



DOMINE SET COMPREHENSIONS NO PYTHON

Domine Sets Comprehensions (ou Compreensão de Sets) com esse ebook completo sobre o assunto! Set Comprehensions possibilita criar e manipular sets de uma maneira concisa e eficiente!

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 

Olá Pythonista!

Nesse post vamos falar sobre uma técnica muito Pythônica de se manipular Sets: as chamadas *Set Comprehensions* ou Compreensão de Sets!

Com ela é possível criar e iterar sobre sets de uma maneira muito eficiente e concisa!

Mas... Se você ainda não domina o assunto, corre no nosso [post completo sobre Sets aqui da Python Academy!](#)

Bora pro post!

Set Comprehension

Set Comprehension é uma técnica presente na Linguagem que nos possibilita criar sets a partir de outros sets de uma maneira bem Pythônica.

Esse conceito também está presente nas listas, com as chamadas *List Comprehensions* ([e veja só se não temos um post completo sobre o assunto 😊](#)) e nos Dicionários, com as chamadas *Dict Comprehensions* ([e não é que TAMBÉM temos um post completo sobre o assunto!](#)).

Sua sintaxe base consiste em utilizar a estrutura de repetição `for ... in ...` dentro de chaves `{}`, da seguinte maneira:

```
{ expressão for variável in iterável }
```

Vamos agora a um exemplo prático:

```
lista = [1, 1, 2, 3, 4]
set_comp = {num for num in lista}

print(set_comp)
print(type(set_comp))
```

Saída:

```
{1, 2, 3, 4}
<class 'set'>
```

Agora vamos explicar passo a passo: - Estamos aplicando *set comprehension* à lista `lista` para gerar um *set*, ao final. - A cada iteração, o *set* resultante estará recebendo o valor da lista. Primeiro 1, depois 1 novamente e **OPA**: sets não permitem dados duplicados, então o segundo 1 **vaza**! Em seguida, o 2, depois o 3 e por fim, o 4. - Ao final, temos o *set* resultante: `{1, 2, 3, 4}`

Agora, vamos aumentar o nível!

Vamos multiplicar por 2 todos os números do iterável:

```
iteravel = [0, 1, 2, 3, 4]

set_comp = {num * 2 for num in iteravel}
print(set_comp)
```

Resultando no seguinte *set*:

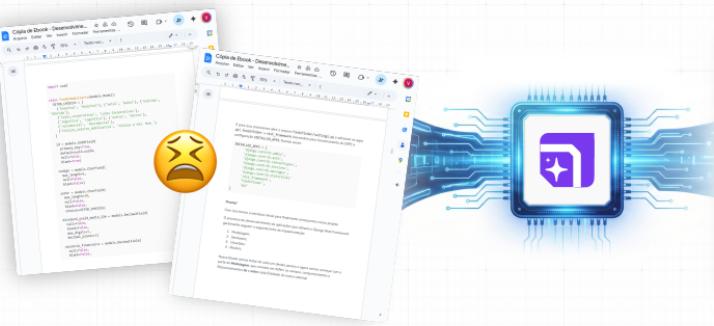
```
{0, 2, 4, 6, 8}
```

 Estou desenvolvendo o **DevBook**, uma plataforma que usa IA para gerar ebooks técnicos profissionais. Não deixe de conferir clicando no botão abaixo!

 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 

Set Comprehensions com Condicional If

Também é possível adicionar expressões condicionais à *Set Comprehensions*.

Sua sintaxe básica é a seguinte:

```
{ expressão for variável in iterável if expressão }
```

Ou seja, só estará presente no *set* resultante as variáveis que passarem na condição `if` do *Set Comprehensions*.

Deixa eu explicar com um exemplo!

Suponha que lhe peçam para desenvolver um código que crie um *set* com apenas os elementos de uma lista de strings que contenham o caracter `0`.

Portanto `100`, `105` e `10` estariam presentes no *set* de saída, mas `5`, `1` ou `2` não.

Uma forma de desenvolver esse código seria a seguinte:

```
iteravel = ['15', '20', '1', '100', '0']

set_comp = { num for num in iteravel if '0' in str(num) }

print(set_comp)
```

Veja a saída:

```
{20, 100, 0}
```

Set Comprehensions com Múltiplos Condicionais If

Podemos ainda adicionar vários condicionais para filtrar os elementos que estarão presentes no *set* de saída, após o processamento pelo *Set Comprehension*.

Sua sintaxe base é:

```
{ expressão for variável in iterável if expressão_1 if expressão_2  
    if ... }
```

É similar ao caso de uma condicional apenas. **Vamos ao exemplo!**

Suponha que lhe tenha sido pedido para encontrar os números divisíveis por 2 e 4 de determinado conjunto de números.

Isso poderia ser feito da seguinte maneira:

```
iteravel = range(20)  
  
set_comp = {num for num in iteravel if num % 2 == 0 if num % 4 == 0}  
  
print(set_comp)
```

Veja a saída:

```
{0, 4, 8, 12, 16}
```

Usa-lo é uma ótima prática para percorrer iteráveis em poucas linhas.

Conclusão

Nesse Post vimos um conceito avançado da linguagem Python: os *Set Comprehensions*!

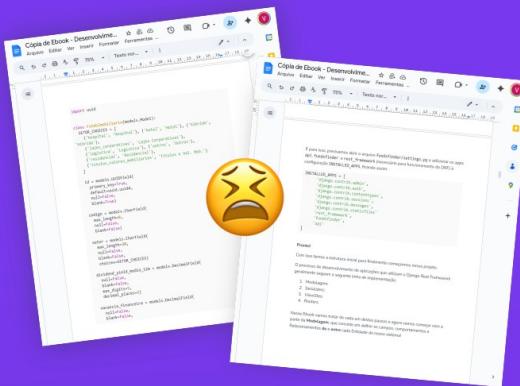
Aprenda a utilizá-los bem pois isso lhe fará um **verdadeiro Pythonista!**

Se ficou com alguma dúvida, fique à vontade para deixar um comentário no box aqui embaixo! Será um prazer te responder! 😊



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



Arquitetura de Software Moderna

A arquitetura de software alvo é profissional contendo o e-mail e produções de software para arquiteturas modernas. Oferece recursos como interface gráfica com interface de usuário.

```
import python
import python

class Arquitetura_de_Software_Moderna:
    ...
    def share(self):
        pass
    ...
    return "Arquitetura de NeXt", "arquitetura_moderna"
    ...

    def __init__(self):
        if user_authenticated():
            self.user_authenticated = user_authenticated()
            self.user_email = user_email()
            self.user_name = user_name()
        ...
        # Envie AI para gerar um código
        return type
    ...
    return self

    
```

AI-generated system

A arquitetura com propósito alvo é software amigável de usuários modernos. Seus recursos incluem conceitos de interface de usuário moderna, interface gráfica com interface de usuário.

AI-generated AI-generated system

```
graph TD
    UserInput[User input] --> DataProcessor[Data processor]
    DataProcessor --> AIEngine[AI engine]
    AIEngine --> GeneratedContent[Generated content]
    GeneratedContent --> OutputSystem[Output system]
    OutputSystem --> ArchDiagram[Architectural diagram]
    OutputSystem --> CodeDiagram[Code diagram]
    OutputSystem --> InfographicDiagram[Infographic diagram]
    
```

Clean layout

Garantimos que o layout é organizado e fácil de ler. O layout é gerado automaticamente, oferecendo uma experiência visual óptima.



</> 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