ESTRUTURAS DE REPETIÇÃO

# ESTRUTURAS DE REPETIÇÃO: LOOPS

Neste vídeo, **Felipe Frigeri**, Sênior Data Scientist do Itaú Unibanco, apresenta as estruturas de repetição (loops) no Python.

08:54

## ESTRUTURAS DE REPETIÇÃO EM PYTHON

Uma habilidade muito importante dentro da programação é conseguir fazer com que um mesmo código seja reaproveitado diversas vezes da maneira mais eficiente possível.

Uma das formas que temos de fazer isso é através das Estruturas de Repetição. Imagine que você tenha uma lista de nomes e queira construir um código para imprimi-los:

```
lista_nomes =
['João','Marcela','Felipe','Ana','Augusto','Flávio',

'Henrique','Juliana','Andrei','Bruno','Marina']
```

Sem um meio de fazer uma estrutura de repetição, precisaríamos escrever:

```
print( Felipe )
```

E repetir para todos os nomes da lista.

## A ESTRUTURA DE REPETIÇÃO *FOR* EM PYTHON

Felizmente, em Python, podemos fazer este trabalho usando uma estrutura de repetição! O comando que indica uma repetição é o equivalente em inglês da palavra “para”: **for**

```
for nome in lista_nomes:
    print(nome)
```

Na estrutura acima, temos alguns elementos importantes:

- **for**: indicativo de uma estrutura de repetição
- **lista_nomes**: Lista de elementos em que faremos a operação a ser repetida
- **nome**: Variável interna da estrutura de repetição, atualizada a cada leitura na lista **lista_nomes**

Veja que podemos escolher o nome da variável interna. O código abaixo terá o mesmo funcionamento do anterior:

```
for x in lista_nomes:
    print(x)
```

Assim como na estrutura condicional ( **if** ), a **indentação/tabulação** faz parte do código, ou seja, os códigos que serão repetidos para cada elemento da lista serão apenas aqueles dentro da indentação.

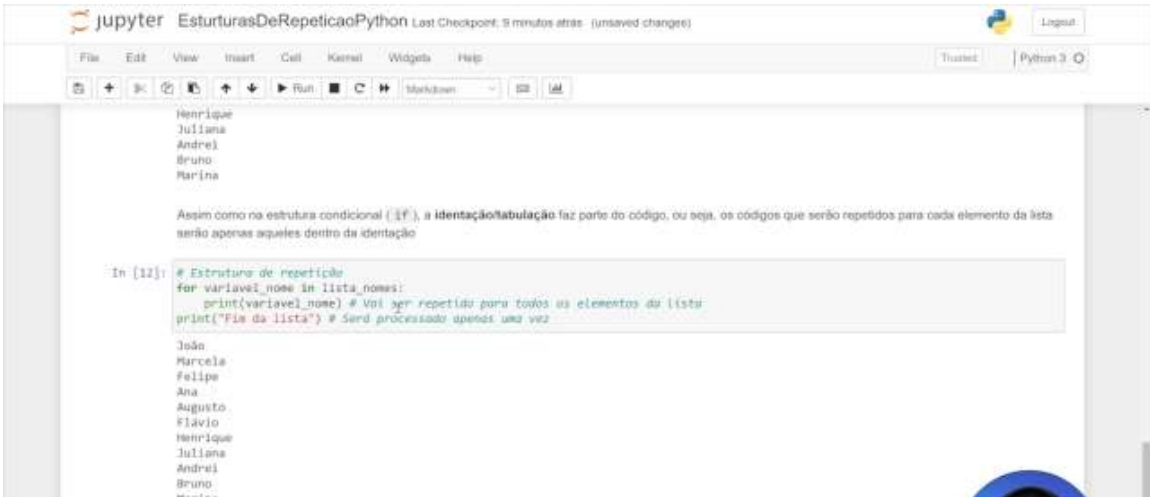
---

AVANÇAR

ESTRUTURAS DE REPETIÇÃO

## COMBINAÇÃO COM CONDICIONAIS

Neste vídeo, Felipe fala sobre a combinação das estruturas de repetição com condicionais.




03:17

## A ESTRUTURA DE REPETIÇÃO WHILE EM PYTHON

Uma outra forma de escrever uma estrutura de repetição é com o **while**, neste caso o código irá se repetir enquanto até que um certo cenário aconteça.

```
contador =  0
while contador < 8:
    contador = contador + 1
    print("Já acabou?")
```

esse contador for menor que **8**.

A cada rodada também incrementamos a variável contador em uma unidade. Quando o contador for igual ou maior que **8** o código será interrompido.

Se a linha que incrementa a variável contador não tivesse sido escrita, teria sido criado um loop infinito que imprimiria a frase repetidas vezes, até o momento que o código tivesse que ser interrompido manualmente.

## COMBINANDO ESTRUTURAS DE REPETIÇÃO E CONDICIONAIS

Como a estrutura de repetição possui um bloco de código próprio, podemos inclusive incluir outras estruturas, como as condicionais (**if/else**).

```
numeros = [0,1,2,3,4,5,6]
for numero in numeros:
    if numero<=2:
        print("É menor ou igual a 2")
    elif numero<=4:
        print("Está entre 2 e 4")
    else:
        print("É maior que 4")
```

No código acima estamos verificando para cada item da lista se o elemento é menor do que 2, se está entre 2 e 4 ou se é maior do que 4.

## (BÔNUS) LIST COMPREHENSION

Se você quer se iterar sobre uma lista e ter outra lista como resultado, a **List Comprehension** oferece uma estrutura que permite que você faça isso de maneira bem sucinta.

**Sintaxe**

## ESTRUTURAS DE REPETIÇÃO

Vamos ver um exemplo:

```
idades = [18, 20, 32, 33, 65]
[2021 - x for x in idades]
[2003, 2001, 1989, 1988, 1956]
```

No código acima temos uma lista com diversas idades. Na segunda linha, estamos fazendo uma operação na qual subtraímos cada elemento do ano atual para ter aproximadamente o ano de nascimento. Podemos também colocar uma condicional na list comprehension com a seguinte sintaxe:

```
[expressão for elemento in lista if condição]
```

Se quisermos retornar uma lista com anos de nascimento, mas somente daqueles com mais de 30 anos, podemos fazer da seguinte forma:

```
[2021 - x for x in idades if x > 30 ]
```

---

AVANÇAR