

Tuplas

Capítulo 10



Python para Todos
es.py4e.com



Las Tuplas Son Como Listas

Las tuplas son otro tipo de secuencia que funciona de forma parecida a una lista – tienen elementos indexados empezando 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”

A diferencia de una lista, una vez que creas una **tupla**,
no puedes alterar su contenido – de forma similar a una cadena

```
>>> 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 deben 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'
```

```
>>>
```

Un Cuento sobre 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

- Puesto que Python no tiene que construir la estructura de una tupla de modo que sea modificable, las tuplas son más simples y eficientes, en términos de uso de memoria y desempeño, que una lista
- Así que en nuestros programas, cuando creamos “variables temporales”, preferimos tuplas en vez de listas

Tuplas y Asignaciones

- También podemos poner una **tupla** en el **lado izquierdo** de una sentencia de asignación
- Incluso podemos omitir los paréntesis

```
>>> (x, y) = (4, 'fred')
```

```
>>> print(y)
```

```
fred
```

```
>>> (a, b) = (99, 98)
```

```
>>> print(a)
```

```
99
```

Tuplas y Diccionarios

El método `items()` en un diccionario retorna una lista de **tuplas** (clave, valor)

```
>>> d = dict()
>>> d['csev'] = 2
>>> d['cwen'] = 4
>>> for (c,v) in d.items():
...     print(c, 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 revisa el siguiente elemento y así sucesivamente, hasta que encuentra elementos diferentes.

```
>>> (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 aprovechar la habilidad de ordenar una lista de **tuplas** para obtener una versión ordenada de un diccionario
- Primero, ordenamos el diccionario basado en las claves utilizando 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()

Incluso podemos hacer esto de forma más directa usando directamente la función nativa `sorted`, la cual toma una secuencia como parámetro y retorna una secuencia ordenada

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

Ordenamiento por Valores en Lugar de Claves

- Si pudiéramos construir una lista de **tuplas** en la forma **(valor, clave)**, podríamos **ordenar (sort)** por valor
- Hacemos esto con un bucle **for** que crea una lista de tuplas

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

```
man_a = open('romeo.txt')
contadores = dict()
for linea in man_a:
    palabras = linea.split()
    for palabra in palabras:
        contadores[palabra] = contadores.get(palabra, 0 ) + 1

lst = list()
for clave, val in contadores.items():
    nuevatup = (val, clave)
    lst.append(nuevatup)

lst = sorted(lst, reverse=True)

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

El top 10 de las
palabras más
comunes

Una Versión Todavía Más Corta

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

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

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

La comprensión de listas crea una lista dinámica. En este caso, creamos una lista de tuplas invertidas y después las ordenamos.

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

Resumen

- Sintaxis de Tuplas
- Inmutabilidad
- Comparabilidad
- Ordenamiento
- Tuplas en sentencias de asignación
- Ordenamiento de diccionarios por clave o valor



Agradecimientos / Contribuciones



Las diapositivas están bajo el Copyright 2010- Charles R. Severance (www.dr-chuck.com) de la Escuela de Informática de la Universidad de Michigan y open.umich.edu, y están disponibles públicamente bajo una Licencia Creative Commons Attribution 4.0. Favor de mantener esta última diapositiva en todas las copias del documento para cumplir con los requerimientos de atribución de la licencia. Si haces un cambio, siéntete libre de agregar tu nombre y organización a la lista de contribuidores en esta página conforme sean republicados los materiales.

Desarrollo inicial: Charles Severance, Escuela de Informática de la Universidad de Michigan.

Traducción al Español por Juan Carlos Pérez Castellanos - 2020-04-26