PYTHON ACADEMY

AS FUNÇÕES MAP E FILTER DO PYTHON

Neste ebook iremos falar sobre as funções map e filter e como elas podem ser usadas para produzir códigos mais pythônicos.

PYTHONACADEMY.COM.BR

Este ebook foi gerado por

DevBook

Crie Ebooks técnicos incríveis em minutos com IA

Conheça a 1ª IA Especializada na criação de Ebooks **com código**!

Chega de formatar código no Google Docs

Deixe que nossa IA faça o trabalho pesado

- Syntax Highlight
- Adicione Banners Promocionais
- Edite em Markdown em Tempo Real
- Infográficos feitos por IA

TESTE AGORA

PRIMEIRO CAPÍTULO 100% GRÁTIS

Olá Pythonista!

Neste artigo, iremos falar sobre as funções `map` e `filter` do Python e como elas podem ser usadas para otimizar seu código sem ter que utilizar laços de repetição!

> *Ainda não domina **Laços de Repetição** utilizando Python? Então acesse agora nosso artigo sobre loops utilizando `for` e loops utilizando `while` AGORA!!*

As funções `map` e `filter` são de extrema importância para a construção de códigos Pythônicos e são contribuições da programação funcional para o Python!

Agora sem enrolação, vamos ao conteúdo!

A função `map`

Muitas vezes, enfrentamos situações em que precisamos executar a **mesma operação** em todos os itens de um iterável de entrada (uma lista, um set, uma tupla) para se construir um novo iterável.

A abordagem mais comum para esse tipo de problema é usar loops `for` ou `while` do Python.

Contudo, você também pode resolver esse problema sem um loop de repetição, usando a função `map()` disponível na biblioteca padrão da própria linguagem (sem precisar instalar algo com `pip` ou importar com `import`).

A sintaxe da função `map` é a seguinte:

```
map(funcao, iteravel1, iteravel2, ..., iteravelN)
```

Nela: - `funcao` é a função que iremos aplicar a cada elemento dos iteráveis de entrada. - `iteravel1, ..., iteravelN` são iteráveis nas quais a função `funcao` será aplicada, elemento a elemento

Para entender melhor, vamos aos **exemplos**!

Exemplos da função `map`

Suponha que você tenha uma lista de números e que lhe pedissem para somar 5 a todos os valores da lista.

A forma mais comum seria utilizar um loop `for`, assim:

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

for item in lista:
    resultado.append(item + 5)
```

É possível conseguir o mesmo resultado utilizando a função `map`!

Veja como o código acima poderia ser reescrito, utilizando esta função:

```
def soma_5(numero):
    return numero + 5

lista = [1, 2, 3, 4, 5]
lesultado = list(map(soma_5, lista))
```

Desta forma, estamos aplicando a função `soma_5` a cada elemento da lista `lista` e o resultado será o mesmo da versão com o loop `for`.

A função `map` com múltiplos iteráveis

Como vimos na sintaxe da função, `map` possibilita passarmos **múltiplos** iteráveis de entrada.

Para isso, é necessária uma função que possua dois ou mais argumentos.

Vamos entender no exemplo a seguir!

```
from math import pow

numeros = [1, 2, 3]
expoentes = [4, 5, 6, 7]

resultado = list(map(pow, numeros, expoentes))
```

Agora acompanhe a execução de cada iteração de `map`:

👉 Na primeira iteração, `map` vai aplicar a função `pow`, utilizando `numeros[0]` e `expoentes[0]` como argumentos, assim: `pow(1,4) = 1`

👉 Em seguida, `map` vai aplicar a função `pow`, utilizando `numeros[1]` e `expoentes[1]`, assim: `pow(2,5) = 32`

👉 Por último, `map` vai aplicar a função `pow`, utilizando `números[2]` e `expoentes[2]`, assim: `pow(3,6) = 729`

EBOOK GERADO POR DevBook
ACESSE DEVBOOK.AI ✨ PRIMEIRO CAPÍTULO 100% GRÁTIS

A iteração termina na menor sequência, portanto o número 7, em `expoentes[3]`, não será utilizado pois a lista `numeros` não possui o elemento `numeros[3]`.

> 💡 *Estou desenvolvendo o **DevBook**, uma plataforma que usa IA para gerar ebooks técnicos profissionais. Depois de ler, dá uma passada no site!*

A função `filter`

A função `filter` do Python serve para **filtrar elementos** de iteráveis que satisfaçam determinadas **condições**.

Essas **condições** são calculadas através de uma função que deve retornar `True` ou `False` para determinado argumento de entrada.

A abordagem mais comum, assim como a função `map`, seria a utilização de **loops** para iterar sobre os elementos verificando elemento a elemento se a condição foi satisfeita ou não.

Contudo, com a função `filter` é possível fazer isso sem ter que se preocupar com as estruturas de repetição `for` ou `while`!

A sintaxe da função filter é:

```
filter(funcao, iteravel)
```

Nela:

- `funcao` é a função de filtragem dos elementos, que deve retornar `True` ou `False`; e
- `iteravel` é a estrutura na qual a função `funcao` será aplicada.

Vamos entender melhor com **exemplos**!

Exemplos da função `filter`

Suponha que você tenha que desenvolver uma função que filtra elementos **maiores que zero** de uma lista.

O código abaixo, utilizando o loop de repetição `for` seria uma possível solução:

```python
def numeros_positivos(numeros):
    numeros_positivos = []

    for num in numeros:
        # Condição de filtragem
        if num > 0:
            numeros_positivos.append(num)

    return numeros_positivos

print(numeros_positivos([-2, -1, 0, 1, 2]))
```

E a saída seria:

```
[1, 2]
```

Veja como seria possível chegar ao mesmo resultado utilizando a função `filter`:

```python
numeros = [-2, -1, 0, 1, 2]

def verifica_numero_positivo(numero):
    return numero > 0

print(list(filter(verifica_numero_positivo, numeros)))
```

E a saída seria exatamente a mesma: com menos linhas de código e com um código muito mais pythônico!

Dica EXTRA: Funções Lambda

Quer deixar seu código ainda mais **pythônico**?

Utilize **funções lambda** em conjunto com as funções `map` e `filter` e tenha os mesmos resultados dos exemplos anteriores com ainda menos linhas de código!

> *Ainda não domina o conceito de **Funções Lambda**?! Não tem problema, acesse nosso artigo sobre Funções Lambda antes de continuar* 😉

Exemplo "soma + 5" do `map` utilizando **funções lambda**:

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

resultado = list(map(lambda x: x+5, lista))
```

Exemplo de filtragem de números positivos do `filter` utilizando **funções lambda**:

```
numeros = [-2, -1, 0, 1, 2]

numeros_positivos = list(filter(lambda x: x > 0, numeros))
```

Use `map`, `filter` em **funções lambda** na sua próxima entrevista de emprego e **impressione** o recrutador!

Conclusão

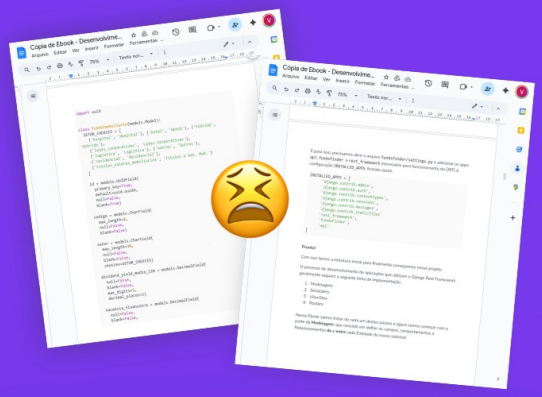
Depois de termos nos aprofundado nas funções **map**, **filter**, além das **funções lambda**, você poderá treinar essas habilidades a fim de se tornar um programador cada vez **melhor**!

Não se esqueça de conferir!

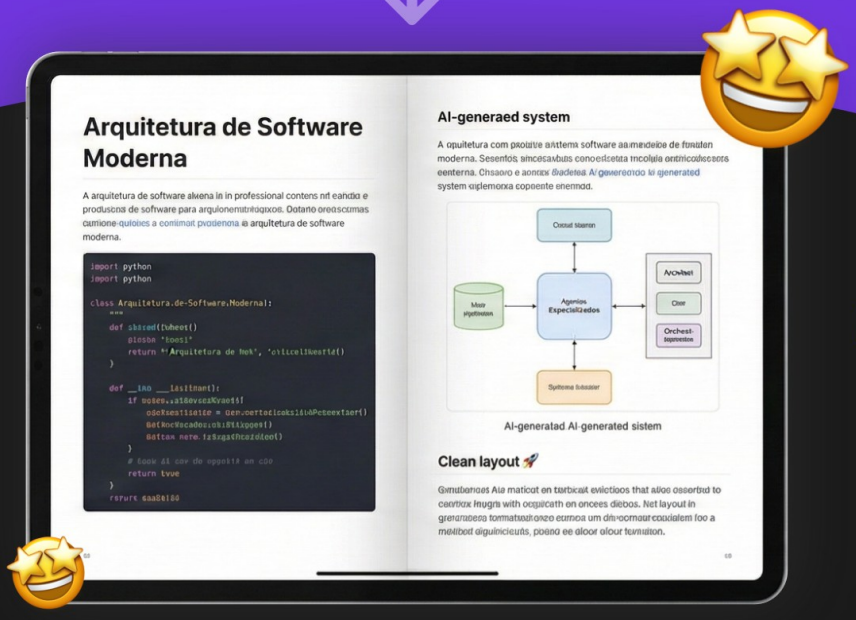
DevBook

Crie Ebooks técnicos em minutos com IA

Conheça a 1ª IA Especializada na criação de Ebooks **com código**!



Chega de formatar código no Google Docs



Syntax Highlight

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