### Examinar los datos GeoJSON

Cuando abras *eq\_1\_day\_m1.geojson*, verás que es muy denso y difícil de leer:

{"type":"FeatureCollection","metadata":{"generated":1649052296000,...  
{"type":"Feature","properties":{"mag":1.6,"place":"63 km SE of Ped...  
{"type":"Feature","properties":{"mag":2.2,"place":"27 km SSE of Ca...  
{"type":"Feature","properties":{"mag":3.7,"place":"102 km SSE of S...  
{"type":"Feature","properties":{"mag":2.92000008,"place":"49 km SE...  
{"type":"Feature","properties":{"mag":1.4,"place":"44 km NE of Sus...  
--snip--

Este archivo está formateado más para máquinas que para humanos. Pero podemos ver que el archivo contiene algunos diccionarios, así como información que nos interesa, como las magnitudes y ubicaciones de los terremotos.

El módulo json proporciona diversas herramientas para explorar y trabajar con datos JSON. Algunas de estas herramientas nos ayudarán a reformatear el archivo para que podamos ver los datos en bruto más fácilmente antes de trabajar con ellos mediante programación.

Empecemos por cargar los datos y mostrarlos en un formato más fácil de leer. Se trata de un archivo de datos largo, así que en lugar de imprimirlo, reescribiremos los datos en un nuevo archivo. Así podremos abrir ese archivo y desplazarnos hacia adelante y hacia atrás por los datos más fácilmente:

**eq\_explore\_data.py**

from pathlib import Path  
import json  
  
# Read data as a string and convert to a Python object.  
path = Path('eq\_data/eq\_data\_1\_day\_m1.geojson')  
contents = path.read\_text()  
❶ all\_eq\_data = json.loads(contents)  
  
# Create a more readable version of the data file.  
❷ path = Path('eq\_data/readable\_eq\_data.geojson')  
❸ readable\_contents = json.dumps(all\_eq\_data, indent=4)  
path.write\_text(readable\_contents)

Leemos el archivo de datos como una cadena, y utilizamos json.loads() para convertir la representación de cadena del archivo en un objeto ❶ de Python . Este es el mismo enfoque que utilizamos en el Capítulo 10. En este caso, todo el conjunto de datos se convierte en un único diccionario, que asignamos a all\_eq\_data. A continuación, definimos un nuevo path en el que podemos escribir estos mismos datos en un formato ❷ más legible . La función json.dumps() que viste en el Capítulo 10 puede tomar un argumento opcional indent ❸, que le indica cuánto sangrar los elementos anidados en la estructura de datos.

Cuando busques en tu directorio *eq\_data* y abras el archivo *readable\_eq\_data.json*, esto es lo primero que verás:

**readable\_eq\_data.json**

{  
 "type": "FeatureCollection",  
❶ "metadata": {  
 "generated": 1649052296000,  
 "url": "https://earthquake.usgs.gov/earthquakes/.../1.0\_day.geojson",  
 "title": "USGS Magnitude 1.0+ Earthquakes, Past Day",  
 "status": 200,  
 "api": "1.10.3",  
 "count": 160  
 },  
❷ "features": [  
 --snip--

La primera parte del archivo incluye una sección con la clave "metadata"❶. Esto nos indica cuándo se generó el archivo de datos y dónde podemos encontrar los datos en Internet. También nos da un título legible por humanos y el número de terremotos incluidos en este archivo. En este periodo de 24 horas, se registraron 160 terremotos.

Este archivo GeoJSON tiene una estructura que resulta útil para los datos basados en la ubicación. La información se almacena en una lista asociada a la clave "features" ❷. Como este archivo contiene datos sobre terremotos, los datos están en forma de lista, donde cada elemento de la lista corresponde a un único terremoto. Esta estructura puede parecer confusa, pero es bastante potente. Permite a los geólogos almacenar toda la información que necesiten en un diccionario sobre cada terremoto, y luego meter todos esos diccionarios en una gran lista.

Veamos un diccionario que representa un único terremoto:

**readable\_eq\_data.json**

--snip--  
 {  
 "type": "Feature",  
❶ "properties": {  
 "mag": 1.6,  
 --snip--  
❷ "title": "M 1.6 - 27 km NNW of Susitna, Alaska"  
 },  
❸ "geometry": {  
 "type": "Point",  
 "coordinates": [  
❹ -150.7585,  
❺ 61.7591,  
 56.3  
 ]  
 },  
 "id": "ak0224bju1jx"  
 },

La clave "properties" contiene mucha información sobre cada terremoto ❶. Nos interesa sobre todo la magnitud de cada terremoto, asociada a la clave "mag". También nos interesa el "title" de cada suceso, que proporciona un buen resumen de su magnitud y localización ❷.

La clave "geometry" nos ayuda a comprender dónde se produjo el terremoto ❸. Necesitaremos esta información para cartografiar cada suceso. Podemos encontrar la longitud ❹ y la latitud ❺ de cada terremoto en una lista asociada a la clave "coordinates".

Este archivo contiene mucho más anidamiento del que utilizaríamos en el código que escribimos, así que si te parece confuso, no te preocupes: Python se encargará de la mayor parte de la complejidad. Sólo trabajaremos con uno o dos niveles de anidamiento cada vez. Empezaremos sacando un diccionario para cada terremoto que se haya registrado en el periodo de 24 horas.

[anterior](c16_17.html)[Subtema 18 de 29: (Ver todo)](c16.html)[siguiente](c16_19.html)