Module 5 Cheatsheet : API's and Data Collection

Package/Method	Description	Code Example

Accessing Element Attribute	Access the value of a specific attribute of an HTML element.	Syntax:
		`attribute = element[(attribute)]`
		Copied!
		Example:
		`href = link_element[(href)]`
		Copied!

BeautifulSoup()	Parse the HTML content of a web page using BeautifulSoup. The parser type can vary based on the project.	Syntax:
		`soup = BeautifulSoup(html, (html.parser))`
		Copied!
		Example:
		`html = (https://api.example.com/data) soup = BeautifulSoup(html, (html.parser))`
		Copied!

delete()	Send a DELETE request to remove data or a resource from the server. DELETE requests delete a specified resource on the server.	Syntax:
		`response = requests.delete(url)`
		Copied!
		Example:
		`response = requests.delete((https://api.example.com/delete))`
		Copied!

find()	Find the first HTML element that matches the specified tag and attributes.	Syntax:
		`element = soup.find(tag, attrs)`
		Copied!
		Example:
		`first_link = soup.find((a), {(class): (link)})`
		Copied!

find_all()	Find all HTML elements that match the specified tag and attributes.	Syntax:
		`elements = soup.find_all(tag, attrs)`
		Copied!
		Example:
		`all_links = soup.find_all((a), {(class): (link)})</td>`
		Copied!

findChildren()	Find all child elements of an HTML element.	Syntax:
		`children = element.findChildren()`
		Copied!
		Example:
		`child_elements = parent_div.findChildren()`
		Copied!

get()	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.	Syntax:
		`response = requests.get(url)`
		Copied!
		Example:
		`response = requests.get((https://api.example.com/data))`
		Copied!

Headers	Include custom headers in the request. Headers can provide additional information to the server, such as authentication tokens or content types.	Syntax:
		`headers = {(HeaderName): (Value)}`
		Copied!
		Example:
		`base_url = (https://api.example.com/data) headers = {(Authorization): (Bearer YOUR_TOKEN)} response = requests.get(base_url, headers=headers)`
		Copied!

Import Libraries	Import the necessary Python libraries for web scraping.	Syntax: `from bs4 import BeautifulSoup` Copied!
json()	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).	Syntax: data = response.json() Example: `response = requests.get((https://api.example.com/data)) data = response.json()` Copied!

next_sibling()	Find the next sibling element in the DOM.	Syntax:
		`sibling = element.find_next_sibling()`
		Copied!
		Example:
		`next_sibling = current_element.find_next_sibling()`
		Copied!

parent	Access the parent element in the Document Object Model (DOM).	Syntax:
		`parent = element.parent`
		Copied!
		Example:
		`parent_div = paragraph.parent`
		Copied!

post()	Send a POST request to a specified URL with data. POST requests are used for creating or updating resources on the server. The data parameter contains the data to send to the server, often in JSON format.	Syntax: response = requests.post(url, data) Example: `response = requests.post((https://api.example.com/submit), data={(key): (value)})` Copied!

put()	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.	Syntax:
		`response = requests.put(url, data)`
		Copied!
		Example:
		`response = requests.put((https://api.example.com/update), data={(key): (value)})`
		Copied!

Query Parameters	Pass query parameters in the URL to filter or customize the request. Query parameters specify conditions or limits for the requested data.	Syntax:
		`params = {(param_name): (value)}`
		Copied!
		Example:
		`base_url = "https://api.example.com/data"`
		`params = {"page": 1, "per_page": 10}`
		`response = requests.get(base_url, params=params)`
		Copied!

select()	Select HTML elements from the parsed HTML using a CSS selector.	Syntax:
		`element = soup.select(selector)`
		Copied!
		Example:
		`titles = soup.select((h1))`
		Copied!

status_code	Check the HTTP status code of the response. The HTTP status code indicates the result of the request (success, error, redirection). It can be used for error handling and decision-making in your code.	Syntax:
		`response.status_code`
		Copied!
		Example:
		`url = "https://api.example.com/data"`
		`response = requests.get(url)`
		`status_code = response.status_code`
		Copied!

tags for find() and find_all()	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.	Tag Example:
		`- (a): Find anchor () tags.`
		`- (p): Find paragraph ((p)) tags.`
		`- (h1), (h2), (h3), (h4), (h5), (h6): Find heading tags from level 1 to 6 ( (h1),n (h2)).`
		`- (table): Find table () tags.`
		`- (tr): Find table row () tags.`
		`- (td): Find table cell ((td)) tags.`
		`- (th): Find table header cell ((td))tags.`
		`- (img): Find image ((img)) tags.`
		`- (form): Find form ((form)) tags.`
		`- (button): Find button ((button)) tags.`
		Copied!

text	Retrieve the text content of an HTML element.	Syntax:
		`text = element.text`
		Copied!
		Example:
		`title_text = title_element.text`
		Copied!

Skills Network