



Introdução à linguagem Python

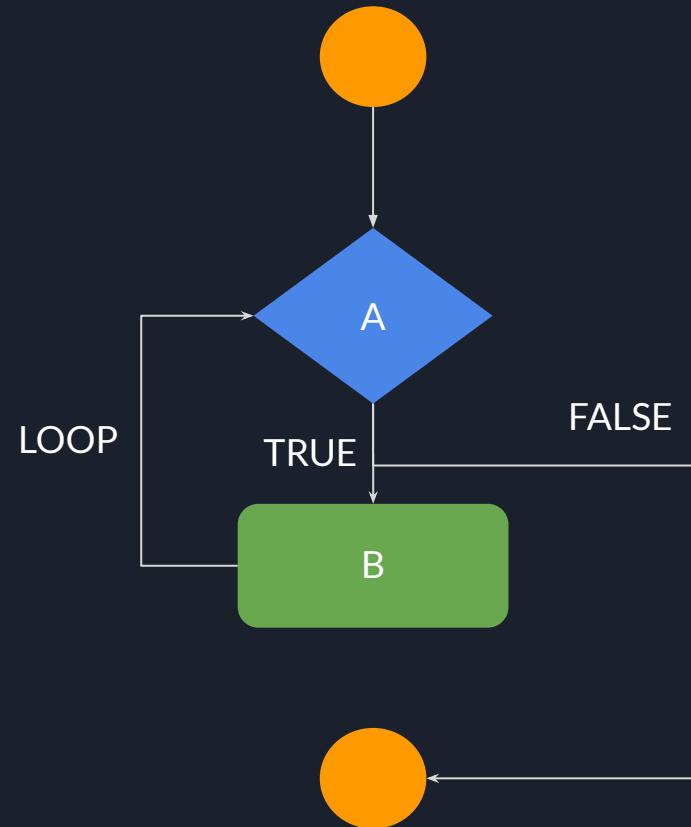
Aula 3 - Estruturas de Repetição (For e While)

Por: Thiago Ribeiro da Silva

Estudante de Ciência da Computação (UFAL)

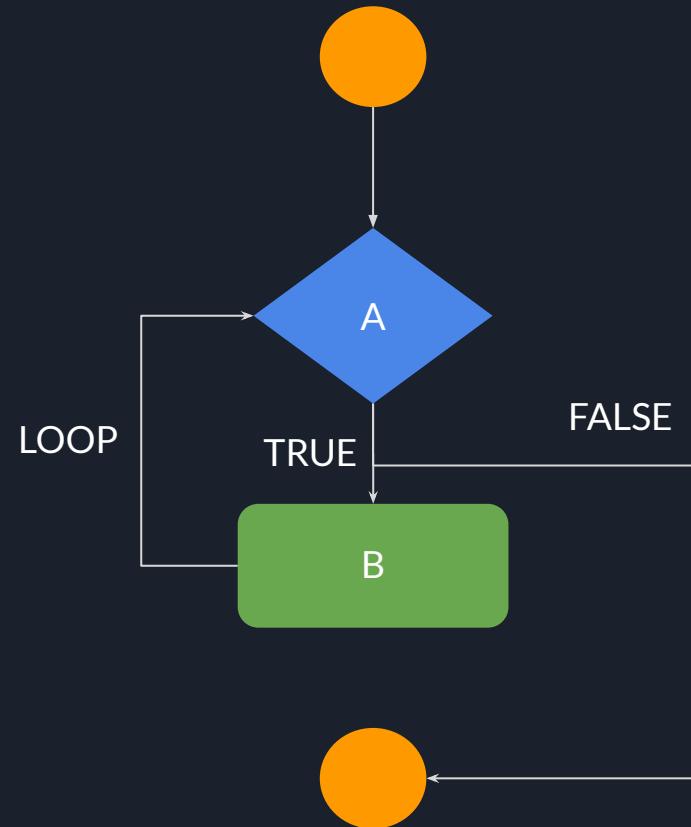
Estruturas de Repetição

- Para além das já discutidas estruturas de condição, a construção de algoritmos também demanda estruturas que possibilitem a repetição de trechos de código para evitar scripts extensos e poluídos.



Estruturas de Repetição

- Desta forma, **enquanto** uma condição for verdadeira, certo bloco de código será executado e o programa retornará a condição, de modo que irá repetir o processo até que aquela condição se torne falsa.



Estruturas de Repetição

- Um dos objetivos do jogo Minecraft é cavar ambientes atrás de minerais, como o ouro. Imaginemos que Steve está em uma caverna e precisa encontrar um bloco com minério de ouro. Desta forma, vamos entender a usabilidade das repetições



Estruturas de Repetição - For Loop

- Uma das formas de resolver o problema de Steve é conhecendo a distância até o ouro. Desta forma, bastaria ele minerar os **N** blocos de pedra até o outro e então, enfim, alcançá-lo.
- Ou seja: “*repita no intervalo de 1 até n a ação de quebrar blocos de pedra*”



```
1 repita no intervalo de 1 até N:  
2     quebrar blocos de pedra  
3  
4     exiba "Enfim um minério de ouro"
```

Estruturas de Repetição - For Loop

- No python, essa estrutura se chama **For Loop**, onde aqui para um valor x num intervalo de 0 até N (sendo aberto em N), repetiremos um bloco de código;
- Para gerar um intervalo, existe a função **range(inicio, fim, passo)**.



```
1 for i in range(0, N):
2     quebrar_blocos_de_pedra()
3
4 print("Enfim um minério de ouro")
```

For Loop - Alguns exemplos

```
▶ # Programa para calcula a tabuada de um número

    num = int(input("Informe o número: "))

    for i in range(1, 11):
        print(f"{num}x{i} = {num*i}")

⇒ Informe o número: 5
5x1 = 5
5x2 = 10
5x3 = 15
5x4 = 20
5x5 = 25
5x6 = 30
5x7 = 35
5x8 = 40
5x9 = 45
5x10 = 50
```

For Loop - Alguns exemplos

```
▶ # Programa que pede 3 números e soma

soma = 0

for i in range(3):
    num = int(input(f"Digite o {i+1}º número: "))
    soma += num

print(f"A soma é {soma}")
```

```
↳ Digite o 1º número: 12
Digite o 2º número: 42
Digite o 3º número: 75
A soma é 129
```

Estruturas de Repetição - While Loop

- Conhecer o caminho torna o objetivo fácil. Para a versão do problema onde só se conhece o final, **enquanto** não se alcança o minério de ouro, Steve deve repetir constantemente a ação de minerar pedra.
- Ou seja: “*enquanto não for ouro, quebre blocos de pedra*”



```
1 enquanto não alcança bloco de ouro repita:  
2     quebrar blocos de pedra  
3  
4     exiba "Enfim um minério de ouro"
```

Estruturas de Repetição - While Loop

- No python, essa estrutura se chama **while loop**, onde enquanto uma condição x for verdadeira, um trecho de código será executado a todo momento.



```
1 while (nao_eh_bloco_de_ouro()):  
2     quebrar_blocos_de_pedra()  
3  
4 print("Enfim um minério de ouro")
```

While Loop - Alguns exemplos

```
# Programa que pede x números para serem digitados e determina o total de pares e ímpares

quantidade = int(input("Digite quantos números quer analisar: "))
total_pares = 0
total_impar = 0

while(quantidade > 0):
    num = int(input("Digite um número: "))
    if num % 2 == 0:
        total_pares += 1
    else:
        total_impar += 1
    quantidade -= 1 # regredindo a quantidade

print(f"Dos números analisados, {total_pares} são pares e {total_impar} são ímpares")
```

```
▷ Dige quantos números quer analisar: 5
Dige um número: 14
Dige um número: 23
Dige um número: 0
Dige um número: 15
Dige um número: 2
Dos números analisados, 3 são pares e 2 são ímpares
```

While Loop - Alguns exemplos



```
# Programa que só aceita entrada se o tipo sanguíneo for 0+  
  
tipo = ''  
  
while(tipo != '0+'):   
    tipo = input("Digite o tipo sanguíneo: ")  
  
print("Finalmente um 0+!")
```

```
Digite o tipo sanguíneo: A  
Digite o tipo sanguíneo: B  
Digite o tipo sanguíneo: D  
Digite o tipo sanguíneo: 0+  
Finalmente um 0+!
```

Estruturas de Repetição - Break e Continue

- Às vezes é necessário dar um parada no caminho, ou até mesmo continuar e ignorar certas instruções.
- Caso Steve saiba que há um bloco de ouro depois de 8 blocos de pedra, mas acabe achando outro no meio, é necessário parar por ali, pois o objetivo foi atingido!

```
1 enquanto blocos quebrados for menor que 8 repita:  
2     quebrar blocos de pedra  
3     se encontrou bloco de ouro:  
4         pare de minerar  
5  
6     exiba "Enfim um minério de ouro"
```

Estruturas de Repetição - Break e Continue

- Para parar um loop em python, usamos o termo **break**, que joga o algoritmo para fora do laço de repetição e continua até o fim do programa.
- Outro comando interessante é o **continue**, que ignora os códigos abaixo do bloco do loop e segue para a próxima conferência do loop.



```
1 blocos_quebrados = 0
2 while(blocos_quebrados < 8):
3     quebrar_blocos_de_pedra()
4     if eh_bloco_de_ouro():
5         break
6
7 print("Enfim um minério de ouro")
```

Break e Continue - Exemplo 1



```
# Usando break
# Uma pessoa precisa do tipo sanguíneo A+
# faça um programa que só aceite o tipo sanguíneo que ela pode receber
tipo = ''

while True:
    tipo = input("Digite o tipo sanguíneo: ")
    if tipo == '0-' or tipo == 'A+':
        break
    else:
        print(f"O tipo {tipo} não serve para ela!")

print("Finalmente um tipo correto!")
```

```
Digite o tipo sanguíneo: B
O tipo B não serve para ela!
Digite o tipo sanguíneo: 0+
O tipo 0+ não serve para ela!
Digite o tipo sanguíneo: 0-
Finalmente um tipo correto!
```

Break e Continue - Exemplo 2

```
# Programa que só printa se a entrada for par
# Ele para se a entrada for -1

while True:
    num = int(input("Digite um valor numérico: "))
    if num == -1:
        break
    if num % 2 != 0:
        continue
    print(f"Seu número é: {num}")
```

```
Digite um valor numérico: 1
Digite um valor numérico: 3
Digite um valor numérico: 19
Digite um valor numérico: 12
Seu número é: 12
Digite um valor numérico: -1
```

Obrigado!

