



PYTHON
ACADEMY

O QUE SÃO FUNÇÕES LAMBDA NO PYTHON

Guia completo de Funções Lambda em Python: sintaxe, casos de uso reais (sort, filter, map, max/min), quando usar funções anônimas e quando evitar complexidade.

[PYTHONACADEMY.COM.BR](https://pythonacademy.com.br)

Gere ebooks como este com



em <https://ebookr.ai>

Crie ebooks profissionais incríveis em minutos com IA



Chega de formatar texto no Google Docs, Word ou ferramentas que só te fazem perder tempo...

E deixe que nossa IA faça o trabalho pesado!



Capas gerados por IA



Infográficos feitos por IA



Edite em Markdown em Tempo Real



Adicione Banners Promocionais

TESTE AGORA

PRIMEIRO CAPÍTULO 100% GRÁTIS

✓ **Atualizado para Python 3.13** (Dezembro 2025) Conteúdo enriquecido com casos de uso do mundo real e análise de quando usar vs funções normais.

Salve salve Pythonista!

Funções lambda são uma ferramenta elegante do Python que permite criar **funções anônimas** (sem nome) de forma concisa. São ideais para operações **simples e únicas** onde uma função completa seria excessiva.

Neste guia, você vai aprender:

- ✓ Sintaxe e criação de funções lambda
- ✓ **Casos de uso reais** (sort, filter, map, max/min)
- ✓ Lambda vs funções normais (def)
- ✓ Quando usar e quando **NÃO usar** lambdas

Como criar Funções lambda

Uma função lambda é criada usando a palavra-chave `lambda`, seguida de um ou mais **argumentos**, e uma **expressão**:

- **argumentos** são os dados de entrada que esta função irá receber
- **expressão** é o código que será executado quando a função lambda for chamada.

Sua sintaxe básica é a seguinte:

```
lambda {argumentos}: {expressão}
```

Veja o exemplo abaixo:

```
soma = lambda x, y: x + y  
print(soma(2, 3))
```

Nela:

- Os **argumentos** são 2: `x` e `y`
- A **expressão** a ser executada: `x + y`

A saída do código acima será o resultado de `soma(2, 3)`:

```
5
```

Podemos armazenar a função lambda em uma variável, como no exemplo acima, ou chamá-la diretamente:

```
print((lambda x, y: x + y)(2, 3))
```

E a saída será a mesma.

Note o conjunto de parentêses para criação da função e outro para os argumentos.

Dessa forma a função lambda está sendo criada e já chamada!

Usando funções lambda com `map`, `filter` e `reduce`

Uma das principais utilidades das funções lambda é quando elas são usadas em conjunto com as funções `map`, `filter` e `reduce` (agora no pacote `functools`).

*Ainda não domina a utilização das funções `map` e `filter`? Então não perca tempo e aprenda essas duas funções incríveis **AGORA** [clcando aqui para ler esse artigo completo!](#)*

Veja o exemplo abaixo, onde usamos a função `map` para aplicar uma função lambda que eleva cada elemento de uma lista ao quadrado:

```
lista = [1, 2, 3, 4]

print(list(map(lambda x: x ** 2, lista)))
```

E a saída será:

```
[1, 4, 9, 16]
```

No exemplo abaixo, usamos a função `filter` para retornar apenas os números pares de uma lista:

```
lista = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

print(list(filter(lambda x: x % 2 == 0, lista)))
```

Gerando a saída:

```
[2, 4, 6, 8, 10]
```

Por fim, veja o exemplo a seguir, onde usamos a função `reduce` para somar todos os elementos de uma lista:

```
from functools import reduce

lista = [1, 2, 3, 4, 5]

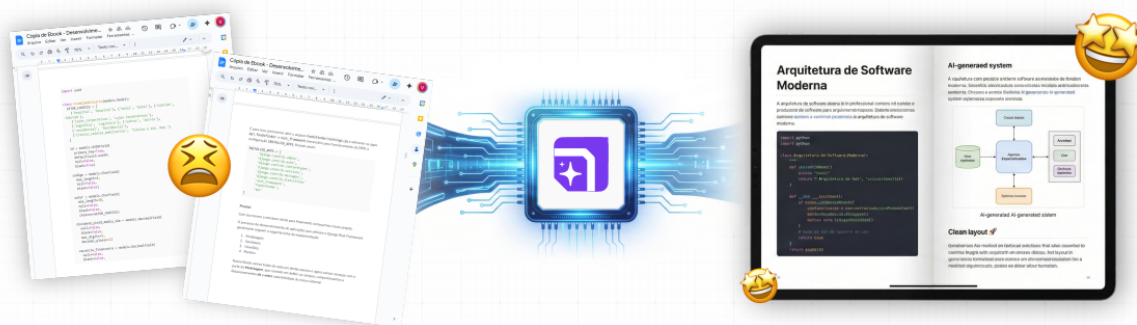
print(reduce(lambda x, y: x + y, lista))
```

```
15
```



Criei o [Ebookr.ai](https://ebookr.ai), uma plataforma que usa IA para gerar ebooks profissionais sobre qualquer tema — com capa gerada por IA, infográficos automáticos e exportação em PDF. Confere!


Crie Ebooks profissionais incríveis em minutos com IA




Chega de formatar texto no Google Docs, Word ou ferramentas que só te fazem perder tempo...


... e deixe que nossa IA faça o trabalho pesado!

TESTE AGORA! PRIMEIRO CAPÍTULO 100% GRÁTIS [↗](#)

 Capas gerados por IA

 Adicione Banners Promocionais

 Edite em Markdown em Tempo Real

 Infográficos feitos por IA

Casos de Uso do Mundo Real

Vamos ver onde lambdas são úteis na prática:

1. Ordenar Listas Complexas

```
# Ordenar lista de dicionários por chave
users = [
    {'name': 'Alice', 'age': 30},
    {'name': 'Bob', 'age': 25},
    {'name': 'Charlie', 'age': 35}
]

# Ordenar por idade
sorted_by_age = sorted(users, key=lambda u: u['age'])
# [{'name': 'Bob', 'age': 25}, ...]

# Ordenar por nome
sorted_by_name = sorted(users, key=lambda u: u['name'])
# [{'name': 'Alice', ...}, ...]

# Ordenar por múltiplos critérios (idade desc, nome asc)
sorted_users = sorted(users, key=lambda u: (-u['age'], u['name']))
```

2. Encontrar Máximo/Mínimo Customizado

```
products = [
    {'name': 'Notebook', 'price': 2500, 'stock': 10},
    {'name': 'Mouse', 'price': 50, 'stock': 100},
    {'name': 'Teclado', 'price': 150, 'stock': 50}
]

# Produto mais caro
most_expensive = max(products, key=lambda p: p['price'])
print(most_expensive) # {'name': 'Notebook', 'price': 2500, ...}

# Produto com menor estoque
lowest_stock = min(products, key=lambda p: p['stock'])
print(lowest_stock) # {'name': 'Notebook', 'stock': 10, ...}
```


3. Filtrar e Transformar Dados

```
# Filtrar números pares e elevar ao quadrado
numbers = [1, 2, 3, 4, 5, 6]

# Com lambda
evens_squared = list(map(lambda x: x**2, filter(lambda x: x % 2 == 0,
                                                numbers)))
# [4, 16, 36]

# Equivalente com list comprehension (mais legível)
evens_squared = [x**2 for x in numbers if x % 2 == 0]
```

4. Funções de Alta Ordem

```
# Criar funções multiplicadoras
def make_multiplier(n):
    return lambda x: x * n

double = make_multiplier(2)
triple = make_multiplier(3)

print(double(5)) # 10
print(triple(5)) # 15
```

Lambda vs Funções Normais (def)

Quando Usar Lambda:

✓ Operações simples e únicas

- key em `sorted()`, `max()`, `min()`

- Callback simples

✓ **Expressão de uma linha**

- Sem lógica condicional complexa
- Fácil de entender

✓ **Usado uma vez**

- Não será reutilizado

Quando Usar def (Função Normal):

✓ **Lógica complexa**

- Múltiplas linhas
- Condições if/else aninhadas

✓ **Reutilização**

- Usada em vários lugares
- Precisa de nome descritivo

✓ **Docstring necessária**

- Precisa documentação
- API pública

Exemplos de Quando NÃO Usar Lambda:

```
# ❌ RUIM: Lambda complexa (ilegível)
calculate = lambda x: x**2 if x > 0 else -x**2 if x < 0 else 0

# ✅ BOM: Função normal (legível)
def calculate(x):
    """Calcula quadrado com sinal"""
    if x > 0:
        return x**2
    elif x < 0:
        return -x**2
    return 0

# ❌ RUIM: Lambda reutilizada (sem nome descritivo)
process_data = lambda data: [x.strip().upper() for x in data]
result1 = process_data(data1)
result2 = process_data(data2)

# ✅ BOM: Função nomeada (clareza)
def normalize_data(data):
    """Normaliza dados: remove espaços e converte para maiúsculas"""
    return [x.strip().upper() for x in data]
```

Conclusão

Neste guia completo sobre **Funções Lambda**, você aprendeu:

✅ **Sintaxe lambda** - Funções anônimas de uma linha ✅ **Casos de uso reais** - sort, filter, map, max/min ✅ **Lambda vs def** - Quando usar cada uma ✅ **Quando NÃO usar** - Complexidade e reutilização

Principais lições:

- Lambdas são ideais para **operações simples e únicas**

- Perfeitas como **key em sorted()** e callbacks
- Use **def** quando lógica é **complexa** ou **reutilizada**
- Lambdas NÃO têm **docstrings** ou **nomes descritivos**
- **Legibilidade > concisão** - se está confuso, use def

Próximos passos:

- Pratique com `sorted()`, `max()`, `min()`
- Explore `functools.partial` para alternativa a lambdas
- Combine com `map()`, `filter()` (mas list comprehension é mais Pythônico)
- Estude funções de alta ordem

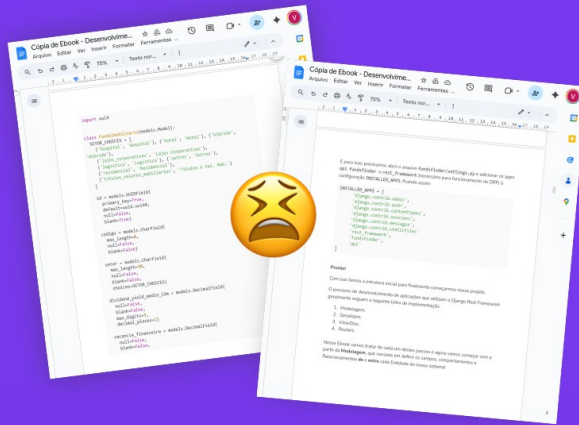
Agora que você domina funções lambda, experimente usá-las em seus próprios projetos e veja como elas podem simplificar seu código!

Não se esqueça de conferir!



Ebookr

Crie Ebooks profissionais em minutos com IA



Chega de formatar código no Google Docs ou Word



Capas gerados por IA



Infográficos feitos por IA



Adicione Banners Promocionais

Deixe que nossa IA faça o trabalho pesado



Edite em Markdown em Tempo Real

TESTE AGORA



PRIMEIRO CAPÍTULO 100% GRÁTIS