

Cheat Sheet: API's and Data Collection

| Package/Method | Description | Code Example |
|-----------------------------|--|--------------|
| Accessing element attribute | Syntax: <pre>1. 1</pre> | |
| | Access the value of a specific attribute of an HTML element. <pre>1. attribute = element[(attribute)]</pre> <div>Copied!</div> | |
| BeautifulSoup() | Example: <pre>1. 1</pre> <pre>1. href = link_element[(href)]</pre> <div>Copied!</div> | |
| | Syntax: <pre>1. 1</pre> <pre>1. soup = BeautifulSoup(html, (html.parser))</pre> <div>Copied!</div> | |
| delete() | BeautifulSoup. The parser type can vary based on the project. <pre>1. 1</pre> <pre>1. html = (https://api.example.com/data) soup = BeautifulSoup(html, (html.parser))</pre> <div>Copied!</div> | |
| | Send a DELETE request to remove data or a resource from the server. DELETE requests delete a specified resource on the server. Syntax: <pre>1. 1</pre> <pre>1. response = requests.delete(url)</pre> <div>Copied!</div> | |
| find() | Example: <pre>1. 1</pre> <pre>1. response = requests.delete((https://api.example.com/delete))</pre> <div>Copied!</div> | |
| | Syntax: <pre>1. 1</pre> <pre>1. element = soup.find(tag, attrs)</pre> <div>Copied!</div> | |
| find_all() | Example: <pre>1. 1</pre> <pre>1. first_link = soup.find((a), {(class): (link)})</pre> <div>Copied!</div> | |
| | Syntax: <pre>1. 1</pre> <pre>1. elements = soup.find_all(tag, attrs)</pre> <div>Copied!</div> | |
| findChildren() | Example: <pre>1. 1</pre> <pre>1. all_links = soup.find_all((a), {(class): (link)})</td></pre> <div>Copied!</div> | |
| | Syntax: <pre>1. 1</pre> <pre>1. children = element.findChildren()</pre> <div>Copied!</div> | |

| | | |
|------------------|--|---|
| | | <div>Example:<pre>1. 1 1. child_elements = parent_div.findChildren()</pre><div>Copied!</div></div> |
| get() | <div>Perform a GET request to retrieve data from a specified URL. GET requests are typically used for reading data from an API. The response variable will contain the server's response, which you can process further.</div> | <div><div>Syntax:<pre>1. 1 1. response = requests.get(url)</pre><div>Copied!</div></div><div>Example:<pre>1. 1 1. response = requests.get((https://api.example.com/data))</pre><div>Copied!</div></div></div> |
| Headers | <div>Include custom headers in the request. Headers can provide additional information to the server, such as authentication tokens or content types.</div> | <div><div>Syntax:<pre>1. 1 1. headers = {(HeaderName): (Value)}</pre><div>Copied!</div></div><div>Example:<pre>1. 1 1. base_url = (https://api.example.com/data) headers = {(Authorization): (Bearer YOUR_TOKEN)} response = requ</pre><div>Copied!</div></div></div> |
| Import Libraries | <div>Import the necessary Python libraries for web scraping.</div> | <div><div>Syntax:<pre>1. 1 1. from bs4 import BeautifulSoup</pre><div>Copied!</div></div></div> |
| json() | <div>Parse JSON data from the response. This extracts and works with the data returned by the API. The response.json() method converts the JSON response into a Python data structure (usually a dictionary or list).</div> | <div><div>Syntax:<pre>1. 1 1. data = response.json()</pre><div>Copied!</div></div><div>Example:<pre>1. 1 2. 2 1. response = requests.get((https://api.example.com/data)) 2. data = response.json()</pre><div>Copied!</div></div></div> |
| next_sibling() | <div>Find the next sibling element in the DOM.</div> | <div><div>Syntax:<pre>1. 1 1. sibling = element.find_next_sibling()</pre><div>Copied!</div></div><div>Example:<pre>1. 1 1. next_sibling = current_element.find_next_sibling()</pre></div></div> |

| | |
|------------------|---|
| | <div>Copied!</div> <div>Syntax:</div> <div>1. 1</div> |
| parent | <div>Access the parent element in the Document Object Model (DOM).</div> <div>1. parent = element.parent</div> <div><div>Copied!</div></div> <div>Example:</div> <div>1. 1</div> <div>1. parent_div = paragraph.parent</div> |
| post() | <div>Send a POST request to a specified URL with data. Create or update POST requests using resources on the server. The data parameter contains the data to send to the server, often in JSON format.</div> <div><div>Copied!</div></div> <div>Syntax:</div> <div>1. 1</div> <div>1. response = requests.post(url, data)</div> <div><div>Copied!</div></div> <div>Example:</div> <div>1. 1</div> <div>1. response = requests.post((https://api.example.com/submit), data={{key): (value}})</div> <div><div>Copied!</div></div> |
| put() | <div>Send a PUT request to update data on the server. PUT requests are used to update an existing resource on the server with the data provided in the data parameter, typically in JSON format.</div> <div><div>Copied!</div></div> <div>Syntax:</div> <div>1. 1</div> <div>1. response = requests.put(url, data)</div> <div><div>Copied!</div></div> <div>Example:</div> <div>1. 1</div> <div>1. response = requests.put((https://api.example.com/update), data={{key): (value}})</div> <div><div>Copied!</div></div> |
| Query parameters | <div>Syntax:</div> <div>1. 1</div> <div>Pass query parameters in the URL to filter or customize the request. Query parameters specify conditions or limits for the requested data.</div> <div>1. params = {(param_name): (value)}</div> <div><div>Copied!</div></div> <div>Example:</div> <div>1. 1</div> <div>2. 2</div> <div>3. 3</div> <div>1. base_url = "https://api.example.com/data"</div> <div>2. params = {"page": 1, "per_page": 10}</div> <div>3. response = requests.get(base_url, params=params)</div> |
| select() | <div><div>Copied!</div></div> <div>Syntax:</div> <div>1. 1</div> <div>1. element = soup.select(selector)</div> <div><div>Copied!</div></div> <div>Example:</div> <div>1. 1</div> <div>1. titles = soup.select((h1))</div> <div><div>Copied!</div></div> |

status_code

Check the HTTP status code of the response. The HTTP status code indicates the result of the request (success, error, redirection). Use the HTTP status codeIt can be used for error handling and decision-making in your code.

Syntax:

```
1. 1
1. response.status_code
```

Copied!

Example:

```
1. 1
2. 2
3. 3
1. url = "https://api.example.com/data"
2. response = requests.get(url)
3. status_code = response.status_code
```

Copied!

Tag Example:

```
1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8
9. 9
10. 10
```

Specify any valid HTML tag as the tag parameter to search for elements of that type. Here are some common HTML tags that you can use with the tag parameter.

```
1. - (a): Find anchor () tags.
2. - (p): Find paragraph ((p)) tags.
3. - (h1), (h2), (h3), (h4), (h5), (h6): Find heading tags from level 1 to 6 ( (h1),n (h2)).
4. - (table): Find table () tags.
5. - (tr): Find table row () tags.
6. - (td): Find table cell ((td)) tags.
7. - (th): Find table header cell ((td))tags.
8. - (img): Find image ((img)) tags.
9. - (form): Find form ((form)) tags.
10. - (button): Find button ((button)) tags.
```

Copied!

tags for find() and find_all()

Syntax:

```
1. 1
1. text = element.text
```

Retrieve the text content of an HTML element.

Example:

```
1. 1
1. title_text = title_element.text
```

Copied!

text



Skills Network

© IBM Corporation. All rights reserved.