Operadores e Expressões em Python

Nesta seção aprenderemos os vários tipos de operadores em Python e mostraremos como criar expressões envolvendo esses operadores. Esta seção será um pouco densa em termos de conteúdo, mas o conteúdo dela é extremamente útil para que você domine Python.

Operadores Aritméticos

A linguagem Python nos permite criar expressões matemáticas com imensa facilidade. No dia a dia, nem todos os programas precisarão de expressões matemáticas complexas, mas é importante saber como usar os operadores aritméticos do Python.

Vejamos alguns exemplos de expressões matemáticas em Python.

```
# Soma e subtração
print(10 + 2 - 7)

# Potência: 2 ^ 4
print(2 ** 4)

# Divisão de ponto flutuante
print(20 / 6)

# Divisão inteira, sem considerar o resto
print(20 // 6)
```

```
5
16
3.3333333333333335
3
```

O operador `/` realiza divisão de ponto flutuante em Python 3, e o operador `//` realiza divisão inteira. Em Python 2 esse mesmo operador realizava divisão inteira. Então é importante ficar atento a esta diferença quando se está programando. Para descobrir a versão do Python que você está usando, basta digitar no terminal (ou prompt de comandos, se você está usando o Windows): `python --version`.

Se quisermos exibir só o resto da divisão, usamos o operador `%`, como mostrado abaixo:

```
20 % 6 # Resto da divisão
```

```
2
```

Expressões matemáticas em Python podem obedecem as mesmas regras de sinal com as quais estamos acostumados:

```
1 * -2 * 3 * -4 * 5 * -6
```

```
-720
```

Como vimos nos exemplos acima, a maioria dos operadores e operações em Python se comportam como os operadores e operações da Matemática:

- Operadores aritméticos: `+`, `-`, **`,`** **`/`**, `//` **(divisão inteira),** `*` (potência)

A tabela abaixo resume o funcionamento dos operadores aritméticos em Python:

Operador	**Descrição**	**Exemplo**
`+`	soma dois valores	`5 + 2` resulta em `7`
`-`	subtrai dois valores	`5 - 2` resulta em `3`

Operador	Descrição	Exemplo
`*`	multiplica dois valores	`5 * 2` resulta em `10`
`/`	divide dois valores (sem arredondar)	`5 / 2` resulta em `2.5`
`//`	divide dois valores (arredondando para baixo)	`5 // 2` resulta em `2`
`%`	resto da divisão	`5 % 2` resulta em `1`
`**`	exponenciação	`5 ** 2` resulta em `25`

Tabela 1. Operadores aritméticos

Operadores Lógicos

O próximo tipo de primitiva da linguagem Python que estudaremos são os operadores lógicos. Os mais importantes são: `and`, `not`, e `or`.

A tabela abaixo resume o funcionamento dos operadores lógicos em Python:

Operador	Descrição	Exemplo

`and`	"e" lógico	`True and True` resulta em `True`
		`True and False` resulta em `False`
		`False and True` resulta em `False`
		`False and False` resulta em `False`

`or`	"ou" lógico	`True or True` resulta em `True`
		`True or False` resulta em `True`
		`False or True` resulta em `True`
		`False or False` resulta em `False`

`not`	"não" lógico	`not True` resulta em `False` `not False` resulta em `True`

Tabela 2. Operadores lógicos

Operadores Relacionais

Python possui também operadores relacionais (de comparação), como os da matemática: `<`, `>`, `<=`, `>=`, `==`, `!=`.

A tabela abaixo resume o funcionamento dos operadores de comparação em Python. Nela assumimos que estamos fazendo `a operador b`, ou seja, que estamos aplicando um operador a dois operandos, `a` e `b`.

Operador	Descrição	Exemplo
`==`	`True` se `a` e `b` são iguais	`5 == 2` resulta em `False`
`!=`	`True` se `a` e `b` são diferentes	`5 != 2` resulta em `True`
`>`	`True` se `a` é maior que `b`	`5 > 2` resulta em `True`
`<`	`True` se `a` é menor que `b`	`5 < 2` resulta em `False`
`>=`	`True` se `a` é maior ou igual a `b`	`5 >= 2` resulta em `True`
`<=`	`True` se `a` é menor ou igual a `b`	`5 <= 2` resulta em `False`

Tabela 3. Operadores relacionais

Talvez os únicos operadores acima que podem causar dúvidas são os seguintes:

- `==` (teste de igualdade): testa se duas coisas são iguais. Por exemplo `10 == 10` retorna `True` e `abacate == melancia` retorna `False`. Tome cuidado: não confunda o operador `==` (teste de igualdade) com o operador `=` (operador de atribuição). Este último é usado para atribuir um valor a uma variável. Por exemplo: `a = 20`. É muito comum as pessoas confundirem esses operadores, principalmente em condicionais (`if` e `else`).
- `!=` (teste de diferença): este operador simplesmente testa se dois valores são diferentes. Exemplo: `10 != 20` retorna `True`. Ele é o oposto do operador `==`.

Operadores de Atribuição

Você já vem usando o principal operador de atribuição do Python (o operador `=`), mas existem outros operadores desse tipo:

- `=`: operador de atribuição. Dada uma variável `x`, ao fazermos `x = valor` atribuímos o valor à variável `x`. Deste momento em diante, `x` é um sinônimo de `valor`. Por exemplo, se fizermos `x = 5`, ao imprimir `x`, o valor `5` será impresso.
- `+=`: é equivalente a fazer `x = x + valor`. Por exemplo, se `x` valer 10 e fizermos `x += 2`, `x` passará a ter o valor 12.
- `-=`, `=`, `/=`, `//=`, `%=`, `*=`: funcionam da mesma forma que o `+=`. Por exemplo, se `x` valer 5 e fizermos `x *= 3` obteremos o valor 15. Na prática, o funcionamento desses operadores é o seguinte: `x op= valor` é equivalente a `x = x op valor`, em que `op` é algum dos operadores listados anteriormente.

Precedência de Operadores

Quando criamos uma expressão em Python, existe em uma ordem em que as subexpressões são avaliadas. Essa ordem é determinada por algo que chamamos de *precedência de operadores*.

Por exemplo, se tivermos a expressão `a ** 2 + b * 3 % 2`, e assumirmos que `a = 4` e `b = 3` antes de avaliarmos a expressão, teremos a seguinte sequência de operações:

- `a ** 2 + b * 3 % 2`: expressão original.
- `4 ** 2 + 3 * 3 % 2`: os valores de `a` e `b` são substituídos no lugar das variáveis.
- `16 + 3 * 3 % 2`: a expressão de potenciação é avaliada (`4 ** 2 = 16`). Ela é a primeira a ser avaliada porque possui a maior precedência.
- `16 + 9 % 2`: a expressão `3 * 3` é avaliada. O operador **é avaliado antes do `%` simplesmente porque apareceu antes na expressão. Tanto o operador** quanto o `%` possuem a mesma precedência. Quando dois operadores possuem a mesma

precedência, o que aparece primeiro (mais à esquerda) na expressão é avaliado primeiro.

- `16 + 1`: a expressão `9 % 2` é avaliada, pois o operador `%` possui maior precedência que o operador `+`.

- `17`: a expressão `16 + 1` é avaliada.

A tabela a seguir resume a precedência dos principais operadores da linguagem Python:

Operador
`**`
`* / % //`
`+ -`
`<= < > >=`
`== !=`
`= %= /= //= -= += = *=`
`not or and`

Tabela 4. Precedência de operadores em Python

É possível inserir parênteses em expressões para indicar a ordem na qual sub-expressões devem ser avaliadas. Por exemplo, se tivermos a expressão `5 * 4 + 3`, teremos como resultado o valor `23`, pois o operador de multiplicação possui maior precedência que o operador de soma, portanto a multiplicação será realizada primeiro. Se quisermos que a soma seja realizada primeiro, podemos escrever `5 * (4 + 3)`, que dará `35` como resultado. Uma forma de ver isso é como se os parênteses tivessem maior precedência que todos os outros operadores, então o que está dentro deles será avaliado primeiro.

Ufa! Chegamos ao final desta seção! Vimos muita coisa, mas precisamos dessa base para avançarmos para os tópicos mais interessantes. Em breve estaremos escrevendo programas não-triviais em Python!

Exercícios

- Qual será o resultado da expressão abaixo?

```
2 + 3 * 5 + 30 // 10
```

▶ Clique para ver a solução

- Qual será o resultado da expressão abaixo?

```
True or False and not True
```

▶ Clique para ver a solução

Playground

script.py

```
# Use este espaço para praticar o que você aprendeu nesta seção.

# Basta digitar o código no espaço abaixo e clicar 'run'.
```

IPython Shell

```
In [1]:
```