

# Hoja de trucos: API y recopilación de datos

Paquete/Método	Descripción	Ejemplo de código
Accediendo al atributo del elemento	Accede al valor de un atributo específico de un elemento HTML.	Sintaxis: <pre>atributo = elemento[(atributo)]</pre> Ejemplo: <pre>href = link_element[(href)]</pre>
BeautifulSoup()	Analiza el contenido HTML de una página web utilizando BeautifulSoup. El tipo de analizador puede variar según el proyecto.	Sintaxis: <pre>soup = BeautifulSoup(html, (html.parser))</pre> Ejemplo: <pre>html = (https://api.example.com/data) soup = BeautifulSoup(html, (html.parser))</pre>
delete()	Envía una solicitud DELETE para eliminar datos o un recurso del servidor. Las solicitudes DELETE eliminan un recurso específico en el servidor.	Sintaxis: <pre>response = requests.delete(url)</pre> Ejemplo: <pre>response = requests.delete((https://api.example.com/delete))</pre>

find()	Encuentra el primer elemento HTML que coincide con la etiqueta y atributos especificados.	<p>Sintaxis:</p> <pre>element = soup.find(tag, attrs)</pre> <p>Ejemplo:</p> <pre>first_link = soup.find((a), {(class): (link)})</pre>
find_all()	Encuentra todos los elementos HTML que coinciden con la etiqueta y atributos especificados.	<p>Sintaxis:</p> <pre>elements = soup.find_all(tag, attrs)</pre> <p>Ejemplo:</p> <pre>all_links = soup.find_all((a), {(class): (link)})&lt;/td&gt;</pre>
findChildren()	Encuentra todos los elementos hijos de un elemento HTML.	<p>Sintaxis:</p> <pre>children = element.findChildren()</pre> <p>Ejemplo:</p> <pre>child_elements = parent_div.findChildren()</pre>
get()	Realiza una solicitud GET para recuperar datos de una URL	<p>Sintaxis:</p> <pre>response = requests.get(url)</pre>

	<p>especificada. Las solicitudes GET se utilizan típicamente para leer datos de una API. La variable de respuesta contendrá la respuesta del servidor, que puedes procesar más adelante.</p>	<p>Ejemplo:</p> <pre>response = requests.get((https://api.example.com/data))</pre>
Headers	<p>Incluye encabezados personalizados en la solicitud. Los encabezados pueden proporcionar información adicional al servidor, como tokens de autenticación o tipos de contenido.</p>	<p>Sintaxis:</p> <pre>headers = {(HeaderName): (Value)}</pre> <p>Ejemplo:</p> <pre>base_url = (https://api.example.com/data) headers = {(Authorization): (Bearer YOUR_TOKEN)} response = requests.ge</pre>
Importar bibliotecas	<p>Importa las bibliotecas de Python necesarias para el web scraping.</p>	<p>Sintaxis:</p> <pre>from bs4 import BeautifulSoup</pre>
json()	<p>Analiza datos JSON de la respuesta. Esto extrae y trabaja con los datos devueltos por la API. El método <code>response.json()</code> convierte la respuesta JSON en una estructura de datos de Python (generalmente un diccionario o lista).</p>	<p>Sintaxis:</p> <pre>data = response.json()</pre> <p>Ejemplo:</p> <pre>response = requests.get((https://api.example.com/data)) data = response.json()</pre>

next_sibling()	Encuentra el siguiente elemento hermano en el DOM.	<p>Sintaxis:</p> <pre>sibling = element.find_next_sibling()</pre> <p>Ejemplo:</p> <pre>next_sibling = current_element.find_next_sibling()</pre>
parent	Accede al elemento padre en el Modelo de Objetos del Documento (DOM).	<p>Sintaxis:</p> <pre>parent = element.parent</pre> <p>Ejemplo:</p> <pre>parent_div = paragraph.parent</pre>
post()	Envía una solicitud POST a una URL especificada con datos. Crea o actualiza solicitudes POST utilizando recursos en el servidor. El parámetro de datos contiene los datos que se enviarán al servidor, a menudo en formato JSON.	<p>Sintaxis:</p> <pre>response = requests.post(url, data)</pre> <p>Ejemplo:</p> <pre>response = requests.post((https://api.example.com/submit), data={{key): (value}})</pre>
put()	Envía una solicitud PUT para actualizar	<p>Sintaxis:</p> <pre>response = requests.put(url, data)</pre>

	datos en el servidor. Las solicitudes PUT se utilizan para actualizar un recurso existente en el servidor con los datos proporcionados en el parámetro de datos, típicamente en formato JSON.	<p>Ejemplo:</p> <pre>response = requests.put((https://api.example.com/update), data={(key): (value)})</pre>
Parámetros de consulta	Envía parámetros de consulta en la URL para filtrar o personalizar la solicitud. Los parámetros de consulta especifican condiciones o límites para los datos solicitados.	<p>Sintaxis:</p> <pre>params = {(param_name): (value)}</pre> <p>Ejemplo:</p> <pre>base_url = "https://api.example.com/data" params = {"page": 1, "per_page": 10} response = requests.get(base_url, params=params)</pre>
select()	Selecciona elementos HTML del HTML analizado utilizando un selector CSS.	<p>Sintaxis:</p> <pre>element = soup.select(selector)</pre> <p>Ejemplo:</p> <pre>titles = soup.select((h1))</pre>
status_code	Verifica el código de estado HTTP de la respuesta. El código de estado HTTP indica el	<p>Sintaxis:</p> <pre>response.status_code</pre>

	<p>resultado de la solicitud (éxito, error, redirección). El código de estado HTTP se puede usar para el manejo de errores y la toma de decisiones en tu código.</p>	<p>Ejemplo:</p> <pre>url = "https://api.example.com/data" response = requests.get(url) status_code = response.status_code</pre>
etiquetas para find() y find_all()	<p>Especifica cualquier etiqueta HTML válida como parámetro de etiqueta para buscar elementos de ese tipo. Aquí hay algunas etiquetas HTML comunes que puedes usar con el parámetro de etiqueta.</p>	<p>Ejemplo de etiqueta:</p> <ul style="list-style-type: none"> <li>- (a): Encontrar etiquetas de ancla ().</li> <li>- (p): Encontrar etiquetas de párrafo ((p)).</li> <li>- (h1), (h2), (h3), (h4), (h5), (h6): Encontrar etiquetas de encabezado del nivel 1 al 6 ( (h1),n (h2)).</li> <li>- (table): Encontrar etiquetas de tabla ().</li> <li>- (tr): Encontrar etiquetas de fila de tabla ().</li> <li>- (td): Encontrar etiquetas de celda de tabla ((td)).</li> <li>- (th): Encontrar etiquetas de celda de encabezado de tabla ((td)).</li> <li>- (img): Encontrar etiquetas de imagen ((img)).</li> <li>- (form): Encontrar etiquetas de formulario ((form)).</li> <li>- (button): Encontrar etiquetas de botón ((button)).</li> </ul>
texto	<p>Recupera el contenido de texto de un elemento HTML.</p>	<p>Sintaxis:</p> <pre>text = element.text</pre> <p>Ejemplo:</p> <pre>title_text = title_element.text</pre>



# Skills Network

© IBM Corporation. Todos los derechos reservados.