UNIVERSIDAD DE MÁLAGA

FGUMA FUNDACIÓN GENERAL UNIVERSIDAD DE MÁLAGA

CURSOS ONLINE

FUNDACIÓN GENERAL UNIVERSIDAD DE MÁLAGA

Tuplas

Unidad 10

Python para principiantes

dgm@uma.es

Las tuplas son como las listas

Las tuplas son otra clase de **secuencia** que se parece mucho a las listas – tienen elementos que pueden ser **indexados desde 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
>>>
```

Pero... las tuplas son “inmutables”

Al contrario que una lista, una vez crees una tupla, **no puedes alterar su contenido** – como un 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
>>>
```

Cosas que no se pueden hacer con 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'
>>>
```

Una historia de dos secuencias

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

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

Las tuplas son más eficientes

- Dado que Python sabe que las tuplas no tienen que modificarse, **son más simples y más eficientes** en términos de uso de memoria y rendimiento en comparación con las listas
- Así que en nuestros programas cuando hagamos "variables temporales" usaremos mejor tuplas en vez de listas

Tuplas y Asignaciones

- También podemos poner una tupla en la parte izquierda de una sentencia de asignación
- Podemos incluso omitir los paréntesis

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

Tuplas y Diccionarios

El método **ítems()** de los diccionarios retorna una lista de tuplas (clave, valor)

```
>>> d = dict()
>>> d['csev'] = 2
>>> d['cwen'] = 4
>>> for (k,v) in d.items():
...     print(k, v)
...
csev 2
cwen 4
>>> tups = d.items()
>>> print(tups)
dict_items([('csev', 2), ('cwen', 4)])
```

Las tuplas son comparables

Los operadores de comparación funcionan con tuplas y otras secuencias. Si el primer elemento es igual, Python continúa con el próximo hasta que algún elemento es diferente.

```
>>> (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 tomar ventaja de la posibilidad de ordenar una lista de tuplas para obtener una versión ordenada de un diccionario

Primero ordenamos el diccionario por la clave usando el método **items()** y la función **sorted()**

```
>>> d = {'a':10, 'b':1, 'c':22}
>>> d.items()
dict_items([('a', 10), ('c', 22), ('b', 1)])
>>> sorted(d.items())
[('a', 10), ('b', 1), ('c', 22)]
```

Usando sorted()

Podemos hacer esto incluso más directamente si usamos la función **sorted**, que toma una secuencia como parámetro y devuelve una secuencia ordenada

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

Ordenar por valores

- Si pudiéramos construir una lista de tuplas de la forma **(valor, clave)**, podríamos ordenar por el valor
- Podemos hacer esto con un **for** que cree una lista de tuplas

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

Las 10 palabras más comunes

```
fhand = open('romeo.txt')
counts = {}
for line in fhand:
    words = line.split()
    for word in words:
        counts[word] = counts.get(word, 0 ) + 1

lst = []
for key, val in counts.items():
     newtup = (val, key)
    lst.append(newtup)

lst = sorted(lst, reverse=True)

for val, key in lst[:10] :
    print(key, val)
```

Una versión más corta

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

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

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

Las **listas por comprensión** crean listas dinámicas. En este caso, hacemos una lista de tuplas invertidas y las ordenamos

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

Resumen

- Sintaxis de las tuplas
- Inmutabilidad
- Comparaciones
- Ordenación
- Tuplas en sentencias de asignación
- Ordenador de diccionarios por clave o valor

Ejercicio

Escribe un programa que lea el archivo mbox-short.txt y muestre la distribución por horas del día de los mensajes.
Se puede extraer la hora de las líneas que comiencen por 'From ', extrayendo la hora completa y dividiendo una segunda vez con los ':' como delimitador.

From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008

Una vez hayas obtenido el total de mensajes en cada hora, muestra los totales ordenados por hora. Deberías obtener el resultado que se muestra a la derecha

```
04 3
06 1
07 1
09 2
10 3
11 6
14 1
15 2
16 4
17 2
18 1
19 1
```

Acknowledgements / Contributions

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

… Insert new Contributors and Translators here
Spanish Version: Daniel Garrido (dgm@uma.es)

Continue…