Python

Guía para ser un Pythonista

Juan José Lozano Gómez – j2logo.com

8
Tipos colección

Índice de contenidos

Introducción

En el tema 4 vimos los tipos de datos básicos de Python: los numéricos (*int*, *float* y *complex*), el booleano (*bool*) y el tipo cadena de caracteres (*str*).

Sin embargo, a lo largo del curso también te he mencionado otros tipos como *list*, *tuple* o *dict*. Además, hemos visto que existen los tipos *iterables* o *secuenciales*.

En este nuevo bloque, formado por el tema actual y los cuatro siguientes, voy a explicarte en detalle cuáles son todos estos tipos, conocidos como tipos colección o contenedor, dado que se utilizan para agrupar elementos.

Pero te los voy a explicar como nunca te los habían explicado, para que sepas de dónde vienen, las relaciones que existen entre ellos y cuáles son sus características principales.

Prepárate una buena taza de café y disfruta.

Tipos colección

Además de los tipos básicos, en Python existen otros tipos de datos complejos, conocidos como colecciones o contenedores.

Un tipo colección o contenedor permite que un objeto guarde o haga referencia a más de un elemento a la vez. Así, podemos tener una colección de números enteros, una colección de cadenas de caracteres, etc.

En programación, estos tipos son muy útiles. Por ejemplo, se pueden utilizar para guardar las notas de las asignaturas de un alumno, un listado de películas vistas por un usuario o los números de teléfono de un cliente.

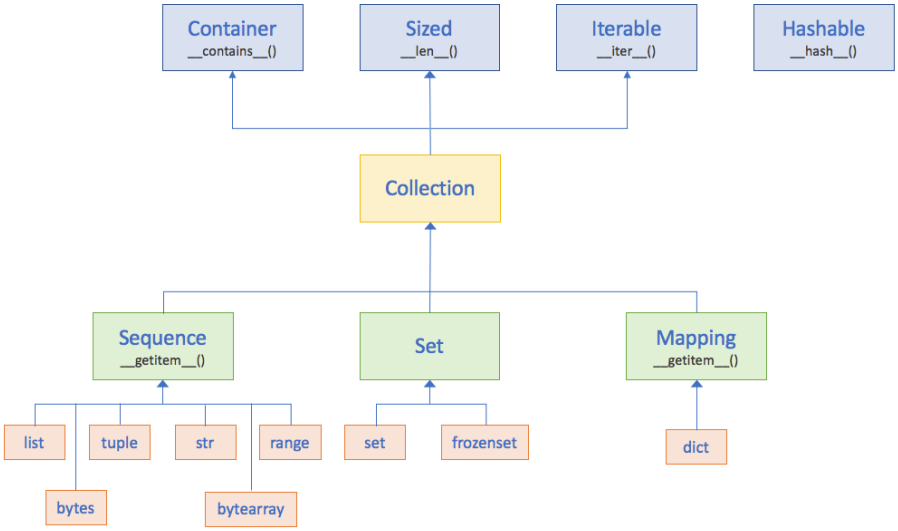
Los tipos colección más habituales de Python se clasifican, principalmente, en ***secuencias***, ***conjuntos*** y ***mapas***.

Pero antes de entrar en detalle con cada uno, te voy a mostrar una simplificación de la jerarquía de clases en las que están basadas estos tipos. Sí, sé que no hemos visto qué es una clase. Si eres nuev@ en esto de la programación y no entiendes a qué me estoy refiriendo, puedes volver aquí después del tema dedicado a ello.

Python define las siguientes clases abstractas (en el módulo `collections.abc`).

Lo que ves en el diagrama es el nombre de la clase y alguno de sus métodos principales (que son abstractos):

Jerarquía de las clases colección



Como puedes observar, los tipos `str`, `list`, `tuple`, `range`, `bytes` y `bytearray` son tipos *secuencia* e *iterables*. Por su parte, los tipos `set` y `frozenset` son *conjuntos iterables*. En cuanto al tipo `dict`, este es un tipo *mapa iterable.*

Además, en Python, los tipos de datos se pueden clasificar como *mutables* e *inmutables*.

En las secciones siguientes veremos cada uno de estos tipos.

Objetos mutables e inmutables

Un objeto es *inmutable* si tiene un valor fijo. Los objetos inmutables incluyen a los de tipo numérico, `str`, `tuple`, `bytes` o `set`. Estos objetos no pueden ser modificados. Se debe crear un nuevo objeto si se quiere almacenar un valor diferente.

Por otro lado, un objeto es *mutable* si puede cambiar su valor. Objetos de tipo `list`, `frozenset` o `dict` son mutables. Una característica de estos objetos es que, aunque se modifiquen, mantienen su `id()`.

Objetos iterables

Un objeto es *iterable* si es capaz de devolver a cada uno de sus miembros de uno en uno. Como hemos visto en el diagrama de arriba, cualquier objeto de un tipo *secuencia*, *set* o *dict* es *iterable*. Pero también lo es un objeto de cualquier clase que implemente el método

`__iter__()` o el método `__getitem__()` con semántica de secuencia.

Los *iterables* se suelen usar en un bucle `for`, aunque también en aquellos lugares donde se necesite una secuencia.

Para iterar sobre los elementos de un *iterable* hace falta un objeto de tipo `iterator`. Este objeto es el que define el mecanismo para recorrer cada uno de los elementos del *iterable*. La forma de crear un objeto `iterator` es pasando el objeto *iterable* como argumento de la función `iter()`.

Generalmente, cuando se trabaja con objetos *iterables* no es necesario llamar a la función `iter()` ni manejar objetos de tipo `iterator`. Por ejemplo, el bucle `for` ya hace esto por ti. Así ocurre también en la mayoría de las ocasiones en que se utilizan.