

INSTITUTO FEDERAL
ALAGOAS

ATLETAS DA PROGRAMAÇÃO

Aula 7



O que veremos na aula de hoje:

- O que são laços e como funciona o flow do código nas repetições;
- A função while;
- Sintaxe da função while;
- Contadores;
- while com estruturas de decisão;

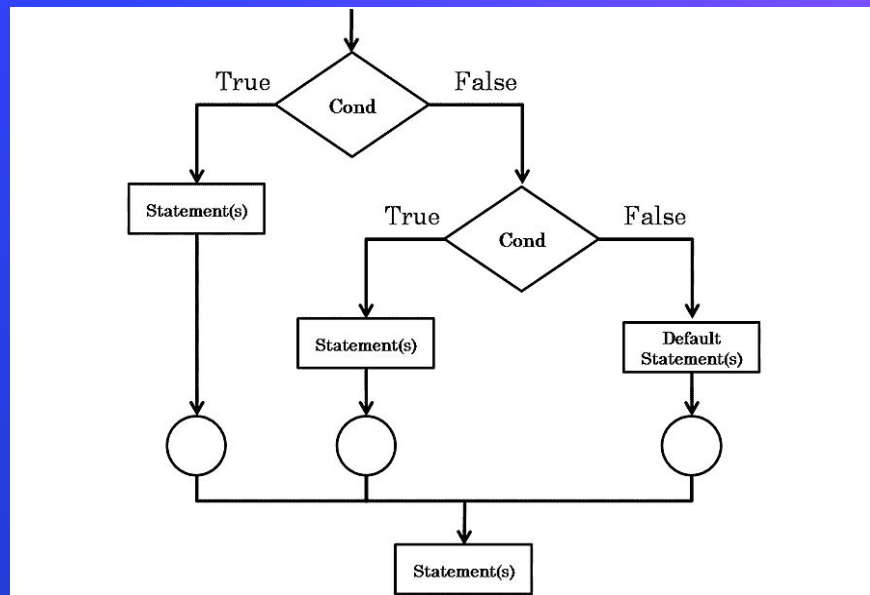
O QUE SÃO LAÇOS E COMO FUNCIONA O FLOW DO CÓDIGO NAS REPETIÇÕES:

- Os laços, ou loops como também são conhecidos, são um recurso útil que é frequentemente utilizado em todas as linguagens de programação modernas.
- E se, em algum momento, precisarmos executar 5 prints iguais? Ou fazer 5 inputs muito semelhantes? Teríamos que ficar usando 5 linhas de código sempre?

Claro que não! É aqui que entram os laços!

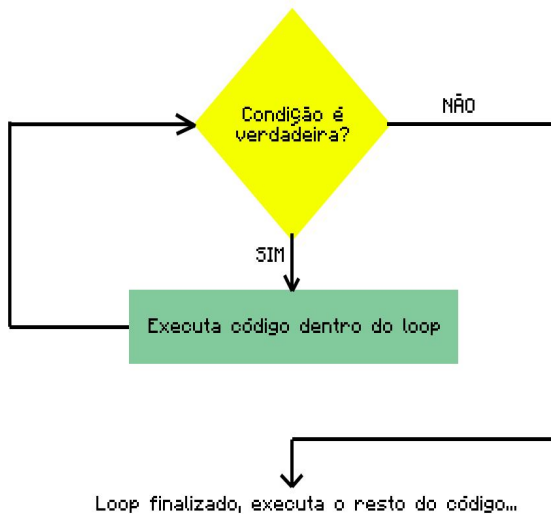
- Se você quiser automatizar uma tarefa específica repetitiva ou evitar escrever código repetido em seus programas, usar um laço é a sua melhor opção.

Lembram que o flow de um código sempre vai para frente, tendo bifurcações ou não? Dessa maneira:



O QUE SÃO LAÇOS E COMO FUNCIONA O FLOW DO CÓDIGO NAS REPETIÇÕES:

Os laços são um conjunto de instruções que são executadas repetidamente enquanto determinada condição for atendida. Dessa forma:



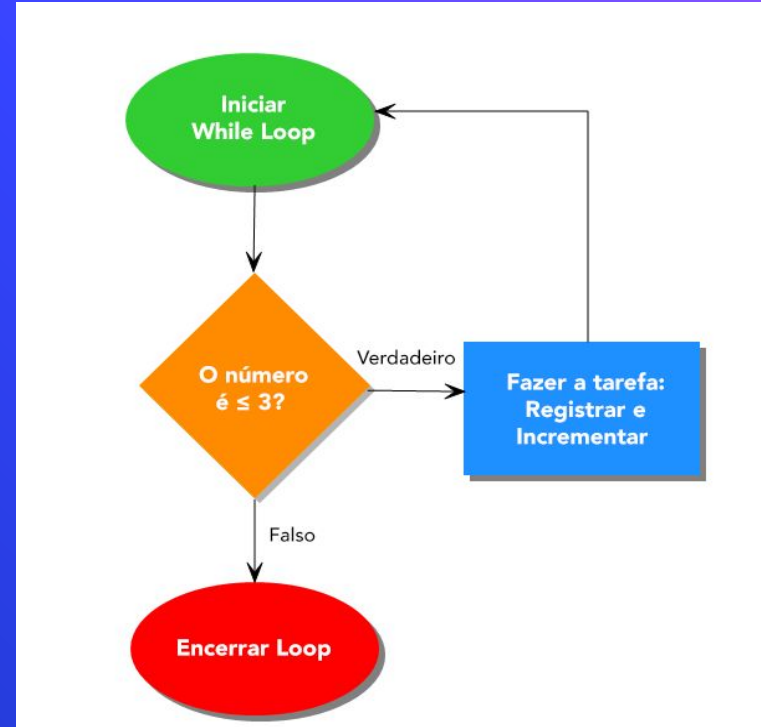
- Há duas funções reservadas para construir laços em Python:

A função **for**
E a função **while**

- Nessa aula vamos aprender como criar um laço utilizando a função **while** e ver como ele funciona.

A FUNÇÃO WHILE

- A função while é usada, geralmente, quando não sabemos quantas vezes um determinado bloco de instruções precisa ser repetido. Com ela, a repetição da execução de um bloco de instruções vai continuar enquanto uma determinada condição for verdadeira.
- Isto é, a condição a ser analisada pela função while deverá retornar um valor boolean (true ou false).
- Com isso o bloco de código será executado repetidamente enquanto uma condição for verdadeira (retornar True) e seguirá executando o conjunto de instruções desejado até que aquela condição deixe de ser True (Verdadeira) e passe a ser False (Falsa).



A SINTAXE DA FUNÇÃO WHILE

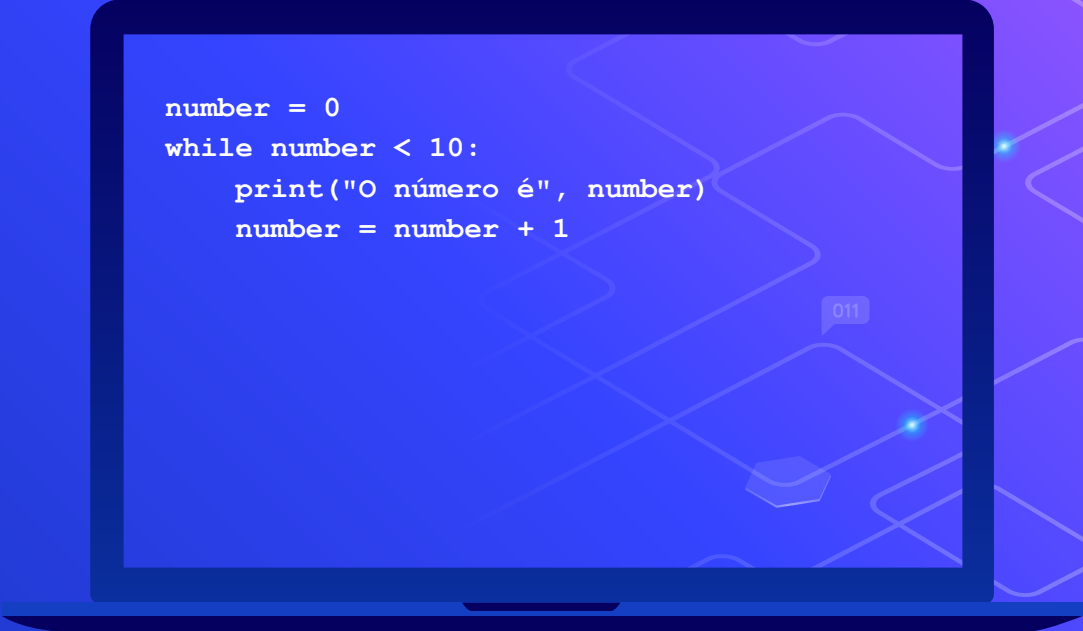
- Um laço while sempre verificará primeiro a condição antes de ser executado.
- Se a condição for avaliada como True, o laço, assim como o código de seu corpo, serão executados.

A stylized illustration of a laptop with a dark blue frame. The screen is light blue and displays Python code for a while loop. The background of the entire slide is a vibrant blue with faint, white, geometric line patterns and small hexagonal shapes, some of which contain binary code (001, 011, 010).

```
while condição:  
    <execute este código dentro do corpo  
do laço>
```

A SINTAXE DA FUNÇÃO WHILE

- Por exemplo, o laço ao lado é executado até que a variável number seja igual ou superior a 10, isto é, enquanto ela for menor que 10:

A laptop is shown with a screen displaying Python code. The code is a while loop that initializes a variable 'number' to 0 and then enters a loop that prints the current value of 'number' and increments it by 1, as long as 'number' is less than 10. The background of the slide features a blue gradient with white geometric lines and small hexagonal shapes, some of which contain binary code (001, 011, 010).

```
number = 0
while number < 10:
    print("O número é", number)
    number = number + 1
```

A SINTAXE DA FUNÇÃO WHILE

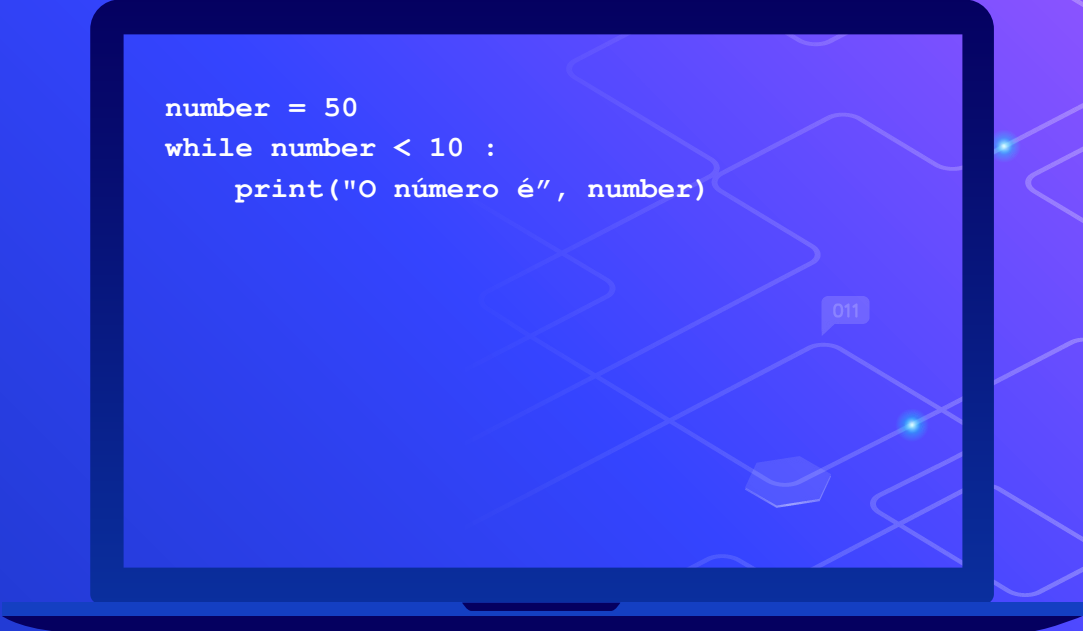
- Veja o output ao lado:
- Uma vez que a variável chega a 10, a função while conclui: “Opa! a variável chegou a 10, essa condição é falsa! Em vez de continuar com o loop, vou quebrá-lo!”

A laptop screen is shown, displaying a series of text outputs from a program. The text is white on a dark blue background. The output consists of ten lines, each starting with 'O número é' followed by a number from 0 to 9. The laptop is a dark blue silhouette with a lighter blue screen area.

```
O número é 0  
O número é 1  
O número é 2  
O número é 3  
O número é 4  
O número é 5  
O número é 6  
O número é 7  
O número é 8  
O número é 9
```


A SINTAXE DA FUNÇÃO WHILE

- Também é possível que o laço while nunca seja executado se não atender à condição, como neste exemplo:

A dark blue laptop is shown from a slightly elevated angle. Its screen displays a Python code snippet. The background of the slide features a network of white lines and nodes on a blue gradient, with some nodes labeled with binary code (001, 011, 010).

```
number = 50
while number < 10 :
    print("O número é", number)
```

CONTADORES

- Para que a função `while` não entre em loop infinito, como vocês puderam ver nos slides anteriores, usamos uma variável que tem o papel de contadora para controlar o número de repetições do loop.

Veja o exemplo ao lado:

- Utilizamos a variável “`i`”, que funciona como um contador e é usada para controlar a repetição do loop `while`. Veja também que incrementamos essa variável dentro das instruções do loop. Isso faz com que a condição de avaliação encontre o ponto de parada e encerre o loop.

```
i = 1
while (i <= 5):
    print(i, end=" ")
    i += 1
'''
```

```
output:
1 2 3 4 5
'''
```

WHILE COM ESTRUTURA DE DECISÃO

- Ao final do **while** podemos utilizar a instrução **else**. O propósito disso é executar alguma instrução ou bloco de código ao final do loop, como podemos ver no exemplo a seguir:

```
x = 0
while x < 10:
    print(x)
    x += 1
else:
    print("fim while")
```

WHILE COM ESTRUTURA DE DECISÃO

- Veja o output ao lado:
- Nesse caso, o while vai checar a condição ($x < 10$, nesse caso) repetidas vezes. Se for verdadeira, ok, senão, ele quebra o loop e pula para o else.

A laptop screen is shown, displaying a list of numbers from 0 to 9, each on a new line. Below the numbers, the text 'fim while' is displayed. The background of the slide features a blue gradient with white geometric lines and small binary code elements (001, 011, 010) scattered around.

```
0  
1  
2  
3  
4  
5  
6  
7  
8  
9  
fim while
```

BREAK

- Outro recurso muito útil em Python é o **break**. Se, dentro da repetição, for executado um **break**, o loop será encerrado imediatamente sem executar o código da cláusula **else**, mesmo se a condição inicial ainda for verdadeira.

```
x = 0
while x < 10:
    print(x, end=" ")
    x = x + 1
    if x == 6:
        print("x é igual a 6")
        break
else:
    print("fim while")
```

BREAK

- Veja o output ao lado:



WHILE TRUE

- A declaração `while True` é usada para especificar um loop infinito while. Um loop infinito é executado indefinidamente até o final do tempo ou quando o programa é interrompido à força.
- Exemplo:

A stylized illustration of a laptop with a dark blue frame. The screen is light blue and displays two lines of Python code in a white, monospaced font. The background of the entire slide is a dark blue gradient with faint, glowing white lines and small hexagonal shapes, some of which contain binary code (001, 011, 010).

```
while True:  
    print("Hello World")
```

WHILE TRUE

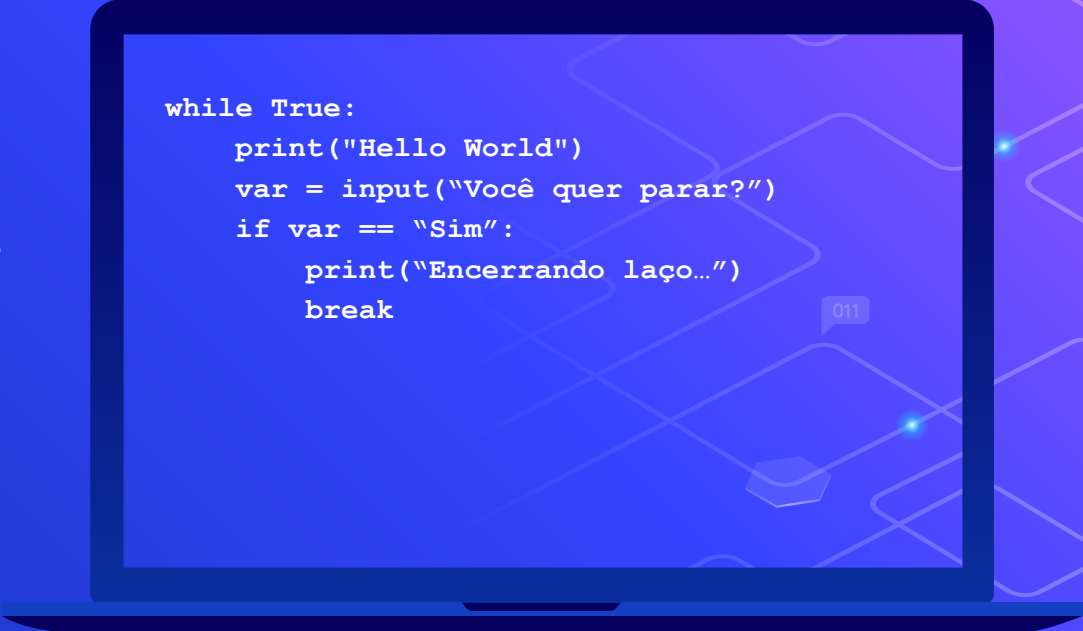
- Veja o output ao lado:
- Essa abordagem não é recomendada e muito menos usada, porque interrompe o código de sua conclusão.



```
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World..
```


WHILE TRUE

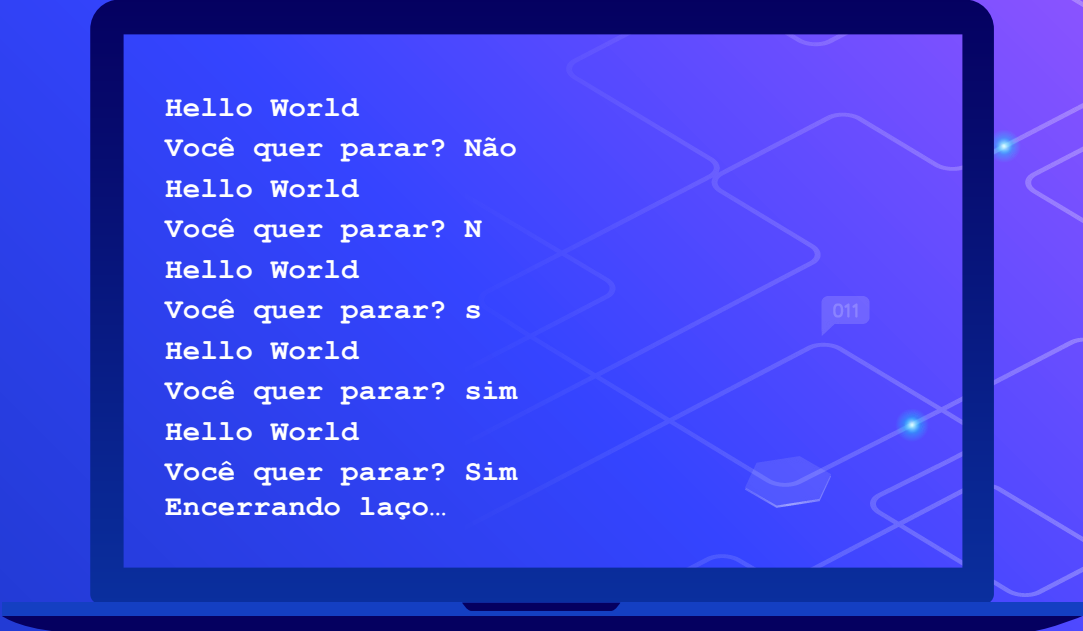
- Uma boa prática é o uso da instrução `break` dentro do loop infinito para interromper o processo quando uma condição específica for satisfeita.
- É daqui que vem a ideia de usarmos o `while` quando não sabemos o número total de loops. O `while` irá repetir até que seja feita a necessidade do programa, que vai acontecer quando ele encontrar um `break` no caminho.
- Exemplo:

A laptop is shown with a screen displaying Python code. The code is a while loop that prints 'Hello World', asks for input, and breaks if the input is 'Sim'. The background of the slide features a blue gradient with white geometric lines and small hexagonal shapes, some containing binary code like '001', '011', and '010'.

```
while True:
    print("Hello World")
    var = input("Você quer parar?")
    if var == "Sim":
        print("Encerrando laço...")
        break
```

WHILE TRUE

- Veja o possível output ao lado:
- Obs: O que aparece ao lado das perguntas não foi geralmente automaticamente pelo programa, foi uma resposta fornecida pelo usuário!



```
Hello World
Você quer parar? Não
Hello World
Você quer parar? N
Hello World
Você quer parar? s
Hello World
Você quer parar? sim
Hello World
Você quer parar? Sim
Encerrando laço...
```

Dúvidas?

Até a próxima aula!

