## La API de Hacker News

Para explorar cómo utilizar las llamadas a la API en otros sitios, echemos un vistazo rápido a Hacker News[(https://news.ycombinator.com)](https://news.ycombinator.com). En Hacker News, la gente comparte artículos sobre programación y tecnología y participa en animados debates sobre esos artículos. La API de Hacker News proporciona acceso a datos sobre todos los envíos y comentarios del sitio, y puedes utilizar la API sin tener que registrarte para obtener una clave.

La siguiente llamada devuelve información sobre el artículo principal actual en el momento de escribir este artículo:

https://hacker-news.firebaseio.com/v0/item/31353677.json

Cuando introduzcas esta URL en un navegador, verás que el texto de la página está encerrado entre llaves, lo que significa que es un diccionario. Pero la respuesta es difícil de examinar sin un formato mejor. Pasemos esta URL por el método json.dumps(), como hicimos en el proyecto terremoto del Capítulo 16, para poder explorar el tipo de información que se devuelve sobre un artículo:

**hn\_article.py**

import requests  
import json  
  
# Make an API call, and store the response.  
url = "https://hacker-news.firebaseio.com/v0/item/31353677.json"  
r = requests.get(url)  
print(f"Status code: {r.status\_code}")  
  
# Explore the structure of the data.  
response\_dict = r.json()  
response\_string = json.dumps(response\_dict, indent=4)  
❶ print(response\_string)

Todo en este programa debería resultarte familiar, porque lo hemos utilizado todo en los dos capítulos anteriores. La principal diferencia aquí es que podemos imprimir la cadena de respuesta formateada ❶ en lugar de escribirla en un archivo, porque la salida no es especialmente larga.

La salida es un diccionario de información sobre el artículo con el ID 31353677:

{  
 "by": "sohkamyung",  
❶ "descendants": 302,  
 "id": 31353677,  
❷ "kids": [  
 31354987,  
 31354235,  
 --snip--  
 ],  
 "score": 785,  
 "time": 1652361401,  
❸ "title": "Astronomers reveal first image of the black hole  
 at the heart of our galaxy",  
 "type": "story",  
❹ "url": "https://public.nrao.edu/news/.../"  
}

El diccionario contiene una serie de claves con las que podemos trabajar. La clave "descendants" nos indica el número de comentarios que ha recibido el artículo ❶. La clave "kids" proporciona los ID de todos los comentarios realizados directamente en respuesta a este envío ❷. Cada uno de estos comentarios puede tener también comentarios propios, por lo que el número de descendientes que tiene un envío suele ser mayor que su número de hijos. Podemos ver el título del artículo que se está discutiendo ❸ y también una URL del artículo que se está discutiendo ❹.

La siguiente URL devuelve una lista simple de todos los ID de los principales artículos actuales de Hacker News:

https://hacker-news.firebaseio.com/v0/topstories.json

Podemos utilizar esta llamada para averiguar qué artículos están en la página principal en este momento, y luego generar una serie de llamadas a la API similares a la que acabamos de examinar. Con este enfoque, podemos imprimir un resumen de todos los artículos que están en la portada de Hacker News en este momento:

**hn\_submissions.py**

from operator import itemgetter  
  
import requests  
  
# Make an API call and check the response.  
❶ url = "https://hacker-news.firebaseio.com/v0/topstories.json"  
r = requests.get(url)  
print(f"Status code: {r.status\_code}")  
  
# Process information about each submission.  
❷ submission\_ids = r.json()  
❸ submission\_dicts = []  
for submission\_id in submission\_ids[:5]:  
 # Make a new API call for each submission.  
❹ url = f"https://hacker-news.firebaseio.com/v0/item/{submission\_id}.json"  
 r = requests.get(url)  
 print(f"id: {submission\_id}\tstatus: {r.status\_code}")  
 response\_dict = r.json()  
  
 # Build a dictionary for each article.  
❺ submission\_dict = {  
 'title': response\_dict['title'],  
 'hn\_link': f"https://news.ycombinator.com/item?id={submission\_id}",  
 'comments': response\_dict['descendants'],  
 }  
❻ submission\_dicts.append(submission\_dict)  
  
❼ submission\_dicts = sorted(submission\_dicts, key=itemgetter('comments'),  
 reverse=True)  
  
❽ for submission\_dict in submission\_dicts:  
 print(f"\nTitle: {submission\_dict['title']}")  
 print(f"Discussion link: {submission\_dict['hn\_link']}")  
 print(f"Comments: {submission\_dict['comments']}")

Primero, hacemos una llamada a la API e imprimimos el estado de la respuesta ❶. Esta llamada a la API devuelve una lista que contiene los ID de hasta 500 de los artículos más populares de Hacker News en el momento en que se realiza la llamada. A continuación, convertimos el objeto respuesta en una lista Python ❷, que asignamos a submission\_ids. Utilizaremos estos ID para construir un conjunto de diccionarios, cada uno de los cuales contiene información sobre uno de los envíos actuales.

Creamos una lista vacía llamada submission\_dicts para almacenar estos diccionarios ❸. A continuación, recorremos en bucle los ID de las 30 mejores presentaciones. Hacemos una nueva llamada a la API para cada envío generando una URL que incluya el valor actual de submission\_id ❹. Imprimimos el estado de cada solicitud junto con su ID, para saber si se ha realizado correctamente.

A continuación, creamos un diccionario para el envío que se está procesando actualmente ❺. Almacenamos el título del envío, un enlace a la página de discusión de ese artículo y el número de comentarios que ha recibido hasta el momento. Después añadimos cada submission\_dict a la lista submission\_dicts ❻.

Cada envío en Hacker News se clasifica según una puntuación general basada en una serie de factores, entre ellos cuántas veces ha sido votado, cuántos comentarios ha recibido y lo reciente que es el envío. Queremos ordenar la lista de diccionarios por el número de comentarios. Para ello, utilizamos una función llamada itemgetter() ❼, que procede del módulo operator. Le pasamos a esta función la clave 'comments', y extrae el valor asociado a esa clave de cada diccionario de la lista. A continuación, la función sorted() utiliza este valor como base para ordenar la lista. Ordenamos la lista en orden inverso, para colocar primero las historias más comentadas.

Una vez ordenada la lista, la recorremos en bucle ❽ e imprimimos tres datos sobre cada uno de los envíos principales: el título, un enlace a la página de discusión y el número de comentarios que tiene actualmente el envío:

Status code: 200  
id: 31390506 status: 200  
id: 31389893 status: 200  
id: 31390742 status: 200  
--snip--  
  
Title: Fly.io: The reclaimer of Heroku's magic  
Discussion link: https://news.ycombinator.com/item?id=31390506  
Comments: 134  
  
Title: The weird Hewlett Packard FreeDOS option  
Discussion link: https://news.ycombinator.com/item?id=31389893  
Comments: 64  
  
Title: Modern JavaScript Tutorial  
Discussion link: https://news.ycombinator.com/item?id=31390742  
Comments: 20  
--snip--

Utilizarías un proceso similar para acceder a la información y analizarla con cualquier API. Con estos datos, podrías hacer una visualización que mostrara qué envíos han inspirado los debates recientes más activos. Esta es también la base de las aplicaciones que proporcionan una experiencia de lectura personalizada para sitios como Hacker News. Para saber más sobre a qué tipo de información puedes acceder a través de la API de Hacker News, visita la página de documentación en [https://github.com/HackerNews/API.](https://github.com/HackerNews/API)

[anterior](c17_16.html)[Subtema 17 de 20: (Ver todo)](c17.html)[siguiente](c17_18.html)