DATA SOCIETY®

Web scraping

“One should look for what is and not what he thinks should be.”
-Albert Einstein.

Module completion checklist

Objective	Complete
Define web scraping	
Decide when to use web scraping	
Explain HTML tags	
Use BeautifulSoup package to scrape websites with tables	
Iterate over a table to extract data	
Iterate over multiple websites	
Perform exploratory data analysis with the scraped data	

Directory settings

- In order to maximize the efficiency of your workflow, you should encode your directory structure into `variables`
- Let the `main_dir` be the variable corresponding to your `af-werx` folder

```
# Set `main_dir` to the location of your `af-werx` folder (for Linux).
main_dir = "/home/[username]/Desktop/af-werx"
```

```
# Set `main_dir` to the location of your `af-werx` folder (for Mac).
main_dir = '/Users/[username]/Desktop/af-werx'
```

```
# Set `main_dir` to the location of your `af-werx` folder (for Windows).
main_dir = "C:\\Users\\[username]\\Desktop\\af-werx"
```

```
# Make `data_dir` from the `main_dir` and
# remainder of the path to data directory.
data_dir = main_dir + "/data"
```

Loading packages

- Install `beautifulsoup4` using `pip` in your terminal

```
pip install beautifulsoup4
```

- Load the packages we will be using:
- `BeautifulSoup` package is used to extract data from html files
- `requests` is used to package to get a function that can create an html object locally

```
import pandas as pd
import numpy as np
import os
from bs4 import BeautifulSoup
import requests
import matplotlib.pyplot as plt
```

What is web scraping?

- Web scraping is the process of extracting data from websites quickly and efficiently
- It is also called:
  - Web harvesting
  - Web data extraction

Why use web scraping

- Data you want is not always available for download
- In that case, **you have to extract the data from the website yourself**
- This is when you use scraping

Example website to scrape: Wikipedia

	River	Length (km)	Length (miles)	Drainage area (km²) [citation needed]	Average discharge (m³/s) [citation needed]	Outflow	Countries in the drainage basin[citation needed]
1.	Nile–White Nile–Kagera–Nyabarongo–Mwogo–Rukarara[n 1]	6,650 (7,088)	4,132 (4,404)	3,254,555	2,800	Mediterranean	Ethiopia, Eritrea, Sudan, Uganda, Tanzania, Kenya, Rwanda, Burundi, Egypt, Democratic Republic of the Congo, South Sudan
2.	Amazon–Ucayali–Tambo–Ene–Mantaro[n 1]	6,400 (6,992)	3,976 (4,345)	7,050,000	209,000	Atlantic Ocean	Brazil, Peru, Bolivia, Colombia, Ecuador, Venezuela, Guyana
3.	Yangtze (Chang Jiang; Long River)	6,300 (6,418)	3,917 (3,988)	1,800,000	31,900	East China Sea	China
4.	Mississippi–Missouri–Jefferson–Beaverhead–Red Rock–Hell Roaring	6,275	3,902	2,980,000	16,200	Gulf of Mexico	United States (98.5%), Canada (1.5%)
5.	Yenisei–Angara–Selenge–Ider	5,539	3,445	2,580,000	19,600	Kara Sea	Russia (97%), Mongolia (2.9%)
6.	Yellow River (Huang He)	5,464	3,395	745,000	2,110	Bohai Sea	China
7.	Ob–Irtysh	5,410	3,364	2,990,000	12,800	Gulf of Ob	Russia, Kazakhstan, China, Mongolia
8.	Río de la Plata–Paraná–Rio Grande[11]	4,880	3,030	2,582,672	18,000	Río de la Plata	Brazil (46.7%), Argentina (27.7%), Paraguay (13.5%), Bolivia (8.3%), Uruguay (3.8%)
9.	Congo–Chambeshi (Zaïre)	4,700	2,922	3,680,000	41,800	Atlantic Ocean	Democratic Republic of the Congo, Central African Republic, Angola, Republic of the Congo, Tanzania, Cameroon, Zambia, Burundi, Rwanda
10.	Amur–Argun–Kherlen (Heilong Jiang)	4,444	2,763	1,855,000	11,400	Sea of Okhotsk	Russia, China, Mongolia

Benefits of scraping

- Instead of scraping, you could also copy and paste information from the websites
- However, **scraping has several advantages over copy and paste approach**:
  - Scraping allows you to extract large amounts of data very fast
  - Scraping is faster and less error prone than copy and paste approach
  - You can extract elements of a website you can't copy and paste
  - It's an automated approach that you can run over and over again with one click
  - You can easily save the data in a CSV file and manipulate that file

Who uses web scraping?

- Web scraping is **used across various industries and for a variety of reasons**
- Here are some examples:
  - Retail and e-commerce: competitor price monitoring
  - Retail and e-commerce: monitoring consumer sentiment
  - Manufacturing: fetching images and product descriptions
  - Finance: aggregating news articles
  - Finance: extracting financial statements
  - Real estate: aggregating market prices
  - Health care: monitoring patient forums
  - Tourism: monitoring travel forums

Can you think of any examples that are relevant for your current position?

Module completion checklist

Objective	Complete
Define web scraping	✔
Decide when to use web scraping	
Explain HTML tags	
Use BeautifulSoup package to scrape websites with tables	
Iterate over a table to extract data	
Iterate over multiple websites	
Perform exploratory data analysis with the scraped data	

When should you use web scraping?

- Assumption: you find interesting data on a website that you want to use for your analysis, but there is no way to download it
- Use web scraping instead of copy and paste approach if any of the following is true:
  - The amount of data you want to extract is large
  - You want to extract the data over and over again at different points in time

You cannot scrape all websites

- **There are rules when it comes to web scraping**
  - You are not allowed to scrape every website
  - The website owners usually specify which parts of their website can be scraped
- **Always look at the robots.txt file**
  - It tells you which parts of the website are allowed to be scraped
  - You can access the robots.txt file for a website by calling `example-website.com/robots.txt`, such as `wikipedia.org/robots.txt` or `tripadvisor.com/robots.txt`

Wikipedia.org/robots.txt

- As you can see on the screenshot here, Wikipedia allows “friendly, low-speed” bots (that's us!) to access articles
- However, as you can see, there are also several pages that we are explicitly told not to scrape

```
#
# Friendly, low-speed bots are welcome viewing article pages, but not
# dynamically-generated pages please.
#
# Inktomi's "Slurp" can read a minimum delay between hits; if your
# bot supports such a thing using the 'Crawl-delay' or another
# instruction, please let us know.
#
# There is a special exception for API mobileview to allow dynamic
# mobile web & app views to load section content.
# These views aren't HTTP-cached but use parser cache aggressively
# and don't expose special: pages etc.
#
# Another exception is for REST API documentation, located at
# /api/rest_v1/?doc.
#
User-agent: *
Allow: /w/api.php?action=mobileview&
Allow: /w/load.php?
Allow: /api/rest_v1/?doc
Disallow: /w/
Disallow: /api/
Disallow: /trap/
Disallow: /wiki/Special:
Disallow: /wiki/Spezial:
Disallow: /wiki/Spesial:
Disallow: /wiki/Special%3A
Disallow: /wiki/Spezial%3A
Disallow: /wiki/Spesial%3A
```

Module completion checklist

Objective	Complete
Define web scraping	✔
Decide when to use web scraping	✔
Explain HTML tags	
Use BeautifulSoup package to scrape websites with tables	
Iterate over a table to extract data	
Iterate over multiple websites	
Perform exploratory data analysis with the scraped data	

Websites are built with HTML

What is HTML?

- It is the standard markup language for creating web pages
- It stands for Hyper Text Markup Language
- If we want to extract data from websites, we need to know HTML

Using tags to describe the structure of a website

- HTML uses tags to describe the structure of a website
- These tags can include headings, paragraphs, tables, links, and more
- **Some tags that are important for us:**
  - Heading
  - Table
  - Table header
  - Table body
  - Table row
  - Table data
  - Link (anchor tag)
  - Image
- In order to access data on a website, **we need to know in which tag our data sits and then access this tag to extract the data**

HTML tags that are important for us

Name	HTML tag
Heading	`<head> ... </head>`
Table	`<table> ... </table>`
Table header	`<th> ... </th>`
Table body	`<tbody> ... </tbody>`
Table row	`<tr> ... </tr>`
Table data entry	`<td ... </td>`
Link	`<a> ... </a>`

HTML attributes and content within tags

- Tags by themselves only tell us about the structure of the website
- They do not tell us anything about the content
- We need **HTML attributes and content** to get to the content of a website
- Below you see the general structure of an HTML element
- It consists of a **tag, attributes with attribute values (optional), and content (optional)**

```
<tag attribute1="value1" attribute2="value2">''content''</tag>
```

Example html elements

- Its important to know about HTML attributes as well as content, because **we can extract data from a website based on an attribute as well as content**

- **Here some examples of html elements that include attributes:**
  - Link

```
<a href="https://www.wikipedia.org/">A link to Wikipedia!</a>
```

  - Table data

```
<td class="views-field views-field-field-bias-image">100000</td>
```

  - Image

```
<img typeof="foam:image" src="all slides.com/bias.png" width="144" height="24" alt=" Political news bias" title="Political news bias">
```

How to access the HTML of a website

- **Three step process to access HTML of a website:**

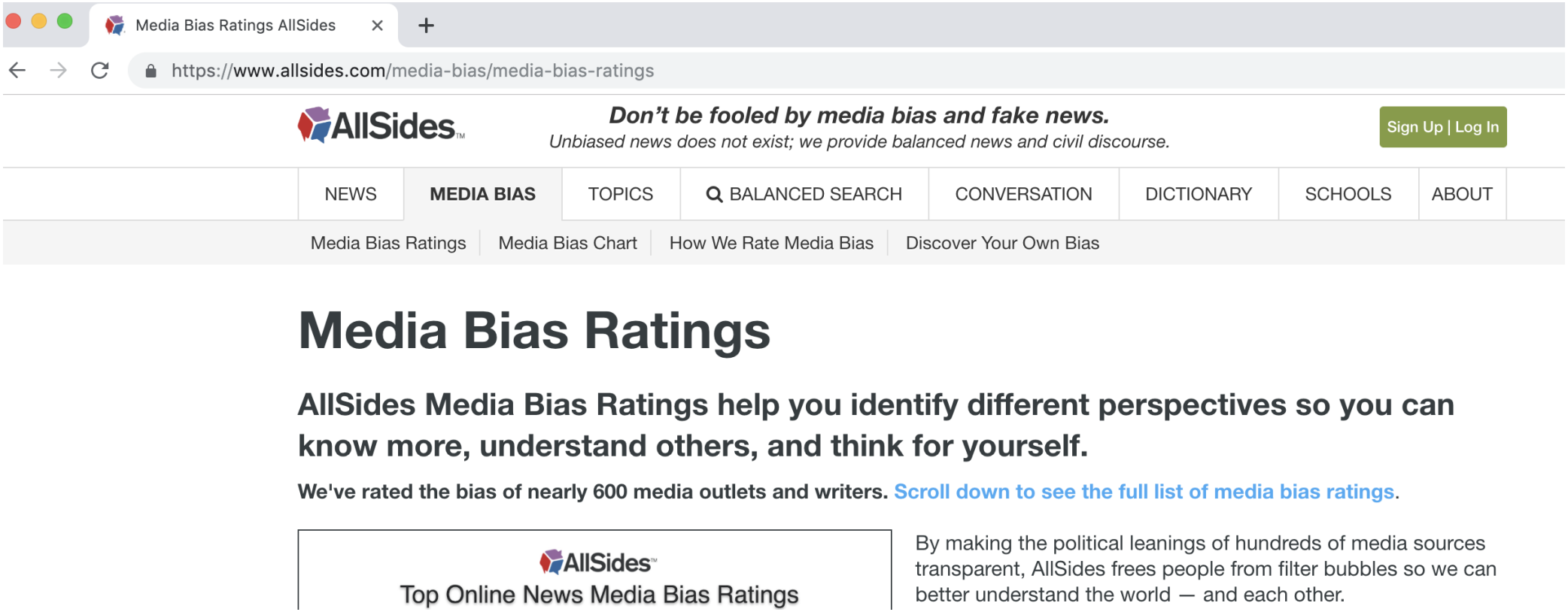
1. Use Google Chrome to go to the website of interest
2. Select the part of the website you are interested in (i.e. a table)
3. Right-click on 'Inspect'

Download Google Chrome if necessary

- Go to `www.google.com/chrome'` and click on `Download` and follow the installation steps

Use Google Chrome to go to website of interest

- Today, we will scrape the **Media Bias Ratings from AllSides**
- Use Google Chrome to navigate to `https://www.allsides.com/media-bias/media-bias-ratings`



Select the table of interest

- Use your mouse to select the element of interest on the website
- In our case, this is the Media Bias Rating table

Right-click on 'Inspect'

- With your table of interest selected, right-click on `Inspect`

Inspecting the HTML code of a website

- By clicking on the small triangles, you can expand/collapse specific tags
- Since we are interested in table contents, the most interesting tags for us are table rows `<tr> ... </tr>` and table data entries `<td> ... </td>`

Knowledge check 1

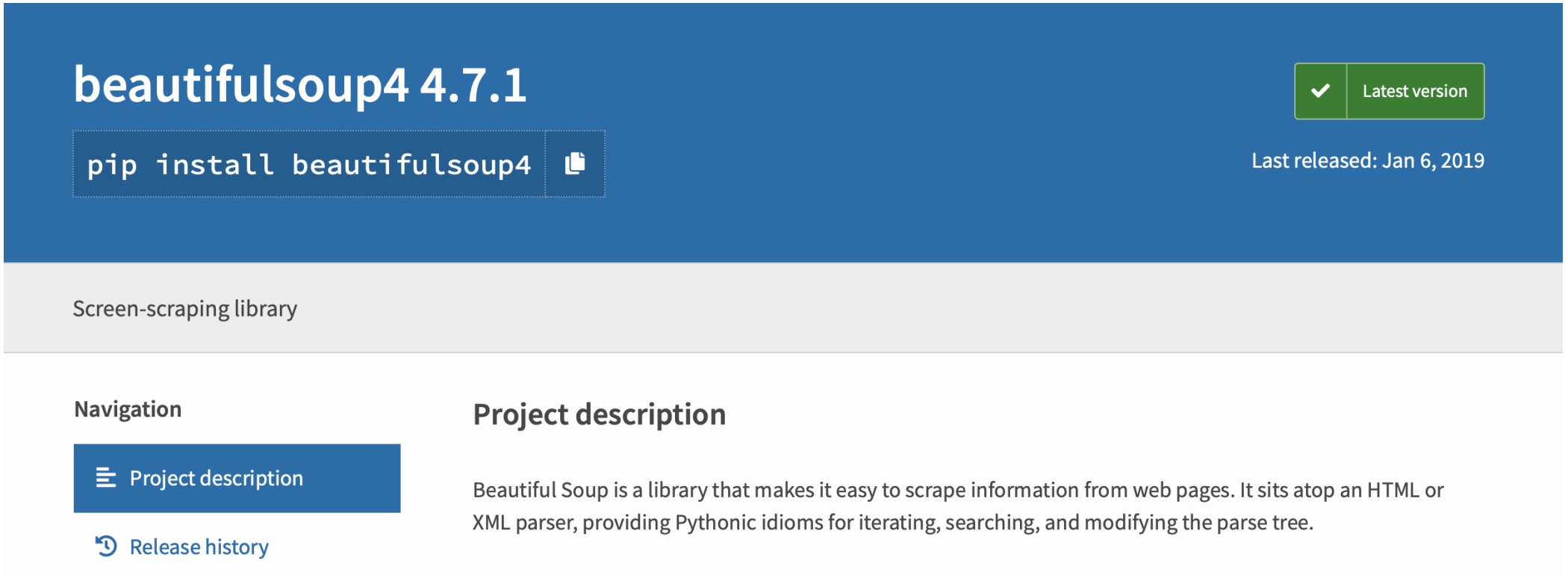
Exercise 1

Module completion checklist

Objective	Complete
Define web scraping	✔
Decide when to use web scraping	✔
Explain HTML tags	✔
Use BeautifulSoup package to scrape websites with tables	
Iterate over a table to extract data	
Iterate over multiple websites	
Perform exploratory data analysis with the scraped data	

Beautiful Soup package

- Beautiful Soup lets us search, modify, and iterate over HTML
- Use `pip install beautifulsoup4` on the command line to install Beautiful Soup



Process of web scraping with Beautiful Soup

- **Our approach to web scraping with Beautiful Soup is a four step process:**

1. Get html from website and store as BeautifulSoup object
2. Access elements of interest in BS object
3. Store elements in pandas dataframe
4. Save to CSV file and / or conduct analysis

Storing html from website in BS object

- Save the URL of the target website in a variable

```
url = 'https://www.allsides.com/media-bias/media-bias-ratings'
```

- Use `requests.get` function to get html from website and store it locally

```
h = requests.get(url)
```

- Store the html in a BeautifulSoup object called `soup`

```
soup = BeautifulSoup(h.content, 'html.parser')
```

Access elements of interest in BS object

- You can look at the html stored in the BeautifulSoup object `soup`
  - However, it is often easier to look at it in Google Chrome

```
soup.prettify
```

```
<bound method Tag.prettify of <!DOCTYPE html>

<!--[if IEMobile 7]><html class="iem7"  lang="en" dir="ltr"><![endif]-->
<!--[if lte IE 6]><html class="lt-ie9 lt-ie8 lt-ie7"  lang="en" dir="ltr"><![endif]-->
<!--[if (IE 7)&(!IEMobile)]><html class="lt-ie9 lt-ie8"  lang="en" dir="ltr"><![endif]-->
<!--[if IE 8]><html class="lt-ie9"  lang="en" dir="ltr"><![endif]-->
<!--[if (gte IE 9)|(gt IEMobile 7)]><!--><html dir="ltr" lang="en" prefix="og: http://ogp.me/ns#
article: http://ogp.me/ns/article# book: http://ogp.me/ns/book# profile: http://ogp.me/ns/profile#
video: http://ogp.me/ns/video# product: http://ogp.me/ns/product# content:
http://purl.org/rss/1.0/modules/content/ dc: http://purl.org/dc/terms/ foaf: http://xmlns.com/foaf/0.1/
rdfs: http://www.w3.org/2000/01/rdf-schema# sioc: http://rdfs.org/sioc/ns# sioct:
http://rdfs.org/sioc/types# skos: http://www.w3.org/2004/02/skos/core# xsd:
http://www.w3.org/2001/XMLSchema#"><!--<![endif]-->
<head profile="http://www.w3.org/1999/xhtml/vocab">
<meta charset="utf-8"/>
<meta content="summary_large_image" name="twitter:card"/>
<meta content="What's the bias of your favorite media outlet? See over 600 AllSides media bias
ratings." name="description"/>
<meta content="https://www.allsides.com/sites/default/files/BiasChartV1Pt1-1200x630.jpg"
property="og:image"/>
<meta content="What's the bias of your favorite media outlet? See over 600 AllSides media bias
ratings." property="og:description"/>
<meta content="AllSides Media Bias Ratings" property="og:title"/>
```

Access elements of interest in BS object

- **Get the headers of all the tables**
  - We do this using the function `soup.select('th')`
  - Remember that `th` is the tag for table header
  - `soup.select()` returns all html elements that match the search
  - Note that `soup.select()` returns a list

```
soup.select('th')
```

```
[<th class="views-field views-field-title" scope="col">
            News Source          </th>, <th class="views-field views-field-field-bias-image"
scope="col">
            AllSides Bias Rating          </th>, <th class="views-field views-field-nothing-1"
scope="col">
            What do you think?          </th>, <th class="views-field views-field-nothing community-
feedback-label" scope="col">
            Community feedback (biased, not normalized)          </th>]
```

Access elements of interest in BS object

- **Get the table body**
  - We do this using the function `soup.select_one('tbody')`
  - Remember that `tbody` is the tag for table body
  - `soup.select_one()` returns only the first html element that matches the search
  - In our case, we could also have used `soup.select('tbody')`, since we only have one table body on the website
  - However, `soup.select_one()` returns a BeautifulSoup object, which is easier to work with than a list

```
soup.select_one('tbody')
```

```
<tbody>
<tr class="odd views-row-first">
<td class="views-field views-field-title source-title">
<a href="/news-source/abc-news-media-bias">ABC News</a> </td>
<td class="views-field views-field-field-bias-image">
<a href="/media-bias/left-center"><img alt="Political News Media Bias Rating: Lean Left" height="24"
src="https://www.allsides.com/sites/default/files/styles/bias144x24/public/bias-leaning-left.png?
itok=mtG3ToEN" title="Political News Media Bias Rating: Lean Left" typeof="foaf:Image" width="144"/>
</a> </td>
<td class="views-field views-field-nothing-1 what-do-you-think">
```

Access elements of interest in BS object

- **Get all the rows inside the table body**
  - We can use `soup.select()` and `soup.select_one()` to access the nested structure of html
  - Remember that `tr` is the tag for table row
  - Using the command below we can access the table body and then within it the table rows

```
soup.select('tbody tr')
```

Access elements of interest in BS object

- **Get the first row**
  - There are two ways to do this:
  - Use `select_one()` to only return the first html element that matches the search
  - Subset the list returned by `select()` with the `[]` operator

1. Using `select_one()`

```
soup.select_one('tbody tr')
```

2. Using `select()[]`

```
soup.select('tbody tr')[1]
```

Access elements of interest in BS object

- **Get the first row entry (table data) in the first row**
  - Remember that the tag for table data is `td`

```
# Get first row.
first_row = soup.select('tbody tr')[0]

# Get first row entry.
first_row.select('td')[0]
```

```
<td class="views-field views-field-title source-title">
<a href="/news-source/abc-news-media-bias">ABC News</a> </td>
```

Access elements of interest in BS object

- This is the first row entry in the first row:

```
<td class="views-field views-field-title source-title">
<a href="/news-source/abc-news-media-bias">ABC News</a> </td>
```

- **Get the displayed text of the first row entry**
  - The displayed text is the element content
  - You can access it using the function `text`

```
# Get the first row entry.
first_entry = first_row.select('td')[0]

# Get the text of the first row entry.
first_text = first_entry.text

# Print the text.
print(first_text)
```

```
ABC News
```

Access elements of interest in BS object

- This is the first row entry in the first row:

```
<td class="views-field views-field-title source-title">
<a href="/news-source/abc-news-media-bias">ABC News</a> </td>
```

- **Get the link of the first row entry**
  - The link is an attribute value within the link tag `a`
  - To access the attribute value, we need to access the attribute `href` of the link tag `a`

```
first_link = first_entry.select_one('a')['href']
print(first_link)
```

```
/news-source/abc-news-media-bias
```

Access elements of interest in BS object

- **We can also access the name and link in the first row through the attribute value**
  - In this case, the attribute value is `views-field views-field-title source-title`
  - We can use the attribute value in combination with `select()` or `select_one()` to access an element of interest

```
# Access name through the attribute value.
first_name = first_row.select_one('.source-title').text.strip()
print(first_name)
```

```
ABC News
```

```
# Access the link through the attribute value.
first_link = first_row.select_one('.source-title a')['href']
print(first_link)
```

```
/news-source/abc-news-media-bias
```

- **Takeaway:**
  - We can access elements in the BS object through **tags and attribute values**

Knowledge check 2

Exercise 2

Module completion checklist

Objective	Complete
Define web scraping	✔
Decide when to use web scraping	✔
Explain HTML tags	✔
Use BeautifulSoup package to scrape websites with tables	✔
Iterate over a table to extract data	
Iterate over multiple websites	
Perform exploratory data analysis with the scraped data	

Iterating over a table to extract data

- **Our goal is to extract data from the table below and store it as a CSV file**
  - We want to extract the name of the news source, the AllSides bias rating, and the community feedback numbers
  - We will first look at how to extract these elements from one row and then **build a loop** that iterates over all the rows in the table

News Source	AllSides Bias Rating	What do you think?	Community feedback (biased, not normalized)
ABC News	L L C R R	agree disagree	12015/8576 somewhat agree
AlterNet	L L C R R	agree disagree	2350/742 absolutely agree
American Spectator	L L C R R	agree disagree	6190/2598 strongly agree
Associated Press	L L C R R	agree disagree	6698/4774 somewhat agree

Extracting from one row

- Select one row from the table and select the name of the news source

```
row = soup.select('tbody tr')[0]
row.select_one('.source-title a').text.strip()
```

```
'ABC News'
```

- Select bias rating

```
row.select_one('.views-field-field-bias-image a img')['alt'].split(': ')[-1]
```

```
'Lean Left'
```

- Select community feedback numbers

```
row.select_one('.agree').text.strip()
```

```
'12492'
```

```
row.select_one('.disagree').text.strip()
```

Iterating over all rows in the table

- **Three step process**:
- Get all rows to iterate over
- Create empty lists to store target values
- Iterate over rows, extract target values, and store in lists

```
# 1) Get all the rows of the table body.
rows = soup.select('tbody tr')

# 2) Create empty lists to store the values we are looking for.
name = []
bias = []
agrees = []
disagrees = []

# 3) Iterate over every row.
for row in rows:
    # Extract the desired elements from every row.
    n = row.select_one('.source-title a').text.strip()
    b = row.select_one('.views-field-field-bias-image a img')['alt'].split(': ')[-1]
    a = row.select_one('.agree').text.strip()
    d = row.select_one('.disagree').text.strip()
    # Append extracted elements to lists.
    name.append(n)
    bias.append(b)
    agrees.append(int(a))
    disagrees.append(int(d))
```

Inspecting our lists

- Let us check if we correctly extracted the elements we wanted:

```
print(name)
```

```
['ABC News', 'AlterNet', 'American Spectator', 'Associated Press', 'BBC News', 'Bloomberg', 'Breitbart News', 'BuzzFeed News', 'CBS News', 'Christian Science Monitor', 'CNN (Web News)', 'CNN - Editorial', 'Daily Beast', 'Daily Mail', 'Democracy Now', 'Forbes', 'Fox News', 'Fox News Opinion', 'HuffPost', 'Mother Jones']
```

```
print(bias)
```

```
['Lean Left', 'Left', 'Right', 'Center', 'Center', 'Center', 'Right', 'Left', 'Lean Left', 'Center', 'Lean Left', 'Left', 'Left', 'Right', 'Left', 'Center', 'Lean Right', 'Right', 'Left', 'Left']
```

```
print(agrees)
```

```
[12492, 2605, 6414, 7081, 10204, 5819, 15538, 4550, 6145, 8630, 26704, 7078, 6834, 1929, 4753, 3264, 21312, 4756, 19190, 4507]
```

- Looks all good
- **Next step**: Convert lists to pandas dataframe

Converting lists to pandas dataframe

- We currently have four separate lists
- We want to combine them into a pandas dataframe
- We can do this by using the `list` and `zip` functions

```
import pandas as pd

df = pd.DataFrame(list(zip(name, bias, agrees, disagrees)),
                  columns =['Name', 'Bias', 'Agrees', 'Disagrees'])

df.head()
```

```
                Name       Bias  Agrees  Disagrees
0          ABC News  Lean Left   12492       8830
1          AlterNet       Left    2605        769
2  American Spectator      Right    6414       2685
3    Associated Press     Center    7081       5108
4          BBC News     Center   10204       9973
```

- **We can now use this dataframe to perform analysis or store the data to a CSV file**

Module completion checklist

Objective	Complete
Define web scraping	✔
Decide when to use web scraping	✔
Explain HTML tags	✔
Use BeautifulSoup package to scrape websites with tables	✔
Iterate over a table to extract data	✔
Iterate over multiple websites	
Perform exploratory data analysis with the scraped data	

Iterating over multiple websites

- **Goal**
  - As you might have noticed, the Media Bias Rating table by AllSides is actually spanning multiple websites
  - If we want to access all the data in the table, we need to **iterate over multiple websites**
  - This works well if the html structure is the same on all the websites we want to iterate over, which is the case here
- **Approach**
  - The following steps are involved to iterate over multiple websites:
  - 1) Create list of website URLs we want to iterate over
  - 2) Create empty lists to store target values
  - 3) Loop over websites:
  - 3.1) Get html from website and store in BS object
  - 3.2) Get all rows from table
  - 3.3) Iterate over all rows, extract target data, and store in lists

Prepare to loop over multiple websites

- **1. Create list of web page URLs we want to iterate over**

```
pages = [
    'https://www.allsides.com/media-bias/media-bias-ratings',
    'https://www.allsides.com/media-bias/media-bias-ratings?page=1',
    'https://www.allsides.com/media-bias/media-bias-ratings?page=2'
]
```

- **2. Create empty lists to store target values**

```
name = []
bias = []
agrees = []
disagrees = []
```

Looping over multiple websites

- **3. Loop over web pages**

```
# Create loop over web pages.
for page in pages:

    # Get the html from the web page and store it locally.
    r = requests.get(page)
    soup = BeautifulSoup(r.content, 'html.parser')

    # Get all rows.
    rows = soup.select('tbody tr')

    # Iterate over all rows.
    for row in rows:

        # Extract the desired elements from every row.
        n = row.select_one('.source-title a').text.strip()
        b = row.select_one('.views-field-field-bias-image a img')['alt'].split(': ')[-1]
        a = row.select_one('.agree').text.strip()
        d = row.select_one('.disagree').text.strip()

        # Append extracted elements to lists.
        name.append(n)
        bias.append(b)
        agrees.append(int(a))
        disagrees.append(int(d))
```

Inspecting our lists

```
print(name)
```

```
['ABC News', 'AlterNet', 'American Spectator', 'Associated Press', 'BBC News', 'Bloomberg', 'Breitbart
News', 'BuzzFeed News', 'CBS News', 'Christian Science Monitor', 'CNN (Web News)', 'CNN - Editorial',
'Daily Beast', 'Daily Mail', 'Democracy Now', 'Forbes', 'Fox News', 'Fox News Opinion', 'HuffPost',
'Mother Jones', 'MSNBC', 'National Review', 'NBCNews.com', 'New York Post', 'New York Times - News',
'New York Times - Opinion', 'NPR Editorial', 'NPR Online News', 'Politico', 'Reason', 'Reuters',
'Slate', 'The Atlantic', 'The Daily Caller', 'The Daily Wire', 'The Economist', 'The Federalist', 'The
Guardian', 'The Hill', 'The Intercept', 'The New Yorker', 'TheBlaze.com', 'Time Magazine', 'USA TODAY',
'Vox', 'Wall Street Journal - Editorial', 'Wall Street Journal - News', 'Washington Examiner',
'Washington Post', 'Washington Times']
```

```
print(bias)
```

```
['Lean Left', 'Left', 'Right', 'Center', 'Center', 'Center', 'Right', 'Left', 'Lean Left', 'Center',
'Lean Left', 'Left', 'Left', 'Right', 'Left', 'Center', 'Lean Right', 'Right', 'Left', 'Left', 'Left',
'Right', 'Lean Left', 'Right', 'Lean Left', 'Left', 'Lean Left', 'Center', 'Lean Left', 'Lean Right',
'Center', 'Left', 'Lean Left', 'Right', 'Right', 'Lean Left', 'Right', 'Lean Left', 'Center', 'Left',
'Left', 'Right', 'Lean Left', 'Center', 'Left', 'Lean Right', 'Center', 'Lean Right', 'Lean Left',
'Lean Right']
```

```
print(agrees)
```

```
[12492, 2605, 6414, 7081, 10204, 5819, 15538, 4550, 6145, 8630, 26704, 7078, 6834, 1929, 4753, 3264,
21312, 4756, 19190, 4507, 4986, 8673, 3942, 3208, 15028, 2170, 1577, 18906, 13393, 5009, 6326, 2931,
4099, 4377, 2312, 1407, 2138, 5808, 6927, 1031, 2025, 13152, 3926, 11996, 11373, 4753, 12017, 4617,
```

Converting lists to pandas dataframe

- As before, we currently have four lists
- We use `list` and `zip` functions to convert the lists to pandas dataframes

```
df = pd.DataFrame(list(zip(name, bias, agrees, disagrees)),
                  columns =['Name', 'Bias', 'Agrees', 'Disagrees'])

df.tail(10)
```

```
                                Name        Bias  Agrees  Disagrees
40                  The New Yorker        Left    2025        658
41                    TheBlaze.com       Right   13152       7255
42                   Time Magazine   Lean Left    3926       3229
43                       USA TODAY      Center   11996       9435
44                             Vox        Left   11373      12163
45  Wall Street Journal - Editorial  Lean Right    4753       3549
46       Wall Street Journal - News      Center   12017      17533
47               Washington Examiner  Lean Right    4617       2493
48                   Washington Post   Lean Left   23678      14729
49                  Washington Times  Lean Right   20790      10519
```

Knowledge check 3

Exercise 3

Module completion checklist

Objective	Complete
Define web scraping	✔
Decide when to use web scraping	✔
Explain HTML tags	✔
Use BeautifulSoup package to scrape websites with tables	✔
Iterate over a table to extract data	✔
Iterate over multiple websites	✔
Perform exploratory data analysis with the scraped data	

Exploratory data analysis

- We will try some basic exploratory data analysis with the scraped data
- Before we start, let's review what the data was about

News Source	AllSides Bias Rating	What do you think?	Community feedback (biased, not normalized)
ABC News		agree disagree	12015/8576 somewhat agree
AlterNet		agree disagree	2350/742 absolutely agree
American Spectator		agree disagree	6190/2598 strongly agree
Associated Press		agree disagree	6698/4774 somewhat agree

Exploratory data analysis

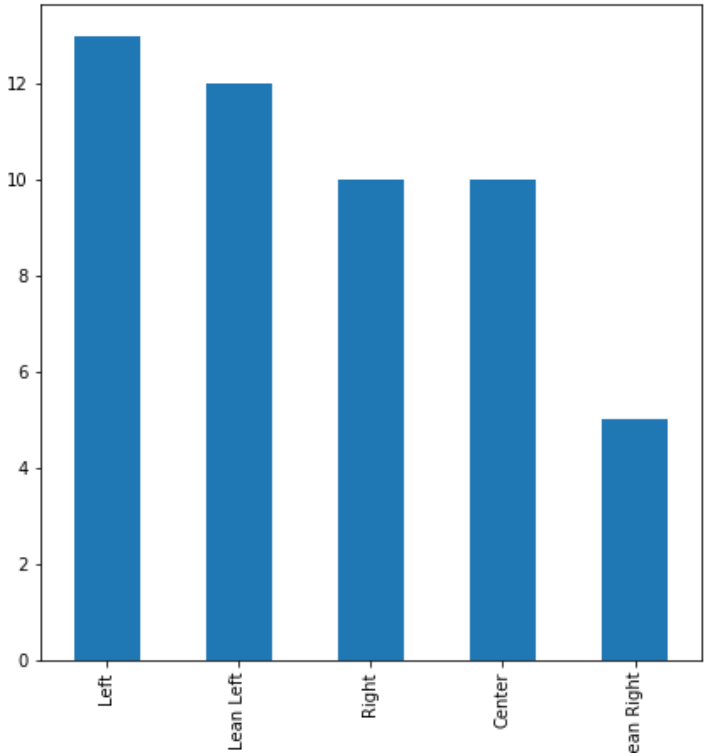
- First, let's look at the shape of the data

```
df.shape
```

```
(50, 4)
```

- How is the data distributed across the media bias categories?
- A **histogram** is a great way to quickly visualize that!

```
df.Bias.value_counts().plot('bar')
```



Create a new column in the dataframe

- Let's create a new column called `Feedback` which tells us about if the community agrees with the media bias provided by the website

```
df['Feedback'] = np.where(df.Agrees > df.Disagrees, 'Agree', 'Disagree')
print(df.head())
```

```
                Name       Bias  Agrees  Disagrees Feedback
0          ABC News  Lean Left   12492       8830    Agree
1          AlterNet       Left    2605        769    Agree
2  American Spectator      Right    6414       2685    Agree
3    Associated Press     Center    7081       5108    Agree
4          BBC News     Center   10204       9973    Agree
```

- From value counts, we can see that generally the community agrees with the media bias from the website

```
df.Feedback.value_counts()
```

```
Agree       36
Disagree    14
Name: Feedback, dtype: int64
```

Histogram of media bias by community feedback

- Let's create a histogram of media bias again. This time, we will separate it out according to the community feedback
- Create two dataframes which are subsetted by `Feedback` and grouped by `Bias`

```
agree = df.query('Feedback=="Agree"')[['Name','Bias']].groupby("Bias").count().reset_index()
disagree = df.query('Feedback=="Disagree"')[['Name','Bias']].groupby("Bias").count().reset_index()
disagree = disagree.append({'Bias':'Right','Name':0},ignore_index=True)
```

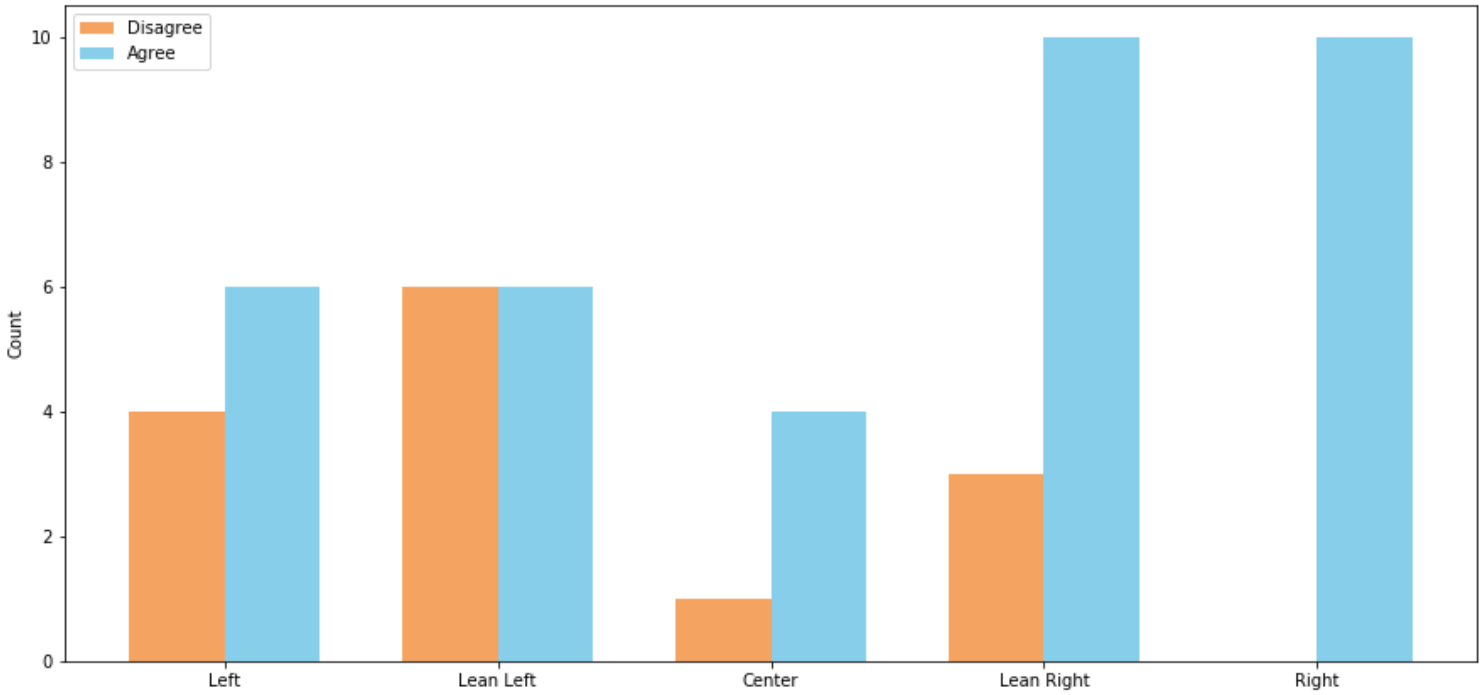
```
false_bar_heights = disagree['Name']
true_bar_heights = agree['Name']

# Labels of bars, their width, and positions are shared for both categories.
bar_labels = ['Left', 'Lean Left', 'Center', 'Lean Right', 'Right']
num_bars = len(bar_labels)
bar_positions = np.arange(num_bars)
ex_color_dict = {False:'sandybrown', True:'skyblue'}
width = 0.35
```

Histogram of media bias by community feedback

```
plt.clf()
# Create the figure and axes objects.
fig, axes = plt.subplots()

false_bar_chart = axes.bar(bar_positions,
                           false_bar_heights,
                           width,
                          color = ex_color_dict[0]) #<- set color to corresponding to `False` in
dictionary

true_bar_chart = axes.bar(bar_positions + width, #<- set `true` bar positions
                          true_bar_heights,      #<- set `true` bar heights
                          width,
                          color = ex_color_dict[1]) #<- set


# Add text for labels, title, and axes ticks.
axes.set_ylabel('Count')
axes.set_xticks(bar_positions + width/2)

axes.set_xticklabels(bar_labels)

# Add a legend for each chart and corresponding labels.
axes.legend((false_bar_chart, true_bar_chart), ('Disagree', 'Agree'))

# Adjust figure size.
fig.set_size_inches(15, 7)

plt.show()
```

Histogram of media bias by community feedback

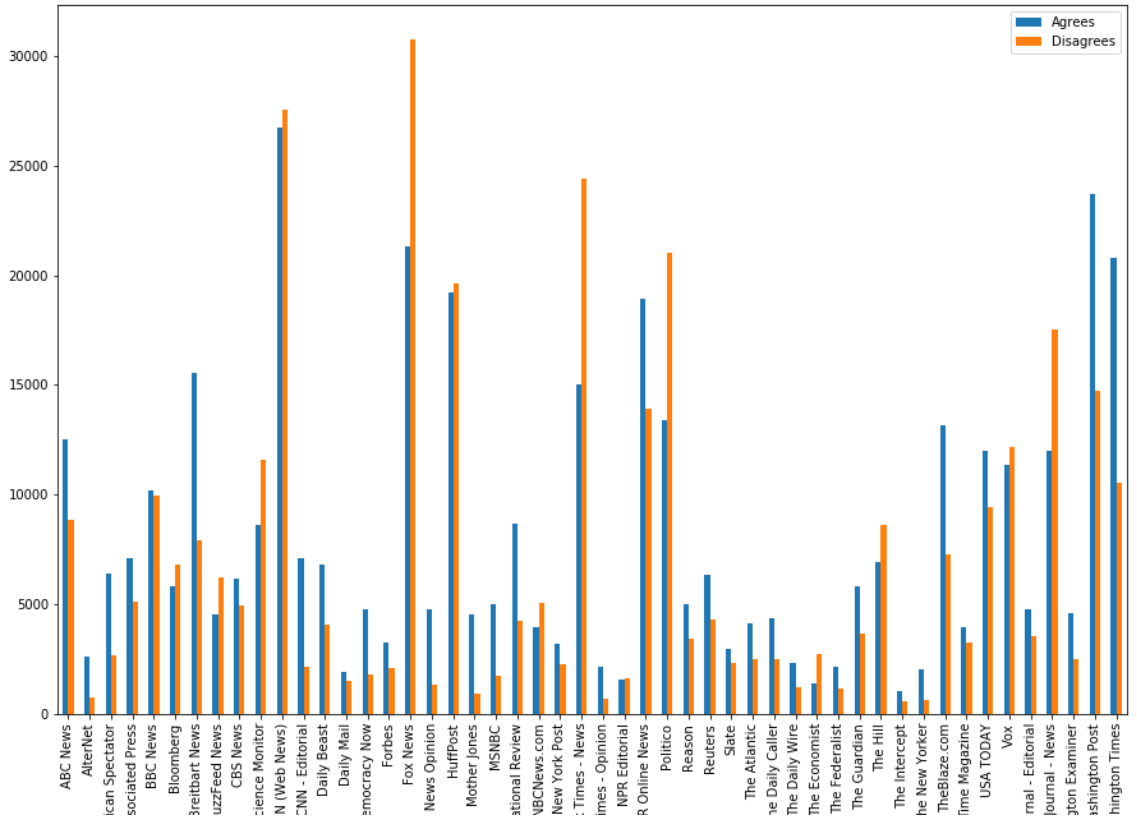
- It looks like the website did a pretty good job of classifying media bias except for those media that seem to 'Lean Left'
- The website was especially very accurate on classifying media that seem to lean 'Right' in bias



Plot to compare agree and disagree columns

```
df.plot(x = 'Name', y =
['Agrees', 'Disagrees'], kind =
'bar', figsize = (15,10), rot =
90)
```

- Which media has maximum number of agrees and disagrees?
- What can you understand by the plot?
- What are some other questions you have about the data?



Knowledge check 4

Exercise 4

Module completion checklist

Objective	Complete
Define web scraping	✔
Decide when to use web scraping	✔
Explain HTML tags	✔
Use BeautifulSoup package to scrape websites with tables	✔
Iterate over a table to extract data	✔
Iterate over multiple websites	✔
Perform exploratory data analysis with the scraped data	✔

Workshop: next steps!

- **Today's after class workshop**
- Workshops are to be completed outside of class and emailed to the instructor by the beginning of class tomorrow
- Make sure to **comment your code** so that it is easy for others to understand what you are doing
- This is an exploratory exercise to **get you comfortable with the content** we discussed today
- Workshop objectives:
  - Find an website with interesting table data that you can scrap
  - Scrape the website using the `BeautifulSoup` package
  - Store your data in a `pandas` dataframe
  - Find some interesting insights from the data you scraped

This completes our module

Congratulations!