

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

```
<!--[if IE 9]><html class="no-js ie9"> <![endif]-->
<!--[if gt IE 9]><!--> <html class="no-js "> <!--<![endif]-->
  <head>
    <title>Weather History for Singapore Payalebar, Senegal | Weather Un
    <link href="//icons.wxug.com/" rel="dns-prefetch" />
    <link href="//api.ak.wunderground.com/" rel="dns-prefetch" />
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
  <meta name="HandheldFriendly" content="True">
  <meta name="MobileOptimized" content="320">
  <meta name="viewport" content="width=device-width, initial-scale=1, minimal-
  <meta http-equiv="cleartype" content="on">
    <meta name="description" content="Weather History for Paya Lebar, SN - G
  uinidity, wind, etc. on Weather Underground." />
    <meta name="keywords" content="weather history Paya Lebar, historical we
  snow history, wind history" />
  <meta name="apple-itunes-app" content="app-id=486154808, affiliate-data=at=1
  <meta name="fb-app-id" content="325331260891611" />
  <meta name="fb-channel-url" content="//www.wunderground.com/php/lib/fb_sdk/c
  <meta name="wui-member-logged-in" content="false" />
  <meta name="com.silverpop.brandeddomains" content="www.pages02.net, www.wunde
    <link rel="canonical" href="https://www.wunderground.com/history/airport
```



	Date	Mean Temp (C)	Max Temp (C)	Min Temp (C)
1	1/1/17	29	32	26
2	1/2/17	28	31	26
3	1/3/17	28	30	26
4	1/4/17	28	31	25
5	1/5/17	29	33	25
6	1/6/17	30	34	26
7	1/7/17	31	35	27
8	1/8/17	31	35	27

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!

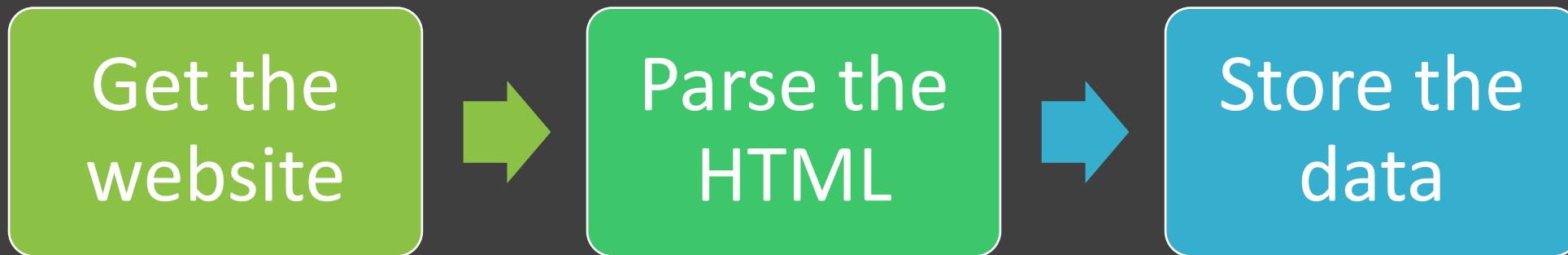
*“The Internet is one giant API – with
a really terrible interface.”*

- 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

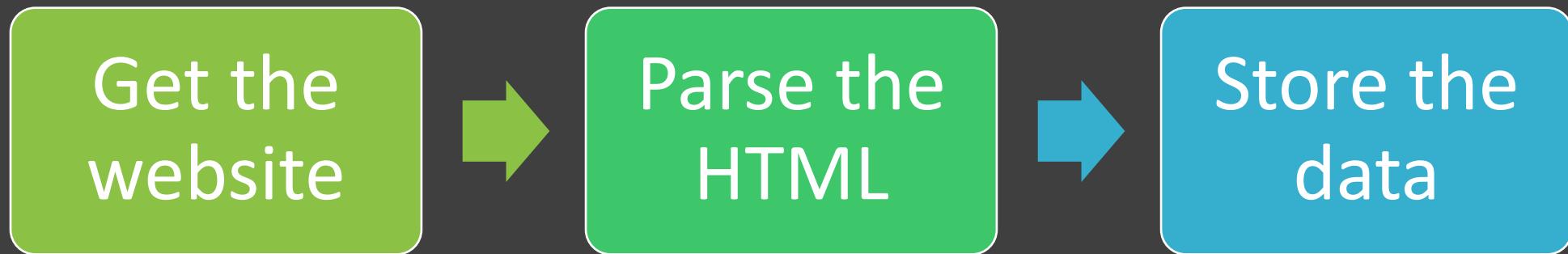
Web Scraping Workflow



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 at the top. Below the header, there are weather cards for San Francisco, CA; Chicago, IL; Boston, MA; and Houston, TX. The main content area is titled "Singapore Payalebar, Singapore" and displays a weather history for March 14, 2018. It includes a "History" tab selected in the navigation bar. Below the title, there's a date selector for "Wednesday, March 14, 2018". A table provides daily temperature statistics:

	Actual	Average (WSSS)
Mean Temperature	30 °C	-
Max Temperature	33 °C	31 °C
Min Temperature	26 °C	24 °C
Cooling Degree Days	20	
Growing Degree Days	34 (Base 50)	

Source Code

```
<!DOCTYPE html>
<!--[if IE 9]&gt;&lt;html class="no-js ie9"&gt; &lt;![endif]--&gt;
<!--[if gt IE 9]&gt;&lt;!--&gt; &lt;html class="no-js "&gt; &lt;!--&lt;![endif]--&gt;
&lt;head&gt;
    &lt;title&gt;Weather History for Singapore Payalebar, Senegal | Weather Underground&lt;/title&gt;
    &lt;link href="//icons.wxug.com/" rel="dns-prefetch" /&gt;
    &lt;link href="//api.ak.wunderground.com/" rel="dns-prefetch" /&gt;
&lt;meta charset="utf-8"&gt;
&lt;meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1"&gt;
&lt;meta name="HandheldFriendly" content="True"&gt;
&lt;meta name="MobileOptimized" content="320"&gt;
&lt;meta name="viewport" content="width=device-width, initial-scale=1, minimal-ui"&gt;
&lt;meta http-equiv="cleartype" content="on"&gt;
&lt;meta name="description" content="Weather History for Paya Lebar, SN - Get Weather History, humidity, wind, etc. on Weather Underground." /&gt;
&lt;meta name="keywords" content="weather history Paya Lebar, historical weather, weather all, snow history, wind history" /&gt;
&lt;meta name="apple-itunes-app" content="app-id=486154808, affiliate-data=at=10101rYB&amp;ct=website&amp;ctc=1" /&gt;
&lt;meta name="fb-app-id" content="325331260891611" /&gt;
&lt;meta name="fb-channel-url" content="//www.wunderground.com/php/lib/fb_sdk/channel.php" /&gt;
&lt;meta name="wui-member-logged-in" content="false" /&gt;
&lt;meta name="com.silverpop.brandeddomains" content="www.pages02.net,www.wunderground.com" /&gt;
&lt;link rel="canonical" href="https://www.wunderground.com/history/airport/WSAP/2018/3/14/DailyHistory.html?temp_f=0&amp;temp_c=1&amp;cityname=Singapore+Paya+Lebar&amp;countrycode=SN&amp;temp_f=0&amp;temp_c=1&amp;cityname=Singapore+Paya+Lebar&amp;countrycode=SN" /&gt;
&lt;link rel="apple-touch-icon" href="//icons.wxug.com/favicon.png" /&gt;
&lt;link rel="shortcut icon" type="image/png" href="//icons.wxug.com/favicon.png" /&gt;
&lt;link rel="stylesheet" href="//icons.wxug.com/css/wu4/core.css?v=2017041301" /&gt;
&lt;link rel="stylesheet" href="//icons.wxug.com/css/wu4/omnibus.css?v=2018022701" /&gt;
&lt;link rel="stylesheet" href="//icons.wxug.com/css/wu4/history.css?v=2015100201" /&gt;
&lt;script src="//icons.wxug.com/scripts/modernizr/2.8.2/modernizr.min.js"&gt;&lt;/script&gt;
&lt;script src="//ajax.googleapis.com/ajax/libs/jquery/1.7.2/jquery.min.js"&gt;&lt;/script&gt;
&lt;script src="//ajax.googleapis.com/ajax/libs/jqueryui/1.10.4/jquery-ui.min.js"&gt;&lt;/script&gt;
&lt;script src="//icons.wxug.com/scripts/wui.js/2.0.3/wui.min.js"&gt;&lt;/script&gt;
&lt;script src="//icons.wxug.com/scripts/wui.login.min.js?v=1.3.18"&gt;&lt;/script&gt;
&lt;script type="text/javascript"&gt;
if (typeof Object.assign != 'function') {
    Object.assign = function(target) {
        'use strict';
        if (target == null) {
            throw new TypeError('Cannot convert undefined or null to object');
        }
        var s = arguments.length;
        var i = 1;
        var o = Object(target);
        while (i &lt; s) {
            var source = arguments[i];
            if (source != null) {
                for (var p in source) {
                    if (Object.prototype.hasOwnProperty.call(source, p)) {
                        o[p] = source[p];
                    }
                }
            }
            i++;
        }
        return o;
    };
}</pre>
```

HTML/CSS

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

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

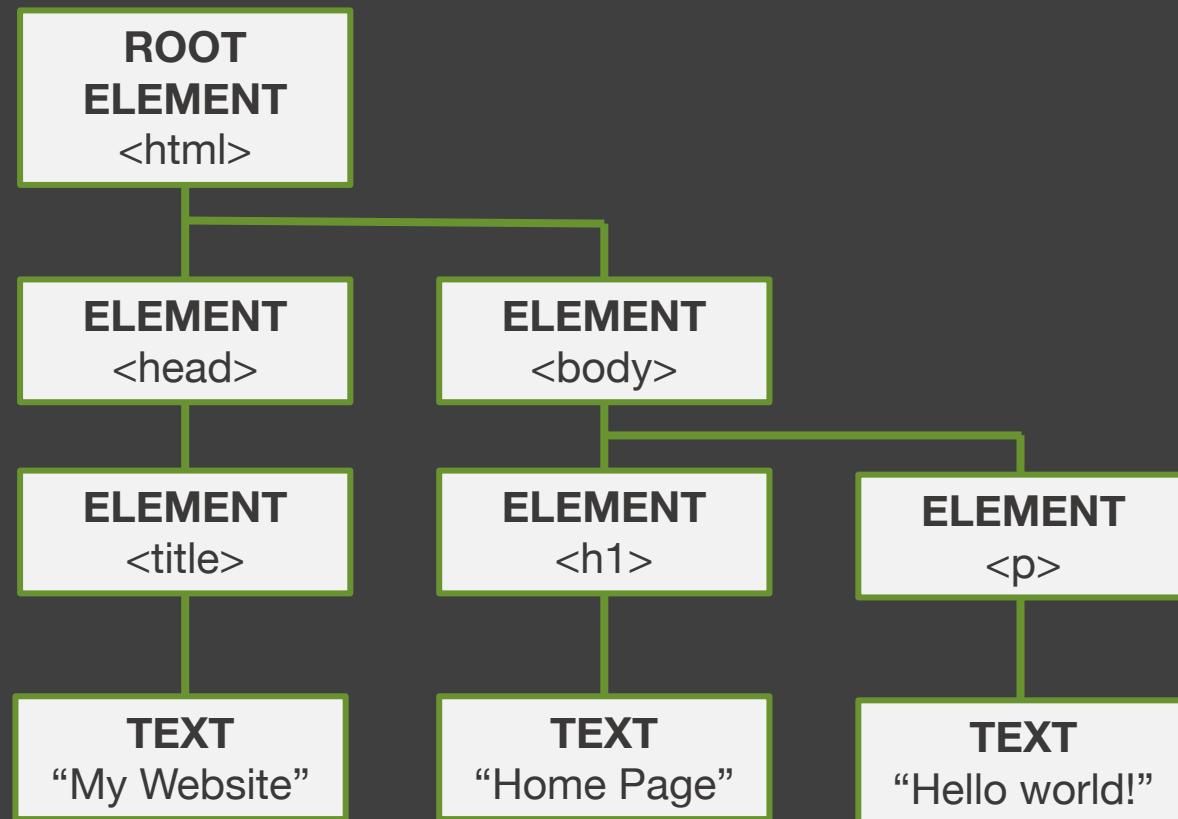
<h1>This is a heading</h1>
<h2>This is a smaller heading</h2>
<a href="https://www.google.com">This is a link</a>


<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

Secure | https://www.wunderground.com/history/airport/WSAP/2017/1/1/DailyHistory.html?req_city=Singapore+Pa... f? :

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		

Elements Console Sources Network Performance

```
<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/sig/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
- Double check the validity of the data!

Review & Results

Thank you!

Resources

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