

Variáveis, Tipos e Operações Aritméticas

MC102

Operações Aritméticas

Apenas o básico

`2 + 3`

`5`

`1 - -5`

`6`

`3 * 5.0`

`15.0`

`3 x 5.0`

`SyntaxError: invalid syntax`

`10 / 4`

`2.5`

`2 / 0`

`ZeroDivisionError: division by zero`

`2.5 ** 3`

`16.625`

Precedência

As operações seguem uma ordem de precedência: $** \rightarrow * / \rightarrow + -$

$$2 + 3 * 7$$

23

Em caso de empate, o operador da esquerda tem a maior precedência.

$$2 + 3 * 7 / 2$$

12.5

Mas podemos “forçar” a precedência utilizando **parêntesis**.

$$(2 + 3) * 7 / 2$$

17.5

Precedência

i Info

Não existe chaves {} ou colchetes [] para simbolizar precedência, eles significam outras coisas!

⚡ Danger

Cuidado quando vamos fazer o *denominador* de frações.

$$2 + 3 / 4 + 5$$

$$2 + \frac{3}{4} + 5$$

7.75

$$(2 + 3) / (4 + 5)$$

$$\frac{2+3}{4+5}$$

0.5555555556

Variáveis

Movimento Uniformemente Variado

Calcular a posição final s de um objeto:

- inicialmente na posição s_0
- que se move com aceleração constante a
- durante t segundos
- com velocidade inicial v_0

Sorvetão! 🍦

$$s = s_0 + v_0 t + a \frac{t^2}{2}$$

Em python, poderíamos fazer algo tipo assim:

```
10 + 2 * 8 + 3 * 8 ** 2 / 2
```

🤢 *Ergh, cringe...* Dá para saber o que cada um desses número significa? E se eu quiser mudar a velocidade inicial, é fácil? E se pudessemos guardar os valores de forma a dar um **nome** para eles?

Movimento Uniformemente Variado

```
1 s0 = 10
2 v0 = 2
3 t = 8
4 a = 3
5 s = s0 + v0 * t + (a * t ** 2) / 2
```

122

Atribuição

Variáveis são caixinhas, com *nomes*, onde guardamos *valores*.

Isso facilita, depois, nos *referirmos* ao valor que ela guarda.

O sinal `=` se refere à operação de **atribuição**:

```
nome = valor
```

- Imagine que, ao realizar essa operação, o python cria uma caixinha, chamada `nome`, com o `valor` dentro dela.
- Agora, caso você queira se referir àquele valor, basta chamar pelo nome!

```
1 x = 10
```

```
2 x + 2
```

```
12
```

Atribuição

```
1 x = 10
2 y = 20
3 y / x
```

```
2.0
```

```
1 a = 3
2 b = 4
3 c = a * b
4 b = -10
```

Qual o valor de `c` na linha 3, antes de executarmos a linha 4? E depois da linha 4?

Na programação, os comandos são executados em ordem, nesse caso, de cima para baixo.

Atribuição

⚡ Danger

O operador de atribuição = não é o sinal de igual da matemática!

```
SyntaxError: cannot assign to expression here.
```

⚡ Danger




Qual o valor de uma variável que ainda não foi definida?

```
NameError: name 'x' is not defined
```

Em Python, a variável passa a existir a partir da primeira atribuição = em seu nome.

Regras

Como vimos, a operação de atribuição possui o formato: `nome = valor`

- O nome precisa começar com uma letra ou `_` (underscore, underline).
 - ▶  Válido: `x` `_y` `a1` `batata_inglesa`
 - ▶  Inválido: `9e` `?o`
- Os demais caracteres podem ser letras, números ou `_`
 - ▶ Mas não pode ser espaços ou outros sinais
 - ▶  Inválido: `x?` `minha variavel`
- Existem diferenças entre minúsculo e maiúsculo
 - ▶ `x` é diferente de `X`
 - ▶ Para facilitar, dê sempre preferência em tudo minúsculo.
- Alguns nomes são protegidos e você não pode usar
 - ▶ `while`, `if`, `break` etc

Regras

Boas práticas:

- Use nomes significativos
 - ▶ `velocidade_inicial` é melhor do que `a`
- Evite nomes grandes
 - ▶ `nome_aluno` é melhor do que `nome_do_aluno_de_mc102`
- Seja consistente
 - ▶ se você possui uma variável chamada `nome_aluno` nada de criar uma `aluno_idade`, prefira `idade_aluno`.

Essas dicas podem parecer “bobas” para quem está começando, mas elas simplificam **muito** o processo de codar.

Tipos

Tipos de dados

Como vimos, a operação de atribuição se dá a partir da sintaxe: `nome = valor`

Em Python, os valores possuem **tipo**.

- `10` é do tipo `int` (inteiro)
- `3.14` é do tipo `float` (decimal)
- `'MC102'` é do tipo `str` (*string* ou literal)

O tipo de um valor define que operações podem ser feitas:

```
100 + 2.0
```

```
102.0
```

```
'mc' + '102'
```

```
'mc102'
```

```
'mc' + 102
```

```
TypeError: can only concatenate str (not "int") to str
```

```
'mc' * 2
```

```
'mc'
```

float

O nome `float` vem de *floating point* (ponto flutuante), que deriva do fato de não haver um número fixo de dígitos antes e depois da vírgula.

Ele é utilizado para representar valores decimais, vulgarmente conhecido como “números com vírgula”.

```
1 15.1
2 3.14
3 -2.7
4 2e-3
```

Danger

Apesar de nós, brasileiros, escrevermos os separador com vírgula, o Python segue o padrão do inglês, separando com ponto.

Assim, `10.1` equivale à 10.1, mas `10,1` o número 10 seguido do número 1.

float

Existe o mito que computadores são maravilhosos em fazer conta... o que não é verdade!

15.3 / 3

5.1000000000000005

5.3 - 2.1

3.1999999999999997

Pelo mesmo motivo que em Física Experimental I é preciso lidar com incerteza, os computadores não possuem memória infinita.

Assim, eles também carregam e acumulam erros conforme fazemos operações com `float`.

Vocês não precisam se preocupar muito com isso, até porque, nós computeiros temos uma matéria inteira só para lidar com isso (Cálculo Numérico, MS211).



Tip

Só usem `float` se for realmente necessário.

Saboor

Digamos que você quer dividir 11 garrafas de Bebida Mansão Maromba™ Saboor Energético® entre seus 5 amigos... e você tá com preguiça de calcular e quer fazer um programa para isso (claro).

```
11 / 5
```

```
2.2
```

Como você vai colocar teu amigo no Uber com 2.2 garrafas de 1 litro?

Os computadores, preocupados com esse dilema, já pensaram nisso: **divisão inteira //**

```
11 // 5
```

```
2
```

Agora você pode ficar tranquilo, mandando cada um dos seus amigos para casa com exatas 2 garrafas.

Mas vai sobrar alguma para você?? Toma esse operador de **resto %**

```
11 % 5
```

```
1
```

str

O tipo que representa texto em Python é o `str`, chamado *string*.

O valor é representado por uma sequência de caracteres envolto por `"` ou `'`

```
1 "sabor energético"  
2 'olha esses gráficos cara'
```

Quando precisamos de um texto com mais de uma linha, usamos `"""` como delimitador.

```
1 texto = """Esse é um texto com  
2 várias linhas, na verdade com  
3 três linhas."""
```

Teremos mais aulas sobre strings ao longo do curso.

Convertendo

Podemos converter entre tipos usando `int()`, `float()` e `str()`

```
str(3)
```

```
'3'
```

```
float("3.5")
```

```
3.5
```

```
int(3.1)
```

```
3
```

```
int('3.1')
```

```
ValueError: invalid literal for int() with base 10
```

Voltando às operações

Lembra quando vimos:

```
2 + 3
```

```
5
```

```
3 * 5.0
```

```
15.0
```

E se...

```
"hot" + "dog"
```

```
'hotdog'
```

```
'sabor' * 5
```

```
'saborsaborsaborsaborsabor'
```

“Ah, então tudo funciona??” – diz o aluno ansioso...

```
"aaa" - "a"
```

```
TypeError: unsupported operand type(s)
```