Tipos de Dados

Prof. Alberto Costa Neto

Programação em Python

Qual é o significado de “Tipo” ?

- Variáveis, Literais e Constantes em Python têm um “tipo”
- Python sabe a diferença entre um número inteiro e uma string
- Por exemplo, “+” significa “somar” se os operandos são números e “concatenar” se for string

```
>>> i = 1 + 4
>>> print(i)
5
>>> h = 'hello ' + 'there'
>>> print(h)
hello there
```

concatenar = juntar, unir

Questões de Tipo

- Python sabe o tipo “type” de tudo
- Algumas operações são proibidas
- Você não pode “adicionar 1” a uma string
- Podemos perguntar a Python qual é o tipo de qualquer coisa através da função type()

```
>>> h = 'hello ' + 'there'
>>> h = h + 1
Traceback (most recent call last):
  File "<stdin>", line 1, in
<module>
TypeError: cannot concatenate
'str' and 'int' objects
>>> type(h)
<class 'str'>
>>> type('hello')
<class 'str'>
>>> type(1)
<class 'int'>
```

Vários Tipos de Números

- Números têm dois tipos principais

  > Inteiros (int) são números inteiros:
  -14, -2, 0, 1, 100, 401233

  > Números em Ponto Flutuante (float)
  são números decimais:
  -2.5 , 0.0, 98.6, 14.0

- Existem outros tipos de números –
  São variações sobre int e float

```
>>> x = 1
>>> type (x)
<class 'int'>
>>> temp = 98.6
>>> type(temp)
<class 'float'>
>>> type(1)
<class 'int'>
>>> type(1.0)
<class 'float'>
>>>
```

Divisão de Inteiros

- Divisão de inteiros trunca resulta em Ponto Flutuante (float)
- Para a divisão inteira, use o operador // que funciona mesmo entre reais

Isto mudou a partir de Python 3.0

```
>>> print(10 / 2)
5.0
>>> print(9 / 2)
4.5
>>> print(10.0 / 2.0)
5.0
>>> print(99 // 100)
0
>>> print(99.0 / 100.0)
0.99
>>> print(9 // 2)
4.0
>>> print(99.0 // 100.0)
0.0
```

Misturando Inteiros e Reais (Ponto Flutuante)

- Na divisão inteira, quando você executa uma operação aonde um operando é um inteiro e o outro operando é um real, o resultado é um real!
- O inteiro é convertido em real antes da operação
- Na divisão real, o resultado sempre é real

```
>>> print(99 // 100)
0
>>> print(99 // 100.0)
0.0
>>> print(99.0 // 100)
0.0
>>> print(1 + 2 * 3 // 4.0 - 5)
-3.0
>>>
```

Conversões de Tipos

- Quando você coloca um inteiro e um real em uma expressão, o inteiro é implicitamente convertido em um real (float)
- Você pode controlar isto com as funções built-in (funções que já vêm no Python) int() e float()

```
>>> print(float(99) // 100)
0.0
>>> i = 42
>>> type(i)
<class 'int'>
>>> f = float(i)
>>> print(f)
42.0
>>> type(f)
<class 'float'>
>>> print(1 + 2 * float(3) // 4 - 5)
-3.0
```

Conversões de String

- Você pode usar também int() e float() para converter entre strings e inteiros/reais
- Você receberá um erro (error) se a string não contiver caracteres numéricos

```
>>> sval = '123'
>>> type(sval)
<class 'str'>
>>> print(sval + 1)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: Can't convert 'int' object
to str implicitly
>>> ival = int(sval)
>>> type(ival)
<class 'int'>
>>> print(ival + 1)
124
>>> ola = 'hello bob'
>>> num = int(ola)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: invalid literal for int()
with base 10: 'hello bob'
```

Operações com String

- Alguns operadores aplicam-se a strings
  + significa “concatenação”
  * significa “concatenação múltipla”
- Python sabe quando está lidando com uma string ou um número e comporta-se de forma apropriada

```
>>> print('abc' + '123’)
abc123
>>> print('Hi' * 5)
HiHiHiHiHi
```