Computadores e Programação

2009–2010 1º semestre — aula 2

Helmut Wolters

helmut@coimbra.lip.pt

2009-10-02

Listas

- As listas são sequências heterogéneas e mutáveis de objectos. Para criar um objecto deste tipo enumeram-se os elementos da lista, separados por vírgulas, entre parêntesis rectos.

  ```
  L = ['abc', 123, 'Python']
  ```

- Uma lista vazia é representada por `[]`.
- Como as listas são objectos mutáveis, podemos alterar um objecto por outro, ou remover um elemento da lista.

Listas

```
>>> stuff = ['123',123,1 +3j,'numbers']
>>> stuff
['123', 123, (1+3j), 'numbers']
>>> del stuff[2]
>>> stuff
['123', 123, 'numbers']
>>> stuff[0] = 2*123
>>> stuff
[246, 123, 'numbers']
```

Listas

- Para acrescentar um objecto a uma lista, utiliza-se o método *append*:

```
>>> clubs = ['Benfica', 'Sporting']
>>> clubs.append('Porto')
>>> clubs
['Benfica', 'Sporting', 'Porto']
```

- Uma lista pode conter outras listas. Uma matriz 2×2 pode ser guardada numa lista de elementos em que cada elemento é uma linha da matriz, também representada numa lista:

```
>>> M = [[1,2,3],[4,5,6],[7,8,9]]
>>> print M[2][1]
8
```

Listas

- Tal como as *strings*, as listas podem ser concatenadas e repetidas usando os operadores + e *:

```
>>> L =['a','ab']
>>> L = L + L
>>> L
['a', 'ab', 'a', 'ab']
>>> 2*L
['a', 'ab', 'a', 'ab', 'a', 'ab', 'a', 'ab']
```

- As listas podem conter referências a outros objectos.

Listas

```
>>> x = 123
>>> L = [x,x*x]
>>> L
[123, 15129]
>>> L  = 2*L
>>> L
[123, 15129, 123, 15129]
>>> L = L[2:] + [2*L[0]]
>>> L
[123, 15129, 246]
>>> L.sort()
>>> L
[123, 246, 15129]
```

Tuplas

- A grande diferença entre as tuplas e as listas é que as primeiras são objectos *imutáveis*. Na construção de uma tupla utilizam-se parêntesis curvos em vez de rectos:

```
>>> t = (1,'a',1j)
>>> type(t)
<type 'tuple'>
>>> print t
(1, 'a', 1j)
>>> type(t[2])
<type 'complex'>
```

- Existe uma notação especial para designar uma tupla de 1 único elemento: `('single',)` e *não* `('single')`!

Tuplas

- Tal como as outras sequências, as tuplas podem ser indexadas e concatenadas. O desempacotamento de uma tupla pode ser feito de uma forma muito simples:

```
>>> x,y,z = ('one','two','three')
>>> print x,y,z
one two three
```

- O número de nomes do lado esquerdo do sinal = têm de ser igual ao dos elementos da sequência, caso contrário ocorre um erro. O desempacotamento também funciona para *strings* e listas.

```
>>> c1,c2,c3,c4 = 'Spam'
>>> print c3
a
```

Tuplas

- As tuplas podem conter sequências que contêm outras sequências. Apesar de serem objectos imutáveis, as tuplas podem conter objectos mutáveis, tais como listas:

```
>>> t = ('Python',['C','Pascal','Perl'])
>>> id(t)
1078912972
>>> lang = t[1]
>>> lang[2] = 'Python'
>>> print t
('Python', ['C', 'Pascal', 'Python'])
>>> id(t)
1078912972
```

Tuplas

- Neste exemplo,a tupla `t` é imutável, mas porque um dos seus elementos é uma lista, e esta foi alterada pelo programa, o resultado é um pouco surpreendente... O adjectivo imutável refere-se à tupla em si, não aos objectos contidos na tupla!
- Embora as tuplas possam parecer redundantes face às listas, é por vezes muito útil ter a certeza que uma certa "lista" de objectos não é alterada. O empacotamento numa tupla é ideal para estes casos.
- Veremos adiante que as tuplas têm um papel muito importante na linguagem Python. Por exemplo, as funções podem devolver objectos múltiplos empacotados numa tupla.