escreva uma função que receba uma lista de strings e retorne uma string única com todas as strings concatenadas. Adicione uma docstring descrevendo os argumentos e o que a função retorna.
9. Crie uma função que receba um número e retorne True se o número for par e False se for ímpar. Adicione uma docstring explicando como usar a função e os argumentos esperados.
10. Escreva uma função que receba um dicionário e retorne uma lista com todas as chaves do dicionário. Adicione uma docstring explicando como a função funciona e o que ela retorna.

Exercícios sobre Funções

1. Escreva uma função que receba dois argumentos posicionais e retorne a soma desses valores.
2. Crie uma função que receba três argumentos posicionais e retorne a média desses valores.
3. Escreva uma função que receba quatro argumentos posicionais e retorne o maior valor entre eles.
4. Crie uma função que receba dois números e retorne a divisão do primeiro pelo segundo. Os argumentos devem ser posicionais.
5. Escreva uma função que receba três números e retorne o menor valor entre eles. Os argumentos devem ser posicionais.
6. Crie uma função que receba um número e uma string e retorne uma nova string com a string original repetida o número de vezes indicado pelo número recebido. Os argumentos devem ser posicionais.
7. Escreva uma função que receba uma lista e um número e retorne uma nova lista com os elementos da lista original multiplicados pelo número recebido. Os argumentos devem ser posicionais.
8. Crie uma função que receba uma string e um número inteiro e retorne uma nova string com a string original invertida, repetida o número de vezes indicado pelo número recebido. Os argumentos devem ser posicionais.
9. Escreva uma função que receba uma lista de números e um número inteiro e retorne uma nova lista com os números da lista original elevados ao número recebido. Os argumentos devem ser posicionais.
10. Crie uma função que receba uma string e dois números inteiros e retorne uma nova string com a string original invertida, cortada a partir do primeiro número e com tamanho igual ao segundo número. Os argumentos devem ser posicionais.
11. Escreva uma função que receba uma lista de strings e um número inteiro e retorne uma nova lista com as strings da lista original concatenadas com o número recebido. Os argumentos devem ser posicionais.
12. Crie uma função que receba uma string e dois números inteiros e retorne uma nova string com a string original repetida o primeiro número de vezes, com as letras cortadas a partir do segundo número. Os argumentos devem ser posicionais.
13. Escreva uma função que receba uma lista de números e dois números inteiros e retorne uma nova lista com os números da lista original elevados ao primeiro número e com os valores cortados a partir do segundo número. Os argumentos devem ser posicionais.
14. Crie uma função que receba duas listas e retorne uma nova lista com os elementos da primeira lista adicionados ao final da segunda lista. Os argumentos devem ser posicionais.
15. Escreva uma função que receba uma string e uma lista de strings e retorne True se a string estiver presente na lista e False caso contrário. Os argumentos devem ser posicionais.
16. Crie uma função que receba uma lista de números e uma lista de strings e retorne uma nova lista com os valores da lista de números elevados ao tamanho das strings correspondentes na lista de strings. Os argumentos devem ser posicionais.
17. Escreva uma função que receba uma string e uma lista de strings e retorne uma nova lista com as strings da lista original que não contêm a string recebida como argumento. Os argumentos devem ser posicionais.
18. Crie uma função que receba duas strings e retorne uma nova string com as duas strings concatenadas e separadas por um espaço em branco. Os argumentos devem ser posicionais.
19. Escreva uma função que receba uma lista de números e retorne uma nova lista com os valores da lista original multiplicados pelo seu índice na lista. Os argumentos devem ser posicionais.
20. Crie uma função que receba duas listas de números do mesmo tamanho e retorne uma nova lista com a soma dos valores correspondentes de cada lista. Os argumentos devem ser posicionais.

21. Crie uma função que receba dois argumentos posicionais e um argumento de palavra-chave opcional chamado "operacao", que indica a operação a ser realizada entre os dois valores (soma, subtração, multiplicação ou divisão).
22. Escreva uma função que receba um número inteiro e um argumento de palavra-chave opcional chamado "base", que indica a base para a conversão do número inteiro.
23. Crie uma função que receba uma string e um argumento de palavra-chave opcional chamado "separador", que indica o caractere que será usado para separar as letras da string.
24. Escreva uma função que receba uma lista de strings e um argumento de palavra-chave opcional chamado "letra", que indica a letra que deve ser removida de cada string da lista.
25. Crie uma função que receba uma lista de números e um argumento de palavra-chave opcional chamado "operacao", que indica a operação a ser realizada com os números (soma, subtração, multiplicação ou divisão).
26. Escreva uma função que receba uma string e dois argumentos de palavra-chave opcionais chamados "inicio" e "fim", que indicam as posições na string onde a substring deve ser extraída.
27. Crie uma função que receba um dicionário e um argumento de palavra-chave opcional chamado "chave", que indica a chave do dicionário que deve ser retornada.
28. Escreva uma função que receba uma lista de números e um argumento de palavra-chave opcional chamado "ordenacao", que indica se a lista deve ser ordenada em ordem crescente ou decrescente.
29. Crie uma função que receba uma string e um argumento de palavra-chave opcional chamado "caractere", que indica o caractere que deve ser removido da string.
30. Escreva uma função que receba uma lista de strings e um argumento de palavra-chave opcional chamado "tamanho", que indica o tamanho máximo das strings na lista.
31. Crie uma função que receba uma lista de números e um argumento de palavra-chave opcional chamado "filtro", que indica um número pelo qual os valores da lista devem ser filtrados.
32. Escreva uma função que receba uma string e um argumento de palavra-chave opcional chamado "substituicao", que indica o caractere que deve ser usado para substituir as letras da string.
33. Crie uma função que receba uma lista de strings e um argumento de palavra-chave opcional chamado "letras", que indica as letras que devem ser removidas de cada string da lista.
34. Escreva uma função que receba um dicionário e dois argumentos de palavra-chave opcionais chamados "inicio" e "fim", que indicam as chaves do dicionário que devem ser retornadas.
35. Crie uma função que receba uma lista de números e um argumento de palavra-chave opcional chamado "media", que indica se a média ou a mediana deve ser retornada.
36. Escreva uma função que receba uma string e um argumento de palavra-chave opcional chamado "substrings", que indica as substrings que devem ser removidas da string.
37. Crie uma função que receba uma lista de strings e um argumento de palavra-chave opcional chamado "palavra", que indica a palavra que deve ser contada em cada string da lista.
38. Escreva uma função que receba um dicionário e um argumento de palavra-chave opcional chamado "valor", que indica o valor que deve ser retornado do dicionário.
39. Crie uma função que receba uma lista de números e um argumento de palavra-chave opcional chamado "excluidos", que indica os números que devem ser excluídos da lista.
40. Escreva uma função que receba uma string e um argumento de palavra-chave opcional chamado "caracteres", que indica os caracteres que devem ser removidos da string.
41. Crie uma função que calcule a média aritmética de uma lista de números. O argumento opcional "pesos" pode ser usado para ponderar a média.
42. Escreva uma função que receba uma string e um argumento opcional "maiuscula" que, se True, retorna a string em letras maiúsculas.
43. Crie uma função que converta uma temperatura de Celsius para Fahrenheit. O argumento opcional "escala" pode ser usado para indicar qual escala deve ser usada na conversão.
44. Escreva uma função que receba um número inteiro e um argumento opcional "base" que, se fornecido, retorna o número na base indicada.
45. Crie uma função que verifique se uma lista contém um determinado elemento. O argumento opcional "inicio" pode ser usado para indicar a posição de início da verificação.

46. Escreva uma função que verifique se uma string é um palíndromo. O argumento opcional "ignorar_espacos" pode ser usado para ignorar espaços em branco na verificação.
47. Crie uma função que calcule a soma de uma lista de números. O argumento opcional "intervalo" pode ser usado para indicar o intervalo de soma.
48. Escreva uma função que verifique se uma lista é ordenada em ordem crescente. O argumento opcional "descendente" pode ser usado para verificar a ordenação em ordem decrescente.
49. Crie uma função que calcule o fatorial de um número inteiro. O argumento opcional "recursivo" pode ser usado para calcular o fatorial de forma recursiva.
50. Escreva uma função que verifique se uma string contém apenas letras maiúsculas. O argumento opcional "ignorar_espacos" pode ser usado para ignorar espaços em branco na verificação.
51. Crie uma função que calcule a potência de um número. O argumento opcional "exponencial" pode ser usado para indicar a exponencial a ser usada na potenciação.
52. Escreva uma função que verifique se uma lista contém elementos duplicados. O argumento opcional "tolerancia" pode ser usado para indicar a tolerância na verificação de duplicados.
53. Crie uma função que verifique se uma lista é simétrica. O argumento opcional "ordenada" pode ser usado para verificar a simetria da lista ordenada.
54. Escreva uma função que verifique se uma string contém apenas dígitos. O argumento opcional "com_sinal" pode ser usado para permitir sinal de positivo ou negativo na string.
55. Crie uma função que calcule o produto escalar de duas listas de números. O argumento opcional "soma" pode ser usado para indicar se o resultado deve ser a soma ou a média dos produtos.
56. Escreva uma função que verifique se uma lista é um palíndromo. O argumento opcional "ordenada" pode ser usado para verificar a igualdade da lista ordenada.
57. Crie uma função que calcule o módulo de um número complexo. O argumento opcional "simplificado" pode ser usado para retornar o módulo em sua forma simplificada.
58. Escreva uma função que verifique se uma string contém apenas letras minúsculas. O argumento opcional "ignorar_espacos" pode ser usado para ignorar espaços em branco na verificação.
59. Crie uma função que calcule o valor absoluto de um número. O argumento opcional "com_sinal" pode ser usado para manter o sinal do número.
60. Escreva uma função que verifique se uma lista é composta apenas por números inteiros. O argumento opcional "tolerancia" pode ser usado para permitir uma margem de erro na verificação.