HACKWAGON
• ACADEMY •

DATA SCIENCE 102:
HTML & WEBSCRAPING

AGENDA

- What is HTML?
- HTML Elements
- Tables
- Web Scraping
  - Requests
  - BeautifulSoup

WHAT IS HTML

- HTML stands for **Hyper Text Markup Language**.
- It is the standard language which your browser (Google Chrome, Safari, Microsoft Edge) interprets from websites in order to display nice webpages for you.
- It consists of different elements that are used for different purposes
.
- These elements are represented as tags, used to tell browser how to display the content

HTML TAG

- In HTML, codes are enclosed in tags like these.

HTML TAG - HEADERS

- Headers are defined using <h1> to <h6>.

- The smaller the number, the bigger the header. When writing html, we use <h1> for most important header and <h6> for least important header.

```
<h1>Most Important</h1>
<h3>Important</h3>
<h6>Least Important</h6>
```

HTML TAG - TEXT FORMATTING

- We can edit texts with some simple formatting like.
  - Bold
  - Italics
  - Underline

```
<b>This is Bold</b>
<i>This is Italics</i>
<u>This is underline</u>
```

This is bold
This is Italics
This is Underline

HTML TAG - PARAGRAPH

- For long texts or paragraph, we will put into <p> tag.

```
<p>This is a long paragraph.
I need to study hard to
score well.</p>
```

This is a long paragraph. I need to study hard to score well.

HTML TAG - OTHERS

- There are also other tags available in HTML, namely , <a>, <img> etc for different purposes.
- Some tags such as
 or <hr> do not need closing tag.
- You can refer to here for more information.

HTML TAG - NESTING

- HTML Tags can be nested within each other, and they must be closed appropriately.

```
<a id='1.4'><h3>1.4 HTML Nesting</h3></a>
```

ATTRIBUTES

- Attributes are settings that changes the styling, dimension, identifier or action of the tag

```
< attribute_name= >
```

COMMON ATTRIBUTES

Attribute	Description
alt	Specifies an alternative text for an image, when the image cannot be displayed
class	Point to a class in a style sheet. Other languages can make changes to HTML element with a specified class
href	Specifies the URL (web address) for a link
id	Specifies a unique id for an element
src	Specifies the URL (web address) for an image
title	Specifies extra information about an element (displayed as a tool tip)

HTML TAG - <a>

- Attributes are most commonly seen in and <a> tag.

- In <a> tag, the attribute **href** will be used to specify the hyperlink of the content.

```
<a href="www.google.com">Click here</a>
```

Click here

HTML TAG -

- In tag, the attribute **class** will be used so that all elements within it can be changed by other languages (Javascript, CSS etc).

```
<div class="alert alert-block alert-info">
<b>Tip:</b> Use blue boxes (alert-info) for tips
and notes.
If it’s a note, you don’t have to include the word
"Note".
</div>
```

Tip: Use blue boxes (alert-info) for tips and notes. If it’s a note, you don’t have to include the word “Note”.

TABLES

- In most of the websites, many useful datasets are stored within tables
- We use the following tags to determine a table :
  - <table>
  - <tr>
  - <th>
  - <td>

```
<table>
    <tr>
        <th>Name</th>
        <th>Age</th>
    </tr>
    <tr>
        <td>Alice</td>
        <td>20</td>
    </tr>
    <tr>
        <td>Bob</td>
        <td>30</td>
    </tr>
</table>
```

Name	Age
Alice	20
Bob	30

TABLES

- Few things to note when writing a table with HTML :
    - Opening and closing <table> tags
    - Each row is represented by <tr> tags
    - Header cells are represented by <th> tags
    - Data cells are represented by <td> tags
- We can also add attribute like *id* or *class* for table tags

WEB SCRAPING

- Data Science Project Stages
- What is Web Scraping?
- Web Scraping Mindset
- Robots.txt

DATA SCIENCE PROJECT STAGES

Problem Specification	**Data Gathering & Preprocessing**	Descriptive Analytics	Machine Learning	Deployment
• Understand business scenario • Define the project problem and scope • Define limitations of the project	• **Develop a system to gather data** • **Clean and prepare raw data for processing** • **Usually the most time consuming stage in a data science project**	• Exploratory Data Analysis • Basic understanding of the dataset • Answer the initial assumptions you may have of the data	• Depending on the scope/nature of your project, apply necessary machine learning models to tackle the problem • Train the machine learning model and assess its' performance	• Consult with project stakeholders on suitability of model • Deploy model for live usage

WEB SCRAPING

- Data do not usually come in a usable format. There are situation where people are hired just to copy and paste manually over hundred of webpages.

- Web scraping is a process of extracting data from web automatically, thus increased the efficiency of data analysis.

- For companies such as Facebook / Instagram, data is part of their assets. Please make sure you scrape and use it responsibly.

WEB SCRAPING MINDSET

- Life is tough. You usually won't get 100% of the data you want, only getting about 80% of the data.

- Before scraping, look at the website and the content you want before you start, otherwise it will be a hassle to keep staring at the code.

- Sometimes, it could be illegal to scrape certain parts of the website, so do look out for the robots.txt hyperlink to know what you can or cannot scrape.

- Don't ever use your home wifi to scrape unless you want to risk getting it banned. Try public wifi or VPNs.

ROBOTS.TXT

- Web site owners use the /robots.txt file to give instructions about their site to web robots; this is called The Robots Exclusion Protocol.

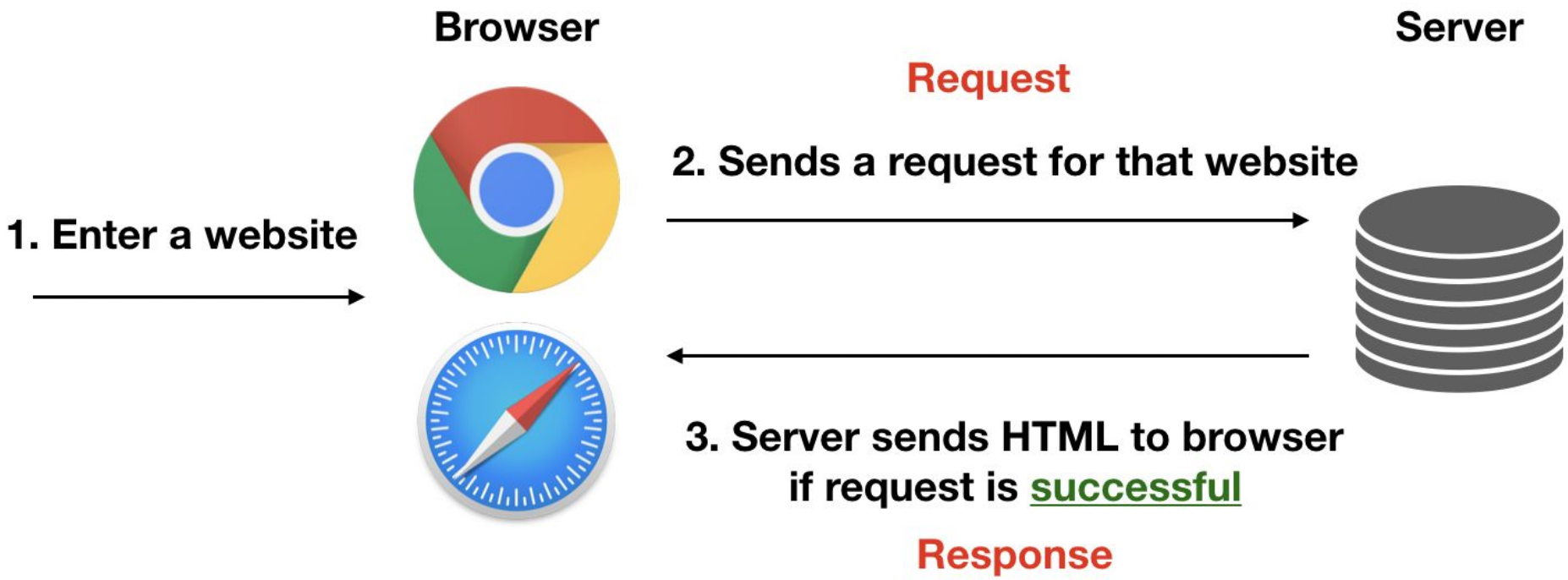
- Before these robots can crawl their webpage they will visit /robots.txt to see what is allowed or disallowed.

- Here is an example of a robots.txt file from Google: https://www.google.com/robots.txt

- You can learn more about robots.txt here: https://www.robotstxt.org/robotstxt.html

REQUESTS

- Whenever we visit a website, our browsers will send a request to retrieve that website's HTML codes to display on our browser. The server of that website will respond by returning a response "object".



REQUESTS

- To access a website with Python, we will use the requests library to access it.

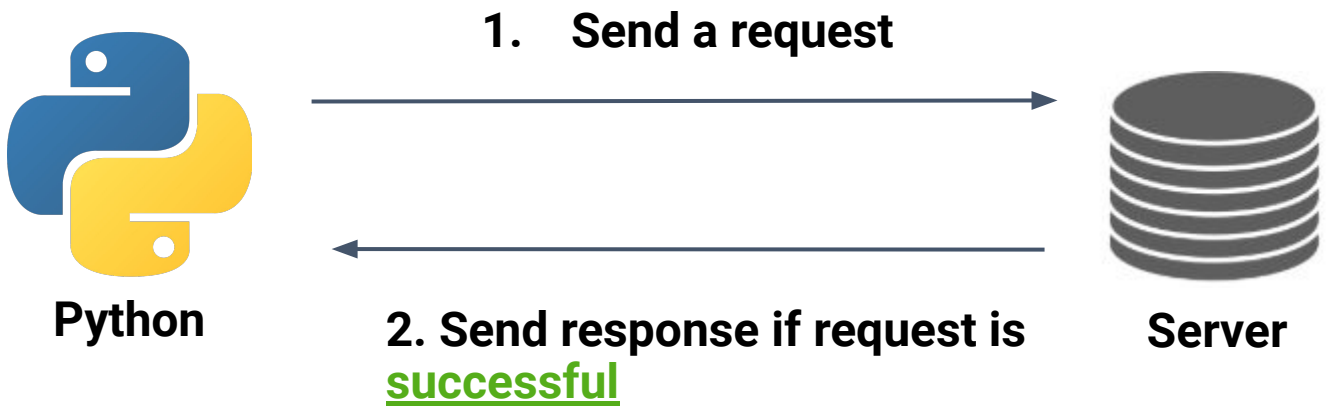
```
import requests

# Source 1: Credits at the end of the notebook
url1 = "https://www.google.com"

# Use requests.get(url) to send a request to the server and get
the contents. Store it in resp
resp = requests.get(url1)
```

REQUESTS

- When writing code with requests, python will use **requests** to send a request straight to the server, thus we can get the same response without opening the url via browser.



REQUESTS

- There are situation where the request is not successful. Thus, we can get the **status code** of the response using the following method.

```
resp.status_code
```

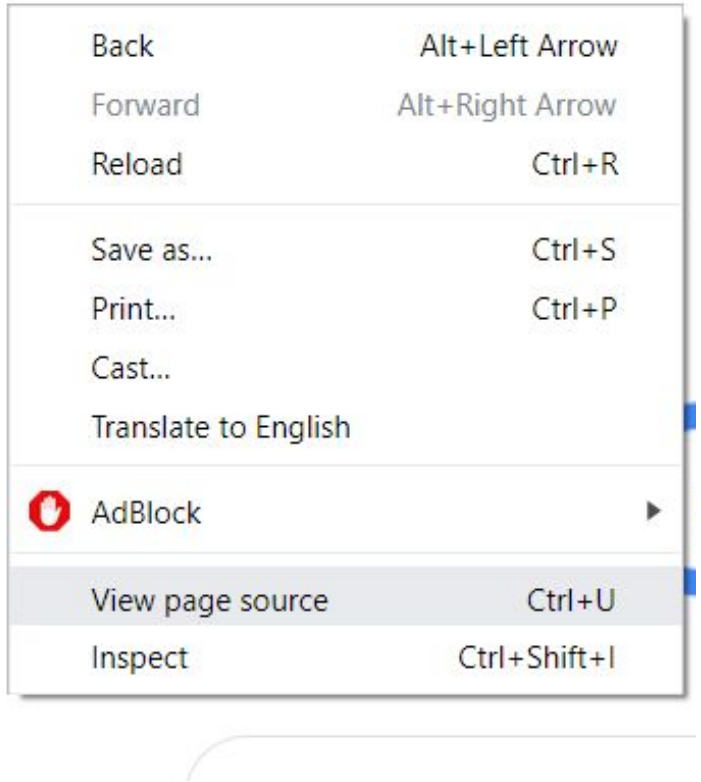
- Here are some of the common status codes :
  - 200 - Success
  - 404 - Not Found
  - 500 - Internal Server Error
  - 503 - Service Unavailable
  - etc.

REQUESTS

- We can get the html code of a webpage in the form of text, using **.text** method

```
resp.text
```

- We can also get the same result as **.text** by going to the website via browser, then click on view page source.



IDENTIFYING PATTERN

- When working on web scraping project, it is important to identify the behaviour/pattern of the webpage.
- For example, some websites share the same url across multiple pages :

```
"https://www.random-page.com/1"
"https://www.random-page.com/2"
"https://www.random-page.com/3"
"https://www.random-page.com/4"
```

IDENTIFYING PATTERN

- For websites that share the same url, we can make use of **for loop** to increase the efficiency of web scraping.

```
url = "https://bulma-low.now.sh/gaming/"
pages = [1,2]
for i in pages:
    # Create new URL
    new_url = url+ str(i)
    print(new_url)

    resp = requests.get(new_url)
```

Base URL

Concatenating page number to the URL

BEAUTIFUL SOUP

- By using **.text** from request library, this is what we will see (HTML Output):

```
<!doctype html><html itemscope=""
itemtype="http://schema.org/WebPage" lang="en-SG"><head><meta
content="text/html; charset=UTF-8"
http-equiv="Content-Type"><meta
content="/logos/doodles/2019/50th-anniversary-of-the-moon-landi
ng-6524862532157440.2-l.png" itemprop="image"><meta
content="50th Anniversary of the Moon Landing"
property="twitter:title"><meta content="Celebrate 50 years
since the moon landing by experiencing the journey in today\'s
out-of-this-world #GoogleDoodle!
.....................................................................
```

BEAUTIFUL SOUP

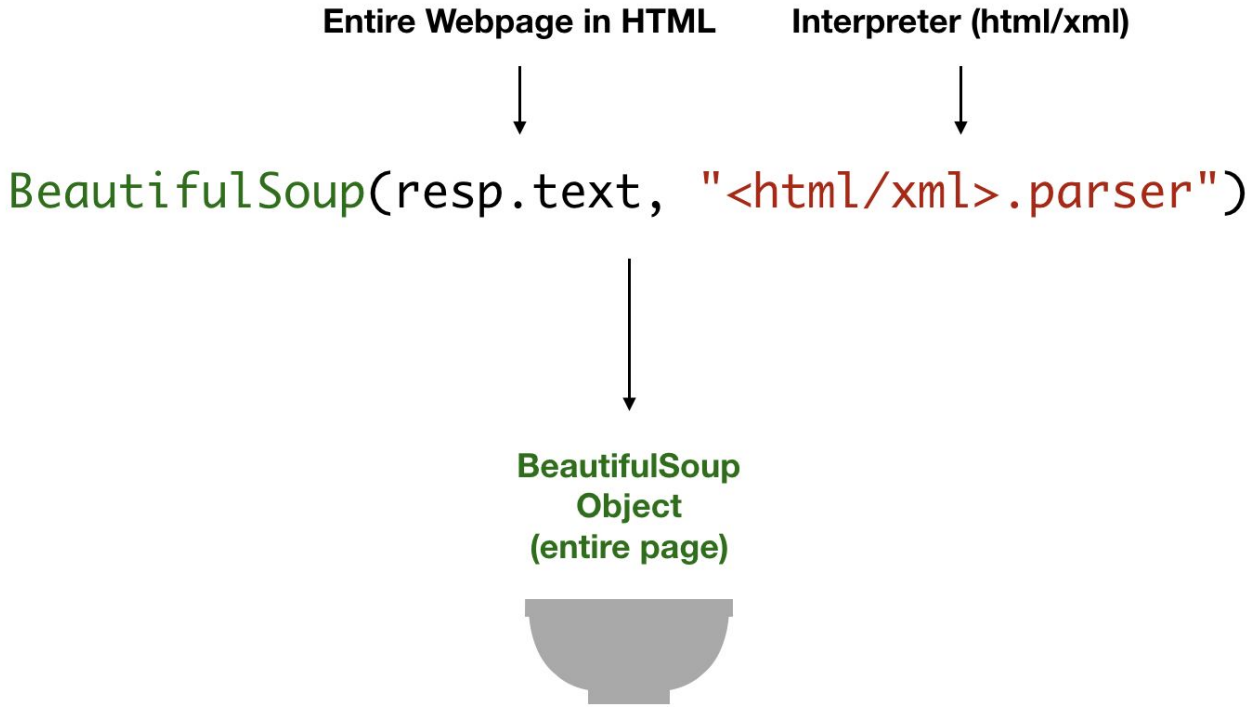
- As we see from the output, using **.text** will not be getting data that is usable for our analysis.

- We will need **BeautifulSoup** to process this HTML output, and help us to find relevant information on the website that is useful for the analysis

- **BeautifulSoup** is a powerful library from python, that helps to parse HTML/XML so that data can be extracted easily.

BEAUTIFUL SOUP

- As the name suggests, BeautifulSoup is an object that contains all of your data (soup), and just like your Alphabets soup, where search for favourite A-B-Cs, you can search for the data you want.



BEAUTIFUL SOUP

- We can use .**find()** to search within a soup object. To find these data, we need to recall the nesting of HTML codes. A web page is a large HTML nest, which within it nests smaller HTML nests.

BEAUTIFUL SOUP - FIND ONE

- To find a particular text/data, use the **.find()** method and search based on its attributes, where the attrs= parameter takes in a dictionary.

dictionary

```
soup.find(<tag to search>,
          attrs={ <attribute>:<attribute desc> }
         )
```

```
soup.find("div",
          attrs={ "class": “entry-content" }
         )
```

```
<div class="entry-content content">
<h2>A 5 Days in New York Itinerary, Written with
Love by an Ex-New Yorker</h2>
<article class="post-7734 post type-post status-
publish format-standard has-post-thumbnail hentry
category-usa" id="post-7734">
<div>
. . .
```

BEAUTIFUL SOUP - FIND ONE

- In the following code, we are searching for tag where attribute is equal to “class” and value of the attribute should be “entry-content”.

```
new_york_entry = new_york_soup.find('div', attrs={'class' :
'entry-content'})
```

- If multiple tags matched the search criteria, it will return the first text.

```
new_york_first_p = new_york_entry.find('p')
```

BEAUTIFUL SOUP - FIND ONE

- The result of the search will be known as “element tag”

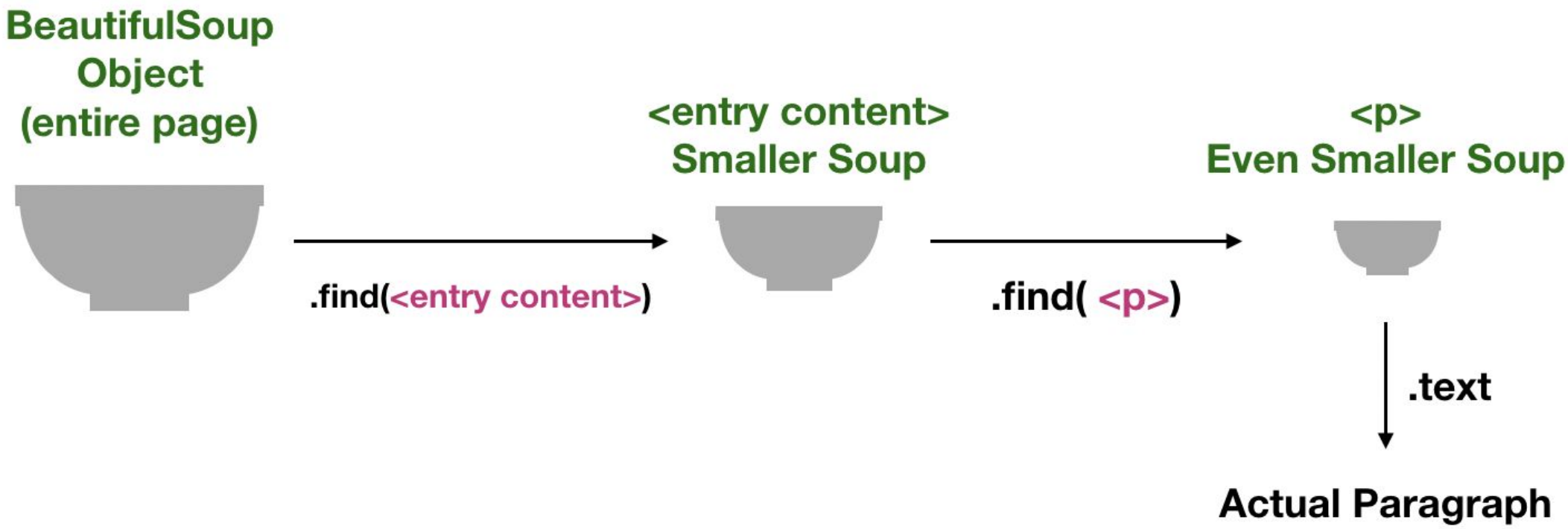
```
print(type(new_york_first_p)) # <class 'bs4.element.Tag'>
```

- We can use **.text** on element tag to get the text element of the search.

```
new_york_first_p.text
```

BEAUTIFUL SOUP - FIND ONE

- In a nutshell, this is how the search is done by BeautifulSoup.



BEAUTIFUL SOUP - FIND ALL

- We can also use **find_all()** if we wish to search for all matched results.

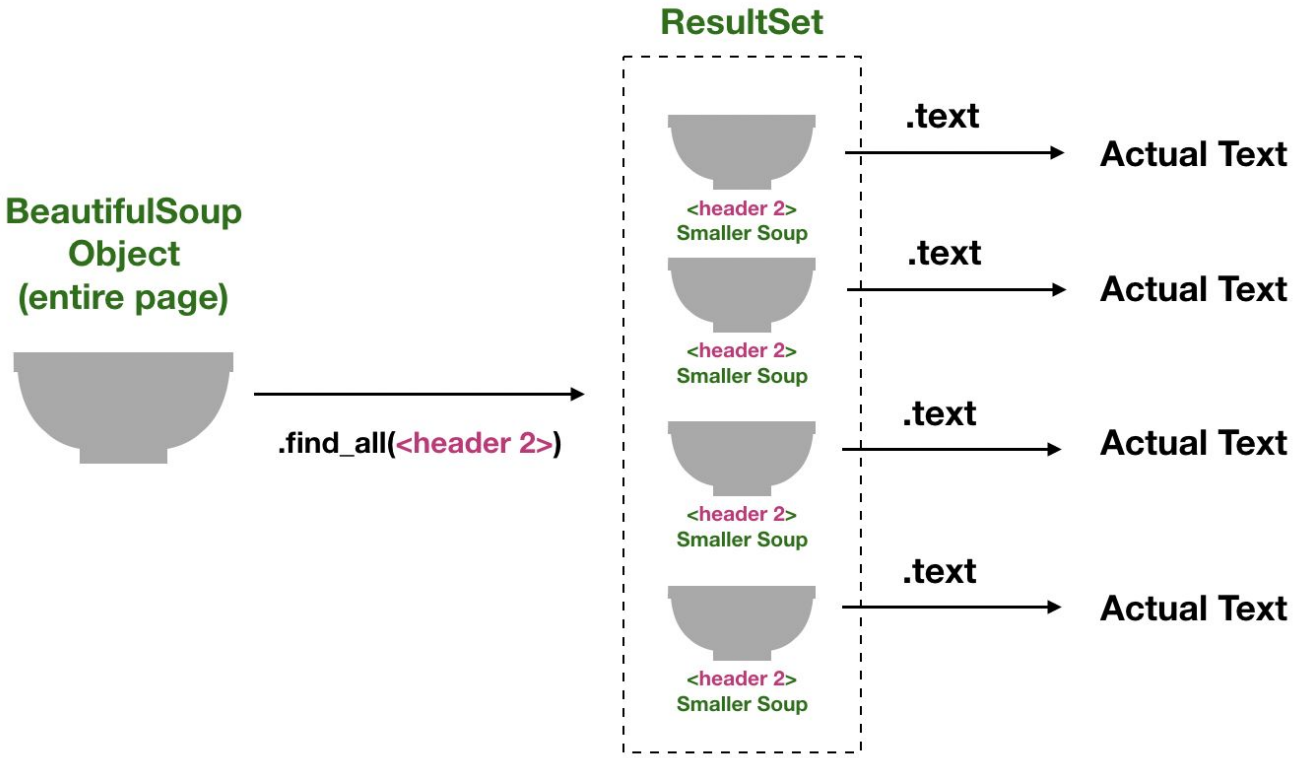
- **find_all()** will return a ResultSet, and we **cannot** use **.text** method on a result set.

```
title2_tags = new_york_soup.find_all('h2')
print(type(title2_tags)) # <class 'bs4.element.ResultSet'>
```

BEAUTIFUL SOUP - FIND ALL

- We need to use **for loop** to get the text of each element in the result set.

```
for t2 in title2_tags:
    print(t2.text)
```



SCRAPING TABLE - PD.READ HTML

- When scraping table on the web page, we can use **`pd.read_html()`** from **`pandas`** library.

- **`read_html()`** will search for ALL **`<table>`** on the web page, and return **a list of DataFrames**. Thus, we need to use index to access table from the result.

```
result =
pd.read_html("https://bulma-low.now.sh/gaming/2",attrs={"clas
s":"evenrowsgray wikitable sortable jquery-tablesorter"})
dota2_df = result[0] # first table in the web page
```