

ESTRUTURAS DE REPETIÇÃO USANDO LOOPS FOR NO PYTHON

Guia de loops for: iterar listas/strings/dicts, range(), enumerate(), break/continue, for-else, casos práticos (processa dados, soma valores), for vs while.

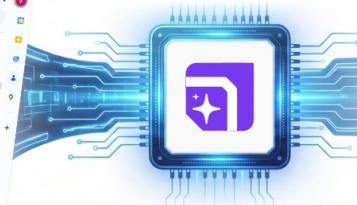
Gere ebooks como este com



Ebookr

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) Loop for com range, enumerate e casos práticos.

Seja muito bem-vindo Pythonista!

Nesse artigo, você vai aprender a criar estruturas de repetição utilizando o `for` do Python.

Estruturas de repetição - como o próprio nome já diz - são trechos de código onde você precisa aplicar determinados comandos repetidas vezes.

Eles são utilizados constantemente em códigos Python, você verá no seu dia a dia de Pythonista! 😊

Está preparado?! Então vamos nessa! 🚀

Estruturas de repetição

Loops ou **estruturas de repetição** são blocos básicos de qualquer linguagem de programação e são muito importantes!

Cada linguagem de programação possui uma sintaxe específica para criação destes *loops*.

Vamos ver nesse post como podemos fazer loops utilizando o `for` !

É essencial **DOMINAR** essa estrutura de repetição para se tornar um verdadeiro Pythonista!

Loops utilizando o `for`

O `for` é utilizado para percorrer ou iterar sobre uma sequência de dados (podendo ser uma lista, uma tupla, uma string), executando um conjunto de instruções em cada item.

Sua sintaxe básica é:

```
for {nome variável} in {variável iteração}:
    {código}
```

Agora vamos à explicação:

- `{variável iteração}` : é o nome da variável que vai receber os elementos do iterável a cada iteração
- `{iterável}` : é o container de dados sobre o qual vamos iterar, podendo ser: uma lista, uma tupla, um dicionário, entre outros.
- `{código}` : é o bloco de código que será executado a cada iteração (*loop*).

Vamos à um exemplo:

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

for item in lista:
    print(item)
```

Vamos entender o passo a passo:

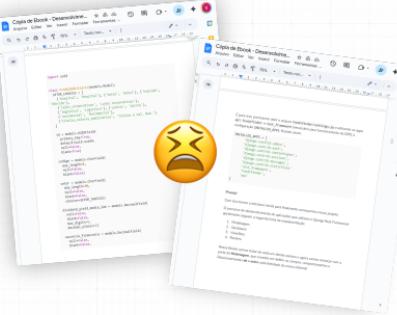
- Na primeira iteração, a variável `item` vai receber o valor do primeiro elemento da lista, que é 1. Portanto `print(item)` vai mostrar o valor 1.
- Na segunda iteração, `item` vai receber o valor do segundo elemento da lista `lista`, que é 2. Portanto `print(item)` vai mostrar o valor 2.

- E assim por diante até o último valor da lista, que é 5.

💡 Criei o [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

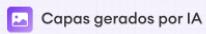


Chega de formatar texto no Google Docs, Word ou ferramentas que só te fazem perder tempo...

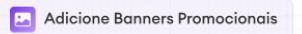


... e deixe que nossa IA faça o trabalho pesado!

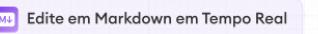
[TESTE AGORA! PRIMEIRO CAPÍTULO 100% GRÁTIS !\[\]\(268d17a79f5913c36d8998dd535a8ce6_img.jpg\)](#)



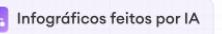
Capas gerados por IA



Adicione Banners Promocionais



Edita em Markdown em Tempo Real



Infográficos feitos por IA

O **break** e o **continue**

Existem duas palavras reservadas da linguagem que servem para auxiliar no controle de fluxo da estrutura de repetição. São elas: **break** e **continue** !

O comando break

Começando pelo `break`, nós o utilizamos para parar a execução de um loop.

Veja um exemplo:

```
for item in [1, 2, 3, 4, 5]:  
    if item == 3:  
        break  
    else:  
        print(item)
```

A saída será:

```
1  
2
```

Note que o loop foi interrompido quando a variável `item` teve seu valor igual à 3.

O comando continue

Já o `continue` serve para pular todo código que estiver **abaixo** da cláusula `continue`, dando sequência a próxima iteração do loop.

Veja como será a saída do código abaixo:

```
for item in [1, 2, 3, 4, 5]:  
    if item == 3:  
        continue  
    print(item)
```

A saída:

```
1  
2  
4  
5
```

Perceba que o número 3 **não aparece!**

Loops utilizando `for` e `else`

Você sabia que podemos usar o `else` em loops `for`?

E a resposta é: **SIM!**

Pois é, esse é um fato que poucos Pythonistas conhecem.

O `else` nos possibilita executar um bloco de código após o iterável ter sido completamente percorrido.

Contudo, o `else` não é executado quando o `for` encontra uma cláusula `break`!

Vamos entender melhor no exemplo:

```
for item in [1, 2, 3, 4, 5]:  
    if item == 6:  
        print('Encontramos o 6')  
        break  
    else:  
        print('Elemento 6 não foi encontrado')
```

Como o número 6 não está presente na lista, o código em `else` será executado!

Conclusão

Agora que você já sabe o funcionamento das estruturas de repetição utilizando `for` teste-as e verifique seus usos em diferentes casos, garanto que você irá aprender bastante!

Qualquer dúvida fique à vontade para utilizar o box de comentários abaixo!

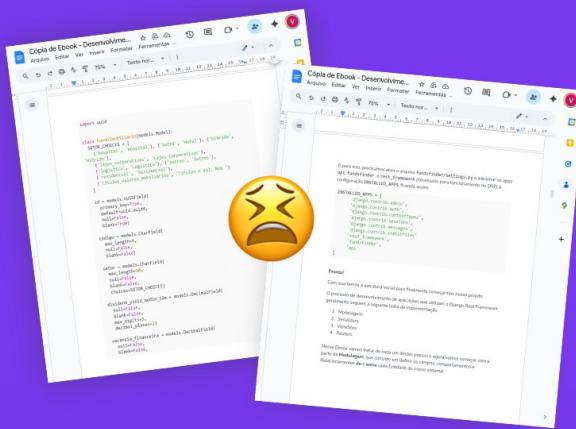
Nos vemos na próxima! 😊

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



Adicione Banners Promocionais

Deixe que nossa IA faça o trabalho pesado



Edite em Markdown em Tempo Real

TESTE AGORA



 PRIMEIRO CAPÍTULO 100% GRÁTIS