

# Intro to Programming for Public Policy Week 7

## Web Scraping and APIs

Eric Potash

May 9, 2018

## Web Resources

# Variety

- ▶ There's a variety of information on the web:
  - ▶ *Unstructured* text
  - ▶ Tables
  - ▶ “Hidden” APIs
  - ▶ Documented APIs

# Examples of Web Resources

1. Google has many straightforward [APIs for mapping](#).
2. Twitter provides a well-known [API](#) that is used heavily by researchers.
3. U.S. Census provides a [data website](#) and [APIs](#).
4. Bureau of Labor Statistics has an [API](#).
5. Lots of Wikipedia articles have nice tables
6. Some websites go to great lengths to keep you out, e.g. [Google Trends](#)

# Tools

1. `requests`: Python module for retrieving web resources
  - ▶ Basic methods for authentication, POSTing, etc.
  - ▶ Basically `curl`/`wget` for Python
2. `beautifulsoup`: Python module for traversing and extracting elements from a web page.
3. `pandas.read_html()`: reads a well-formatted html table into a pandas DataFrame.
4. `selenium` is similar but actually launches a web browser like Firefox
  - ▶ Works with JavaScript heavy pages
5. Command line tools (`curl`, `wget`)

Scraping

# What is scraping?

- ▶ Look at HTML and individual requests (e.g. using developers tools in your browser)
- ▶ Identify patterns in HTML and URLs that allow you to download the right resources
- ▶ Extract data from those resources
- ▶ Relatively ad-hoc, need to write new scraping tools for each (part of each) site

# Requests

```
import requests

base = 'https://www.nytimes.com'
path = '/interactive/projects/guantanamo/detainees/current'

response = requests.get(base + path)
```



## Responses

```
>>> type(response)
requests.models.Response
>>> response.status_code
200
>>> print(response.text)
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" la
  <head>
    <meta http-equiv="Content-Type" content="text/html; cha
    <meta name="viewport" content="width=device-width, user
    <title>The Detainees - The Guantanamo Docket</title>
    <meta name="hdl" content="The Detainees - The Guantanamo
```

# Identify

The first step in scraping is figuring out a way to identify the information you want.

```
97 <table class="s-full s-datatable">
98   <tr class="bar"><th>Name</th><th>Citizenship</th></tr>
99   <tr class="divide nytint-first">
100     <td><a href="/interactive/projects/guantanamo/detainees/694-sufyan-barhoumi">Barhoumi, Sufyan</a></td>
101     <td><a href="/interactive/projects/guantanamo/country/algeria">Algeria</a></td>
102   </tr>
103   <tr class="divide ">
104     <td><a href="/interactive/projects/guantanamo/detainees/244-abdul-latif-nasir">Nasir, Abdul Latif</a></td>
105     <td><a href="/interactive/projects/guantanamo/country/morocco">Morocco</a></td>
106   </tr>
107   <tr class="divide ">
108     <td><a href="/interactive/projects/guantanamo/detainees/38-ridah-bin-saleh-al-yazidi">al Yazidi, Ridah Bin Saleh</td>
109     <td><a href="/interactive/projects/guantanamo/country/tunisia">Tunisia</a></td>
110   </tr>
111   <tr class="divide ">
112     <td><a href="/interactive/projects/guantanamo/detainees/893-tolfig-nassar-ahmed-al-bihani">al Bihani, Tolfig Nassa</td>
113     <td><a href="/interactive/projects/guantanamo/country/yemen">Yemen</a></td>
114   </tr>
115   <tr class=" ">
116     <td><a href="/interactive/projects/guantanamo/detainees/309-muieen-a-deen-jamal-a-deen-abd-al-fusal-abd-al-sattar">
117     <td><a href="/interactive/projects/guantanamo/country/unknown-national-origin">Unknown National Origin</a></td>
118   </tr>
119 </table>
```

# Parsing HTML

- ▶ You could manually search for `response.text` for

`<a href="/interactive/projects/gauntanamo/detainees/...`

- ▶ Then, to get the detainee's name you'd need to find the next `>` and the closing tag `</a>` and extract the text in between.
- ▶ This would be cumbersome

# Beautiful Soup

- ▶ The BeautifulSoup library makes this much easier by converting the text into a data structure that is easy to “traverse”

```
>>> from bs4 import BeautifulSoup
>>> page = BeautifulSoup(response.text, 'html.parser')
>>> type(page)
bs4.BeautifulSoup
```

## find all()

Use the `find_all()` function to find all tags of a particular type:

```
>>> page.find_all('a')
[<a href="http://www.nytimes.com" target="_blank">Related
  <a href="https://www.nytimes.com/interactive/projects/guar
  ...
```

# Tag

- ▶ Each element in the list returned by `find_all()` is a special Element data type:

```
>>> a = page.find_all('a')[0]
>>> type(a)
bs4.element.Tag
```

- ▶ You can see its HTML by simply printing it:

```
>>> a
<a href="http://www.nytimes.com" target="_blank"><img alt="New York Times logo" data-bbox="132 640 1000 726"/>
```

## Tag details

- ▶ You can see a dictionary of its attributes:

```
>>> a.attrs
{'href': 'http://www.nytimes.com',
 'target': '_blank'}
```

- ▶ You can see its contents:

```
>>> a.contents
[<img alt="The New York Times" src="https://int.nyt.com
>>> a.text
''
```

## Detainee links

- ▶ But we only want the detainee links
- ▶ These can be identified as having an href that starts with '/interactive/projects/guantanamo/detainees'
- ▶ So one way to get them would be:

```
detainee_links = []
prefix = '/interactive/projects/guantanamo/detainees'
for a in page.find_all('a'):
    if a.attrs['href'].startswith(prefix):
        detainee_links.append(a)
```



# Issues

```
>>> detainee_links
[<a href="/interactive/projects/guantanamo/detainees/current">
  <a href="/interactive/projects/guantanamo/detainees/694-suspects">
  <a href="/interactive/projects/guantanamo/detainees/244-alphabetical">
  <a href="/interactive/projects/guantanamo/detainees/38-rich">
  <a href="/interactive/projects/guantanamo/detainees/893-top">
  ...
```

- ▶ This includes the links to detainees/current and detainees/country
- ▶ We could manually remove them but there's a better way

# Regex

- ▶ Regular expressions are a language for expressing patterns that can be matched to text
- ▶ For example the regular expression `\d` matches any numeric digit
- ▶ The regular expression `detainees/\d` matches `detainees/` followed by a digit

# Python re

The re module provide regular expression matching in Python:

```
>>> import re
>>> pattern = re.compile('d+')
>>> type(pattern)
_sre.SRE_Pattern
>>> pattern.findall('1')
['1']
>>> pattern.findall('abc')
[]
>>> pattern.findall('Chicago, IL 60637')
['6', '0', '6', '3', '7']
```

## More regex

- ▶ Match either “gray” or “grey”:

`gray|grey`

- ▶ Same:

`gr(a|e)y`

# Regex quantifiers

- ▶ `?`: match zero or one occurrence
  - ▶ `colou?r` matches both “color” and “colour”
- ▶ `*`: match any number of occurrences
  - ▶ `1\d*` matches any number whose first digit is 1
- ▶ `+`: match one or more occurrences
  - ▶ `\d` matches any number

## Regex find\_all()

The BeautifulSoup.find\_all() function can filter an attribute to match a regular expression:

```
page.find_all('a', href=re.compile('detainees/\d'))
```

## Print prisoner names

```
import requests

base = 'https://www.nytimes.com'
path = '/interactive/projects/guantanamo/detainees/current'

response = requests.get(base + path)
page = BeautifulSoup(response.text, 'html.parser')

detainee_links = page.find_all('a', href=re.compile('detainee'))

for a in detainee_links:
    print(a.text)
```

## Country

- ▶ After each prisoner link in the HTML there is a link to their country.
- ▶ We can scrape this, too like so:

```
>>> a
<a href="/interactive/projects/guantanamo/detainees/694-suf
>>> a.find_next('a')
<a href="/interactive/projects/guantanamo/country/algeria">
>>> a.find_next('a').text
'Algeria'
```



## More information

- ▶ What if we want to get more information about each detainee?
  - ▶ For example, how long they've been detained
- ▶ We'll need to request each detainee's page and scrape that:

```
>>> detainee_path = detainee_links[0].attrs['href']
>>> detainee_path
'/interactive/projects/guantanamo/detainees/694-sufyian-ba
>>> detainee_page = requests.get(base + detainee_path)
```

# Detainee HTML

```
77     <div class="nytint-detainee-fullcol">
78         <h1 class="nytint-detainee-header nytint-mainheader">
79             Sufyian Barhoumi
80         </h1>
81
82         <p>
83             Sufyian Barhoumi
84             is a 44-year-old
85             citizen of
86             <a href="https://www.nytimes.com/interactive/projects/guantanamo/country/algeria">Algeria</a>.
87
88
89
90
91             As of January 2010, the <a href="http://www.justice.gov/ag/guantanamo-review-final-
92             report.pdf">Guantánamo Review Task Force</a> had recommended him for prosecution.
93             A parole-like <a href="http://www.prs.mil/Home.aspx">Periodic Review Board</a> later
94             recommended him for transfer.
95             As of May 9, 2018, he has been <a href="https://www.nytimes.com/interactive/projects/guantanamo
96             /detainees/held">held at Guantánamo</a> for 15 years 11 months.
97
98
99             War crimes charges against Mr. Barhoumi have been dismissed but may be refiled.
100         </p>
```

## Extract time detained

```
▶ >>> time_pattern = re.compile('\d+ year')  
>>> time_pattern.findall(detainee_div.text)  
['15 year']
```

How can we get the number 15?

## Extract time detained

```
▶ >>> time_pattern = re.compile('\d+ year')  
>>> time_pattern.findall(detainee_div.text)  
['15 year']
```

How can we get the number 15?

```
▶ >>> matches = time_pattern.findall(detainee_div.text)  
>>> int(matches[0].rstrip(' year'))  
15
```

## Put it in a function

```
time_pattern = re.compile('\d+ year')

def get_years(det_page):
    div = det_page.find('div',
                        class_='nytint-detainee-fullcol')
    matches = time_pattern.findall(div.text)
    return int(matches[0].rstrip(' year'))
```

## Another function

```
def get_detainee_page(detainee_link):  
    detainee_response = requests.get(  
        base + detainee_link.attrs['href'])  
  
    return BeautifulSoup(detainee_response.text,  
        'html.parser')
```

## Putting it together

```
import time

names, countries, years = [], [], []

for a in detainee_links:
    print(a.text)
    names.append(a.text)
    countries.append(a.find_next('a').text)

    detainee_page = get_detainee_page(a)
    years.append(get_years(detainee_page))


    time.sleep(2)

detainees = pd.DataFrame({'name': names,
                          'country': countries,
                          'years': years})
```

Pandas read\_html()



# HTML Table



WIKIPEDIA  
The Free Encyclopedia

- Main page
- Contents
- Featured content
- Current events
- Random article
- Donate to Wikipedia
- Wikipedia store

Interaction

- Help
- About Wikipedia
- Community portal
- Recent changes
- Contact page

Tools


- What links here
- Related changes
- Upload file
- Special pages
- Permanent link
- Page information

Not logged in

Article **Table** Read Edit View history

## Table of biofuel crop yields

From Wikipedia, the free encyclopedia



This article **needs additional citations for verification**. Please help improve this article by adding citations to reliable sources. Unsourced material may be challenged and removed. *(April 2008)* [\(Learn how and when to remove this message\)](#)

The following table shows the [vegetable oil](#) yields of common [energy crops](#) associated with [biodiesel](#) production. This is unrelated to [ethanol](#) and [cellulose](#) content instead of oil yields.

Crop	kg oil/ha/yr	litres oil/ha	lbs oil/acre	US gal/acre
<a href="#">maize</a> (corn)	147	172	129	18
<a href="#">cashew nut</a>	148	176	132	19
<a href="#">oats</a>	183	217	163	23
<a href="#">lupin</a> (lupine)	195	232	175	25
<a href="#">kenaf</a>	230	273	205	29
<a href="#">calendula</a>	256	305	229	33
<a href="#">cotton</a>	273	325	244	35
<a href="#">hemp</a>	305	363	272	39
<a href="#">soybean</a>	375	446	335	48
<a href="#">coffee</a>	386	459	345	49

Figure 1: [https://en.wikipedia.org/wiki/Table\\_of\\_biofuel\\_crop\\_yields](https://en.wikipedia.org/wiki/Table_of_biofuel_crop_yields)

# read\_html()

```
In [21]: import pandas as pd  
pd.read_html('https://en.wikipedia.org/wiki/Table_of_biofuel_crop_yields')[0]
```

```
Out[21]:
```

	0	1
0	NaN	This article needs additional citations for ve...

```
In [22]: pd.read_html('https://en.wikipedia.org/wiki/Table_of_biofuel_crop_yields')[1]
```

```
Out[22]:
```

	0	1	2	3	4
0	Crop	kg oil/ha/yr	litres oil/ha	lbs oil/acre	US gal/acre
1	maize (corn)	147	172	129	18
2	cashew nut	148	176	132	19
3	oats	183	217	163	23
4	lupin (lupine)	195	232	175	25
5	kenaf	230	273	205	29
6	calendula	256	305	229	33
7	cotton	273	325	244	35

## read\_html() arguments

```
In [23]: import pandas as pd
pd.read_html('https://en.wikipedia.org/wiki/Table_of_biofuel_crop_yields',
             match='Crop',
             header=0)[0]
```

Out[23]:

	Crop	kg oil/ha/yr	litres oil/ha	lbs oil/acre	US gal/acre
0	maize (corn)	147.0	172.0	129.0	18.0
1	cashew nut	148.0	176.0	132.0	19.0
2	oats	183.0	217.0	163.0	23.0
3	lupin (lupine)	195.0	232.0	175.0	25.0
4	kenaf	230.0	273.0	205.0	29.0
5	calendula	256.0	305.0	229.0	33.0
6	cotton	273.0	325.0	244.0	35.0
7	hemp	305.0	363.0	272.0	39.0
8	soybean	375.0	446.0	335.0	48.0
9	coffee	386.0	459.0	345.0	49.0
10	flax (linseed)	402.0	478.0	359.0	51.0

APIs

# API Overview

- ▶ Application Programming Interfaces (APIs) are web-based resources that serve data directly
- ▶ Typically in json, csv, or xml format
- ▶ You can access them with requests in python or curl
- ▶ The parameters are typically provided in the URL with a query string (?param1=value1&param2=value2)

# REST

REST is a standard style for organizing API resources. The philosophy is that:

- ▶ Client and server are “stateless”, i.e. each request is independent and the server does not save state between requests
- ▶ Service is scalable and cacheable
- ▶ Typically use HTTP methods (GET, DELETE, PUT, etc.) meaningfully

# Census API

- ▶ U.S. Census data available through [API](#)
- ▶ For example the 5-year ACS estimates
  - ▶ List of variables

[https://api.census.gov/data/2014/acs5/profile?get=DP02\\_0037PE,NAME&for=state:\\*](https://api.census.gov/data/2014/acs5/profile?get=DP02_0037PE,NAME&for=state:*)

- ▶ Can simply `curl`

curl

```
$ curl "https://api.census.gov/data/2014/acs5/profile?get=I"
-o fertility.json
$ head fertility.json
[["DP02_0037PE", "NAME", "state"],
["41.7", "Alabama", "01"],
["34.2", "Alaska", "02"],
["39.7", "Arizona", "04"],
["37.7", "Arkansas", "05"],
["33.2", "California", "06"],
["28.3", "Colorado", "08"],
["33.7", "Connecticut", "09"],
["39.3", "Delaware", "10"],
["47.7", "District of Columbia", "11"],
...]
```



## python

```
▶ >>> path = "https://api.census.gov/data/2014/acs5/profiles"
>>> j = requests.get(path).json()
>>> j
[[u'DP02_0037PE', u'NAME', u'state'],
 [u'41.7', u'Alabama', u'01'],
 [u'34.2', u'Alaska', u'02'],
 [u'39.7', u'Arizona', u'04'],
 [u'37.7', u'Arkansas', u'05'],
 [u'33.2', u'California', u'06'],
 ...]
```

How can we put this data in a pandas DataFrame?

# python

```
▶ >>> path = "https://api.census.gov/data/2014/acs5/profi
>>> j = requests.get(path).json()
>>> j
[[u'DP02_0037PE', u'NAME', u'state'],
 [u'41.7', u'Alabama', u'01'],
 [u'34.2', u'Alaska', u'02'],
 [u'39.7', u'Arizona', u'04'],
 [u'37.7', u'Arkansas', u'05'],
 [u'33.2', u'California', u'06'],
 ...
```

How can we put this data in a pandas DataFrame?

```
▶ >>> pd.DataFrame(j[1:], columns=j[0])
```