As **lambda expressions** (ou funções lambda) em Python são uma forma de criar **funções anônimas** (sem nome) de maneira curta e direta, tudo em uma única linha.

Sintaxe básica

```python
lambda argumentos: expressão
```

- `lambda` → palavra-chave que cria a função
- `argumentos` → zero ou mais parâmetros (separados por vírgula)
- `:` → separa os argumentos do corpo da função
- `expressão` → o que a função retorna (é avaliado e retornado automaticamente)

Exemplos simples

```python
# Função normal equivalente a uma lambda
def soma(x, y):
    return x + y

# A mesma coisa com lambda
soma = lambda x, y: x + y

print(soma(5, 3))  # 8
```

Python

Copiar

```python
# Lambda sem argumentos
ola = lambda: print("Olá mundo!")
ola()  # Olá mundo!

# Lambda com um argumento
quadrado = lambda x: x ** 2
print(quadrado(5))  # 25

# Lambda com argumentos default
potencia = lambda x, exp=2: x ** exp
print(potencia(3))     # 9  (usa exp=2)
print(potencia(3, 4))  # 81
```

Principais usos de lambda (onde elas brilham)

1. **Com funções de ordem superior (higher-order functions)**
    - `map()`
    - `filter()`
    - `sorted()` / `sort()`
    - `key=` em várias funções

```python
numeros = [1, 2, 3, 4, 5]

# map: aplicar uma função a cada item
quadrados = list(map(lambda x: x**2, numeros))
print(quadrados)  # [1, 4, 9, 16, 25]

# filter: manter apenas itens que satisfazem uma condição
pares = list(filter(lambda x: x % 2 == 0, numeros))
print(pares)  # [2, 4]

# sorted com chave personalizada
pessoas = [('Ana', 25), ('Bia', 20), ('Carlos', 30)]
ordenadas_por_idade = sorted(pessoas, key=lambda pessoa: pessoa[1])
print(ordenadas_por_idade)
# [('Bia', 20), ('Ana', 25), ('Carlos', 30)]
```

1. **Dentro de funções que recebem outras funções como parâmetro**

```python
def fazer_operacao(a, b, operacao):
    return operacao(a, b)

soma = fazer_operacao(10, 5, lambda x, y: x + y)      # 15
subtracao = fazer_operacao(10, 5, lambda x, y: x - y) # 5
multiplicacao = fazer_operacao(10, 5, lambda x, y: x * y) # 50
```

Limitações importantes

- Só podem conter **uma única expressão** (não podem ter vários statements como `if` com bloco, `for`, `while`, etc.)
- Não podem ter anotações de tipo nem docstrings
- São menos legíveis quando ficam muito complexas

Ruim (difícil de ler):

```python
lista = sorted(dados, key=lambda x: x[2] if x[2] is not None else 0)
```

Melhor (mais legível):

```python
def chave(x):
    return x[2] if x[2] is not None else 0

lista = sorted(dados, key=chave)
```

Resumo rápido

Característica	Função normal (`def`)	Lambda
Nome	Tem nome	Anônima (pode ser atribuída)
Tamanho	Várias linhas possíveis	Apenas uma expressão
Uso comum	Lógica complexa, reutilização	Passar funções rápidas inline
Legibilidade em código longo	Alta	Baixa quando complexa

As lambdas são uma ferramenta poderosa para deixar o código mais conciso, especialmente quando você precisa de uma função simples só naquele momento (com `map`, `filter`, `sorted`, etc.). Use com moderação para não prejudicar a legibilidade! 😊

Se quiser mais exemplos práticos ou exercícios, é só pedir!

↳ Exemplos com map e filter

↳ Funções de ordem superior