Lab #4: Web Scraping

Or, “How to get banned from Yelp”

Oct 17, 2019

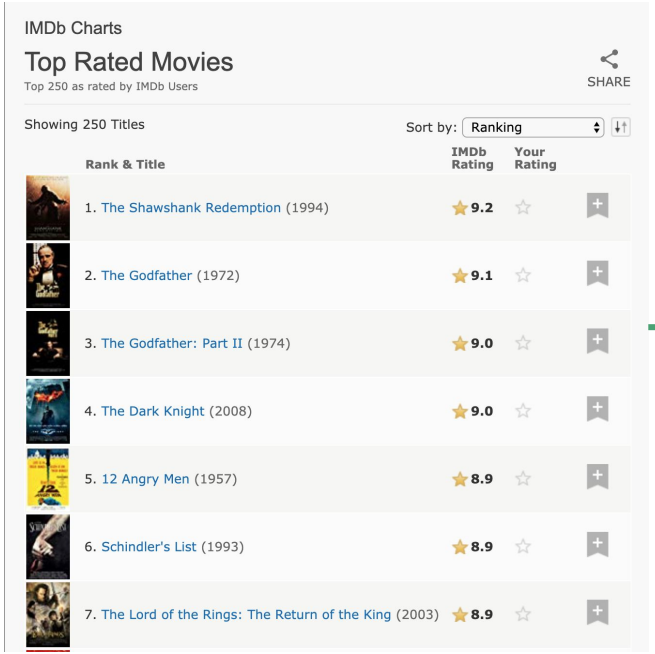
What is web scraping?

What is web scraping?

- you need data (for research, personal interest, etc.)
- it’s on the web
- there doesn’t seem to be an API for it

What is web scraping?

Get from here...



...to here

	title	rating
0	The Shawshank Redemption	9.2
1	The Godfather	9.1
2	The Godfather: Part II	9.0
3	The Dark Knight	9.0
4	12 Angry Men	8.9
5	Schindler's List	8.9
6	The Lord of the Rings: The Return of the King	8.9
7	Pulp Fiction	8.9
8	The Good, the Bad and the Ugly	8.8
9	Fight Club	8.8
10	The Lord of the Rings: The Fellowship of the Ring	8.8
11	Joker	8.8
12	Forrest Gump	8.8
13	Inception	8.7
14	Star Wars: Episode V - The Empire Strikes Back	8.7
15	The Lord of the Rings: The Two Towers	8.7
16	The Matrix	8.6
17	One Flew Over the Cuckoo's Nest	8.6
18	Goodfellas	8.6
19	Seven Samurai	8.6
20	Se7en	8.6
21	City of God	8.6
22	Life Is Beautiful	8.6
23	The Silence of the Lambs	8.6
24	It's a Wonderful Life	8.6
25	Star Wars: Episode IV - A New Hope	8.6

Before we begin

All materials available at:

https://github.com/5harad/css/tree/master/web-scraping

Credit to Jongbin Jung (2017 TA) and Joe Nudell (SCPL) for materials and inspiration.

Outline

- Motivation
- Approaches
- Example 1: IMDB Top 250
  - HTML, live demo
  - Exercise: Wikipedia – Lakes of California
- Example 2: Public Notices
  - Chrome devtools, live demo
- Scraping ethics
- Postscript: Selenium WebDriver

Some motivating examples

- Building a business directory for the United Kingdom
  - sources: tripadvisor, timeout, etc.
- Building a graph of actors (nodes) and movies (edges)
  - “six degrees of Kevin Bacon”
  - sources: IMDB, ...
- Writing a bot to check Craigslist for apartments
- Determining which subreddits tend to contain more hate speech, etc.

Approaches

- **Standard**
  - load page, parse HTML
- **Front door**
  - public API
- **Back door**
  - decipher internal structure (e.g. hidden API)
- **The imposter**
  - control a browser automatically

Approaches

- **Standard**
  - load page, scrape HTML
- ~~**Front door**~~ ← We've seen this before (Twitter API)
  - ~~public API~~
- **Back door**
  - decipher internal structure (e.g. hidden API)
- **The imposter**
  - control a browser automatically

Approaches

- **Standard**
  - load page, scrape HTML
- ~~**Front door**~~ ← We've seen this before (Twitter API)
  - ~~public API~~
- **Back door**
  - decipher internal structure (e.g. hidden API)
- **The imposter**
  - control a browser automatically ← This is really hard!

Example 1: IMDB Top 250

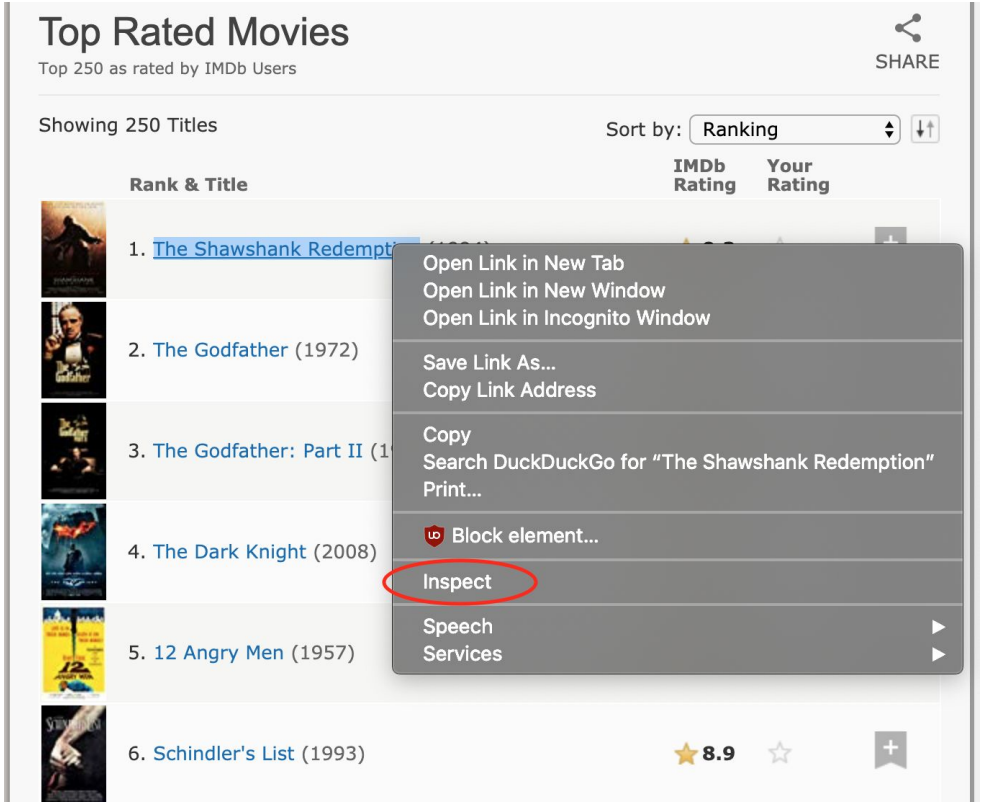
Example 1: IMDB Top 250

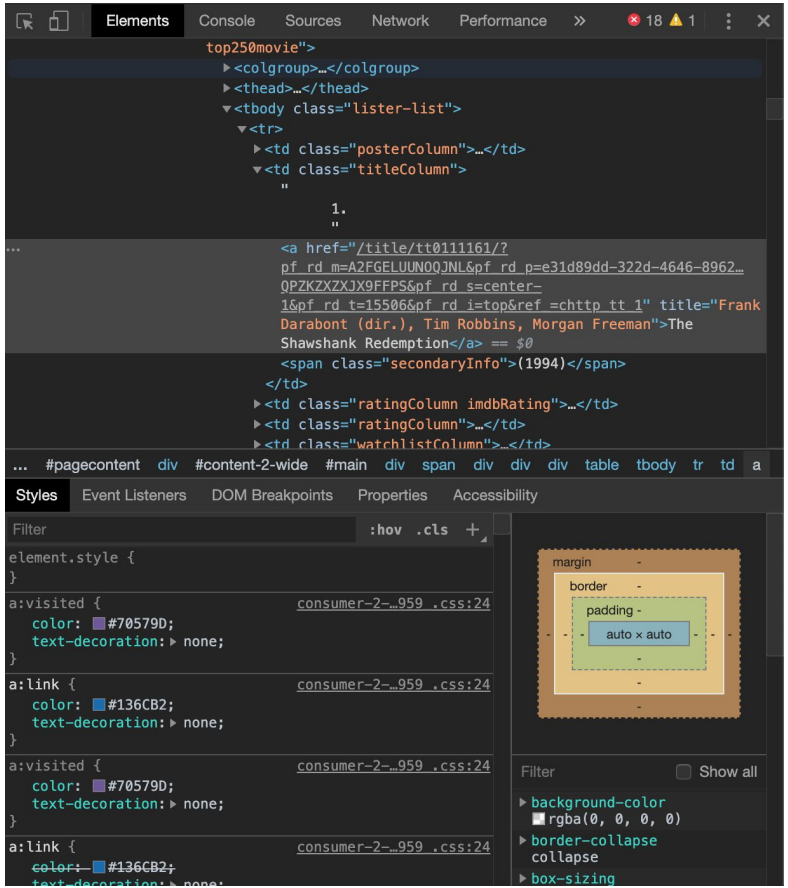
The “standard approach”

Goal: Collect the **cast overview** (actor + character played) for each of the **top 10 movies** on the **IMDB Top 250** list.

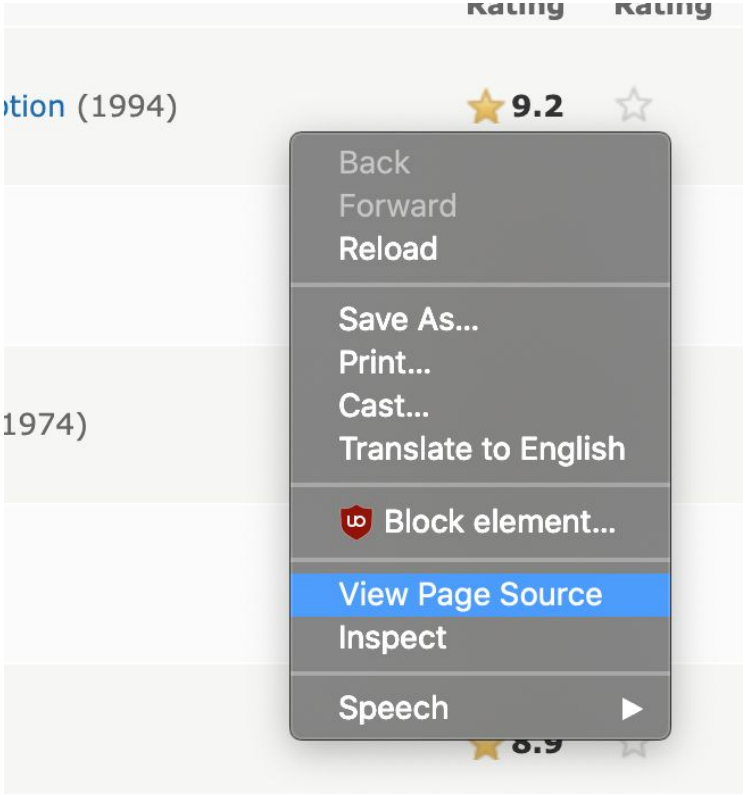
https://www.imdb.com/chart/top?ref_=nv_ch_250_4

(live demo)





Inspecting HTML



view-source:https://www.imdb.com/chart/top?ref_=nv_ch_250_4

```
        <tbody class="lister-list">

  <tr>
    <td class="posterColumn">

    <span name="rk" data-value="1"></span>
    <span name="ir" data-value="9.222197604788654"></span>
    <span name="us" data-value="7.791552E11"></span>
    <span name="nv" data-value="2147579"></span>
    <span name="ur" data-value="-1.7778023952113458"></span>
<a href="/title/tt0111161/?pf_rd_m=A2FGELUUNOQJNL&pf_rd_p=e31d89dd-322d-4646-8962-327b42fe94b1&pf_rd_r=
1&pf_rd_t=15506&pf_rd_i=top&ref_=chttp_tt_1"
> <img src="https://m.media-amazon.com/images/M/MV5BMDFkYTc0MGEtZmNhMC00ZDIzLWFmNTEtODM1ZmRlYWMwMWFmXkE
width="45" height="67" alt="The Shawshank Redemption"/>
</a>    </td>
    <td class="titleColumn">
      1.
      <a href="/title/tt0111161/?pf_rd_m=A2FGELUUNOQJNL&pf_rd_p=e31d89dd-322d-4646-8962-327b42fe94b1&pf
1&pf_rd_t=15506&pf_rd_i=top&ref_=chttp_tt_1"
title="Frank Darabont (dir.), Tim Robbins, Morgan Freeman" >The Shawshank Redemption</a>
        <span class="secondaryInfo">(1994)</span>
    </td>
    <td class="ratingColumn imdbRating">
            <strong title="9.2 based on 2,147,579 user ratings">9.2</strong>
    </td>
    <td class="ratingColumn">
    <div class="seen-widget seen-widget-tt0111161 pending" data-titleid="tt0111161">
        <div class="boundary">
            <div class="popover">
<span class="delete"> </span><ol><li>1<li>2<li>3<li>4<li>5<li>6<li>7<li>8<li>9<li>10</ol>
        </div>
        <div class="inline">
            <div class="pending"></div>
            <div class="unseeable">NOT YET RELEASED</div>
            <div class="unseen"> </div>
            <div class="rating"></div>
            <div class="seen">Seen</div>
        </div>
    </div>
    </td>
    <td class="watchlistColumn">
        <div class="wlb_ribbon" data-tconst="tt0111161" data-recordmetrics="true"></div>
    </td>
  </tr>

  <tr>
```

Viewing HTML source

Learning how HTML works...

Parsing HTML

```
import requests
import bs4

IMDB_PAGE = "https://www.imdb.com/chart/top?ref_=nv_ch_250_4"

response = requests.get(IMDB_PAGE)
soup = bs4.BeautifulSoup(response.content)

# for example
soup.find_all('td', attrs={'class': 'titleColumn'})
```

Exercise: Wikipedia

Using the “standard approach”

Goal: Collect the **name**, **county**, **type**, and **surface area** for each of the lakes in California (from Wikipedia).

https://en.wikipedia.org/wiki/List_of_lakes_in_California

Example 2: Public Notices

Public Notices: Background

- City of Mountain View is required by law to post public notices of certain city proceedings in a newspaper
- It does so in the *San Jose Post Record*
- The *San Jose Post Record* has a total print distribution of...

49 copies

- Public notices also available on legaladstore.com (not a spam site, i swear!)

Ref: https://www.mv-voice.com/news/2019/10/04/for-public-notices-mountain-view-turns-to-obscure-newspaper

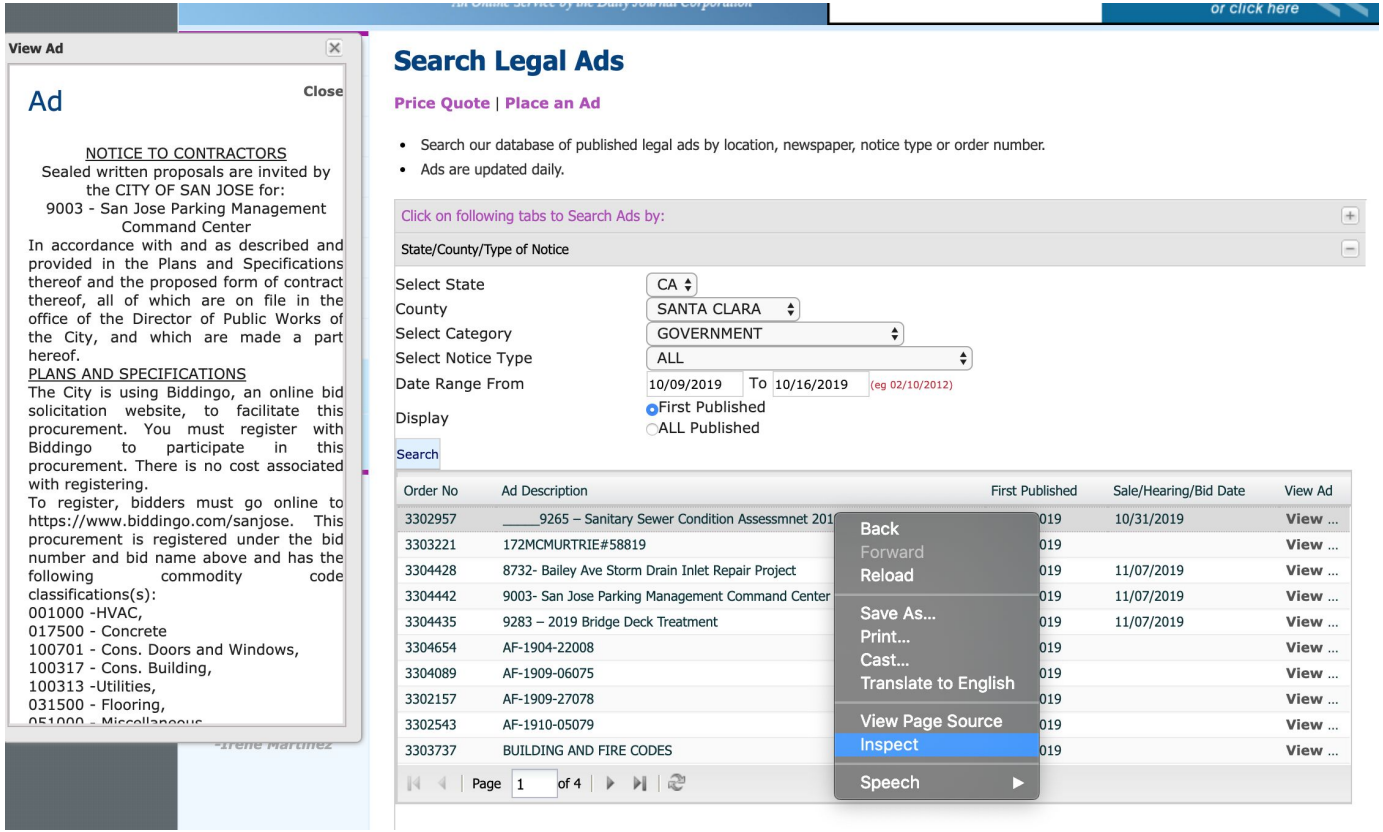
Example 2: Public Notices

The “back door” approach

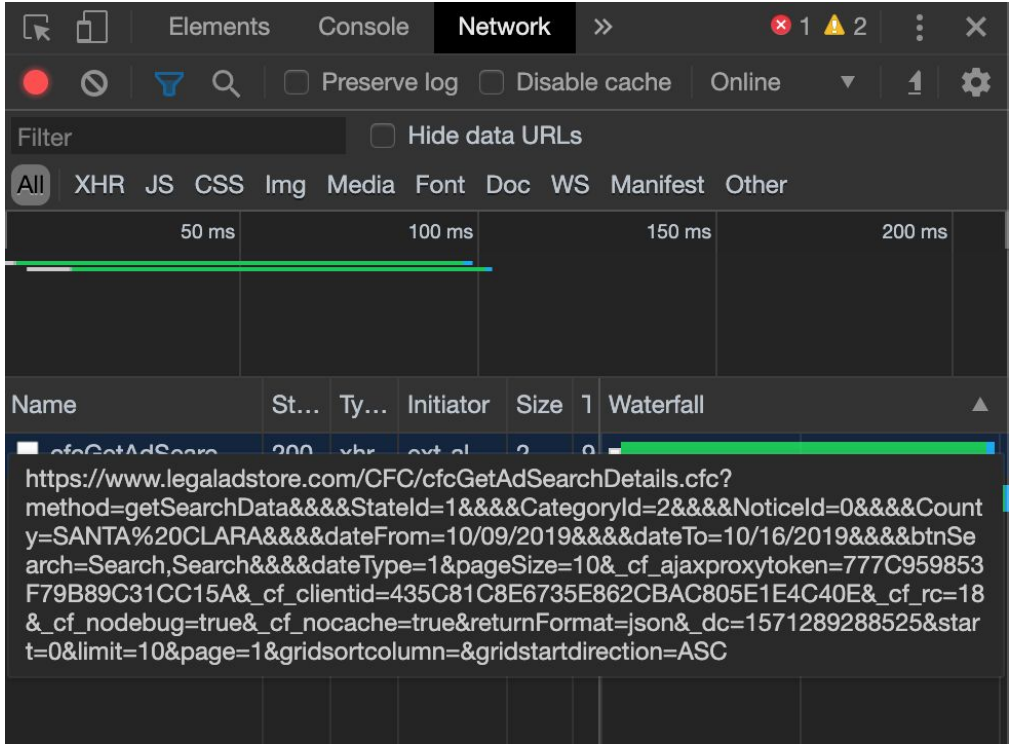
Goal: Collect **all government** notices for **Santa Clara County** posted since **2019-01-01**.

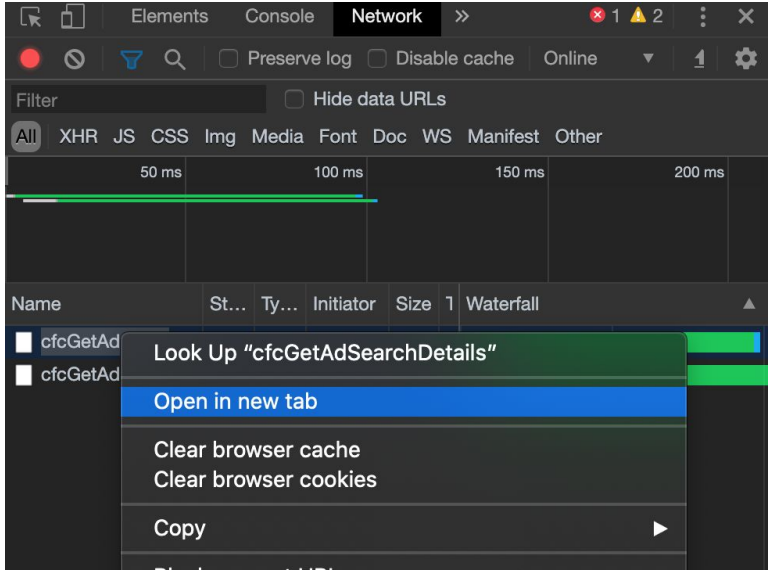
https://www.legaladstore.com/

(live demo)

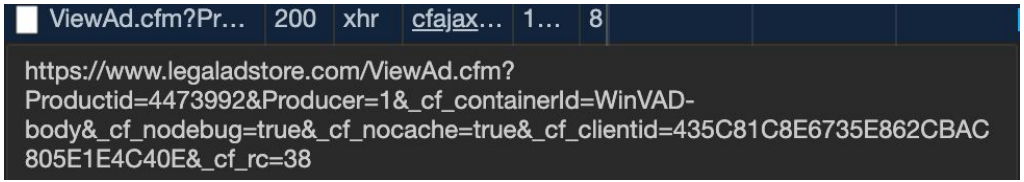


Head to the console again (⌥-⌘-C, ⇧-⌘-C)





Watch the “Network” tab



Through the back door we go!

```
import requests

search_results = requests.get(
    "https://www.legaladstore.com/CFC/cfcGetAdSearchDetails.cfc ",
    params={
        "method": "getSearchData",
        "CategoryId": 2,
        "County": "SANTA CLARA",
        "dateFrom": "01/01/2019",
        "dateTo": "10/17/2019",
        "limit": 10000,
        # ... more parameters ...
    })
```

Through the back door we go!

```
for record in search_results.json()["QUERY"]["DATA"]:
    ad_download = requests.get(
        "https://www.legaladstore.com/ViewAd.cfm ",
        params = {
            "Productid": record[4],
            "Producer": 1,
            # ... more fields ...
        })
    # save the ad somewhere, e.g. write to a file or print
    print(ad_download.text)
```

Scraping Ethics

Scraping Ethics

- Throttling
  - Your script can visit web pages a lot faster than you can!
  - Compute resources are not unlimited
  - Fix: using `time.sleep` in a Python script, e.g.
  - If not careful, you can get a whole IP block banned from a site
- Robots
  - The `robots.txt` file exists as a guideline for web spiders, etc.; be a good internet citizen and abide by them!
  - Check out https://en.wikipedia.org/robots.txt for an extensive example
- Terms of service
  - Certain sites (Yelp, i'm looking at you!) explicitly prohibit scraping in their terms of service
  - However, this likely has no legal ramifications

Postscript: Selenium WebDriver

Selenium WebDriver

- Programmatically “drive” the browser
- Allows more flexibility (closer to human interaction)
- Possible applications:
  - Multi-step login
  - Filling out a CAPTCHA
- Difficult/error-prone. Prefer other methods when possible.

Want more?

Materials for this section are at:
https://github.com/5harad/css/tree/master/web-scraping

A more in-depth tutorial on web scraping can be found here:
https://github.com/stanford-policylab/iriss-workshop/blob/master/py/python_scraping-exercises.ipynb
(rendered version can be viewed here)