

[◀ VOLTAR](#)

# Estruturas de repetição: while e for

Apresentar os comandos da estrutura de repetição da linguagem Python.

## NESTE TÓPICO

[> WHILE \(CONDICIONAL\)](#)[> FOR \(INCONDICIONAL\)](#)[> Dê uma olhada nos links  
abaixo para saber mais sobre a  
Marcar  
tópico](#)

Olá alunos,

Vamos ver um dos principais recursos da lógica de programação: os comandos de repetição que junto com os comandos de decisão, praticamente conseguimos resolver qualquer problema do mundo real, utilizando esses recursos da lógica.

O loop também é um bloco de instruções dentro de um código e também devemos abrir e fechar este bloco.

Em português estruturado temos:

**ENQUANTO** condição verdadeira

**REPITA**

Comandos;

**FIM ENQUANTO;**

Ou

**PARA** número de passos

**REPITA**

Comandos;

**FIM PARA;**

Os comandos ficam em um loop (laço, repetição) e em algumas linguagens temos que colocar a palavra LOOP (repita) para iniciar a repetição e END LOOP; para finalizar o loop.

Agora em Python:

## WHILE (CONDICIONAL)

Repete um bloco de comandos (entra em loop) enquanto a condição que segue o comando **while** for verdadeira, ou seja, o laço (loop) está sob a condição do **while** (é condicional) e parará de repetir o comando ou os comandos quando a condição deixar de ser verdadeira.

**while** condição:

..... comandos

Para abrir o loop: basta colocar o comando **while** (em letras minúsculas) e utilizar os dois pontos ":" depois da condição e o interpretador então irá executar o comando ou os comandos que estão logo após os **dois pontos**.

Agora, note também que os comandos do **while** estão com um recuo (indentação) e para encerrar o bloco basta, na próxima linha do código, tirar o recuo.

**Exemplo 1:**

**Imprimindo os números de 1 a 30:**

```
1. vNum = 1
2. while vNum <= 30:
3.     print(vNum)
4.     vNum = vNum + 1
5. input('FIM DO PROGRAMA!')
```

```
1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8
9. 9
10. 10
11. 11
12. 12
13. 13
14. 14
15. 15
16. 16
17. 17
18. 18
19. 19
20. 20
21. 21
22. 22
23. 23
24. 24
25. 25
26. 26
27. 27
28. 28
29. 29
30. 30
31. FIM DO PROGRAMA!
```

Na linha **1**, criamos a variável **vNum** e iniciamos com um valor, já determinando que a variável é do tipo número inteiro.

Na linha **2**, abrimos o bloco **loop** com o **while** colocando a condição que enquanto a variável **vNum** for menor ou igual a 30, o conjunto de comandos dentro do bloco se repetirá.

A linha **3** é o primeiro comando do laço (loop) e imprimirá o valor da variável **vNum**.

A linha **4** (que é o último comando do laço (loop)) vai somar mais 1 ao valor da variável e atribuirá o resultado na própria variável **vNum**, que é conhecido como incremento da variável.

No primeiro “giro” do loop, **vNum** estará valendo 1 e por causa do comando **print**, será impresso na tela este valor, na sequência (linha **4**) a variável mudará o valor para 2, a execução voltará para cima, testará a condição e como o valor de **vNum** agora está valendo 2 e como 2 é menor que 30, então o loop executará novamente as linhas **3 e 4** dos comandos, e assim por diante, até que, na linha **4** o valor de **vNum** passe a valer 31 e como 31 não é menor e nem igual a 30, o interpretador encerrará a execução do bloco de comandos do loop e executará a última linha (**5**) do código. Esta linha **5** não pertence ao bloco do loop (**while**).

Outros operadores que poderemos utilizar são os operadores booleanos: **True** ou **False**.

Podemos também utilizar o comando **break** para sair do loop.

## Exemplo 2:

### Sorteando três números diferentes de 1 a 50:

```
1. import random
2. print("\n* * * SORTEANDO TRÊS NÚMEROS DIFERENTES ENTRE 1 E 50 * * *\n")
3. input("Pressione ENTER para gerar os números: ")
4. while True:
5.     num1 = random.randint(1,50)
6.     num2 = random.randint(1,50)
7.     num3 = random.randint(1,50)
8.     if num1 not in (num2,num3) and num2 != num3:
9.         break
10. print("Os números são:\n ",num1,num2,num3)
11. input("Pressione ENTER para sair...")
```

Em Python temos uma vasta biblioteca, que são rotinas já implantadas na linguagem, para utilizarmos, basta utilizar o comando **import** e o nome da rotina, no caso, a biblioteca **random** que possui várias funções (métodos) para gerar números aleatórios (randômicos).

Na linha **4**, utilizamos o **while** com o operador booleano (George Boole): **True** (verdadeiro), note que a primeira letra T é em maiúsculo e o restante em minúsculo.

Nas próximas linhas, criamos três variáveis e utilizamos a função **randint** que pertence a biblioteca **random** (temos que fazer a referência com o ponto: **random.randint**) para gerar números aleatórios entre 1 a 50 (1,50) e os valores serão atribuídos às variáveis.

Na linha **8**, temos um comando **if** com uma condição que analisa se todos os números gerados são diferentes para evitar de gerar dois ou mais números iguais.

Com o comando **while** em **True**, o loop será realizado infinitamente, ou seja, irá gerar três números aleatórios, mas com o comando **if**, se todos os números gerados forem diferentes (daí a condição se torna verdadeira), o comando **break** (breque!), será executado e o interpretador interromperá o loop e executará a última linha com o comando **print**, mostrando todos os números do sorteio. Caso contrário, se a função randômica gerar dois ou mais números iguais, o loop irá gerar números, até forem todos diferentes.

```
1. * * * SORTEANDO TRÊS NÚMEROS DIFERENTES ENTRE 1 E 50 * * *
2.
3. Pressione ENTER para gerar os números:
4. Os números são:
5.     28 14 33
6. Pressione ENTER para sair...
```

## FOR (INCONDICIONAL)

Repete um bloco de comando n vezes, ou seja, até que a variável contadora atinja o seu valor final ou receba um comando **break**.

Diferente do **while**, o **for** não depende de uma condição (é incondicional) para o loop.

**for contador in lista:****..... bloco de comandos**

Com o comando **for** determinamos quantas vezes o comando ou comandos vão repetir (loop). Logo depois da palavra **for** tem uma variável tipo número inteiro (contador), que a cada loop ela vai alterando o valor automaticamente, acrescentando mais um.

Mesmo Exemplo 1 com **for**:

```
1. for vNum in range(1,31):
2.     print(vNum)
3. input('FIM DO PROGRAMA!')
```

No primeiro “giro” a variável **vNum** vale 1 e este valor será impresso até o número 30, pois o comando **print** será executado 30 vezes. Algo a observar é que utilizamos a função **range** onde determinamos o valor inicial e o valor final (**1,31**). Lembre-se que, a exemplo do que também acontece na linguagem C, a primeira posição não inicia com 1 e sim com 0 (zero), então nesse caso, o 1 é a segunda posição. Assim o **range** de 1 até 31, será repetido 30 vezes.

**Exemplo 3:**

```
1. for atriz in ["Angelina Jolie","Julia Roberts","Megan Fox","Ivete Sangalo"]:
2.     if atriz != "Ivete Sangalo":
3.         print("A",atriz,"é uma atriz de Hollywood!")
4.     else:
5.         print("A", atriz,"NÃO é uma atriz de Hollywood!\n")
6. input('Pressione ENTER para sair...')
```

Neste exemplo, utilizamos o que é conhecido como lista, que em Python, basta colocar os valores em **colchetes** separados com **vírgula**. Automaticamente o interpretador irá, a cada loop, atribuir o valor a variável **atriz** de acordo com a fila em que os elementos estão na lista e como os elementos são do tipo string (caractere), a variável **atriz** já é considerada tipo **str** (string).

Dentro do bloco **for** tem um bloco **if** (observe a indentação) e os comandos **print** pertencem ao bloco **if** com o **else** (observe a indentação).

Analise o código do exemplo e tente perceber qual será o resultado:

```
1. A Angelina Jolie é uma atriz de Hollywood!
2. A Julia Roberts é uma atriz de Hollywood!
3. A Megan Fox é uma atriz de Hollywood!
4. A Ivete Sangalo NÃO é uma atriz de Hollywood!
5.
6. Pressione ENTER para sair...
```

## SAIBA MAIS...

Dê uma olhada nos links abaixo para saber mais sobre a linguagem Python:

<https://www.python.org/doc/> (<https://www.python.org/doc/>)

<https://wiki.python.org/moin/PythonBooks>  
(<https://wiki.python.org/moin/PythonBooks>)

Neste tópico vimos como implantar estruturas de repetição, utilizando os comandos while e for que representam um loop (laço, repetição).

## Quiz

Exercício Final

Estruturas de repetição: while e for

INICIAR ➤

## Referências

SUMMERFIELD, M. *Programação em Python 3: Uma introdução completa à linguagem Python*. Rio de Janeiro Alta Books, 2012. 495 p.

MENEZES, N. N. C. *Introdução à programação com Python: algoritmos e lógica de programação para iniciantes*. 2. ed. São Paulo: Novatec, 2014. 328 p.

SWEIGART, AL. *Automatize tarefas maçantes com Python: programação prática para verdadeiros iniciantes*. São Paulo: Novatec, 2015. 568 p.

PYTHON, doc. Disponível em: <<https://www.python.org/doc/>>. Acesso em: Junho/2018.

PYTHON, books. Disponível em: <<https://wiki.python.org/moin/PythonBooks>>. Acesso em: Junho/2018.



Avalie este tópico



ANTERIOR

Estruturas de decisão: IF, ELIF e ELSE



Índice

Biblioteca

(<https://www.uninove.br/conheca->

a-

uninove/biblioteca/sobre-

a-

biblioteca/apresentacao/)

Portal Uninove

(<http://www.uninove.br>)

Mapa do Site

Ajuda?

PRÓXIMO  
(<https://ava.uninove.br/ava-uninove/>)

Tratamento de exceções

© Todos os direitos reservados

