Python/Conceitos básicos/Tipos e operadores

< Python | Conceitos básicos

Python é uma linguagem de tipos dinâmicos, ou seja, não é necessário fazer casting como em Java, Pascal ou C.

Índice

Objetos

Em Python tudo é objeto. Isso quer dizer que um objeto do tipo `string`, por exemplo, tem seus próprios métodos.

O conceito de variável é uma associação entre um nome e um valor, mas não é necessário declarar o tipo da variável, portanto, o tipo relacionado a variável pode variar durante a execução do programa isto implica em muitos aspectos no uso da linguagem.

Este conceito é chamado em programação de "duck typing" (tipagem do pato) - baseado na expressão, em inglês, devida a James Whitcomb Riley:

> Quando eu vejo uma ave que caminha como um pato, nada como um pato e grasna como um pato, eu chamo esta ave de "pato"[1]

Tipos numéricos

Existem 4 tipos numéricos:

- inteiro (int)
- ponto flutuante (float)
- booleano (bool)
- complexo (complex)

Suportam adição, subtração, multiplicação e divisão e também podem se relacionar.

Mesmo os tipos não sendo declarados explicitamente, eles sempre irão assumir um tipo de dado, abaixo, exemplos de retornos de tipos:

- Tipo inteiro:

```
>>> a = 1
>>> type(a)
```

```
<type 'int'>
```

Um cuidado que se deve tomar é que o tipo inteiro é de *precisão infinita*, ou seja, um programador descuidado pode gerar um número inteiro que ocupe toda a memória do computador. Por exemplo, vimos anteriormente o arquivo fatorial.py:

```
# Arquivo fatorial.py
def fat(n):
  if n <= 1:
    return 1
  return fat(n-1) * n
```

Python consegue calcular o fatorial de *qualquer* inteiro, retornando sempre um inteiro, com precisão total. Os limites são apenas o tempo de processamento e a memória do computador:

```
>>> import fatorial
>>> a = fatorial.fat(5)
>>> a
>>> b = fatorial.fat(a)
>>> b
>>> c = fatorial.fat(b)  # nao faça isso!!!
>>> c  # nem chega aqui
```

- Tipo ponto flutuante:

```
>>> a = 1.0
>>> type(a)
<type 'float'>
```

- Tipo booleano:

```
>>> a = True
>>> type(a)
<type 'bool'>
```

- Tipo complexo:

```
>>> a = 4+3j
>>> type(a)
<type 'complex'>
```

E eles mudam de tipo dinamicamente por exemplo, a variável a:

```
>>> a = 1
>>> type(a)
<type 'int'>
>>> a = 1.0
>>> type(a)
<type 'float'>
>>>
```

Operadores são símbolos que atuam sobre variáveis e valores. Existem:

- Operadores aritméticos (+, -, *, /, %, **, +=, -=, *=, /=, %=, **=):

```
>>> a = 1
```

```
>>> b = 2
>>> print a + b
3

ou

>>> a = 5
>>> print a**2
25
```

- Operadores de comparação (>, <, ==, >=, <=, <>, !=, is, in):

```
>>> a = 1
>>> b = 2
>>> a == b
False
>>> print a > b
False
```

- Operadores lógicos (and, or, not):

```
>>> nome = "leonardo"
>>> idade = 22
>>> nome == "leonardo" and idade < 23
True
```

- Operadores de atribuição (=):

```
>>> a = 1
```

Strings

- Substituição em strings:

A substituição em strings acontece com o operador %, para substituir strings usa-se %s, para substituir decimais usa-se %d e para substituir floats usa-se %f.

Exemplo simples:

```
print "Bom dia! Hoje e' %02d/%02d/%04d" % (26, 1, 2011)
```

Exemplo, escrevendo a saída formatada em um arquivo:

```
from codecs import *
f = open ("arquivo.txt", "a+", encoding='utf-8' )
f.write (` %s ` % nome_artigo() ) #A referência %s para strings
f.close()
```

Referências

1. Para mais detalhes sobre "duck typing", consulte a Wikipedia em inglês: w:en:Duck typing