### Utilización de json.dumps() y json.loads()

Vamos a escribir un breve programa que almacene un conjunto de números y otro programa que vuelva a leer estos números en la memoria. El primer programa utilizará json.dumps() para almacenar el conjunto de números, y el segundo programa utilizará json.loads().

La función json.dumps() toma un argumento: un dato que debe convertirse al formato JSON. La función devuelve una cadena, que podemos escribir en un archivo de datos:

**number\_writer.py**

from pathlib import Path  
import json  
  
numbers = [2, 3, 5, 7, 11, 13]  
  
❶ path = Path('numbers.json')  
❷ contents = json.dumps(numbers)  
path.write\_text(contents)

Primero importamos el módulo json, y luego creamos una lista de números con los que trabajar. A continuación, elegimos un nombre de archivo en el que almacenar la lista de números ❶. Es habitual utilizar la extensión de archivo *.json* para indicar que los datos del archivo se almacenan en formato JSON. A continuación, utilizamos la función json.dumps() ❷ para generar una cadena que contenga la representación JSON de los datos con los que estamos trabajando. Una vez que tenemos esta cadena, la escribimos en el archivo utilizando el mismo método write\_text() que utilizamos antes.

Este programa no tiene salida, pero abramos el archivo *numbers.json* y echémosle un vistazo. Los datos se almacenan en un formato parecido al de Python:

[2, 3, 5, 7, 11, 13]

Ahora escribiremos un programa independiente que utilice json.loads() para volver a leer la lista en la memoria:

**number\_reader.py**

from pathlib import Path  
import json  
  
❶ path = Path('numbers.json')  
❷ contents = path.read\_text()  
❸ numbers = json.loads(contents)  
  
print(numbers)

Nos aseguramos de leer del mismo archivo que escribimos en ❶. Como el archivo de datos es sólo un archivo de texto con un formato específico, podemos leerlo con el método read\_text() ❷. A continuación, pasamos el contenido del archivo a json.loads() ❸. Esta función recibe una cadena con formato JSON y devuelve un objeto Python (en este caso, una lista), que asignamos a numbers. Por último, imprimimos la lista de números recuperada y vemos que es la misma lista creada en *number\_writer.py*:

[2, 3, 5, 7, 11, 13]

Ésta es una forma sencilla de compartir datos entre dos programas.

[anterior](c10_32.html)[Subtema 33 de 37: (Ver todo)](c10.html)[siguiente](c10_34.html)