PROGRAMAÇÃO I

Curso Técnico Subsequente em Informática Lucas Sampaio Leite





- O Python permite que uma função receba um número variável de argumentos, sem precisar defini-los previamente.
- Para isso, utiliza-se um asterisco * antes do nome do parâmetro na definição da função.
- Todos os argumentos fornecidos após o asterisco são coletados em uma tupla, que pode ser manipulada dentro da função.

```
def soma_numeros(*numeros):
   total = sum(numeros)
   return total
```

Uma tupla é uma estrutura de dados ordenada e imutável que permite armazenar múltiplos valores de diferentes tipos em uma única variável.



```
def soma_numeros(*numeros):
    total = sum(numeros)
    return total

print(soma_numeros(2, 4, 6))
print(soma_numeros(1, 3, 5, 7, 9))
print(soma_numeros())
```



```
def listar_nomes(*nomes):
    print("Lista de nomes:")
    for nome in nomes:
        print(f"- {nome}")

listar_nomes("Ana", "Carlos", "Maria")
listar_nomes("Lucas")
Lista de nomes:
- Carlos
- Maria
Lista de nomes:
- Lucas
```



- Para criar um dicionário com base em uma lista de argumentos, utiliza-se o duplo asterisco ** antes do nome do parâmetro na definição da função.
- Dessa forma, todos os argumentos fornecidos devem ser passados como pares chave=valor, e serão coletados dentro de um dicionário que pode ser manipulado na função.

```
def imprimir_info(**informacoes):
    for chave, valor in informacoes.items():
        print(f"{chave}: {valor}")
```

Um dicionário é uma estrutura de dados semelhante a listas, mas que armazena valores de forma não ordenada e mutável, por meio de pares chave:valor, permitindo acesso rápido aos elementos pelas suas chaves.



```
def imprimir_info(**informacoes):
    for chave, valor in informacoes.items():
        print(f"{chave}: {valor}")

imprimir_info(nome="Lucas", idade=25, curso="Informática")
imprimir_info(pais="Brasil", cidade="Aracaju")
```



nome: Lucas

idade: 25

curso: Informática

pais: Brasil

cidade: Aracaju



- Em Python, é possível definir valores padrão para os parâmetros de uma função.
- Isso significa que a função pode ou não receber um valor para esses parâmetros no momento da chamada.
 - Se um valor for fornecido, ele será utilizado normalmente.
 - Caso contrário, a função assumirá automaticamente o valor padrão definido na sua declaração.

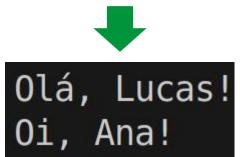
```
def saudacao(nome, mensagem = "0lá"):
    print(f"{mensagem}, {nome}!")

saudacao("Lucas")
saudacao("Ana", mensagem="0i")
```



```
def saudacao(nome, mensagem = "Olá"):
    print(f"{mensagem}, {nome}!")

saudacao("Lucas")
saudacao("Ana", mensagem="Oi")
```





```
def aplicar_desconto(preco, desconto = 0.10):
    valor_final = preco * (1 - desconto)
    return valor_final

print(aplicar_desconto(100))
print(aplicar_desconto(200, 0.25))
```



90.0 150.0



- Em Python, podemos indicar os tipos esperados dos parâmetros e o tipo de retorno de uma função utilizando anotações de tipo (*type annotations*).
- As anotações de tipo são opcionais e servem para documentar o código, tornando-o mais legível e facilitando a detecção de erros por ferramentas externas (como *linters* e *type checkers*).
- Elas não alteram a execução do programa em tempo de execução, funcionando apenas como uma forma de especificar os tipos esperados dos

argumentos e do valor de retorno.

```
def soma(a: int, b:int) -> int:
    resultado = a + b
    return resultado
x = soma(5, 6)
print(x)
```



```
def soma_lista(numeros: list[int]) -> int:
    resultado = sum(numeros)
    return resultado

valores = [2, 4, 6, 8]
print(soma_lista(valores))
```



- As anotações de tipo em Python são opcionais e não alteram o comportamento da função em tempo de execução.
- Servem como uma ferramenta de documentação e apoio, ajudando a compreender melhor a função e o modo correto de utilizá-la.
- Por padrão, essas anotações são apenas uma convenção e não são verificadas automaticamente pelo interpretador Python.
- Para validar se os tipos dos argumentos e do valor de retorno estão de acordo com as anotações, é possível utilizar ferramentas externas, como o mypy, que realizam a verificação de consistência de tipos durante o desenvolvimento.



- Funções da biblioteca padrão do Python:
 - São funções que já vêm incluídas com o Python quando você o instala.
 - Não é necessário instalar nada extra para usá-las; basta importar o módulo correspondente.

• Exemplos:

- math.sqrt() → calcula a raiz quadrada.
- random.randint() → gera números aleatórios.
- os.listdir() → lista arquivos de um diretório.
- Elas permitem realizar operações comuns sem precisar escrever o código do zero.



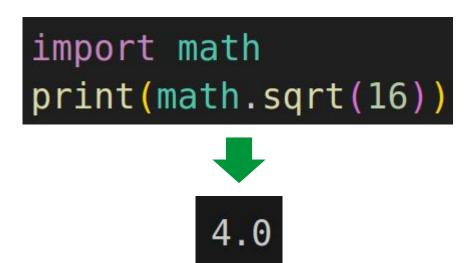
- Funções de terceiros
 - São funções criadas por outras pessoas ou comunidades, que não vêm instaladas junto com o Python.
 - Para utilizá-las, é necessário instalar a biblioteca usando o pip.

• Exemplos:

- numpy.array() → biblioteca NumPy para trabalhar com arrays e matemática avançada.
- pandas.DataFrame() → biblioteca Pandas para manipulação de dados.
- requests.get() → biblioteca Requests para fazer requisições HTTP.
- Elas permitem expandir muito as funcionalidades do Python, trazendo soluções prontas para problemas específicos.



- Importação de módulos:
 - Para acessar funções de outros módulos (bibliotecas), usamos a instrução import.
 - Após importar, chamamos as funções usando a sintaxe módulo.função().
- Exemplo:





- Importando funções específicas:
 - Podemos importar apenas a função desejada.
- Exemplo:

```
from math import sqrt print(sqrt(16))

4.0
```



- Atribuindo apelidos (alias):
 - Útil para abreviar nomes longos de módulos.

area = area circulo(raio)

```
• Exemplo: import math as m

def area_circulo(raio: float) -> float:
    return m.pi * m.pow(raio, 2)

raio = 5
```



print(f"A área do círculo de raio {raio} é {area:.2f}")

A área do círculo de raio 5 é 78.54

Exercícios



- 1. Escreva uma função que receba um número de parâmetros indefinido. Imprima a quantidade de parâmetros recebidos de cada tipo de dado. A função também deve imprimir o maior e o menor valor numérico recebido.
- 2. Faça um programa que simule um lançamento de dados. Lance o dado 100 vezes e armazene os resultados em uma lista. Utilize uma lista de contadores (1-6) e depois da execução dos sorteios, mostre quantas vezes cada valor foi conseguido.
 - Dica: sorteios em python podem ser realizados através da biblioteca random, pelas funções randint() e randrange().
 - Documentação: https://docs.python.org/pt-br/3.8/library/random.html

Exercícios



- 3. Construa uma função que receba uma string como parâmetro e devolva outra string com os carateres embaralhados.
 - Por exemplo: se função receber a palavra python, pode retornar npthyo, ophtyn ou qualquer outra combinação possível, de forma aleatória.
 - Padronize a sua função para que todos os caracteres sejam devolvidos em caixa alta ou caixa baixa, independentemente de como foram digitados.

Dúvidas





PROGRAMAÇÃO I

Curso Técnico Subsequente em Informática Lucas Sampaio Leite

