Capítulo 2: Visualizar datos con gráficos

### **Trabajar con listas y tuplas**

Al hacer gráficos con Python, trabajaremos con *listas* y *tuplas*. En Python, se trata de dos formas distintas de almacenar grupos de valores. Las tuplas y las listas son muy similares en su mayor parte, con una diferencia importante: después de crear una lista, es posible añadirle valores y cambiar el orden de los valores. En cambio, los valores de una tupla se fijan inmediatamente y no pueden modificarse. Utilizaremos listas para almacenar *las coordenadas* *x* e *y* de los puntos que queremos trazar. Las tuplas aparecerán en "[Personalizar gráficos](ch02.html#ch02lev2sec05)", en la [página 41](ch02.html#page_41), cuando aprendamos a personalizar el rango de nuestros gráficos. En primer lugar, repasemos algunas características de las listas.

Puedes crear una lista introduciendo valores, separados por comas, entre corchetes. La siguiente sentencia crea una lista y utiliza la etiqueta simplelist para referirse a ella:

>>> simplelist = [1, 2, 3]

Ahora puedes referirte a los números individuales-1, 2 y 3-utilizando la etiqueta y la posición del número en la lista, que se denomina *índice*. Así, simplelist[0] se refiere al primer número, simplelist[1] al segundo y simplelist[2] al tercero:

>>> simplelist[0]  
1  
>>> simplelist[1]  
2  
>>> simplelist[2]  
3

Observa que el primer elemento de la lista está en el índice 0, el segundo en el 1, y así sucesivamente, es decir, las posiciones de la lista empiezan a contar desde 0, no desde 1.

Las listas también pueden almacenar cadenas:

>>> stringlist = ['a string','b string','c string']  
>>> stringlist[0]  
'a string'  
>>> stringlist[1]  
'b string'  
>>> stringlist[2]  
'c string'

Una ventaja de crear una lista es que no tienes que crear una etiqueta distinta para cada valor; basta con crear una etiqueta para la lista y utilizar la posición del índice para referirte a cada elemento. Además, puedes añadir a la lista siempre que necesites almacenar nuevos valores, por lo que una lista es la mejor opción para almacenar datos si no sabes de antemano cuántos números o cadenas puedes necesitar almacenar.

Una lista *vacía* es precisamente eso -una lista sin elementos- y puede crearse así:

>>> emptylist = []

Las listas vacías son útiles sobre todo cuando no conoces de antemano ninguno de los elementos que habrá en tu lista, pero piensas rellenar los valores durante la ejecución de un programa. En ese caso, puedes crear una lista vacía y luego utilizar el método append() para añadir elementos más adelante:

➊ >>> emptylist  
[]  
➋ >>> emptylist.append(1)  
>>> emptylist  
[1]  
➌ >>> emptylist.append(2)  
>>> emptylist  
➍ [1, 2]

En ➊, emptylist comienza vacía. A continuación, añadimos el número 1 a la lista en ➋ y luego añadimos el 2 en ➌. En la línea ➍, la lista es ahora [1, 2]. Ten en cuenta que cuando utilizas .append(), el valor se añade al final de la lista. Ésta es sólo una forma de añadir valores a una lista. Hay otras, pero no las necesitaremos en este capítulo.

Crear una tupla es similar a crear una lista, pero en lugar de corchetes, utilizas paréntesis:

>>> simpletuple = (1, 2, 3)

Puedes referirte a un número individual en simpletuple utilizando el índice correspondiente entre paréntesis, igual que con las listas:

>>> simpletuple[0]  
1  
  
>>> simpletuple[1]  
2  
>>> simpletuple[2]  
3

También puedes utilizar *índices negativos* tanto con listas como con tuplas. Por ejemplo, simplelist[-1] y simpletuple[-1] se referirían al último elemento de la lista o de la tupla, simplelist[-2] y simpletuple[-2] se referirían al penúltimo elemento, y así sucesivamente.

Las tuplas, como las listas, pueden tener cadenas como valores, y puedes crear una *tupla vacía* sin elementos como emptytuple=(). Sin embargo, no existe ningún método append() para añadir un nuevo valor a una tupla existente, por lo que no puedes añadir valores a una tupla vacía. Una vez creada una tupla, su contenido no puede modificarse.

#### ***Iterar sobre una lista o tupla***

Podemos recorrer una lista o tupla utilizando un bucle for como se indica a continuación:

>>> l = [1, 2, 3]  
>>> for item in l:  
print(item)

Esto imprimirá los elementos de la lista:

1  
2  
3

Los elementos de una tupla pueden recuperarse del mismo modo.

A veces puedes necesitar conocer la posición o el índice de un elemento de una lista o tupla. Puedes utilizar la función enumerate() para iterar sobre todos los elementos de una lista y devolver el índice de un elemento, así como el propio elemento. Utilizamos las etiquetas index y item para referirnos a ellas:

>>> l = [1, 2, 3]  
>>> for index, item in enumerate(l):  
print(index, item)

Esto producirá el siguiente resultado:

0 1  
1 2  
2 3

Esto también funciona para tuplas.

[anterior](ch02_2.html)[Subtema 3 de 7: (Ver todo)](ch02.html)[siguiente](ch02_4.html)