

# Data 2.0 Hackathon

## Web Scraping Workshop

Sophie Janaskie

Yale School of Forestry and Environmental Studies

March 17, 2018

# What is web scraping?

- A method of extracting information from websites programmatically
- A way to convert semi-structured web data into usable, structured data that we can analyze

```
# convert contents to nested data structure
soup = BeautifulSoup(content, "xml")

# parse HTML
date = soup.find('div', attrs={'class':'daily-history-select'}).h2.text
meantemp = soup.find_all('tr')[2].find_all('td')[1].find(attrs={"class":"wx-value"}).text
maxtemp = soup.find_all('tr')[3].find_all('td')[1].find(attrs={"class":"wx-value"}).text
mintemp = soup.find_all('tr')[4].find_all('td')[1].find(attrs={"class":"wx-value"}).text
precip = soup.find_all('tr')[13].find_all('td')[1].find(attrs={"class":"wx-value"}).text
wind = soup.find_all('tr')[17].find_all('td')[1].find(attrs={"class":"wx-value"}).text

# write results
with open("WeatherScrapeOnePage.csv", "w") as file:
    csv_writer = csv.writer(file)
    csv_writer.writerow(['Date', 'Mean Temp (C)', 'Max Temp (C)', 'Min Temp (C)', 'Precipitation (in)', 'Wind Speed (m/s)'])
    csv_writer.writerow([dateString, meantemp, maxtemp, mintemp, precip, wind])
```



```
import requests
from bs4 import BeautifulSoup

url = "https://www.wunderground.com/we

def load(url):
    attempts = 5
    for i in range(attempts):
        try:
            response = requests.get(url)
            return response.content
        except requests.HTTPError:
            if i+1 == attempts:
                raise
            else:
                time.sleep(60)

content = load(url)
soup = BeautifulSoup(content, "html.parser")

# move on to parsing!
```

# Why is it useful, and when to use it?

- No API or readily available data
- Programmatically pull data from the web to save time from manual collection
- Useful for projects and capstones!

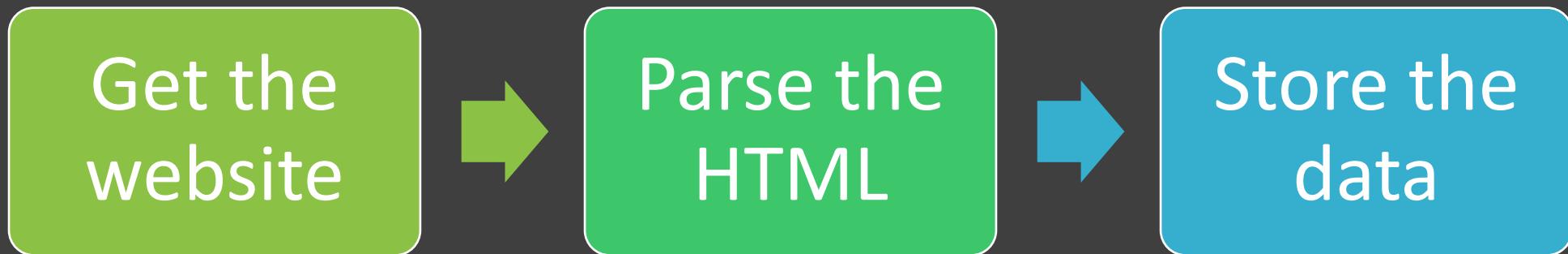
*“The Internet is one giant API – with  
a really terrible interface.”*

- Ryan Mitchell, author of Web Scraping with Python

# Agenda

1. Scraping workflow
2. HTML Basics
3. Pulling the web page with Requests
4. Parsing the HTML with BeautifulSoup
5. Storing your data
6. Other considerations

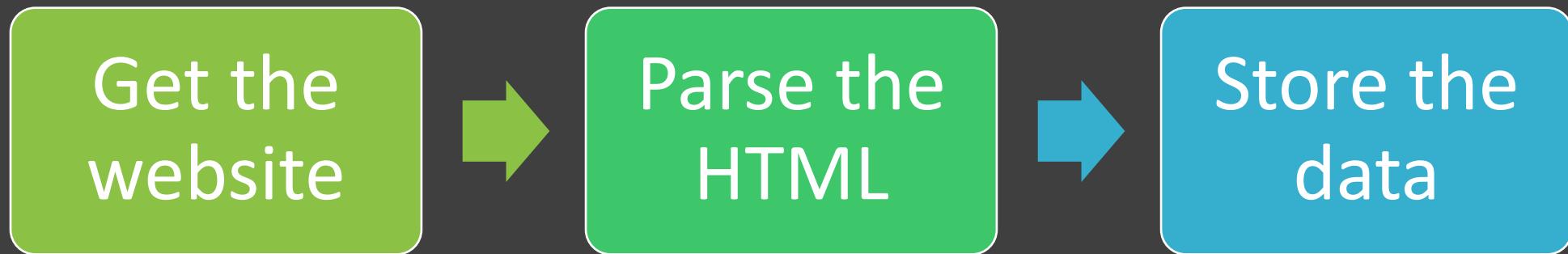
# Web Scraping Workflow



# Web Scraping Workflow



# Web Scraping Workflow



*Requests*

*BeautifulSoup*

*csv*

```
import requests  
from bs4 import BeautifulSoup  
import csv
```

# HTML Basics

# Web Page Structure

## Rendered HTML

The screenshot shows the Weather Underground homepage with a navigation bar for Maps & Radar, Severe Weather, News & Blogs, Photos & Video, and Action. Below the navigation are local weather cards for San Francisco, CA (9.7°C Partly Cloudy), Chicago, IL (-5.5°C Mostly Cloudy), Boston, MA (-0.5°C Clear), and Houston, TX (13.5°C Partly Cloudy). The main content area displays the weather history for WSAP (Singapore Payalebar) on Wednesday, March 14, 2018. The page includes a date selector for March 14, 2018, and tabs for Forecast, History, Calendar, Rain / Snow, and Health. The History tab is selected, showing a table for Temperature with rows for Mean Temperature (30 °C), Max Temperature (33 °C), Min Temperature (26 °C), Cooling Degree Days (20), and Growing Degree Days (34 (Base 50)).

## Source Code

```
import requests
from bs4 import BeautifulSoup
import csv, time

# initial GET request
url = "https://www.wunderground.com/history/airport/WSAP/2017/1/1/DailyHistory.html"
response = requests.get(url)
content = response.content

# convert contents to nested data structure
soup = BeautifulSoup(content, "lxml")

# parse HTML
date = soup.find('div', attrs={'class':'daily-history-select'}).h2.text
meantemp = soup.find_all('tr')[2].find_all('td')[1].find(attrs={"class":"wx-value"}).text
maxtemp = soup.find_all('tr')[3].find_all('td')[1].find(attrs={"class":"wx-value"}).text
mintemp = soup.find_all('tr')[4].find_all('td')[1].find(attrs={"class":"wx-value"}).text
precip = soup.find_all('tr')[13].find_all('td')[1].find(attrs={"class":"wx-value"}).text
wind = soup.find_all('tr')[17].find_all('td')[1].find(attrs={"class":"wx-value"}).text

# write results
with open("WeatherScrapeOnePage.csv", "w") as file:
    csv_writer = csv.writer(file)
    csv_writer.writerow(['Date', 'Mean Temp (C)', 'Max Temp (C)', 'Min Temp (C)', 'Precipitation (in)', 'Wind'])
    csv_writer.writerow([dateString, meantemp, maxtemp, mintemp, precip, wind])

print("Scrape complete!")
```

# HTML/CSS

- *Hyper Text Markup Language provides structure for web pages.*
- *CSS provides styling and design.*

```
<!DOCTYPE html>
<html>
<body>

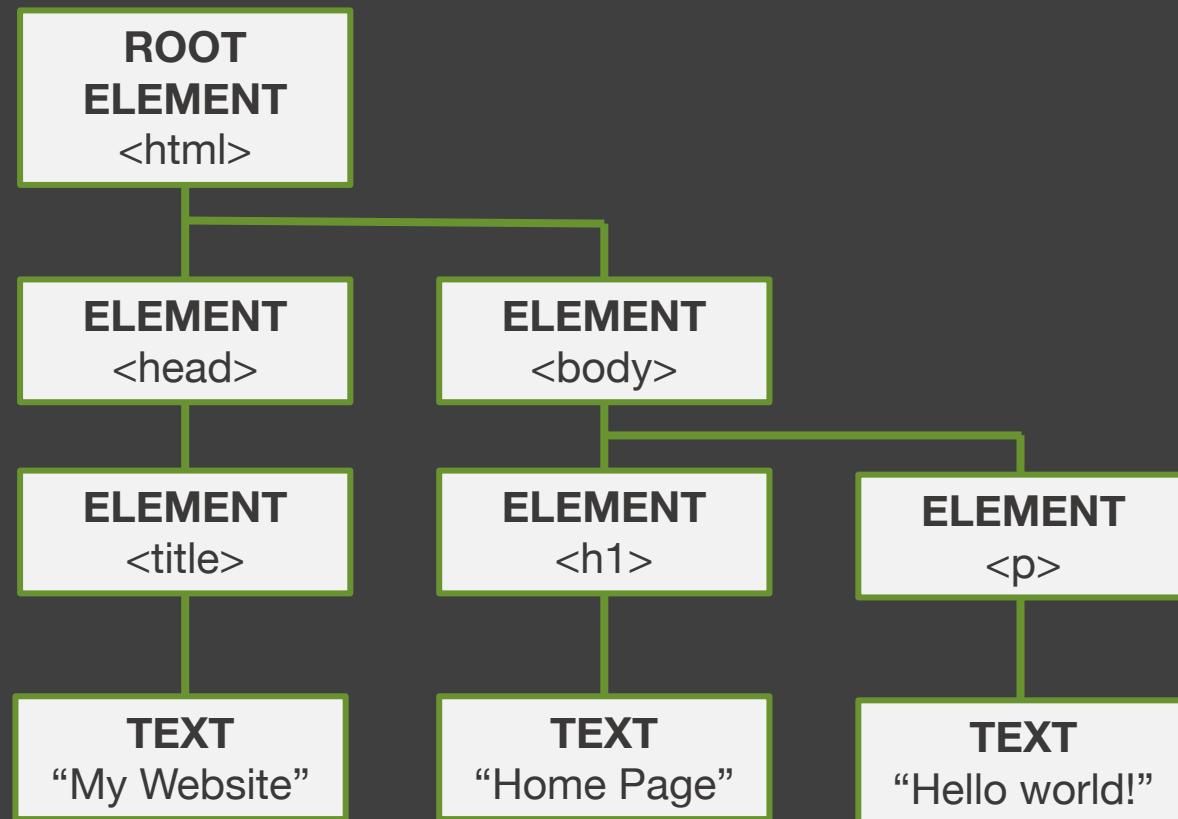
<h1>This is a heading</h1>
<h2>This is a smaller heading</h2>
<a href="https://www.google.com">This is a link</a>


<ul>
    <li>This</li>
    <li>is an</li>
    <li>unordered list</li>
</ul>

</body>
</html>
```

# Document Object Model

- *The DOM creates an object-oriented model of the information to enable easier manipulation.*



# Get the website

*Using the Requests library*

# Requests library

- Sending an HTTP request

```
url = "https://www.wunderground.com/history/airport/WSAP/" + startDate + "/DailyHistory.html"  
response = requests.get(url)
```

- Access the response's body of content

```
content = response.content
```

- Check to make sure the request was successful (200 code)

```
status = response.status_code
```

# Parse the HTML

*Using the BeautifulSoup library*

# BeautifulSoup

## Make the Soup

```
soup = BeautifulSoup(content, "lxml")
```

## Navigate the Tree

- Using .title, .contents, .children, .parent, .next\_sibling, .text etc.

## Search the Tree

- Using find( ), find\_all( ), select( ), etc.

## Modify the Tree

- Using decompose( ), insert( ), NavigableString( ), etc.

# Using Inspect Element

The screenshot shows a web browser window displaying the weather history for Singapore Payalebar in January 2017. The browser's developer tools are open, specifically the 'Elements' tab, which is focused on the `h2.history-date` element. This element represents the heading for the date Sunday, January 1, 2017. The browser's address bar shows the URL [https://www.wunderground.com/history/airport/WSAP/2017/1/1/DailyHistory.html?req\\_city=Singapore+Pa...](https://www.wunderground.com/history/airport/WSAP/2017/1/1/DailyHistory.html?req_city=Singapore+Pa...). The developer tools interface includes tabs for Elements, Console, Sources, Network, and Performance, along with various status indicators and search/filter options.

Singapore Payalebar, Singapore ★  
Paya Lebar

Forecast History Calendar Rain / Snow Health

Weather History for WSAP - January, 2017

Change the Weather History Date:  
January 1 2017 View

Sunday, January 1, 2017  
« Previous Day Next Day »

Daily Weekly Monthly Custom

	Actual	Average (WSSS)	Record (WSSS)
Temperature			
Mean Temperature	29 °C	-	
Max Temperature	32 °C	29 °C	31 °C (2000)
Min Temperature	26 °C	23 °C	22 °C (2013)
Cooling Degree Days	18		
Growing Degree Days	34 (Base 50)		
Moisture			
Dew Point	24 °C		
Average Humidity	79		

```
><div id="location">...</div>
><div class="row collapse city-nav">...</div>
<div id="profile_overlay" class="reveal-modal" data-reveal></div>
><div class="row collapse">...</div>
<div class="row collapse">
  ::before
  <div class="column large-8 right-spacing">
    <div class="daily-history-select">
      <div class="change-date">Change the Weather History Date:</div>
      <div class="history-date-select">...</div>
      <h2 class="history-date">Sunday, January 1, 2017</h2> == $0
      <div class="previous-link">...</div>
      <div class="next-link">...</div>
      <div id="#history-select-tabs">...</div>
      <script type="text/javascript">...</script>
    </div>
    <table cellspacing="0" cellpadding="0" id="historyTable" class="responsive airport-history-summary-table">...</table>
    <div class="taC red b">Averages and records for this station are not official NWS values. </div>
    <table cellspacing="0" cellpadding="3" class="full" style="border-bottom: 1px solid #CCC;">...</table>
    <div class="taC tm10">...</div>
    <center class="clearfix">...</center>
    <h2>Daily Weather History Graph</h2>
    <div id="history-graph-image">...</div>
  </div>

```

html body #content-wrap #inner-wrap #inner-content div div div div h2.history-date

Styles Event Listeners DOM Breakpoints Properties Accessibility

Filter :hov .cls +

Console What's New

# Example 1: Wunderground

```
date = soup.find('div', attrs={'class':'daily-history-select'}).h2.text  
meantemp = soup.find_all('tr')[2].find_all('td')[1].find(attrs={"class":"wx-value"}).text  
maxtemp = soup.find_all('tr')[3].find_all('td')[1].find(attrs={"class":"wx-value"}).text  
mintemp = soup.find_all('tr')[4].find_all('td')[1].find(attrs={"class":"wx-value"}).text  
precip = soup.find_all('tr')[13].find_all('td')[1].find(attrs={"class":"wx-value"}).text  
wind = soup.find_all('tr')[17].find_all('td')[1].find(attrs={"class":"wx-value"}).text
```

# Further Examples: Wunderground

```
date = soup.find('div', attrs={'class':'daily-history-select'}).h2.text
```

```
nextlink = soup.find('div', attrs={'class':'daily-history-select'}).find_all('a')[1].get('href')
```

# Store the results

*Using the csv library*

# Example: Wunderground

- *Create a csv file to write your results to.*
- *Create headers.*
- *Write your results in a new row.*

```
with open("WeatherScrapeOnePage.csv", "w") as file:  
    csv_writer = csv.writer(file)  
    csv_writer.writerow(['Date', 'Mean Temp (C)', 'Max Temp (C)', 'Min Temp (C)', 'Precipitation (in)', 'Wind Speed (m/s)'])  
    csv_writer.writerow([date, meantemp, maxtemp, mintemp, precip, wind])
```

```
import requests
from bs4 import BeautifulSoup
import csv, time

# initial GET request
url = "https://www.wunderground.com/history/airport/WSAP/2017/1/1/DailyHistory.html"
response = requests.get(url)
content = response.content

# convert contents to nested data structure
soup = BeautifulSoup(content, "lxml")

# parse HTML
date = soup.find('div', attrs={'class':'daily-history-select'}).h2.text
meantemp = soup.find_all('tr')[2].find_all('td')[1].find(attrs={"class":"wx-value"}).text
maxtemp = soup.find_all('tr')[3].find_all('td')[1].find(attrs={"class":"wx-value"}).text
mintemp = soup.find_all('tr')[4].find_all('td')[1].find(attrs={"class":"wx-value"}).text
precip = soup.find_all('tr')[13].find_all('td')[1].find(attrs={"class":"wx-value"}).text
wind = soup.find_all('tr')[17].find_all('td')[1].find(attrs={"class":"wx-value"}).text

# write results
with open("WeatherScrapeOnePage.csv", "w") as file:
    csv_writer = csv.writer(file)
    csv_writer.writerow(['Date', 'Mean Temp (C)', 'Max Temp (C)', 'Min Temp (C)', 'Precipitation (in)', 'Wind Speed (m/s)'])
    csv_writer.writerow([dateString, meantemp, maxtemp, mintemp, precip, wind])

print("Scrape complete!")
```

# Scaling up

*And other important considerations*

# Traversing links

- Handling URL generation / pagination

```
#find next link
nextlink = soup.find('div', attrs={'class':'daily-history-select'}).find_all('a')[1].get('href')
url = 'https://www.wunderground.com'+nextlink
```

- Alternative methods for traversing pages in this scraper

# Handling exceptions

- Try/except logic
- Filling in N/A's
- Consider special cases

```
import requests
from bs4 import BeautifulSoup

url = "https://www.wunderground.com/weather/sg/singapore"

def load(url):
    attempts = 5
    for i in range(attempts):
        try:
            response = requests.get(url)
            return response.content
        except requests.HTTPError:
            if i+1 == attempts:
                raise
            else:
                time.sleep(60)

content = load(url)
soup = BeautifulSoup(content, "html.parser")

# move on to parsing!
```

# Data Cleaning

- Unnecessary characters may be present in your scraped data
- Can use the `.replace()` function, among other techniques, to remove and edit outputs

# Review & Results

Thank you!

# Resources

- [Requests documentation](#)
- [BeautifulSoup documentation](#)
- [csv documentation](#)