# FUNDAMENTOS DE PROGRAMAÇÃO

## SUMÁRIO


EXPRESSÕES BOOLEANAS ..... 2
PQ NÃO = ? ..... 2
OPERADORES LÓGICOS ..... 2
EXECUÇÃO CONDICIONAL (IF) ..... 4
EXECUÇÃO ALTERNATIVA (ELSE) ..... 5
CONDICIONAIS ENCADEADOS ..... 5
CONDICIONAIS ANINHADOS ..... 6
ANINHAMENTO CORRETO ..... 7
FLUXOGRAMA DO IF ..... 8

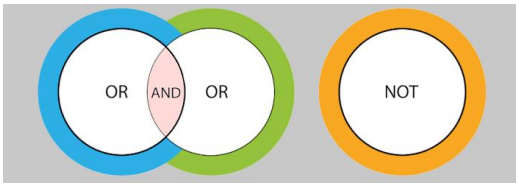

## PQ NÃO = ?

Embora esses operadores provavelmente sejam familiares a você, os símbolos em Python são diferentes dos símbolos da matemática. Um erro comum é usar um sinal de igual sozinho (=) em vez de um duplo (==). Lembre-se de que = é um operador de atribuição e == é um operador de comparação. Também não existem coisas como =< ou =>.

## OPERADORES LÓGICOS

Existem três operadores lógicos: and, or, not (e, ou, não). A semântica (significado) destes operadores é similar aos seus significados em inglês (ou português). Por exemplo, x > 0 and x < 10 é verdadeiro somente se x for maior que 0 e menor que 10.

n%2 == 0 or n%3 == 0 é verdadeiro se qualquer das condições for verdadeira, quer dizer, se o número n for divisível por 2 ou por 3.




## EXPRESSÕES BOOLEANAS

Uma expressão booleana é uma expressão que é verdadeira (True) ou é falsa (False). Em Python, uma expressão que é verdadeira tem o valor True, e uma expressão que é falsa tem o valor False.

O operador == compara dois valores e produz uma expressão booleana:

```
>>> 5 == 5

True

>>> 5 == 6

False
```

No primeiro comando, os dois operadores são iguais, então a expressão avalia como True (verdadeiro); no segundo comando, 5 não é igual a 6, então temos False (falso).

O operador == é um dos operadores de comparação; os outros são:

x != y # x é diferente de y
x > y # x é maior que y
x < y # x é menor que y
x >= y # x é maior ou igual a y
x <= y # x é menor ou igual a y

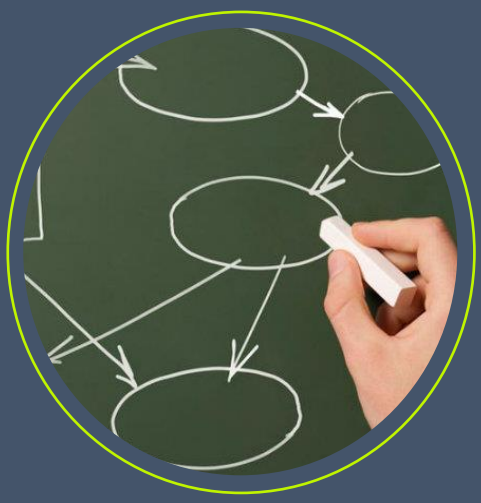
Finalmente, o operador lógico not nega uma expressão booleana, assim, not(x > y) é verdadeiro se (x > y) for falso, quer dizer, se x for menor ou igual a y.

A rigor, os operandos de operadores lógicos deveriam ser expressões booleanas, mas Python não é muito rigoroso. Qualquer número diferente de zero é interpretado como verdadeiro (True):

```
>>> x = 5

>>> x and 1

1

>>> y = 0

>>> y and 1

0
```

Em geral, esse tipo de coisa não é considerado de bom estilo. Se você precisa comparar um valor com zero, deve fazê-lo explicitamente.

Os operandos de operadores lógicos deveriam ser expressões booleanas.

# EXECUÇÃO CONDICIONAL (IF)

Para poder escrever programas úteis, quase sempre precisamos da habilidade de checar condições e mudar o comportamento do programa de acordo com elas. As instruções condicionais nos dão essa habilidade. A forma mais simples é a instrução if (se):

```
if x > 0
    print ("x é positivo")
```

A expressão booleana depois da instrução if é chamada de condição. Se ela é verdadeira (true), então a instrução endentada é executada. Se não, nada acontece.

Assim como outras instruções compostas, a instrução if é constituída de um cabeçalho e de um bloco de instruções:

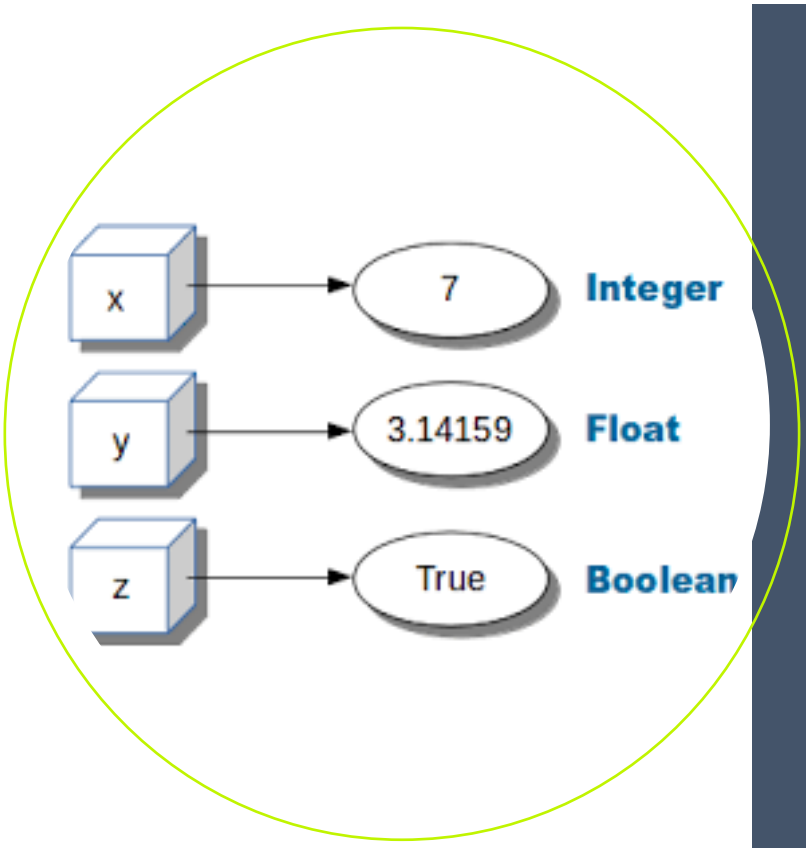
```
CABECALHO:
    PRIMEIRO COMANDO
    ...
    ULTIMO COMANDO
```

O espaçamento para indicar o pertencimento ao bloco aberto é de **4 espaços.**

O cabeçalho começa com uma nova linha e termina com dois pontos (:). Os comandos ou instruções endentados que seguem são chamados de bloco. A primeira instrução não endentada marca o fim do bloco. Um bloco de comandos dentro de um comando composto ou instrução composta é chamado de corpo do comando.

Não existe limite para o número de instruções que podem aparecer no corpo de uma instrução if, mas tem que haver pelo menos uma. Ocasionalmente, é útil ter um corpo sem nenhuma instrução (usualmente, como um delimitador de espaço para código que você ainda não escreveu). Nesse caso, você pode usar o comando pass, que indica ao Python:

```
"passe por aqui sem fazer nada".
```




## CONDICIONAIS ENCADEADOS

Às vezes existem mais de duas possibilidades e precisamos de mais que dois ramos. Uma condicional encadeada é uma maneira de expressar uma operação dessas:

```
if x < y:
    print (x, "é menor que", y)
elif x > y:
    print (x, "é maior que", y)
else:
    print (x, "e", y, "são iguais")
```

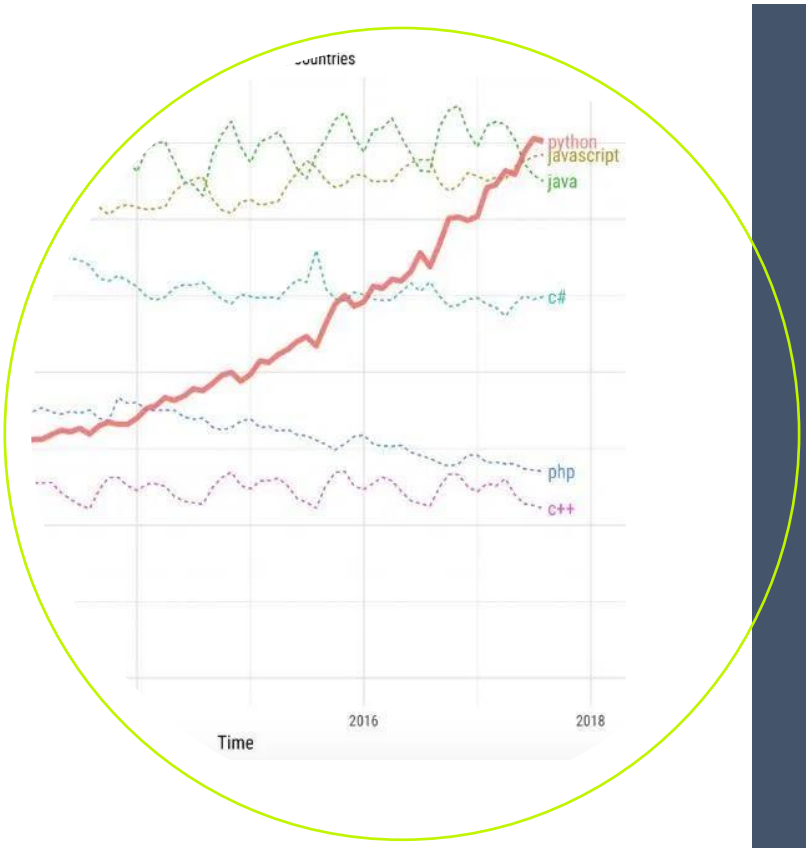
elif é uma abreviação de “else if” (“senão se”). De novo, precisamente um ramo será executado. Não existe limite para o número de instruções elif, mas se existir uma instrução else ela tem que vir por último.

## EXECUÇÃO ALTERNATIVA (ELSE)

Um segundo formato da instrução if é a execução alternativa, na qual existem duas possibilidades e a condição determina qual delas será executada. A sintaxe se parece com:

```
if x % 2 == 0:
    print (x, "é par")
else:
    print (x, "é impar")
```

Se o resto da divisão de x por 2 for 0, então sabemos que x é par, e o programa exibe a mensagem para esta condição. Se a condição é falsa, o segundo grupo de instruções é executado. Desde que a condição deva ser verdadeira (True) ou falsa (False), precisamente uma das alternativas vai ser executada. As alternativas são chamadas ramos (branches), porque existem ramificações no fluxo de execução.




```
if escolha == 'A':
    funcaoA()
elif escolha == 'B':
    funcaoB()
elif escolha == 'C':
    funcaoC()
else:
    print ("Escolha inválida.")
```

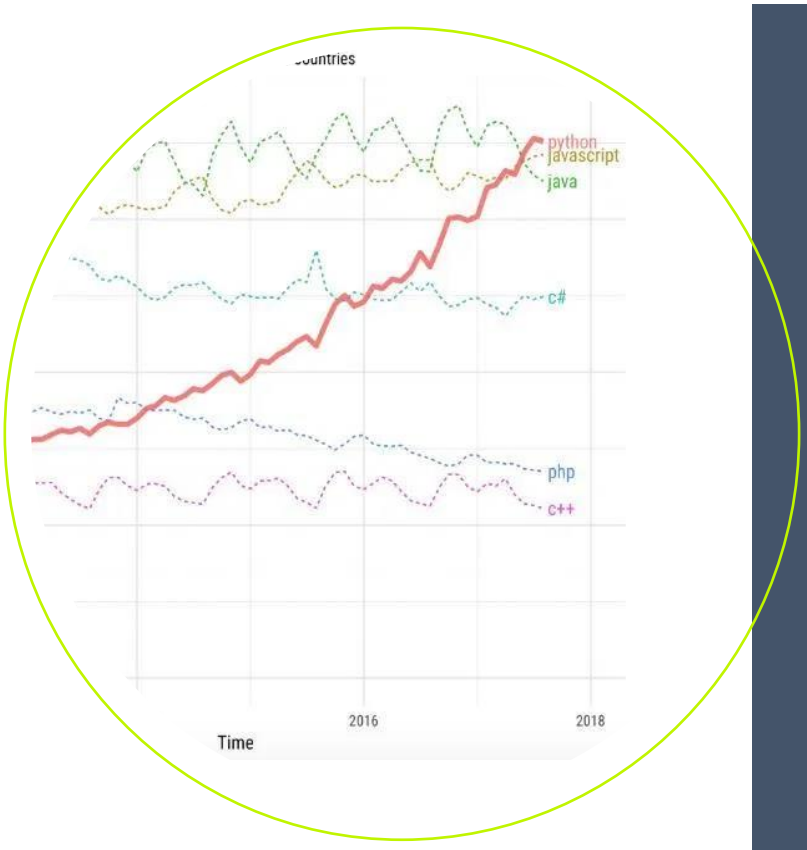
Cada condição é checada na ordem. Se a primeira é falsa, a próxima é checada, e assim por diante.

Se uma delas é verdadeira, o ramo correspondente é executado, e a instrução termina. Mesmo que mais de uma condição seja verdadeira, apenas o primeiro ramo verdadeiro executa.

## CONDICIONAIS ANINHADOS

Um condicional também pode ser aninhado dentro de outra. Poderíamos ter escrito o exemplo tricotômico (dividido em três) como segue:

```
if x == y:
    print (x, "e", y, "são iguais")
else:
    if x < y:
        print (x, "é menor que", y)
    else:
        print (x, "é maior que", y)
```




## ANINHAMENTO CORRETO

A instrução print é executada somente se a fizermos passar por ambos os condicionais, então, podemos usar um operador and:

```
if 0 < x and x < 10:

    print ("x é um número positivo de um só algarismo.")
```

Esses tipos de condições são comuns, assim, Python provê uma sintaxe alternativa que é similar à notação matemática:

```
if 0 < x < 10:

    print ("x é um número positivo de um só algarismo.")
```

O condicional mais externo tem dois ramos. O primeiro ramo contém uma única instrução de saída. O segundo ramo contém outra instrução if, que por sua vez tem dois ramos. Os dois ramos são ambas instruções de saída, embora pudessem conter instruções condicionais também.

Embora a endentação das instruções torne a estrutura aparente, condicionais aninhados tornam-se difíceis de ler rapidamente. Em geral, é uma boa ideia evitar o aninhamento quando for possível.

Operadores lógicos frequentemente fornecem uma maneira de simplificar instruções condicionais aninhadas. Por exemplo, podemos reescrever o código a seguir usando uma única condicional:

```
if 0 < x:

    if x < 10:

        print ("x é um número positivo de um só algarismo.")
```

## FLUXOGRAMA DO IF