Introducción a la Programación con PYTHON

José Rodriguez

Unidad 2 - Datos Compuestos

UNIDAD 2 – COLECCIONES DE DATOS

ÍNDEX

En el capítulo anterior vimos los tipos de datos simples, toca el turno ahora de ver los tipos de datos compuestos por varios datos.

En Python existen tres tipos de datos compuestos llamados genéricamente colecciones: listas, tuplas y colecciones.

1.- Listas.

Una lista es un conjunto de datos ordenado, Es lo que en otros lenguajes de programación se denomina array. Las listas pueden contener cualquier tipo de datos, incluso otras listas y no son, en general, homogéneos en el sentido de que todos los datos deben pertenecer al mismo tipo.

Los elementos de la lista van encerrados entre corchetes y separados por coma.

```
>>> lista=[1,"dos",[1,2]]
>>> print lista
[1, 'dos', [1, 2]]
```

Cada elemento es identificado por un índice comenzando por el cero.

```
>>> print lista[0]
1
>>> print lista[2]
[1, 2]
```

cuando un elemento es a su vez una lista hay que indicar los dos índices.

```
>>> print lista[2][0]
1
```

Obviamente, una lista puede modificarse en tiempo de ejecución

```
>>> lista[2][0]=45
>>> print lista
```

```
[1, 'dos', [45, 2]]
```

Como puedes ver, el mecanismo de asignación y lectura de una lista es similar a la de cualquier lenguaje, sin embargo, Python dispone de dos características adiciones:

- Si indicamos un índice negativo la búsqueda se inicia desde el final. Podemos verlo en el siguiente ejemplo.

```
>>> lista=[0,1,2,3,4]
>>> print lista[0]
0
>>> print lista[-1]
4
```

Python permite seleccionar sublistas de una lista dada indicando para ello los índices de inicio y final separados por el signo ":"

```
>>> lista=[0,1,2,3,4]
>>> sublista=lista[1:3]
>>> print sublista
[1, 2]
```

Si omitimos alguno de los extremos se asume que estos son el principio o final de la lista, por ejemplo:

```
>>> sublista = lista[2:]
>>> print sublista
[2, 3, 4]
>>> sublista= lista[:2]
>>> print sublista
[0, 1]
```

Observa el comportamiento de los extremos: el valor ínicial indica desde donde debemos empezar mientras que el valor final indica hasta donde debemos seleccionar sin incluir este valor.

Este mecanismo de selección se puede utilizar también para asignar valores

```
>>> lista1=[1,1,1,1,1]
>>> lista2=[2,2,2,2,2]
>>> lista1[1:3]=lista2[0:2]
>>> print lista1
[1, 2, 2, 1, 1]
```

En el caso de que omitamos los índices se crea una nueva lista con los valores de la lista copiada

```
>>> lista1[:]=lista2[1:3]
```

```
>>> print lista1
[2, 2]
```

Por último, señalemos que las listas forman un tipo de datos denominado “list”

```
>>> type(lista)
<type 'list'>
```

2.- Tuplas

Las tuplas son conjuntos de datos similares a las listas pero con una particularidad: su constructor es la coma.

```
>>> tupla = 1,2,3
>>> print tupla
(1, 2, 3)
>>> type(tupla)
<type 'tuple'>
```

Como puedes ver, las tuplas se suelen encerrar entre paréntesis pero no olvides que el elemento distintivo es la coma.

```
>>> tupla=(2,3,4)
>>> print tupla
(2, 3, 4)
```

Para referenciar a un elemento de la tupla, al igual que las listas, utilizamos el operador []

```
>>> print tupla[0]
2
```

Entonces, ¿Cuál es la diferencia entre tupla y lista?. Las tuplas no admiten redefiniciones en tiempo de ejecución.

```
>>> tupla[0]=4
Traceback (most recent call last):
  File "<pyshell#39>", line 1, in <module>
    tupla[0]=4
TypeError: 'tuple' object does not support item assignment
```

Ni tampoco soportan redimensionamiento

```
>>> tupla[3]=3
```

Por tanto, podemos utilizar indistintamente ambos tipos con las limitaciones indicadas. Las tuplas son más ligeras y son aconsejables en el caso de conjuntos de datos fijos.

3.- Diccionarios

Los diccionarios son matrices asociativas, es decir, se trata de parejas de valores en los cuales el primer valor es el índice y el segundo el contenido. Cada pareja va separada por una coma. Para representar los diccionarios se utiliza el constructor “{ }”

```
>>> capital={"España" : "Madrid","Francia" : "Paris"}
```

Para acceder a un elemento del diccionario indicamos como índice la primera palabra de la pareja.

```
>>> print capital["España"]
Madrid
```

Son colecciones de datos que pueden permiten modificar su contenido pero sólo de los datos, no el valor de los índices:

```
capital["Francia"]="Roma"
```

Podemos definir un diccionario vacío

```
capital = {"Primero":"" , "segundo": ""}
```

y llenarlo de contenido después

```
capital["primero"]="Madrid"
```

también podemos redimensionar el diccionario incluyendo nuevos términos

```
capital["tercero"]="Roma"
```

Por ultimo, indiquemos que el diccionario es un tipo de datos

```
>>> type(capital)
<type 'dict'>
```