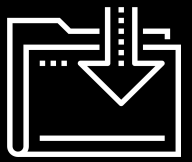
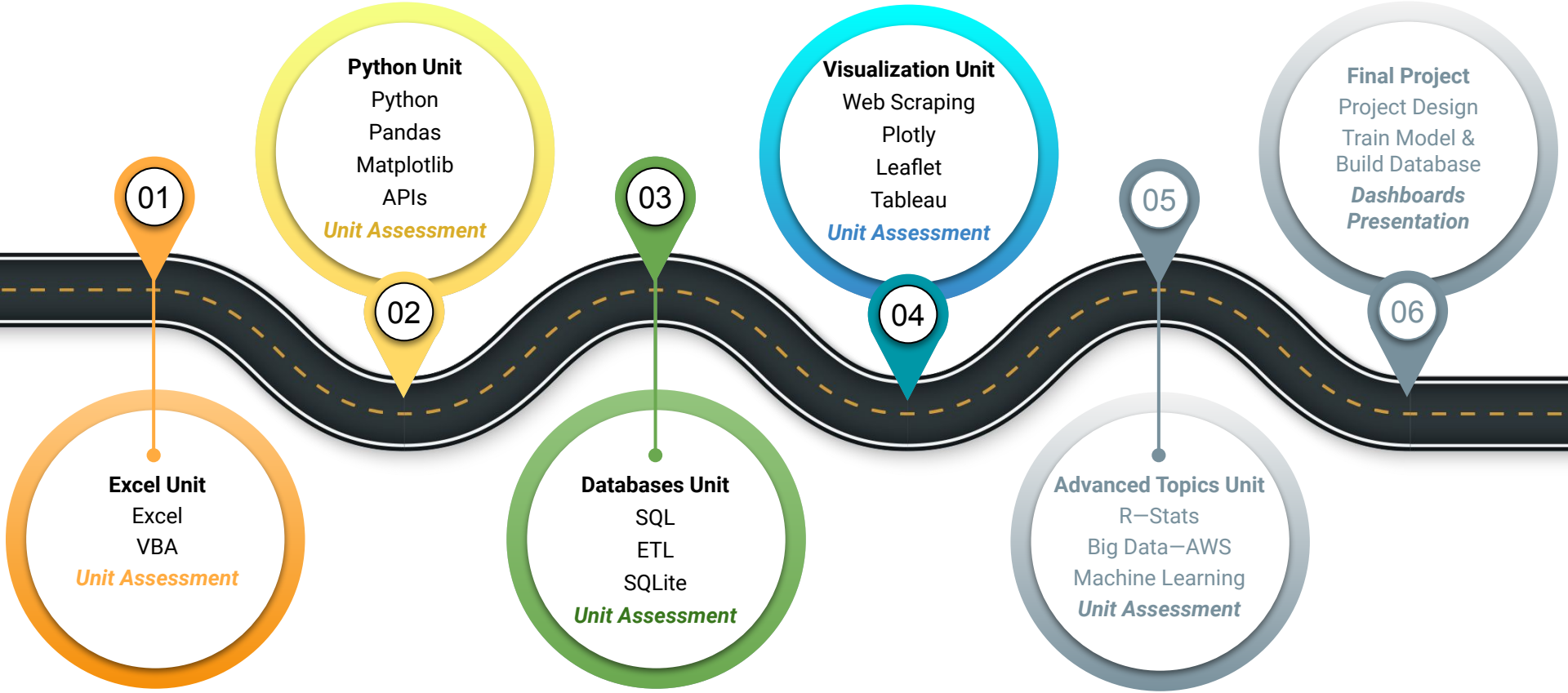
Web Scraping With HTML & CSS

Data Boot Camp

Lesson 10.1

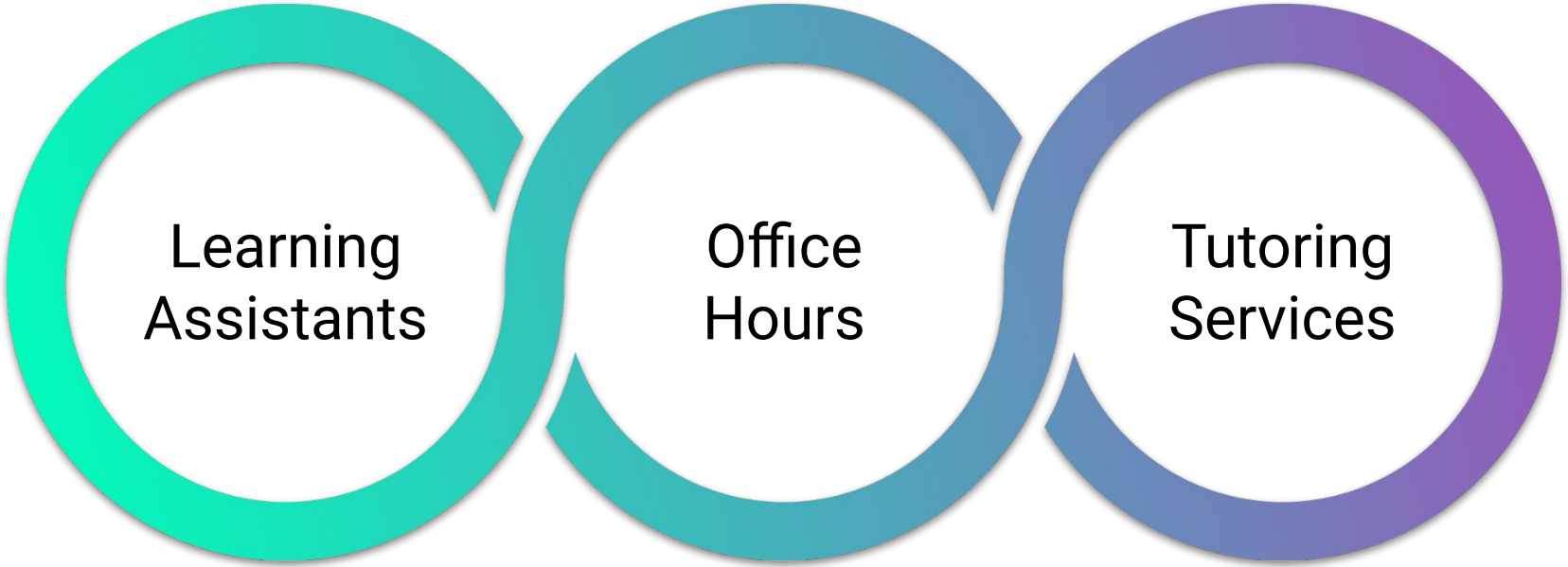


The Big Picture



Bootcamp Pointers

If you're struggling with this module, remember that you have the following resources to help you:



Module 10

This Week:
Web Scraping with HTML & CSS

This Week: Web Scraping with HTML & CSS

By the end of this week, you'll know how to:

- Recognize and use HTML elements, and class and id attributes, to identify content for web scraping
- Use Beautiful Soup and Splinter to automate a web browser and perform a web scrape
- Create a MongoDB database to store data from the web scrape
- Create a web application with Flask to display the data from the web scrape
- Create an HTML/CSS portfolio to showcase projects
- Use Bootstrap components to polish and customize the portfolio

This Week's Challenge

Using the skills learned throughout the week, create a web app that displays scraped images of Mars' hemispheres, complete with titles.

Career Connection

How will you use this module's content in your career?

Module 10

How to Succeed This Week

Quick Tip for Success:

We’re introducing a lot of new material at once, and that can be a little overwhelming. Trust in the process!

Module 10

Today’s Agenda

Today's Agenda

By completing today's activities, you'll learn the following skills:

01 Scrape with Beautiful Soup

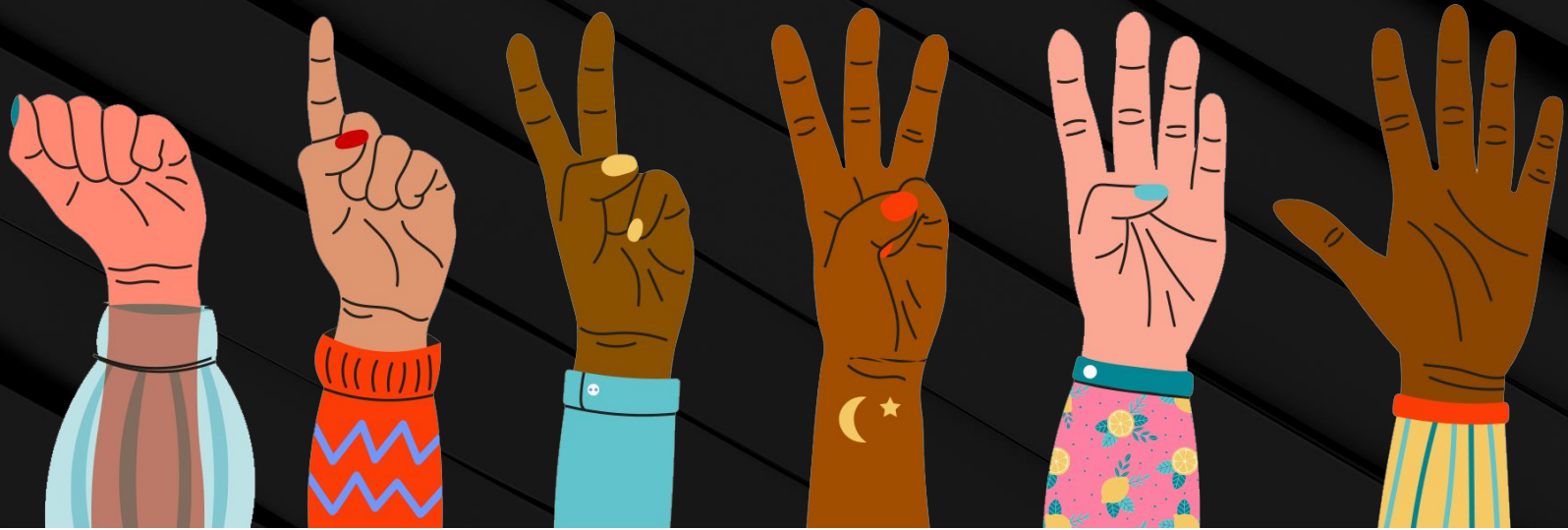
02 Store data in a Mongo database

03 Automate scraping with Splinter

Make sure you've downloaded any relevant class files!

FIST TO FIVE:

How comfortable do you feel with this topic?

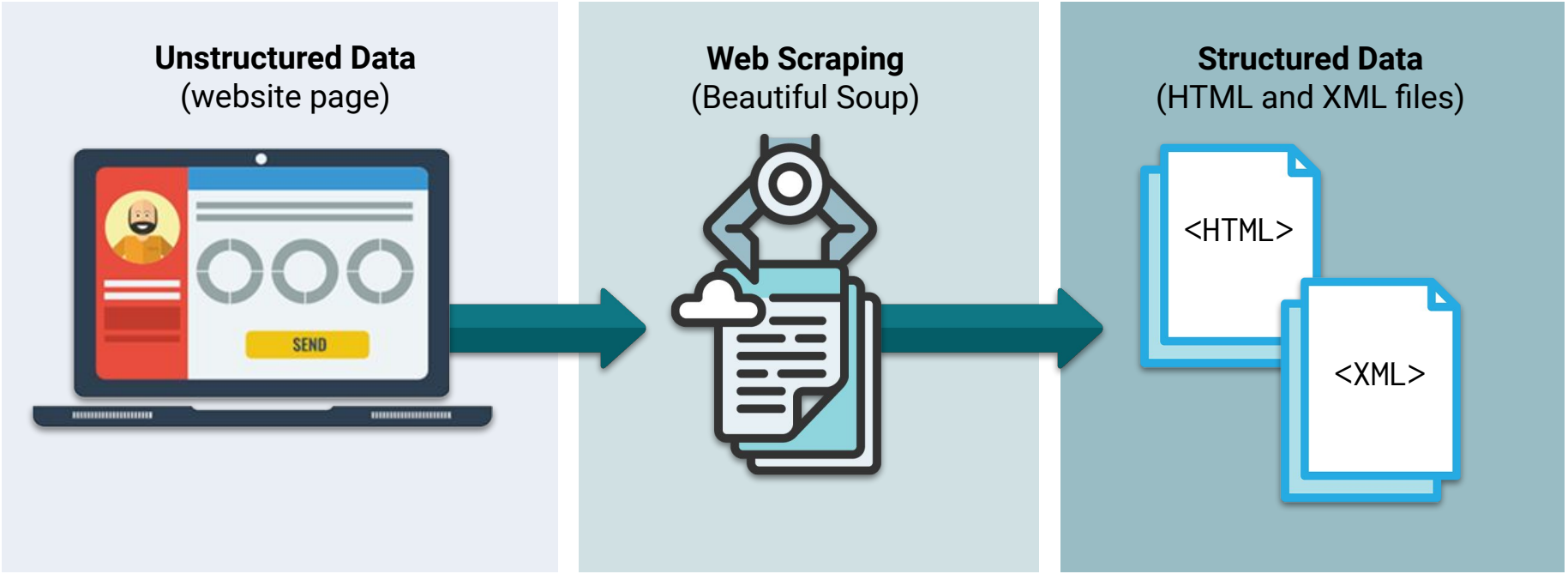


Introduction to Beautiful Soup

Beautiful Soup is a Python library created to pull data out from HTML and XML files.

Beautiful Soup

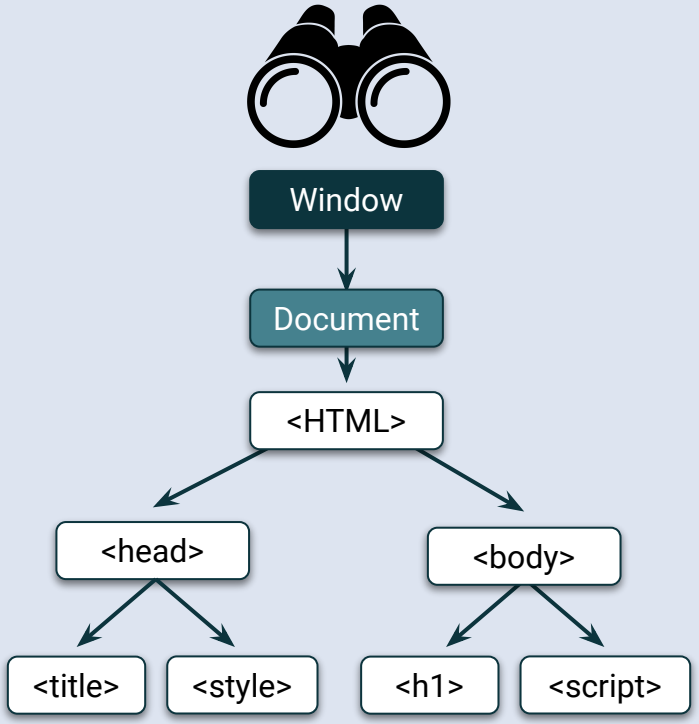
Beautiful Soup is a Python library created to pull data out from HTML and XML files.



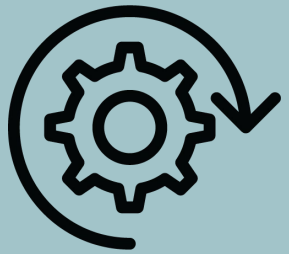
DOM, or Document Object Model, is a tree whose structure is defined by the nesting of tags.

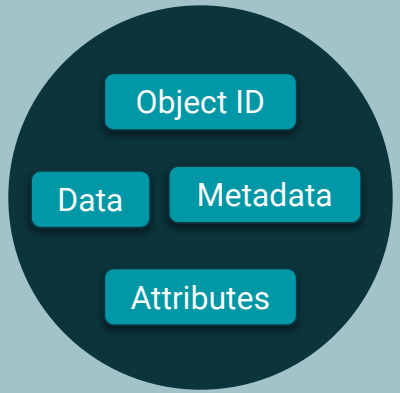
Beautiful Soup

Beautiful Soup looks through the tree

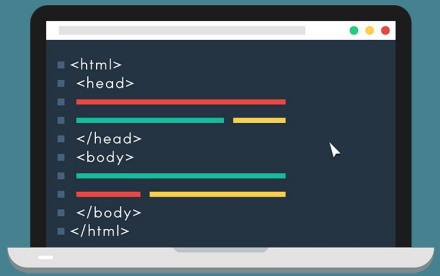


Beautiful Soup converts it to a specialized object





The specialized object can traverse and search the HTML for attributes, text, etc.



Time to Code

Run `pip install bs4` within the terminal

Suggested Time:

5 minutes

Instructor Demonstration

Beautiful Soup

Questions?

Introduction to Beautiful Soup

Beautiful Soup - The Basics

➔ `from bs4 import BeautifulSoup as bs`

```
In [1]: # import dependency
        from bs4 import BeautifulSoup as bs
```

➔ `bs(html_string, 'html.parser')` - This line of code creates a Beautiful Soup object. The object is assigned to a variable, in this case `soup`.

```
In [2]: html_string = """
        <html>
        <head>
        <title>
        A Simple HTML Document
        </title>
        </head>
        <body>
        <p>This is a very simple HTML document</p>
        <p>It only has two paragraphs</p>
        </body>
        </html>
        """
```

```
In [3]: # create a BS object
        soup = bs(html_string, 'html.parser')
```

Introduction to Beautiful Soup

Beautiful Soup - The Basics

- ➔ `type()` - This method will return the type of …
- ➔ `prettify()` - This method returns a more readable way of the ...

```
In [4]: type(soup)

Out[4]: bs4.BeautifulSoup

In [5]: print(soup.prettify())

<html>
 <head>
  <title>
   A Simple HTML Document
  </title>
 </head>
 <body>
  <p>
   This is a very simple HTML document
  </p>
  <p>
   It only has two paragraphs
  </p>
 </body>
</html>
```

Introduction to Beautiful Soup

Beautiful Soup - The Basics

- ➔ `soup.title` - You can extract only the title from a BS object. This, in particular, will return the whole 'title' element, including the tags.
- ➔ `.text` - Adding this to the end of the call returns the text contained within the title element. Note that this will return the string surrounded by white spaces.
- ➔ `.strip()` - If you desire to further clean the return, this can be added to the end of the chain.

```
In [7]: soup.title

Out[7]: <title>
        A Simple HTML Document
        </title>

In [8]: soup.title.text

Out[8]: '\nA Simple HTML Document\n'

In [9]: soup.title.text.strip()

Out[9]: 'A Simple HTML Document'
```

Introduction to Beautiful Soup

Beautiful Soup - The Basics

- ➔ `soup.body` - This will return the entire body of the HTML file, and adding the below will return
- ➔ `.text`
- ➔ `.strip()`
- ➔ `.p.text`

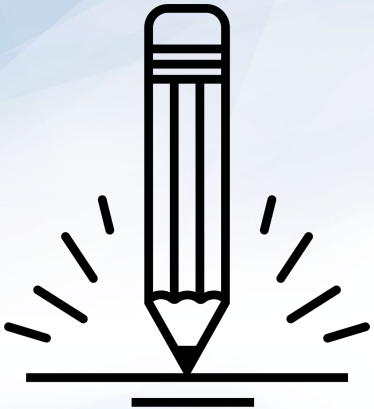
```
In [10]: soup.body

Out[10]: <body>
         <p>This is a very simple HTML document</p>
         <p>It only has two paragraphs</p>
         </body>

In [11]: soup.body.text

Out[11]: '\nThis is a very simple HTML document\nIt only has two paragraphs\n'

In [14]: soup.body.text.strip()

Out[14]: 'This is a very simple HTML document\nIt only has two paragraphs'

In [15]: soup.body.p.text

Out[15]: 'This is a very simple HTML document'
```

Introduction to Beautiful Soup

Beautiful Soup - The Basics

- ➔ `find_all(<element>)` - method returns a list containing all of the HTML elements of a specific type.
- ➔ `('p')` - this element will return all the paragraphs.
- ➔ `['an index number]` - you can also return a specific paragraph using the index number.

```
In [17]: soup.body.find_all('p')

Out[17]: [<p>This is a very simple HTML document</p>, <p>It only has two paragraph
         s</p>]

In [18]: soup.body.find_all('p')[0]

Out[18]: <p>This is a very simple HTML document</p>

In [19]: soup.body.find_all('p')[1]

Out[19]: <p>It only has two paragraphs</p>
```



Activity: CNN Soup

In this activity, you will take your first step in web scraping by taking an external HTML file, parsing it, and then printing out specific elements to the console.

Suggested Time:
15 Minutes

Activity: CNN Soup

Instructions	Hint	Bonus
Believe it or not, CNN's website for 1996: Year in Review is still alive on the web! We have, however, stored the HTML document as a string in your starter file. Your task, should you accept it (and you should), is to use Beautiful Soup to scrape and print the following pieces of information: • The title of the webpage • All paragraph text on the page • The top 10 headlines for the year. This last one is a bit tricky and may not come out perfectly!	For the third task in this activity, you will need a means of filtering the data … perhaps over multiple iterations … with a loop … HINT, HINT! 	If you finish early, head over to the Beautiful Soup documentation to read up on accessing attributes and navigating the DOM.

Let's Review

Website Scraping

So far, we have only parsed HTML strings with Beautiful Soup. Now it is time to scrape a live website!

Website Scraping

The goal will be to scrape the results for an item, such as a guitar, on Craigslist and return each relevant listing's title, price, and URL.

Instructor Demonstration

Craig’s Wishlist

Craig's Wishlist

Beautiful Soup - Live Website!

- So far, you have only parsed HTML strings with Beautiful Soup. Now it is time to scrape a live website!

➔ Look at the code below. Notice anything different from what we have learned so far?

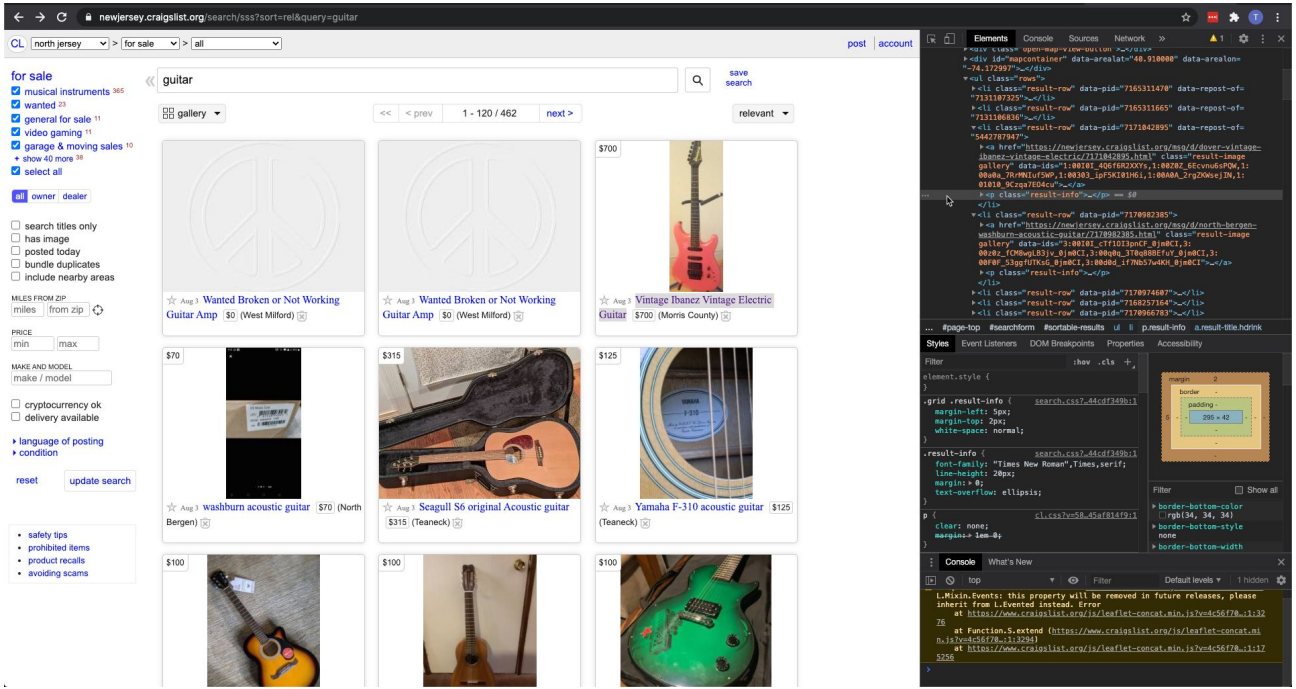
```
In [2]: # import dependency
        from bs4 import BeautifulSoup as bs
        import requests

In [3]: # URL of page to be scraped
        url = 'https://newjersey.craigslist.org/search/sss?sort=rel&query=guitar'

In [4]: # Retrieve page with the requests module
        response = requests.get(url)

In [6]: # Create BeautifulSoup object; parse with 'html.parser'
        soup = bs(response.text, 'html.parser')

In [7]: # Examine the results, then determine element that contains sought info
        print(soup.prettify())
```

```
<!DOCTYPE html>
<html class="no-js">
 <head>
  <title>
   north jersey for sale "guitar" - craigslist
  </title>
  <meta content='north jersey for sale "guitar" - craigslist' name="descr
iption"/>
  <meta content="IE=Edge" http-equiv="X-UA-Compatible">
   <link href="https://newjersey.craigslist.org/search/sss" rel="canonica
l"/>
   <link href="https://newjersey.craigslist.org/search/sss?format=rss&am
p;query=guitar&amp;sort=rel" rel="alternate" title='RSS feed for craigsli
st | north jersey for sale "guitar" - craigslist' type="application/rss+x
ml"/>
   <link href="https://newjersey.craigslist.org/search/sss?s=120&amp;quer
y=guitar&amp;sort=rel" rel="next"/>
   <meta content="width=device-width,initial-scale=1" name="viewport"/>
```

Craig's Wishlist

Beautiful Soup - Live Website!

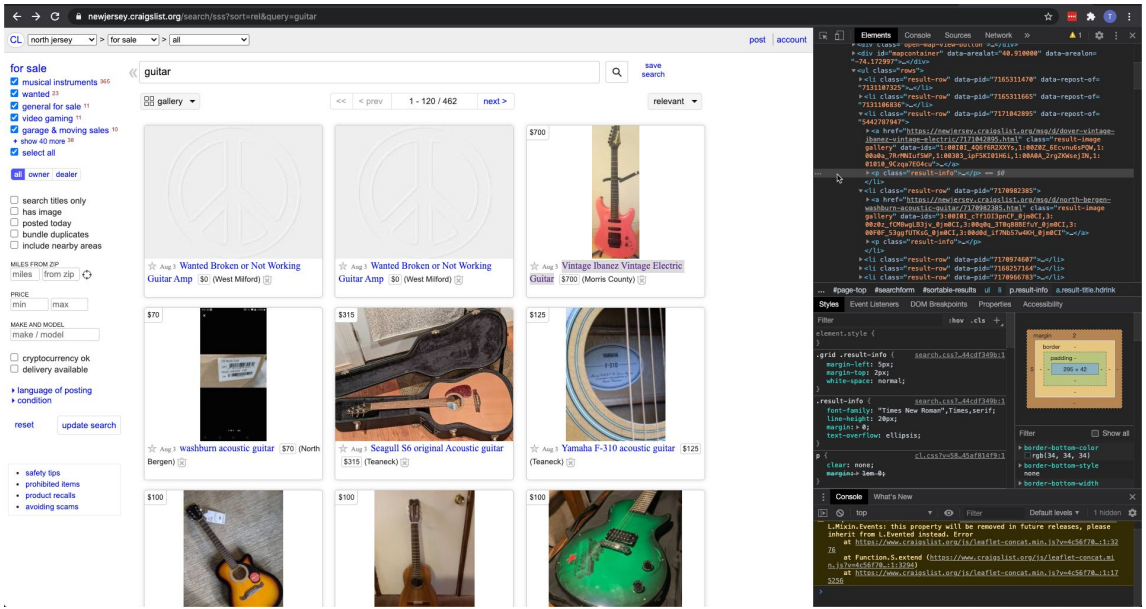
➔ Another way to look into the HTML is to view the page in a browser and open up its source within the inspector.



Craig's Wishlist

BeautifulSoup - Live Website!

- ➔ The listing price can also be found to exist within a `span` element whose class is `result-price`.
- ➔ To retrieve the content of these elements, you can call the `find_all('li', class_='result-row')` method on the `soup` object.



Craig's Wishlist

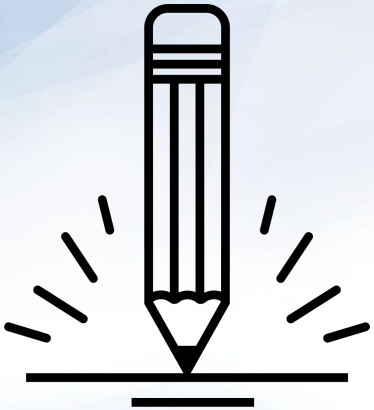
Beautiful Soup - Live Website!

- ➔ By iterating through the listings, specific information can then be pulled from the Beautiful Soup object.
- ➔ Each listing's title can, therefore, be gathered using `result.find('a', class_='result-title')`, their prices collected with `result.a.span.text`, and their links retrieved by accessing the "href" attribute of each listing using `result.a["href"]`.

```
In [12]: for result in results:
             try:
                 title = result.find('a', class_="result-title").text
                 price = result.a.span.text
                 link = result.a['href']

                 if (title and price and link):
                     print('--------------')
                     print(title)
                     print(price)
                     print(link)
             except AttributeError as e:
                     print(e)
```

```
'NoneType' object has no attribute 'text'
'NoneType' object has no attribute 'text'
--------------
Vintage Ibanez Vintage Electric Guitar
$700
https://newjersey.craigslist.org/msg/d/dover-vintage-ibanez-vintage-electric/7171042895.html
--------------
washburn acoustic guitar
$70
https://newjersey.craigslist.org/msg/d/north-bergen-washburn-acoustic-guitar/7170982385.html
--------------
Seagull S6 original Acoustic guitar
$315
https://newjersey.craigslist.org/msg/d/teaneck-seagull-s6-original-acoustic/7170974607.html
--------------
Yamaha F-310 acoustic guitar
$125
https://newjersey.craigslist.org/msg/d/matawan-yamaha-310-acoustic-guitar/7168257164.html
--------------
```

Questions?



Activity: Reddit Scraper

In this activity, you will scrape the Python Reddit for potentially interesting content. You will also have to filter for threads with 20 or more comments in them.

Suggested Time:
20 minutes

Reddit Scraper Instructions

In this activity, you will scrape the Python Reddit using Beautiful Soup. Scrape only threads that have 20 or more comments, and then print each thread's title, number of comments, and URL.

```
----------------

Doing conditionals
Comments: 258
https://www.reddit.com/r/ProgrammerHumor/comments/7pw5qk/doing_conditionals/

----------------

Perfect date
Comments: 58
https://www.reddit.com/r/ProgrammerHumor/comments/7pyyl2/perfect_date/

----------------

The truth about java.
Comments: 61
https://www.reddit.com/r/ProgrammerHumor/comments/7pxod4/the_truth_about_java/

----------------
```

Let's Review

Questions?

Mongo Scraping

Suggested Time:

5 minutes

Mongo Craig

Translate the results of a web scraper to a MongoDB database

➔ For this particular application, we are going to use the `lxml` parser instead of `html.parser`. Check out the Beautiful Soup documentation and check an informative table on the various parsers available to BS.

In [6]:
```
# Dependencies
from bs4 import BeautifulSoup
import requests
import pymongo
```

In [7]:
```
# Initialize PyMongo to work with MongoDBs
conn = 'mongodb://localhost:27017'
client = pymongo.MongoClient(conn)
```

In [8]:
```
# Define database and collection
db = client.craigslist_db
collection = db.items
```

In [9]:
```
# URL of page to be scraped
url = 'https://newjersey.craigslist.org/search/sss?sort=rel&query=guitar'

# Retrieve page with the requests module
response = requests.get(url)
# Create BeautifulSoup object; parse with 'lxml'
soup = BeautifulSoup(response.text, 'lxml')
```

Mongo Craig

Translate the results of a web scraper to a MongoDB database

➔

In [5]:
```
# Examine the results, then determine element that contains sought info
# results are returned as an iterable list
results = soup.find_all('li', class_='result-row')

# Loop through returned results
for result in results:
    # Error handling
    try:
        # Identify and return title of listing
        title = result.find('a', class_='result-title').text
        # Identify and return price of listing
        price = result.a.span.text
        # Identify and return link to listing
        link = result.a['href']

        # Run only if title, price, and link are available
        if (title and price and link):
            # Print results
            print('-------------')
            print(title)
            print(price)
            print(link)

            # Dictionary to be inserted as a MongoDB document
            post = {
                'title': title,
                'price': price,
                'url': link
            }

            collection.insert_one(post)

    except Exception as e:
        print(e)
```

```
-------------
Aucostic Guitar (Savanah sgd 10bk)
$60
https://newjersey.craigslist.org/for/d/aucostic-guitar-savanah-sgd/6564697500.html
-------------
Aucostic Guitar (Savanah sgd 10bk)
$60
https://newjersey.craigslist.org/ele/d/aucostic-guitar-savanah-sgd/6564699565.html
-------------
3/4 Size Acoustic Guitar Gig Bag
$10
https://newjersey.craigslist.org/msg/d/3-4-size-acoustic-guitar-gig/6566731315.html
-------------
Attn Guitar Players! - Fuzz Face Germanium Mini Distortion Pedal
$95
https://newjersey.craigslist.org/msg/d/attn-guitar-players-fuzz-face/6566698424.html
-------------
Beautiful Custom Left Handed Carvin Guitar w/synth pickup $1000 off!
$1799
https://newjersey.craigslist.org/msg/d/beautiful-custom-left-handed/6543383668.html
-------------
```

Mongo Craig

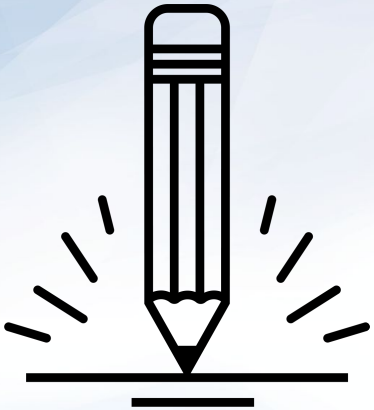
Translate the results of a web scraper to a MongoDB database

➔ After the application has pushed all of the scraped data into the Mongo database, this can be verified by querying the database one and printing out all of the results to the console.

```
In [6]: # Display items in MongoDB collection
        listings = db.items.find()

        for listing in listings:
            print(listing)
```

```
{'_id': ObjectId('5ada79c6ee61f93d19698abb'), 'title': 'Aucostic Guitar (Savanah sgd 10bk)', 'price': '$60', 'url':
'https://newjersey.craigslist.org/for/d/aucostic-guitar-savanah-sgd/6564697500.html'}
{'_id': ObjectId('5ada79c7ee61f93d19698abc'), 'title': 'Aucostic Guitar (Savanah sgd 10bk)', 'price': '$60', 'url':
'https://newjersey.craigslist.org/ele/d/aucostic-guitar-savanah-sgd/6564699565.html'}
{'_id': ObjectId('5ada79c7ee61f93d19698abd'), 'title': '3/4 Size Acoustic Guitar Gig Bag', 'price': '$10', 'url': 'ht
tps://newjersey.craigslist.org/msg/d/3-4-size-acoustic-guitar-gig/6566731315.html'}
{'_id': ObjectId('5ada79c7ee61f93d19698abe'), 'title': 'Attn Guitar Players! - Fuzz Face Germanium Mini Distortion Pe
dal', 'price': '$95', 'url': 'https://newjersey.craigslist.org/msg/d/attn-guitar-players-fuzz-face/6566698424.html'}
{'_id': ObjectId('5ada79c7ee61f93d19698abf'), 'title': 'Beautiful Custom Left Handed Carvin Guitar w/synth pickup $10
00 off!', 'price': '$1799', 'url': 'https://newjersey.craigslist.org/msg/d/beautiful-custom-left-handed/6543383668.ht
ml'}
{'_id': ObjectId('5ada79c7ee61f93d19698ac0'), 'title': 'Peavey Special 112 Guitar Amp 80s / 90s - 160W Made in USA',
'price': '$140', 'url': 'https://newjersey.craigslist.org/msg/d/peavey-special-112-guitar-amp/6549026986.html'}
{'_id': ObjectId('5ada79c7ee61f93d19698ac1'), 'title': 'flea market guitar frenzy 20 40 100 150 GUITARS', 'price':
'$1', 'url': 'https://newjersey.craigslist.org/for/d/flea-market-guitar/6551234119.html'}
{'_id': ObjectId('5ada79c7ee61f93d19698ac2'), 'title': 'Ibanez IBZ10G Guitar Amplifier', 'price': '$49', 'url': 'http
s://newjersey.craigslist.org/msg/d/ibanez-ibz10g-guitar-amplifier/6566608081.html'}
{'_id': ObjectId('5ada79c7ee61f93d19698ac3'), 'title': 'Gibson Acoustic Electric Guitar', 'price': '$1500', 'url': 'h
ttps://newjersey.craigslist.org/msg/d/gibson-acoustic-electric/6566602165.html'}
```

Questions?



Activity: Hockey Headers

In this activity, you will scrape the news page of the NHL website for articles and then post the title/header of each code block to the best of your ability.

Suggested Time:
15 Minutes

Hockey Headers Instructions

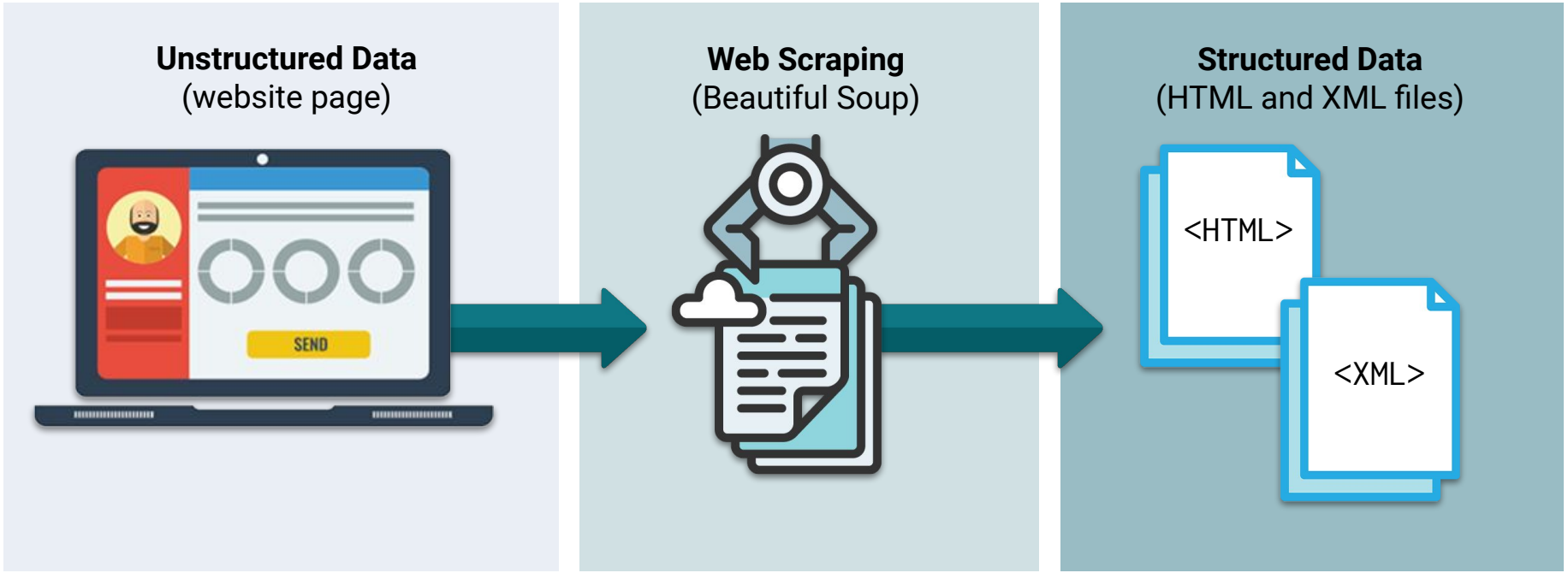
- Teamwork! Speed! Mental and physical toughness! Passion! Excitement! Unpredictable matchups down to the wire! What could be better? While these terms could easily be applied to a data science hackathon, we're talking about the magnificent sport of hockey.
- Your assignment is to scrape the articles on the news page of the NHL website—which is frequently updated—and then post the results of your scraping to MongoDB.
- Use Beautiful Soup and requests to scrape the header and subheader of each article on the front page.
- Post the above information as a MongoDB document and then print all of the documents on the database to the console.
- In addition to the above, post the date of the article's publication as well.

Let's Review

Splinter

Splinter

Up to this point, we have used Beautiful Soup to scrape one static page at a time.

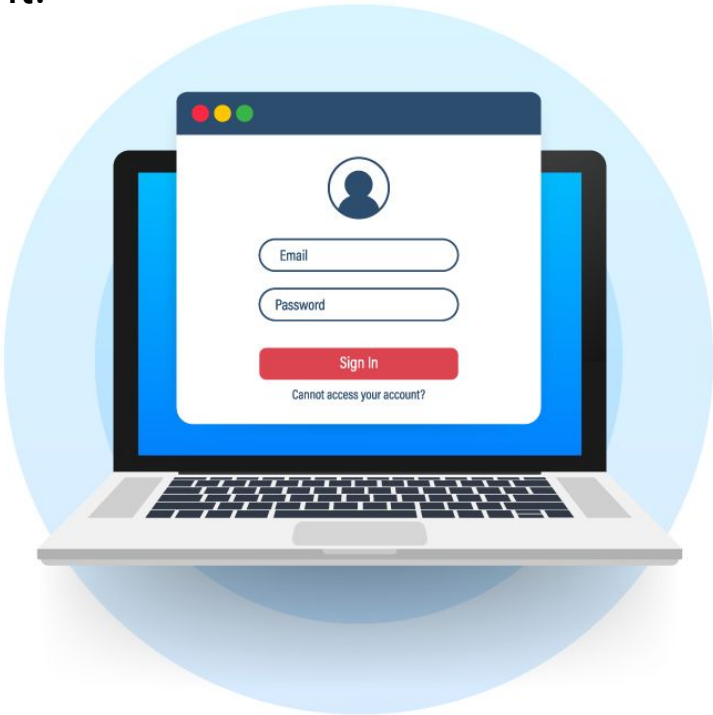


Splinter

Developers often can only access interesting parts of a website after engaging in some kind of interaction with it.

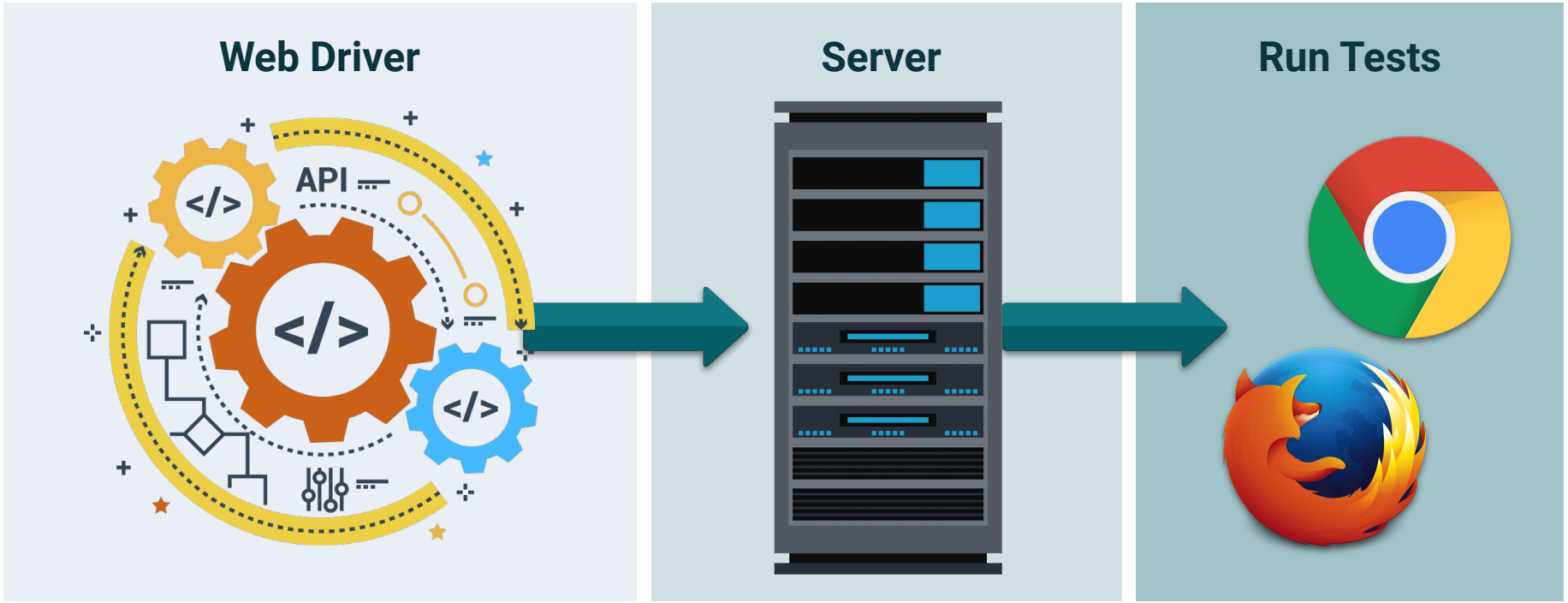
Typically, these interactions are pretty easy to automate:

- Logging in
- Filling out and submitting forms



Splinter

When the data is "buried" behind such dynamic interactions, a web driver can be used to write scripts for the browser!



Splinter allows developers to simulate user interactions programmatically and scrape multiple pages along the way.

Install the web-driver manager

```
pip install webdriver_manager
```

Suggested Time:

2 minutes

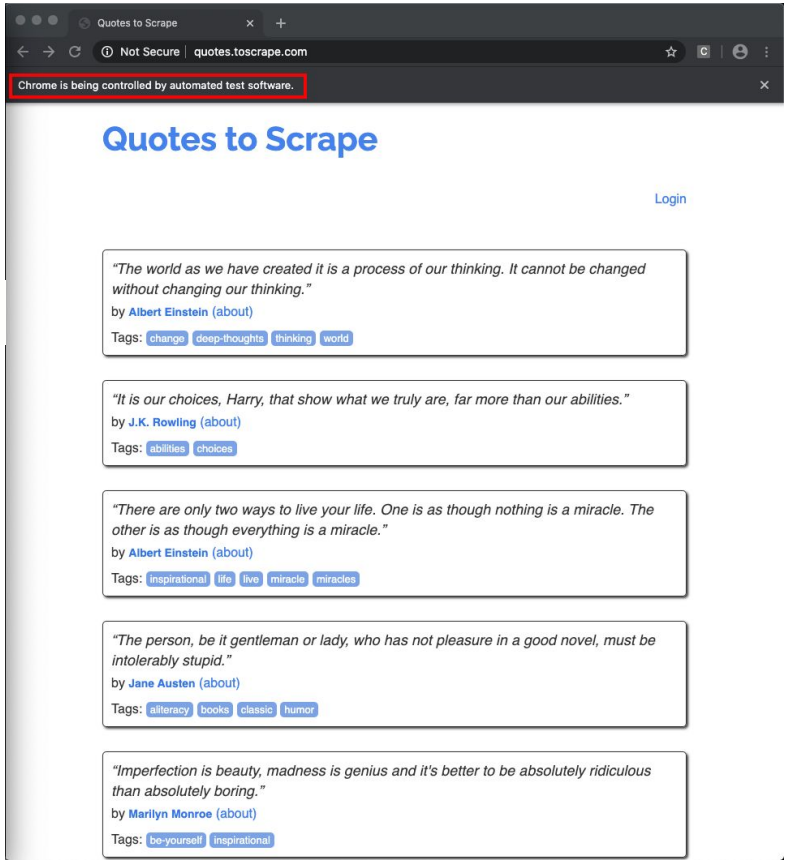
Instructor Demonstration

Introduction to Splinter

Introduction to Splinter

```
from splinter import Browser
from bs4 import BeautifulSoup
from webdriver_manager.chrome import ChromeDriverManager
```

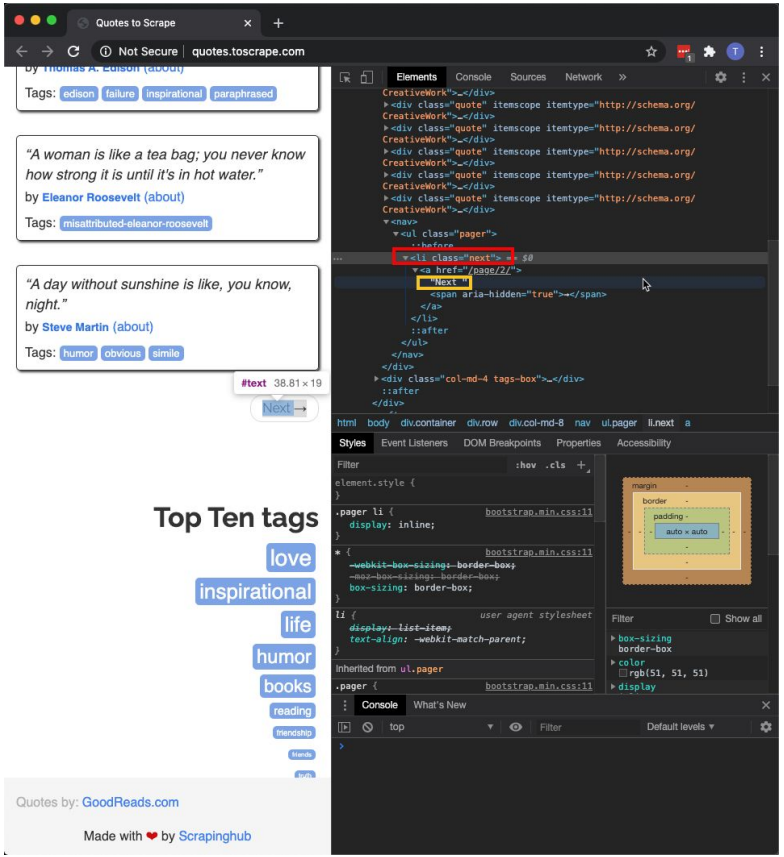
```
# Setup splinter
executable_path = {'executable_path': ChromeDriverManager().install()}
browser = Browser('chrome', **executable_path, headless=False)
```



Note: You might need to install splinter using `pip install splinter`

Introduction to Splinter

➔ Open the Chrome inspector to identify the element that the application will need to click.

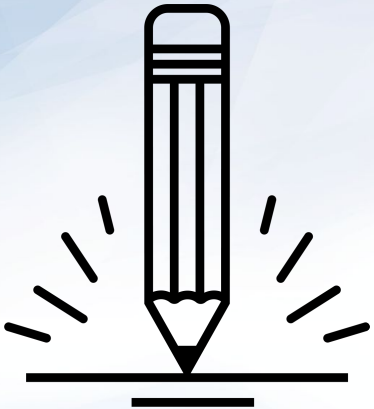


Introduction to Splinter

➔ The `click_link_by_partial_text()` method.

```
In [18]: for x in range(1, 6):

             html = browser.html
             soup = BeautifulSoup(html, 'html.parser')

             quotes = soup.find_all('span', class_='text')

             for quote in quotes:
                 print('page:', x, '-------------')
                 print(quote.text)

             browser.links.find_by_partial_text('Next')
```

```
page: 1 -------------
“The world as we have created it is a process of our thinking. It cannot be changed without changing our thinking.”
page: 1 -------------
“It is our choices, Harry, that show what we truly are, far more than our abilities.”
page: 1 -------------
“There are only two ways to live your life. One is as though nothing is a miracle. The other is as though everything
is a miracle.”
page: 1 -------------
“The person, be it gentleman or lady, who has not pleasure in a good novel, must be intolerably stupid.”
page: 1 -------------
“Imperfection is beauty, madness is genius and it's better to be absolutely ridiculous than absolutely boring.”
page: 1 -------------
“Try not to become a man of success. Rather become a man of value.”
page: 1 -------------
“It is better to be hated for what you are than to be loved for what you are not.”
page: 1 -------------
“I have not failed. I've just found 10,000 ways that won't work.”
page: 1 -------------
“A woman is like a tea bag; you never know how strong it is until it's in hot water.”
page: 1 -------------
```



Activity: Bookscraper

In this activity, you will practice your web-scraping skills on a site similar to the one shown in the instructor demonstration.

Suggested Time:
15 minutes

Bookscraper Instructions

Go to http://books.toscrape.com/

01

Scrape the titles and the URLs to all books on this fictional online bookstore.

02

Display the results in console.

03

Bonus: try scraping all books by category.

Good luck!

Let's Review

Questions?