Websites

Hidden websites: Websites hiding information in which need authenticate to view
Static websites: Websites where HTML code doesn't change unless there is interaction
Dynamic websites: Website HTML code is changing, and thus a requests.get(...) call may not work. Perhaps a requests.get(...) call could return **JavaScript that executes said HTML page contents**.

requests

What is requests?: A PIP package that allows for the obtaining of a website's HTML information, such as the HTML code of the website.

request's main command: **get(...)**

- Sends an HTTP GET request to the website to acquire it's information
- Stores the GET request into a Python object
    - obj**.text** ~ Returns the website's HTML code
        - Example: print(src.text)

1. get(url)
2. get(url, auth("username", "password"))
    a. For websites that need authentication, requests has other forms of authentication handling

requests-html

Allows for the execution of javascript code for dealing with **dynamic websites**

Beautiful Soup

What is Beautiful Soup?: A PIP package that allows for the parsing of structured data.

Parsing data with Beautiful Soup

1. obj = BeautifulSoup(...)
    a. Creates a BeautifulSoup object (a string of the raw HTML code)
    b. Function variations
        i. BeautifulSoup(requestsResult.content)
            1. requestsResult: A requests.get(...) call
            2. Defaults to the HTML parser
        ii. BeautifulSoup(requestsResult.content, "html.parser")
            1. requestsResult: A requests.get(...) call
            2. Choose your parser
2. Find & scrape ~ find(...) & find_all(...)
    a. BeautifulSoulObj.find(...)
        i. Returns a string of the first instance of HTML code queried

ii. Function variations
    1. find(id)
        a. Finds a block of code of the first instance of the ID specified
    2. find(html_tag, class_ = "")
        a. Finds a block of code of the first instance of the html tag of the class specified

b. BeautifulSoulObj.find_all(...)
    i. Returns an array of every instance of the HTML code queried
    ii. Function variations
        1. find_all(id)
        2. find(html_tag, class_ = "")

Misc Functions

- **.prettify()** Making HTML parse results legible

  ```
  obj = BeautifulSoup(requestsResult.content, "html.parser")
  obj.find(id="Something")
  print(obj.prettify())
  ```

- **.text** Returns only the text of a given block of HTML code

  ```
  BeautifulSoupObj = "<a> Text </a>"
  print(BeautifulSoupObj.text) // Text
  ```

    - Usually use this in tandem with strip(), which removes whitespaces in Python