





Loops são estruturas de controle que permitem a repetição de um bloco de código várias vezes. Python oferece dois tipos principais de loops:

• FOR: Utilizado para iterar sobre uma sequência (como listas, strings, ou ranges).

```
python

for variavel in sequencia:
    # bloco de código

python

python

python

python

python

Copy code

copy code

copy code

copy code

copy code

copy code
```

Loops são estruturas de controle que permitem a repetição de um bloco de código várias vezes. Python oferece dois tipos principais de loops:

• FOR: Utilizado para iterar sobre uma sequência (como listas, strings, ou ranges).

```
python

for _ in range(5):

print("Esta mensagem será exibida 5 vezes")
```



Quando você precisa apenas repetir um bloco de código um certo número de vezes e não precisa do valor de cada iteração, é comum usar _ para mostrar que o valor não é relevante.

• WHILE: Utilizado para iterar sobre uma sequência (como listas, strings, ou ranges).



CONTROLE DE FLUXO "BREAK" E "CONTINUE"

A instrução break interrompe a execução do loop imediatamente, saindo do loop, mesmo que ainda existam iterações restantes.

```
python

for i in range(1, 10):
    if i == 5:
        break
    print(i)
```

Saída: 1, 2, 3, 4 (o loop para ao atingir 5).

CONTROLE DE FLUXO "BREAK" E "CONTINUE"

A instrução continue pula a iteração atual do loop e continua com a próxima. Qualquer código que apareça após continue na iteração atual é ignorado.

```
python

for i in range(1, 10):
    if i == 5:
        continue
    print(i)
```

Saída: 1, 2, 3, 4, 6, 7, 8, 9 (5 é pulado).

INTRUÇÃO PASS

A instrução pass é uma operação nula; ela não faz nada e o loop continua normalmente. É usada quando uma declaração é sintaticamente necessária, mas você não deseja executar nenhum código.

```
python

for i in range(1, 10):
    if i % 2 == 0:
        pass # Placeholder para código futuro
    else:
        print(i)
```

Saída: 1, 3, 5, 7, 9

RESUMO

- break: Sai do loop imediatamente.
- continue: Pula a iteração atual e vai para a próxima
- pass: Não faz nada, mas é útil como um placeholder quando você está planejando adicionar código posteriormente ou deseja manter a sintaxe correta.

INTERPOLAÇÃO DE STRINGS

Interpolação de strings é o processo de inserir o valor de variáveis ou expressões dentro de uma string. Isso é útil quando você deseja construir strings dinâmicas que incluam dados de variáveis.

- Concatenação Simples (com o operador +)
- Interpolação com o Método format()
- Interpolação com f-strings (Formato Literal)
- Interpolação com o % (antigo método de formatação)

CONCATENAÇÃO SIMPLES

A forma mais básica de combinar strings e variáveis é usando o operador + para concatenálas. No entanto, essa abordagem pode se tornar confusa e difícil de ler quando se trabalha com muitas variáveis.

```
python

nome = "Maria"
idade = 30
mensagem = "Meu nome é " + nome + " e eu tenho " + str(idade) + " anos."
print(mensagem)

Copy code

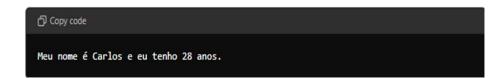
Meu nome é Maria e eu tenho 30 anos.
```

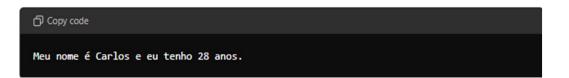
INTERPOLAÇÃO COM FORMAT()

O método format() permite que você insira valores dentro de uma string usando {} como placeholders.









INTERPOLAÇÃO COM F-STRINGS

Introduzidas no Python 3.6, as f-strings são a forma mais moderna e recomendada de interpolar strings. Elas permitem inserir expressões dentro de strings prefixando-as com f.

```
python

nome = "Carlos"

idade = 28

mensagem = "Meu nome é {} e eu tenho {} anos.".format(nome, idade)

print(mensagem)

Copy code

Meu nome é Carlos e eu tenho 28 anos.
```

INTERPOLAÇÃO COM % (ANTIGO)

Este método é mais antigo e menos comum nas versões modernas de Python, mas ainda é válido.

```
python

nome = "Lucas"
idade = 35
mensagem = "Meu nome é %s e eu tenho %d anos." % (nome, idade)
print(mensagem)

① Copy code

Meu nome é Lucas e eu tenho 35 anos.
```

