Tuplas

Prof. Alberto Costa Neto

Programação em Python

Tuplas são parecidas com listas

- Uma Tupla é outro tipo de seqüência que funciona de forma parecida com uma lista – pois tem elementos que são indexados iniciando de 0

```
>>> x = ('jose', 'fred', 'maria')
>>> print(x[2])
maria
>>> y = ( 1, 9, 2 )
>>> print(y)
(1, 9, 2)
>>> print(max(y))
9
```

```
>>> for iter in y:
...     print(iter)
...
1
9
2
>>>
```

mas... Tuplas são “imutáveis”

Diferentemente de uma lista, uma vez criada a tupla não pode ter seu conteúdo alterado, assim como uma string

```
>>> x = [9, 8, 7]
>>> x[2] = 6
>>> print(x)
>>>[9, 8, 6]
>>>
```

```
>>> y = 'ABC'
>>> y[2] = 'D'
Traceback:'str'
object does
not support item
Assignment
>>>
```

```
>>> z = (5, 4, 3)
>>> z[2] = 0
Traceback:'tuple'
object does
not support item
Assignment
>>>
```

Coisas que não podemos fazer com tuplas

```
>>> x = (3, 2, 1)
>>> x.sort()
Traceback:
AttributeError: 'tuple' object has no attribute 'sort'
>>> x.append(5)
Traceback:
AttributeError: 'tuple' object has no attribute 'append'
>>> x.reverse()
Traceback:
AttributeError: 'tuple' object has no attribute 'reverse'
>>>
```

Comparativo de funções disponíveis (métodos) em Tuplas e Listas

```
>>> l = list()
>>> dir(l)
['append', 'clear', 'copy', 'count', 'extend', 'index',
'insert', 'pop', 'remove', 'reverse', 'sort']

>>> t = tuple()
>>> dir(t)
['count', 'index']
```

Tuplas são mais eficientes

- O interpretador de Python não tem que criar uma estrutura modificável para uma tupla, como nas listas
- Portanto, a estrutura usada é mais simples e eficiente em termos de memória e desempenho que a da lista
- Logo, em nossos programas, ao criar “variáveis temporárias”, é preferível usar tuplas no lugar de listas

Tuplas e Atribuições

- Podemos também colocar uma tupla no lado esquerdo de um comando de atribuição
- Podemos até omitir parênteses

```
>>> (x, y) = (4, 'fred')
>>> print(y)
fred
>>> a, b = 99, 98
>>> print(a)
99
```

Tuplas e Dicionários

- A função (método) items() dos dicionários retorna uma lista de tuplas (chave, valor)

```
>>> d = dict()
>>> d['jose'] = 2
>>> d['fred'] = 4
>>> for (c,v) in d.items():
...     print(c, v)
...
jose 2
fred 4
>>> tuplas = list(d.items())
>>> print(tuplas)
[('jose', 2), ('fred', 4)]
```

Tuplas são Comparáveis

- Os operadores relacionais funcionam com tuplas, listas e outras seqüências
- Se o primeiro item é igual, compare-se o próximo elemento, e assim por diante, até achar o Python elementos que diferem.

```
>>> (0, 1, 2) < (5, 1, 2)
True
>>> (0, 1, 2000000) < (0, 3, 4)
True
>>> ('Jose', 'Fred') < ('Jose', 'Maria')
True
>>> ('Jose', 'Fred') > ('Aldo', 'Maria')
True
```