



- IF com OR
- Negação e Diferença
- IF aninhados
- Função Range

Python

Lógica de Programação

Principais conceitos da Linguagem Python

01 If com Or (Ou)

- Como vimos na *sheet* anterior, os operadores And (E) e Or (OU) funcionam da maneira ilustrada a seguir. O operador And (E), só é verdadeiro se o lado esquerdo representado por A bem como o lado direito representado por B sejam verdadeiros.

A	B	A e B	A ou B
F	F	F	F
F	V	F	V
V	F	F	V
V	V	V	V

- Por outro lado, o operador Or (OU), só é verdadeiro se um dos lados (esquerdo ou direito) forem verdadeiros. Para finalizar, analise o código a seguir e tire suas conclusões.

```
1 print(1==1 and 1==2) False
2 print(2==1 and 1==2) False
3 print(3>=1 and 1<=2) True
4
5 print(1==1 or 1==2) True
6 print(2==1 or 1==2) False
7 print(3>=1 or 1<=2) True
```

Saída

- Considerando a idade = 18, no código a seguir, as linhas 1,2,3,4 e 5 serão executadas, pois:

- O lado esquerdo (destacado em laranja) é verdadeiro, pois idade é maior ou igual a 18
- E, o lado direito (destacado em verde) também é verdadeiro, pois idade é menor que 65.
- Então, como os dois lados são verdadeiros, dá certo, pois o operador And exige que os dois lados (laranja e verde) sejam verdadeiros.

```
if idade >= 18 and idade < 65 :
```

true
linha 1
linha 2
linha 3
linha 4
linha 5

- E se fosse o OR (Ou)? No exemplo a seguir, se a idade for 70, as linhas 1,2,3,4,5 serão apresentadas, pois a condição à esquerda (laranja) será verdadeira e à direita (verde) será falsa. E, para o operador OR (Ou), apenas um dos lados precisa ser verdadeiro.

```
if idade >= 18 or idade < 65 :
```

true
linha 1
linha 2
linha 3
linha 4
linha 5

02 Combinando And e Or

- Podemos combinar os operadores And e Or, como veremos no exemplo a seguir, em que a linha 5 é executada.

```
1 idade = 13
2 salary = 10000
3
4 if (idade > 18 and idade < 65) or salary >= 10000:
5     print('adulto ou qualquer um com salario acima de 10k')
```

- A primeira expressão a ser executada é a que está entre os parênteses, destacado em vermelho, devido a precedência. A condição esquerda (idade > 18) é falsa e a direita (idade < 65) é verdadeira, de modo que falso e verdadeiro resulta em falso.
- Uma vez analisado os parênteses, é hora de analisar o lado direito (salary >= 10000), que é verdadeiro. Por fim, devido a estarmos em uma condição de OR, o resultado final é verdadeiro.

VERDADEIRO

FALSO OU VERDADEIRO

FALSO E VERDADEIRO VERDADEIRO

```
4 if (idade > 18 and idade < 65) or salary >= 10000:
5     print('adulto ou qualquer um com salario acima de 10k')
```

03 Negação e Diferença

- O operador not permite que você inverta o valor de expressões e objetos booleanos. Ou seja, se algo era true, vira false e vice-versa.
- Você pode usar esse operador em contextos booleanos, como if e while, como também não booleanos.

03 Negação e Diferença (Continuação)

- Chamamos o operador NOT como negação e, conforme dito anteriormente, inverte os valores booleanos. Na sequência apresentamos o funcionamento do operador NOT na prática.

```
1 print(True)           True
2 print(not True)       False
3
4 print(False)          False
5 print(not False)      True
6
7 estaEmpregado = True
8 print(estaEmpregado)  True
9 print(not estaEmpregado) False
```

Saída

- O operador NOT pode estar associado a IFs, como no exemplo a seguir. Caso o operador NOT não tivesse sido utilizado, a linha 4 não seria apresentada.

```
1 idade = 17
2
3 if not idade >= 18:
4     print('menor de idade')
```

Saída menor de idade

- Por fim, vamos apresentar o operador de DIFERENTE (!=). Se a idade for exatamente 18, a linha 4 será apresentada. Caso a idade seja diferente (!=) de 18, a linha 6 será apresentada.

```
1 idade = 17
2
3 if idade == 18:
4     print('tem 18 anos')
5 elif idade != 18:
6     print('não tem 18 anos')
```

Saída não tem 18 anos

04 IFs aninhados

- É possível ter uma instrução if/elif/else dentro de outra instrução if/elif/else, o que chamamos de IFs aninhados, em inglês, nested IFs statements. Este fato, pode diminuir a legibilidade do código e aumentar a confusão no entendimento, de modo que deve ser evitado.
- Tomemos como exemplo o código apresentado a seguir.

```
1 num = float(input("Informe um número "))
2
3 if num >= 0:
4     if num == 0:
5         print("Zero")
6     else:
7         print("Número positivo")
8 else:
9     print("Número negativo")
```

- Notem que, dentro do IF num >= 0, temos duas instruções:
 - IF num == 0 e um else
 - Caso na linha 1, informássemos o valor 0 no teclado, teríamos o seguinte funcionamento:
 - **Linha 3:** If num >= 0 será verdadeiro, de forma que entra no IF
 - **Linha 4:** If num == 0 será executado, uma vez que a linha 3 é verdadeira. E, como num é zero, a sentença é verdadeira e a linha 5 é executada.
 - A seguir, destacamos em vermelho o fluxo da execução para o cenário supracitado.

```
1 num = float(input("Informe um número "))
2
3 if num >= 0:
4     if num == 0:
5         print("Zero")
6     else:
7         print("Número positivo")
8 else:
9     print("Número negativo")
```

- Para completar o entendimento, experimente executar o programa acima com valores positivos e negativos.

05 Função Range

- A função range retorna uma sequência de números e é comumente utilizada com o comando FOR, que aprenderemos na sequência. Existe 3 opções (“sabores”) para a função range:
 - **range(n)** na qual gera números de 0 até n-1
 - Por exemplo, range(3) gera 0, 1 e 2
 - **range(a,n)** na qual gera números de a até n-1
 - Por exemplo, range(1,3) gera 1 e 2
 - **range(a,n,s)** na qual gera números de a até n-1, pulando de s em s
 - Por exemplo, range(0,5,2) em teoria geraria 0, 1, 2, 3, 4. Entretanto, o último parâmetro (s) que tem o valor 2 significa que ele pula de 2 em 2. Como assim? No 0, ele pula diretamente para 2 e do 2 exatamente para 4, de modo que os 1 e 3 são desconsiderados.

```
1 for n in range(0,5,2):
2     print(n)
3
4 for n in range(0,5):
5     print(n)
6
7 for n in range(5):
8     print(n)
```