Web Scraping with Beautiful Soup

PS 452: Text As Data

Fall 2014

Department of Political Science, Stanford University

Adapted from **Rebecca Weiss**'s VAM tutorial (http://www.rebeccaweiss.info/VAM-Python/VAM.html) by **Frances Zlotnick**

This tutorial will provide a very basic introduction to using the `Beautiful Soup` package to scrape text data from the web. This assumes that you already have the necessary packages installed.

HTML and the DOM

The general idea behind web scraping is to retrieve data that exists on a website, and convert it into a format that is usable for analysis. Webpages are rendered by the brower from HTML and CSS code, but much of the information included in the HTML underlying any website is not interesting to us.

We begin by reading in the source code for a given web page and creating a Beautiful Soup object with the `BeautifulSoup` function.

In [1]:
```
from bs4 import BeautifulSoup
import urllib
r = urllib.urlopen('http://www.aflcio.org/Legislation-and-Politics/Legislative-Alerts').read()
soup = BeautifulSoup(r)
print type(soup)
```

```
<class 'bs4.BeautifulSoup'>
```

The soup object contains all of the HTML in the original document.

In [2]:
```
print soup.prettify()[0:1000]
```

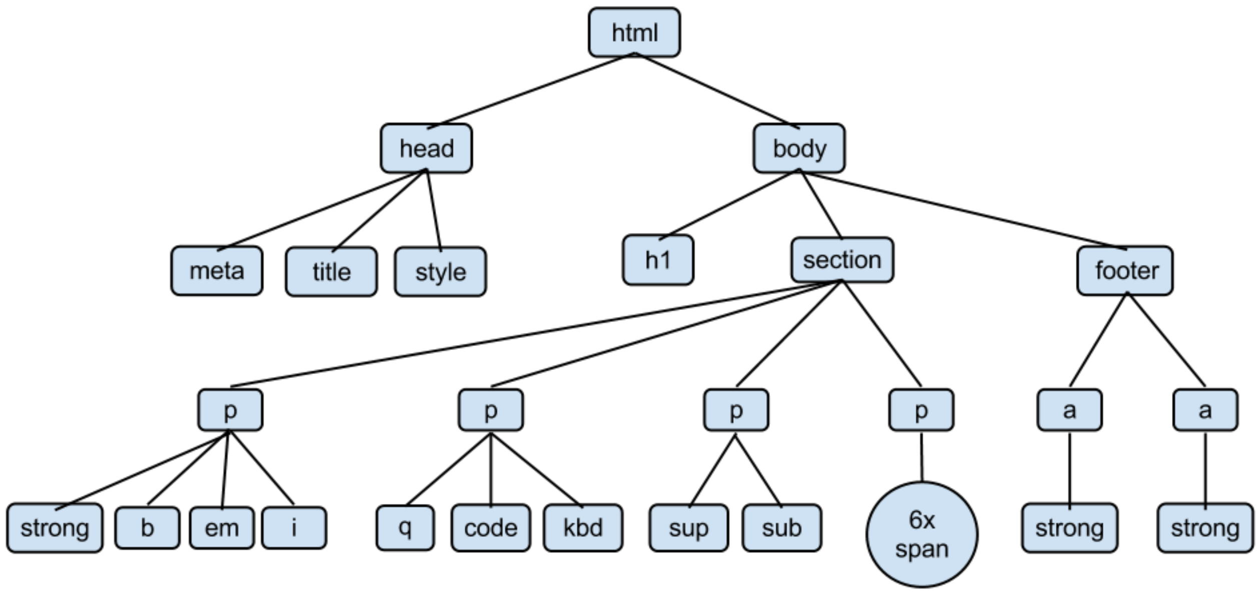
```
<!DOCTYPE html>
<html class="no-js" lang="en-US" xml:lang="en-US" xmlns="http://www.w3.org/1999/xhtml" xmlns:fb="http://ogp.me/ns/fb#">
 <head>
  <title>
   Legislative Alerts
  </title>
  <meta content="text/html; charset=utf-8" name="Content-Type"/>
  <meta content="en-US" name="Content-language"/>
  <meta content="" name="author"/>
  <meta content="" name="copyright"/>
  <meta content="" name="description"/>
  <meta content="" name="keywords"/>
  <meta content="TRUE" name="MSSmartTagsPreventParsing"/>
  <meta content="eZ Publish" name="generator"/>
  <meta content="Legislative Alerts" property="og:title"/>
  <meta content="http://www.aflcio.org/Legislation-and-Politics/Legislative-Alerts" property="og:url"/>
  <meta content="AFL-CIO" property="og:site_name"/>
  <meta content="http://www.aflcio.org/extension/aflcio/design/aflcio_user/images/facebook_aflcio_200x200.jpg" property="og:image"/>
  <meta content="non_profit" property="og:type"/>
  <meta content="288636237825618" property="
```

The HTML tags contained in the angled brackets provide structural information (and sometimes formatting), which we probably don't care about in and of itself but is useful for selecting only the content relevant to our needs.

Beautiful Soup is essentially a set of wrapper functions that make it simple to select common HTML elements.

In [3]:
```python
from IPython.display import Image
Image('http://www.openbookproject.net/tutorials/getdown/css/images/lesson4/HTMLDOMTree.png')
```

Out[3]:



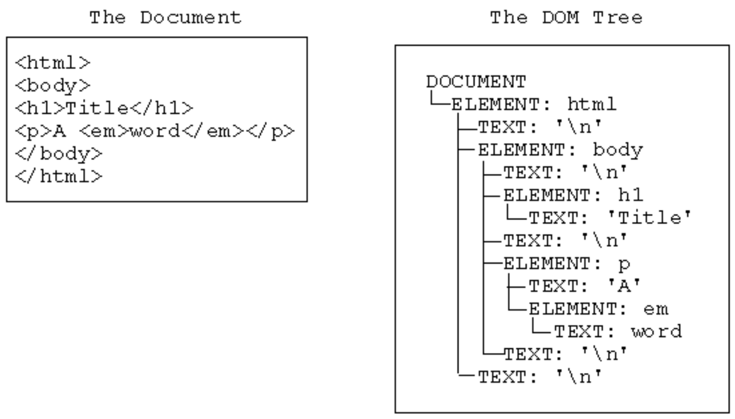
Most modern browsers have a *parser* that reads in the HTML **document**, parses it into a DOM (Document Object Model) structure, and then renders the **DOM** structure.

Much like HTTP, the DOM is an agreed-upon standard (http://www.w3.org/DOM/).

The DOM is much more than what I've described (https://developer.mozilla.org/en-US/docs/DOM), but for our purposes, what is most important to understand is that the text is only one part of an HTML element, and we need to select it explicitly.

In [4]:
```python
Image('http://www.cs.toronto.edu/~shiva/cscb07/img/dom/treeStructure.png')
```

Out[4]:



```
The Document

<html>
<body>
<h1>Title</h1>
<p>A <em>word</em></p>
</body>
</html>
```

```
The DOM Tree

DOCUMENT
└─ELEMENT: html
  ├─TEXT: '\n'
  ├─ELEMENT: body
  │ ├─TEXT: '\n'
  │ ├─ELEMENT: h1
  │ │ └─TEXT: 'Title'
  │ ├─TEXT: '\n'
  │ ├─ELEMENT: p
  │ │ ├─TEXT: 'A'
  │ │ └─ELEMENT: em
  │ │   └─TEXT: word
  │ └─TEXT: '\n'
  └─TEXT: '\n'
```

Parsing HTML

I study interest group political activity, and I'd like to know what issues organized labor was active on this year. Fortunately, the largest American labor federation has a website where they document the letters* that the national organization has sent to federal legislators. This is the page we loaded in above.

*Some things to always think about, regardless of your source: Who created this website? Do we think this is likely to be a complete list? Might this organization have some reason to selectively post information?

In [5]:
```
from IPython.display import HTML
HTML('<iframe src=http://www.aflcio.org/Legislation-and-Politics/Legislative-Alerts width=700 height=500></iframe>')
```

Out[5]:

Our goal here is to get the name of each item in this list, the URL that it links to, and the associated date. This sounds easy, but there is alot of junk to wade through to find what we want. By looking at the source code, I found that all of the items I am interested in are in a single `div` element of `class` "legisalerts_listing", and that each individual item is a member of `class` "ec_statements". Here is what the first few look like in HTML.

In [6]:
```
print soup.prettify()[28700:30500]
```

```
;
                                });
       </script>
      </div>
     </div>
     <div class="legisalerts_listing">
      <div class="ec_statements">
       <div id="legalert_title">
        <a href="/Legislation-and-Politics/Legislative-Alerts/Letter-to-Senators-Urging-Them-to-Support-Cloture-and-Final-Passage-of-the
-Paycheck-Fairness-Act-S.2199">
         Letter to Senators Urging Them to Support Cloture and Final Passage of the Paycheck Fairness Act (S.2199)
        </a>
       </div>
       <div id="legalert_date">
        September 10, 2014
       </div>
      </div>
      <div class="ec_statements">
       <div id="legalert_title">
        <a href="/Legislation-and-Politics/Legislative-Alerts/Letter-to-Representatives-Urging-Them-to-Vote-on-the-Highway-Trust-Fund-Bi
ll">
         Letter to Representatives Urging Them to Vote on the Highway Trust Fund Bill
        </a>
       </div>
       <div id="legalert_date">
        July 30, 2014
       </div>
      </div>
      <div class="ec_statements">
       <div id="legalert_title">
        <a href="/Legislation-and-Politics/Legislative-Alerts/Letter-to-Representatives-Urging-Them-to-Vote-No-on-the-Legislation-Provid
ing-Supplemental-Appropriations-for-the-Fiscal-Year-Ending-Sept.-30-2014">
         Letter to Representatives Urging Them to Vote No on the Legislation Providing Supplemental Appropriations for the Fiscal Year E
nding Sept. 30, 2014
        </a>
       </div>
       <div id="legalert_date">
        July 30, 2014
       </div>
      </div>
      <div class="ec_statements">
       <div id="legalert_title">
        <a href="/Legislation-and-Politics/Legislative-Alerts/Letter-to-Senators-Urging-Them-to-Vote-Yes-on-the-Motion-to-Proceed-to-the
-Emergency-Supplemental-Appropriations-Act-of-2014-S.2648">
         Letter to Senators Urging The
```

I can use the heirarchical nature of HTML structure to grab precisely the content that I am interested in. I will grab all of the elements that are within `div` tags and are also members of `class` "ec_statements."

```
In [7]: letters = soup.find_all("div", class_="ec_statements")
```

This returns a collection of tag objects. This is not one of the normal python collections covered in the other tutorials, it is an object specific to the `Beautiful Soup` library. It can be iterated over, but most other standard methods won't work on it. We'll have to do some preprocessing to get out the content that we want.

```
In [8]: print type(letters)

        <class 'bs4.element.ResultSet'>
```

By examining one of the elements in this collection, we can see that the information we want inside these objects, but we'll need to use more of Beautiful Soup's functionality and know something about HTML strucure to access them.

```
In [9]: letters[0]

Out[9]: <div class="ec_statements">
        <div id="legalert_title"><a href="/Legislation-and-Politics/Legislative-Alerts/Letter-to-Senators-Urging-Them-to-Support-Cloture-and-Fin
        al-Passage-of-the-Paycheck-Fairness-Act-S.2199">Letter to Senators Urging Them to Support Cloture and Final Passage of the Paycheck Fair
        ness Act (S.2199)</a></div>
        <div id="legalert_date">September 10, 2014</div>
        </div>
```

Recall that we want 3 things: the text of the item as it appears on the website, the URL that is linked to (so we can scrape and analyze it later), and the date the letter was sent.

First we should consider how we are going to store this data. Since we want to maintain the association between the 3 things from each observation. A natural way to store this is as a nested `dict`. `Dict` keys must be unique, and some of our items have the same associated date, so we'll have to use one of the other items. We'll use the name.

Visible text is always placed between tags. On the rendered page, the name we want is an active link, but we can get just the associated text by using the `get_text` method on the `a` tags. You can also use `contents`, which returns a `list` instead of a `string`.

We'll go through all of the items in our letters collection, and for each one, pull out the name and make it a key in our `dict`. The value will be another `dict`, but we haven't yet found the contents for the other items yet so we'll just create assign an empty `dict` object.

```
In [10]: lobbying = {}
         for element in letters:
             lobbying[element.a.get_text()] = {}
```

Now we want to add to our inner dictionary by the linked URL.

In HTML, links are always enclosed in `a` tags as the `href` attribute. We can use this fact to easily pull out the links. But if we look at the URLs, we'll find that they are incomplete. Like many websites, this site uses relative paths.

```
In [11]: letters[0].a["href"]

Out[11]: u'/Legislation-and-Politics/Legislative-Alerts/Letter-to-Senators-Urging-Them-to-Support-Cloture-and-Final-Passage-of-the-Paycheck-Fairn
         ess-Act-S.2199'
```

If we want to visit these links later we'll need the complete path, so we will add in a prefix to complete these paths.

```
In [12]: prefix = "www.aflcio.org"
```

Now we'll go through our letters collection again, and add a link key to our lobbying dictionaries.

```
In [13]: for element in letters:
             lobbying[element.a.get_text()]["link"] = prefix + element.a["href"]
```

Now we'll do the same thing with the date. Note that all of the dates are found within `div` tags with `id` "legalert_date". We can use this structure to grab the date information just as we did the link text. We know there is only one of these per item so we can use `find` instead of `find_all`.

```
In [14]: letters[0].find(id="legalert_date")

Out[14]: <div id="legalert_date">September 10, 2014</div>
```

Now we'll add it into our lobbying dictionary. We can see above that we only want the text between the tags, so we'll pull that out before adding it into the dictionary.

```
In [15]: for element in letters:
             date = element.find(id="legalert_date").get_text()
             lobbying[element.a.get_text()]["date"] = date
```

Let's look at what we've made.

```
In [16]: for item in lobbying.keys():
             print item + ": " + "\n\t" + "link: " + lobbying[item]["link"] + "\n\t" + "date: " + lobbying[item]["date"] + "\n\n"
```

```
In Opposition of "Regulatory Reform" Bills – H.R. 2804 and H.R. 899:
        link: www.aflcio.org/Legislation-and-Politics/Legislative-Alerts/In-Opposition-of-Regulatory-Reform-Bills-H.R.-2804-and-H.R.-899
        date: February 25, 2014


Letter to Representatives Opposing the Customer Protection and End User Relief Act:
        link: www.aflcio.org/Legislation-and-Politics/Legislative-Alerts/Letter-to-Representatives-Opposing-the-Customer-Protection-and-
End-User-Relief-Act
        date: June 20, 2014


Letter to Representatives Urging Them to Vote on the Highway Trust Fund Bill:
        link: www.aflcio.org/Legislation-and-Politics/Legislative-Alerts/Letter-to-Representatives-Urging-Them-to-Vote-on-the-Highway-Tr
ust-Fund-Bill
        date: July 30, 2014


Letter to representatives in support of the Westmoreland-DeFazio Amendment to H.R. 4745:
        link: www.aflcio.org/Legislation-and-Politics/Legislative-Alerts/Letter-to-representatives-in-support-of-the-Westmoreland-DeFazi
o-Amendment-to-H.R.-4745
        date: June 09, 2014


In Opposition to H.R. 4320 and H.R. 4321:
        link: www.aflcio.org/Legislation-and-Politics/Legislative-Alerts/In-Opposition-to-H.R.-4320-and-H.R.-4321
        date: April 08, 2014


In Support of the Confirmation of Judge Robert L. Wilkins:
        link: www.aflcio.org/Legislation-and-Politics/Legislative-Alerts/In-Support-of-the-Confirmation-of-Judge-Robert-L.-Wilkins
        date: January 07, 2014


Letter to Senators Urging Them to Support Cloture and Final Passage of the Paycheck Fairness Act (S.2199):
        link: www.aflcio.org/Legislation-and-Politics/Legislative-Alerts/Letter-to-Senators-Urging-Them-to-Support-Cloture-and-Final-Pas
sage-of-the-Paycheck-Fairness-Act-S.2199
        date: September 10, 2014


In Support of the Conference Agreement on H.R. 3080, the Water Resources Reform and Development Act of 2013:
        link: www.aflcio.org/Legislation-and-Politics/Legislative-Alerts/In-Support-of-the-Conference-Agreement-on-H.R.-3080-the-Water-R
esources-Reform-and-Development-Act-of-2013
        date: May 22, 2014


In support of the Moore and Wilson Amendments to the Success and Opportunity through Quality Charter Schools Act:
        link: www.aflcio.org/Legislation-and-Politics/Legislative-Alerts/In-support-of-the-Moore-and-Wilson-Amendments-to-the-Success-an
d-Opportunity-through-Quality-Charter-Schools-Act
        date: May 08, 2014


Letter to Senators in Support of the "Workforce Innovation and Opportunity Act (H.R. 803)":
        link: www.aflcio.org/Legislation-and-Politics/Legislative-Alerts/Letter-to-Senators-in-Support-of-the-Workforce-Innovation-and-O
pportunity-Act-H.R.-803
        date: June 25, 2014


Letter to Senators Harkin and Alexander in Support of the "Workforce Innovation and Opportunity Act":
        link: www.aflcio.org/Legislation-and-Politics/Legislative-Alerts/Letter-to-Senators-Harkin-and-Alexander-in-Support-of-the-Workf
orce-Innovation-and-Opportunity-Act
        date: June 04, 2014


Letter to Senators in Support of the Klobuchar-Coats-Schatz-Blunt Amendment:
        link: www.aflcio.org/Legislation-and-Politics/Legislative-Alerts/Letter-to-Senators-in-Support-of-the-Klobuchar-Coats-Schatz-Blu
nt-Amendment
        date: June 18, 2014


In Opposition of the Republican FY 2015 Budget Resolution :
        link: www.aflcio.org/Legislation-and-Politics/Legislative-Alerts/In-Opposition-of-the-Republican-FY-2015-Budget-Resolution
        date: April 09, 2014


Letter to Senators in Support of the Medicare Protection Act (S. 2491):
        link: www.aflcio.org/Legislation-and-Politics/Legislative-Alerts/Letter-to-Senators-in-Support-of-the-Medicare-Protection-Act-S.
-2491
        date: June 25, 2014


Letter to Senators Urging Them to Vote Yes on the Motion to Proceed to the Emergency Supplemental Appropriations Act of 2014 (S.2648):
        link: www.aflcio.org/Legislation-and-Politics/Legislative-Alerts/Letter-to-Senators-Urging-Them-to-Vote-Yes-on-the-Motion-to-Pro
ceed-to-the-Emergency-Supplemental-Appropriations-Act-of-2014-S.2648
        date: July 30, 2014


Letter to Representatives in Opposition to the Meadows Amendment to the FY 2015 Financial Services Appropriations Bill (H.R. 5016):
        link: www.aflcio.org/Legislation-and-Politics/Legislative-Alerts/Letter-to-Representatives-in-Opposition-to-the-Meadows-Amendmen
t-to-the-FY-2015-Financial-Services-Appropriations-Bill-H.R.-5016
```

```
        date: July 15, 2014


Letter to Representatives in Opposition to the Child Tax Credit Improvement Act of 2014 (H.R. 4935). :
        link: www.aflcio.org/Legislation-and-Politics/Legislative-Alerts/Letter-to-Representatives-in-Opposition-to-the-Child-Tax-Credit
-Improvement-Act-of-2014-H.R.-4935-.
        date: July 23, 2014


In Opposition of the Expatriate Health Coverage Clarification Act:
        link: www.aflcio.org/Legislation-and-Politics/Legislative-Alerts/In-Opposition-of-the-Expatriate-Health-Coverage-Clarification-A
ct
        date: April 08, 2014


Letter to Representatives in Support of the "Workforce Innovation and Opportunity Act (H.R. 803)":
        link: www.aflcio.org/Legislation-and-Politics/Legislative-Alerts/Letter-to-Representatives-in-Support-of-the-Workforce-Innovatio
n-and-Opportunity-Act-H.R.-803
        date: July 08, 2014


In Opposition of the Save American Workers Act (H.R. 2575):
        link: www.aflcio.org/Legislation-and-Politics/Legislative-Alerts/In-Opposition-of-the-Save-American-Workers-Act-H.R.-2575
        date: April 01, 2014


Letter to representatives opposing the elimination of Saturday mail delivery to replenish the federal highway trust fund.:
        link: www.aflcio.org/Legislation-and-Politics/Legislative-Alerts/Letter-to-representatives-opposing-the-elimination-of-Saturday-
mail-delivery-to-replenish-the-federal-highway-trust-fund
        date: June 05, 2014


Letter to Representatives Urging Passage of a Multiyear Reauthorization of the Highway Trust Fund:
        link: www.aflcio.org/Legislation-and-Politics/Legislative-Alerts/Letter-to-Representatives-Urging-Passage-of-a-Multiyear-Reautho
rization-of-the-Highway-Trust-Fund
        date: July 15, 2014


In Support of the Paycheck Fairness Act (S.84):
        link: www.aflcio.org/Legislation-and-Politics/Legislative-Alerts/In-Support-of-the-Paycheck-Fairness-Act-S.84
        date: April 03, 2014


Letter to Representatives in Support of the Reauthorization of the Export-Import Bank:
        link: www.aflcio.org/Legislation-and-Politics/Legislative-Alerts/Letter-to-Representatives-in-Support-of-the-Reauthorization-of-
the-Export-Import-Bank
        date: June 24, 2014


Letter to Representatives Urging Them to Vote No on the Legislation Providing Supplemental Appropriations for the Fiscal Year Ending Sep
t. 30, 2014:
        link: www.aflcio.org/Legislation-and-Politics/Legislative-Alerts/Letter-to-Representatives-Urging-Them-to-Vote-No-on-the-Legisla
tion-Providing-Supplemental-Appropriations-for-the-Fiscal-Year-Ending-Sept.-30-2014
        date: July 30, 2014


Letter to Representative Cummings expressing concern about possible legislation that would curtail currently permissible union activity
by Administrative Law Judges:
        link: www.aflcio.org/Legislation-and-Politics/Legislative-Alerts/Letter-to-Representative-Cummings-expressing-concern-about-poss
ible-legislation-that-would-curtail-currently-permissible-union-activity-by-Administrative-Law-Judges
        date: June 20, 2014


Letter to Senators in Support of the Protect Women’s Health from Corporate Interference Act (S. 2578):
        link: www.aflcio.org/Legislation-and-Politics/Legislative-Alerts/Letter-to-Senators-in-Support-of-the-Protect-Women-s-Health-fro
m-Corporate-Interference-Act-S.-2578
        date: July 14, 2014


In Support of the Protecting Volunteer Firefighters and Emergency Responders Act of 2014:
        link: www.aflcio.org/Legislation-and-Politics/Legislative-Alerts/In-Support-of-the-Protecting-Volunteer-Firefighters-and-Emergen
cy-Responders-Act-of-2014
        date: April 06, 2014


Letter to Representatives Kline and Miller in Support of the "Workforce Innovation and Opportunity Act":
        link: www.aflcio.org/Legislation-and-Politics/Legislative-Alerts/Letter-to-Representatives-Kline-and-Miller-in-Support-of-the-Wo
rkforce-Innovation-and-Opportunity-Act
        date: June 04, 2014


In Support of the Water Resources Reform and Development Act of 2013:
        link: www.aflcio.org/Legislation-and-Politics/Legislative-Alerts/In-Support-of-the-Water-Resources-Reform-and-Development-Act-of
-2013
        date: May 19, 2014


Urging Representatives to Oppose the Save American Workers Act and the Forty Hours is Full Time Act:
        link: www.aflcio.org/Legislation-and-Politics/Legislative-Alerts/Urging-Representatives-to-Oppose-the-Save-American-Workers-Act-
and-the-Forty-Hours-is-Full-Time-Act
```

```
        date: January 28, 2014


In support of workforce management accountability efforts during consideration of the 2015 National Defense Authorization Act:
        link: www.aflcio.org/Legislation-and-Politics/Legislative-Alerts/In-support-of-workforce-management-accountability-efforts-durin
g-consideration-of-the-2015-National-Defense-Authorization-Act
        date: May 21, 2014
```

Writing to File

Now we can save it for later use. Typically you will use the `DictWriter` class of the `csv` module to write nested dictionaries, however if you want to write the keys as well as the values, you can't use that package. Because we want our keys in this case, we'll use the more general version.

In [17]:
```
import os, csv
os.chdir("/Users/franceszlotnick/Dropbox/TextAsData/Sections/tutorials/")

with open("lobbying.csv", "w") as toWrite:
    writer = csv.writer(toWrite, delimiter=",")
    writer.writerow(["name", "link", "date"])
    for a in lobbying.keys():
        writer.writerow([a.encode("utf-8"), lobbying[a]["link"], lobbying[a]["date"]])
```

A better format for recording nested dictionaries is JSON (Java Script Object Notation). It is trivially easy to write this type of structure with nested dictionaries in python. This is a format you should get comfortable with, because this is how most data from APIs is provided. But we'll save that for another tutorial!

In [18]:
```
import json

with open("lobbying.json", "w") as writeJSON:
    json.dump(lobbying, writeJSON)
```