Tuplas

Prof. Alberto Costa Neto

Programação em Python

CC BY

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 = ('Glenn', 'Sally', 'Joseph')
>>> print x[2]
Joseph
>>> 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', 'count', 'extend', 'index', 'insert', 'pop',
'remove', 'reverse', 'sort']

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

Tuplas são mais eficientes

- Como o interpretador de Python não tem que criar uma estrutura modificável para uma tupla, a estrutura usada é mais simples e eficiente em termos de memória de desempenho, quando comparada à de uma 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['csev'] = 2
>>> d['cwen'] = 4
>>> for (c,v) in d.items():
...     print c, v
...
csev 2
cwen 4
>>> tuplas = d.items()
>>> print tuplas
[('csev', 2), ('cwen', 4)]
```

Tuplas são Comparáveis

- Os operadores de comparação funcionam com tuplas e outras seqüências. Se o primeiro item é igual, Python vai para o próximo elemento, e assim por diante, até achar elementos que diferem.

```
>>> (0, 1, 2) < (5, 1, 2)
True
>>> (0, 1, 2000000) < (0, 3, 4)
True
>>> ( 'Jones', 'Sally' ) < ('Jones', 'Sam')
True
>>> ( 'Jones', 'Sally') > ('Adams', 'Sam')
True
```

Ordenando Listas de Tuplas

- Podemos tirar vantagem da habilidade de ordenar uma lista de tuplas para obter uma versão ordenada de um dicionário
- Primeiro, ordenamos o dicionário pela chave usando a função (método) sort()

```
>>> d = {'a':10, 'b':1, 'c':22}
>>> t = d.items()
>>> t
[('a', 10), ('c', 22), ('b', 1)]
>>> t.sort()
>>> t
[('a', 10), ('b', 1), ('c', 22)]
```

Usando sorted()

Podemos fazer isto de uma forma ainda mais direta usando a função built-in sorted que recebe uma seqüência como parâmetro e retorna a seqüência ordenada

```
>>> d = {'a':10, 'b':1, 'c':22}
>>> d.items()
[('a', 10), ('c', 22), ('b', 1)]
>>> t = sorted(d.items())
>>> t
[('a', 10), ('b', 1), ('c', 22)]
>>> for c, v in sorted(d.items()):
...     print c, v
...
a 10
b 1
c 22
```

Ordenando pelos valores ao invés de pela chave

- Se pudéssemos construir uma lista de tuplas da forma (valor, chave), poderíamos ordenar pelo valor
- Fazemos isso com um laço for que cria uma lista de tuplas

```
>>> c = {'a':10, 'b':1, 'c':22}
>>> temp = list()
>>> for c, v in c.items() :
...      temp.append( (v, c) )
...
>>> print temp
[(10, 'a'), (22, 'c'), (1, 'b')]
>>> temp.sort(reverse=True)
>>> print temp
[(22, 'c'), (10, 'a'), (1, 'b')]
```

```
arq = open('palavras.txt')
conts = dict()
for linha in arq:
    palavras = linha.split()
    for palavra in palavras:
        conts[palavra] = conts.get(palavra, 0) + 1

lista = list()
for chave, valor in conts.items():
    lista.append( (valor, chave) )

lista.sort(reverse=True)

for valor, chave in lista[:10] :
    print chave, valor
```

Mostrando as 10 palavras mais comuns em um arquivo, ordenadas pela freqüência

Uma versão ainda mais curta

```
>>> c = {'a':10, 'b':1, 'c':22}

>>> print sorted( [ (v,c) for c,v in c.items() ] )

[(1, 'b'), (10, 'a'), (22, 'c')]
```

Compreensão de Listas cria uma lista dinâmica. Neste caso, criamos uma lista invertida das tuplas dos pares (chave, valor) de um dicionário e então a ordenamos (pelo valor e não pela chave).

http://wiki.python.org/moin/HowTo/Sorting

Acknowledgements / Contributions
Agradecimentos / Contribuições

These slides are Copyright 2010- Charles R. Severance (www.dr-chuck.com) of the University of Michigan School of Information and open.umich.edu and made available under a Creative Commons Attribution 4.0 License. Please maintain this last slide in all copies of the document to comply with the attribution requirements of the license. If you make a change, feel free to add your name and organization to the list of contributors on this page as you republish the materials.

...

Initial Development: Charles Severance, University of Michigan School of Information

These slides were translated and adapted by Alberto Costa Neto (albertocn.sytes.net) of the Federal University of Sergipe