



# Resumo - Aula 2

## Operadores de comparação

Todos os exemplos são expressões "verdadeiras"

 Nome	 Exemplo
<u>Igual</u>	<code>10 == 10</code>
<u>Maior</u>	<code>5 &gt; 3</code>
<u>Maior ou igual</u>	<code>3 &gt;= 3</code>
<u>Menor</u>	<code>2 &lt; 3</code>
<u>Menor ou igual</u>	<code>2 &lt;= 2</code>
<u>Diferente</u>	<code>1 != 2</code>



Quando escrevemos `10 == 10` estamos perguntando se essa expressão é verdadeira. O resultado dessa expressão (ou de qualquer outra utilizando os operadores de comparação) é sempre um valor *booleano*, ou seja `True` ou `False`

## Condicionais: `if/else/elif`



O Python utiliza as palavras `if/elif/else` para executar instruções caso uma expressão seja `verdadeira` (ou seja, `True`)

Exemplo:

```
idade = int(input("Qual a sua idade?")) # Aqui estamos transformando o input em int
if idade > 18: # Se idade é maior que 18, execute o código "dentro" do "if"
    print("Você é um adulto(a)")
elif idade >= 12: # Se idade não é maior que 18 mas é maior ou igual a 12...
    print("Você é um(a) adolescente!")
elif idade >= 4: # Podemos utilizar quantos elif's quisermos
    print("Você é uma criança!")
else: # Nenhuma das condições acima foi considerada True
    print("Você é um bebê.")
```

Algumas linguagens utilizar `{ }` para identificar o bloco de código dentro de um `if/else` que tem que ser executado. Por exemplo:

```
if idade > 18 {
    print(idade)
} else {
    print("Só falo com maiores de idade.")
}
```

No caso, Python utiliza a identificação por **identação**, logo, a quantidade de "espaço" antes de uma linha de código importa.

```
if (10 % 5 == 0):
    print("É divisível")
```

Vai dar um erro se você tentar executar, dizendo que "espera-se um bloco indentado", ou seja, com tabulação/espaçamento para identificar o bloco de código a ser executado caso a expressão `(10 % 5 == 0)` seja verdadeira.

# Operadores lógicos

Os três principais operadores lógicos do Python são: `and`, `or` e `not`.

Eles são utilizados para "conectar" expressões que resultam em `True` ou `False`. Por exemplo:

```
x = 10
if (x == 10) and (x % 2 == 0):
    print("X é igual a 10 e divisível por 2")
elif (x == 10) or (type(x) == int):
    print("X é igual a 10 e x é um inteiro")
else:
    print("Nenhuma das anteriores...")
```

Já o `not`, é simplesmente uma inversão de uma expressão. Ou seja `not True`  $\Rightarrow$  `False` e `not False`  $\Rightarrow$  `True`.

```
if not (idade >= 18): # Observe que (idade >= 18) resulta em True ou False, `not` inverte isso
    print("Somente maiores de 18 anos podem dirigir.")
```

# Funções

A sintaxe para definir uma função no Python é a seguinte.

```
# Essa é uma função muito boba: recebe um valor e retorn ele mesmo.
# equivalente a f(x) = x na matemática
def f(x):
    return x

print(f(10)) # Imprime 10 na tela
```

## Função que não retorna valor nenhum

```
def imprime_1_2_3():
    print(1)
    print(2)
    print(3)

imprime_1_2_3() # => 1 2 3
```

## Composição de função

```
def f(x):
    return 2 * x

def g(x)
    return x + 10

f(g(10)) # => 40
# g(10) => 20, f(20) => 40

g(f(10))
```

## Função chamando outra função

```
def dizer_oi():
    print("oi")

def dizer_tchau():
    print( "tchau")

def conversar():
    dizer_oi()
    dizer_tchau()
```

## Função com múltiplos argumentos

```
def soma(a, b):  
    return a + b
```

## Desafio FizzBuzz

Peça para o usuário digitar um número inteiro.

Se esse número for múltiplo de 3, imprimir "Fizz" na tela.

Se esse número for múltiplo de 5, imprimir "Buzz" na tela.

Se esse número for múltiplo de 3 E de 5, imprimir "FizzBuzz" na tela.

## Gabarito

Para verificar que seu programa funciona, execute ele 15 vezes, de 1 a 15. Os resultados devem ser os seguintes: (⇒ sem nada ao lado significa que nada deve ser impresso na tela).

1 ⇒

2 ⇒

3 ⇒ "Fizz"

4 ⇒

5 ⇒ "Buzz"

6 ⇒ "Fizz"

7 ⇒

8 ⇒

9 ⇒ "Fizz"

10 ⇒

11 ⇒

12 ⇒ "Fizz"

13 ⇒

14 ⇒

15 ⇒ "FizzBuzz"