Loops em Python

Domine a arte de repetir ações de forma inteligente e eficiente no seu código



O que são Loops?

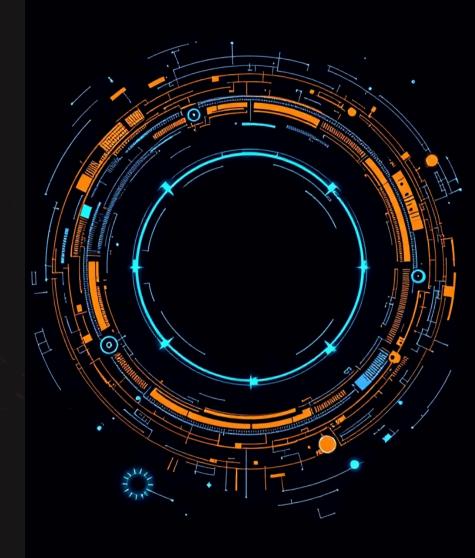
Conceito

Loops são estruturas que permitem **executar um bloco de código repetidamente**, sem precisar escrevê-lo várias vezes.

Imagine que você precisa imprimir "Olá" 100 vezes. Em vez de escrever 100 linhas, você usa um loop!

Por que usar?

- Economizam tempo e código
- Processam listas e coleções
- Automatizam tarefas repetitivas
- Tornam o código mais limpo





Tipos de Loops em Python

Loop for

Usado quando você sabe **quantas vezes** quer repetir algo ou quando quer percorrer uma sequência (lista, string, range).

Exemplo: Processar cada item de uma lista

Loop while

Usado quando você quer repetir **enquanto uma condição for verdadeira**. O número de repetições pode ser desconhecido.

Exemplo: Continuar até o usuário digitar "sair"

Loop for — Estrutura Básica

Como funciona?

O loop for percorre cada elemento de uma sequência, executando o bloco de código para cada item.

Estrutura:

for variavel in sequencia:
código a executar

Exemplo Prático

Imprimindo números de 1 a 5:

```
for numero in [1, 2, 3, 4, 5]:
    print(numero)

# Resultado:
# 1
# 2
# 3
# 4
# 5
```



Loop while - Estrutura Básica

Como funciona?

O loop while continua executando **enquanto a condição for verdadeira**.

Cuidado: Se a condição nunca se tornar falsa, você terá um loop infinito!

Exemplo Prático

Contador que para em 5:

```
contador = 1
```

while contador <= 5:
 print(contador)
 contador = contador + 1</pre>

Resultado: 1, 2, 3, 4, 5

Usando range() no Loop for

A função range() é uma ferramenta poderosa para gerar sequências numéricas automaticamente.

01

range(fim)

Gera números de 0 até fim -1

for i in range(5): print(i) # 0, 1, 2, 3, 4 02

range(início, fim)

Gera números do início até fim -1

for i in range(2, 7): print(i) # 2, 3, 4, 5, 6 03

range(início, fim, passo)

Gera números pulando de acordo com o passo

```
for i in range(0, 10, 2):
print(i)
# 0, 2, 4, 6, 8
```



Percorrendo Listas com for

Lista de Frutas

Uma das aplicações mais comuns é processar cada item de uma lista.

O Python percorre automaticamente cada elemento, um por vez.

Exemplo Completo

frutas = ['maçã', 'banana', 'laranja', 'uva']

for fruta in frutas: print(f'Eu gosto de {fruta}!')

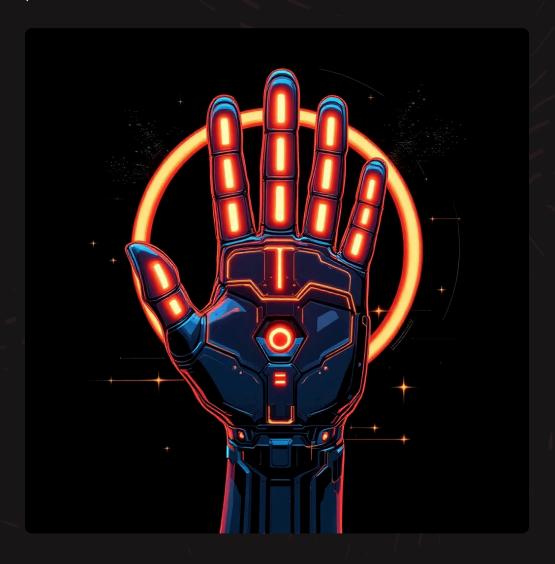
Resultado:# Eu gosto de maçã!# Eu gosto de banana!# Eu gosto de laranja!# Eu gosto de uva!

Comando break — Interrompendo o Loop

O que faz?

O comando break **interrompe completamente** o loop, mesmo que a condição ainda seja verdadeira.

É útil quando você encontrou o que procurava e não precisa continuar.



Exemplo Prático

Procurando um número específico:

```
numeros = [1, 3, 5, 7, 9, 11]

for num in numeros:
  if num == 7:
  print('Encontrei o 7!')
  break
  print(num)

# Resultado:
# 1
# 3
# 5
# Encontrei o 7!
```

Comando continue — Pulando Iterações

O que faz?

O comando continue **pula a iteração atual** e vai direto para a próxima.

O loop não termina, apenas ignora o restante do código naquela rodada.



Exemplo Prático

Pulando números pares:

```
for num in range(1, 8):
    if num % 2 == 0:
        continue
    print(num)

# Resultado:
# 1
# 3
# 5
# 7
# (pula 2, 4 e 6)
```



Resumo e Boas Práticas

Use for quando souber a quantidade

Ideal para listas, ranges e sequências conhecidas

Use while para condições

Perfeito quando não sabe quantas vezes vai repetir

Evite loops infinitos

Sempre garanta que a condição do while possa se tornar falsa

Use break e continue com sabedoria

Eles tornam o código mais eficiente quando bem aplicados

Dica final: Escreva nomes de variáveis claros nos loops. Use for fruta in frutas em vez de for x in lista — seu código ficará muito mais fácil de entender! **%**