# Web Scraping

## Part #1: Scraping the Fremd Webpage

In this notebook we will look at 2 different ways of using Python to gain access to an HTML page (web page) in order to find information.

### Method #1: Exploring Webpages Using the 'requests' Library ¶

In [1]:
```
import requests
```

```
**Question \#1:** What is the *requests* library used for?

*Note: In order to answer this question, you will need to read the first page
of the documentation found at the following link:
 https://pypi.python.org/pypi/requests/*

Your Answer: Allows you to send HTTP files to places easily
```

Now that you know what the *requests* library is used for, we will it to get the data from the Fremd Wikipedia page:

In [2]:
```
# Get the Wikipedia Fremd page, store in the variable req
req = requests.get("https://en.wikipedia.org/wiki/William_Fremd_High_School")
```

```
**Question \#2**: Print the value of *req* to see what data is stored in this
variable. Then use the *type* method to print the data type stored in *req*.
What do you see from the output of these two print statements?

Your Answer: 1st one says tghat it is response 200, and the second one says req
is a response model
```

In [3]:
```
# Your first print statement here to print value of 'req':
print(req)
```

```
<Response [200]>
```

In [4]:
```
# Your second print statement here to print the type of data 'req' holds:
type(req)
```

Out[4]:
```
requests.models.Response
```

```
**Question \#3:** Look at the directory of req by typing in *dir(req)*.  What
do you think this directory shows you?

Your Answer: All the codes and special stuff in the webpage pulled
```

In [5]:
```
# Your code here
dir(req)
```

Out[5]:
```
['__attrs__',
 '__bool__',
 '__class__',
 '__delattr__',
 '__dict__',
 '__dir__',
 '__doc__',
 '__enter__',
 '__eq__',
 '__exit__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getstate__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__iter__',
 '__le__',
 '__lt__',
 '__module__',
 '__ne__',
 '__new__',
 '__nonzero__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__setattr__',
 '__setstate__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 '__weakref__',
 '_content',
 '_content_consumed',
 '_next',
 'apparent_encoding',
 'close',
 'connection',
 'content',
 'cookies',
 'elapsed',
 'encoding',
 'headers',
 'history',
 'is_permanent_redirect',
 'is_redirect',
 'iter_content',
 'iter_lines',
 'json',
 'links',
 'next',
 'ok',
 'raise_for_status',
 'raw',
 'reason',
 'request',
 'status_code',
 'text',
 'url']
```

```
**Question \#4:**  Add a print statement to the cell below to look at the
actual webpage text. Do you see some familiar text from the work we did with
HTML/CSS/Javascript first semester?  List 3 or 4 things you can pick out from
the text that look familiar to you:

Your Answer: 1. <title>
             2. <script>
             3. <div>
             4. <body>
             5. <html>
```

In [7]:

```python
w_page=req.text

# print the value of w_page here
print(w_page)
```

```
11-13</span></span>.</cite><span title="ctx_ver=Z39.88-2004&amp;rft_val_fmt=i
nfo%3Aofi%2Ffmt%3Akev%3Amtx%3Abook&amp;rft.genre=unknown&amp;rft.btitle=%22Yo
nkers%22+%28Single+Version%29+-+Prefixmag.com&amp;rft.au=Andrew_Martin&amp;rf
t_id=http%3A%2F%2Fwww.prefixmag.com%2Fmedia%2Ftyler-the-creator%2Fyonkers-sin
gle-version%2F49494%2F&amp;rfr_id=info%3Asid%2Fen.wikipedia.org%3AWilliam+Fre
md+High+School" class="Z3988"></span><link rel="mw-deduplicated-inline-style"
href="mw-data:TemplateStyles:r951705291"/></span>
</li>
<li id="cite_note-43"><span class="mw-cite-backlink"><b><a href="#cite_ref-4
3">^</a></b></span> <span class="reference-text"><cite class="citation web"><
a rel="nofollow" class="external text" href="http://www.xxlmag.com/the-break/
2011/10/the-break-presents-brandun-deshay/">"The Break Presents: brandUn DeSh
ay - XXL"</a>. <i>XXL Mag</i>.</cite><span title="ctx_ver=Z39.88-2004&amp;rft
_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Ajournal&amp;rft.genre=unknown&amp;rf
t.jtitle=XXL+Mag&amp;rft.atitle=The+Break+Presents%3A+brandUn+DeShay+-+XXL&am
p;rft_id=http%3A%2F%2Fwww.xxlmag.com%2Fthe-break%2F2011%2F10%2Fthe-break-pres
ents-brandun-deshay%2F&amp;rfr_id=info%3Asid%2Fen.wikipedia.org%3AWilliam+Fre
md+High+School" class="Z3988"></span><link rel="mw-deduplicated-inline-style"
href="mw-data:TemplateStyles:r951705291"/></span>
```

**Task #1:** Use the *requests* library again, but this time get the data from the actual FHS webpage, http://fhs.d211.org/ (http://fhs.d211.org/). Store this data in the variable *req2*.

In [8]:

```python
# Get the FHS page, http://fhs.d211.org/, and store as req2
# HINT: There is useful code in the second cell!
req2 = requests.get("http://fhs.d211.org/")
```

For the AP test, you may want to know the names of the various pieces of the URL http://fhs.d211.org (http://fhs.d211.org). Here they are:

- .org: top-level domain
  - can be .com, .org, .net, and many more
- d211: second-level domain
  - a second-level domain must be registered
  - the d211 second-level domain is unique to District 211 webpages
- fhs: subdomain
  - fhs is subdomain of d211
  - this structure reflects the fact that Fremd is a part of District 211
- http: the protocol for data exchange
  - you learned a lot about this protocol last semester

## Metadata from the Fremd Webpage

We can look at information about the FHS web page (metadata) by accessing the headers property of the Response object:

In [9]:
```
print(req2.headers)
```

```
{'Date': 'Mon, 27 Apr 2020 14:45:08 GMT', 'Content-Type': 'text/html; charset=u
tf-8', 'Transfer-Encoding': 'chunked', 'Connection': 'keep-alive', 'Cache-Contr
ol': 'private', 'Content-Encoding': 'gzip', 'Vary': 'Accept-Encoding', 'Serve
r': 'Microsoft-IIS/8.5', 'Strict-Transport-Security': 'max-age=31536000; includ
eSubDomains;', 'X-XSS-Protection': '1; mode=block', 'X-AspNet-Version': '4.0.30
319', 'Set-Cookie': 'PSN=+IsCigvKEHFzYB1hRU3cjA==; path=/; secure; HttpOnly, PS
DB=get+ikDKUSgz5GSCEtbxNpOV6Ak/U0I1RXwREP4/87E=; path=/; secure; HttpOnly, Acco
untID=Xogon24LhVEF1Gfd40nUZQ==; path=/; secure; HttpOnly, APIKey=6bbe9abb-2ca5-
42ca-ac9c-a4a52d8c9ccb; path=/; secure; HttpOnly, SWSessionID=2391eaac-0d4a-468
e-8f30-77d382255533; path=/; secure; HttpOnly, RedirectTo=http%3A%2F%2Fadc.d21
1.org%2Fsite%2Fdefault.aspx%3FDomainID%3D9; path=/; secure, CancelRedirectTo=;
expires=Mon, 27-Apr-2020 06:45:08 GMT; path=/; secure', 'X-Powered-By': 'ASP.NE
T', 'X-Frame-Options': 'SAMEORIGIN'}
```

```
**Question \#5:**  What do you see in the metadata above? What domain is this
web page hosted by?

Your Answer: I see a binch of meta data, like time, date, and the page is
hosted by ASP.net
```

## Data from the Fremd Webpage

Files are comprised of data and metadata (data about the data). You saw the metadata for the Fremd webpage above. Now here's the main data:

In [10]:
```
fhs_page = req2.text   # Save the text of the webpage in a variable
print(fhs_page)
```

```
                    e.preventDefault();

                    // This is the first item
                    if ($(this).prev('.sw-mystart-nav').length == 0) {
                        $(this).parents('div').find('.sw-mystart-nav').last().foc
us();
                    } else {
                        $(this).prev('.sw-mystart-nav').focus();
                    }
                } else if (e.keyCode == 38) { //key up
                    e.preventDefault();

                    // show school dropdown
                    if ($(this).find('ul').length > 0) {
                        $(this).find('div.sw-dropdown').css('display', 'block').f
ind('ul').attr('aria-hidden', 'false').find('a').attr('tabIndex', 0).last().f
ocus();
                    }
```

```
**Question \#6:**  What do you see in the output above? What is the type of
data stored in fhs_page?

*Note: Use the *type* method to answer the second part of this question.*

Your Answer: strings/html
```

In [12]:
```
# Your code here to print the type of data 'fhs_page' holds:
type(fhs_page)
```

Out[12]:
```
str
```

To find specific instances of HTML tags in fhs_page we would now need to use methods of the string object. While this can be done, there is a better and more efficient way to traverse our way through a web page. String manipulation alone often involves the use of regular expressions (and the Python *re* module).

Regular expressions are used in many different programming languages (including Javascript). Regular expressions are very useful, but can also get quite complicated when used properly. You will see an example of using regular expressions and the *re* module later in this notebook.

```
**Question \#7:**  What is a "regular expression" and what is it used for?

*Note: You will need to refer to
https://docs.python.org/3/howto/regex.html#regex-howto for more information
about regular expressions.*

Your Answer: A code within python that looks for patterns withing strings
```

We will now look at a second way we can gain access to HTML pages for manipulation. This is a more elegant (and much simpler) way to access different parts of a web page than using the builtin Requests library:

## Method #2: Exploring Webpages Using the Beautiful Soup Library

A library for easily getting data out of HTML and XML files.

```
**Question \#8:** Visit the Beautiful Soup documentation at this link:
 https://www.crummy.com/software/BeautifulSoup/. What are the three features
that make Beautiful Soup so powerful?

Your Answer:
1. Helps you manipulate parse trees
2. Converts incomeing documents into Unicode and outgoing to UTF-8
3. Allows you to try diffrent parsing stratigies
```

In [13]:
```
from bs4 import BeautifulSoup  # Import BeautifulSoup
```

Now we will use Beautiful Soup to parse a web page document. To get a web page into our notebook we can either have BS4 read in an html file from our root directory or, in this case, we can just use the string we created earlier when we opened the FHS page using the Requests library (stored in fhsPage):

```
**Question \#9:** Add a print statement to the code below. What similarities do
you notice with *print(fhs_page)* from earlier?

Your Answer: Both are code, and show information in the form of strings to us
in the python cell
```

In [14]:
```
fhs_soup = BeautifulSoup(fhs_page, 'html.parser')   # Beautiful Soup will allow y

# Print the contents of 'fhs_soup' to see what it looks like
print(fhs_soup)
```

```
        });

    // ADA SKIP NAV
    $(document).ready(function () {
        $(document).on('focus', '#skipLink', function () {
            $("div.sw-skipnav-outerbar").animate({
                marginTop: "0px"
            }, 500);
        });


        $(document).on('blur', '#skipLink', function () {
            $("div.sw-skipnav-outerbar").animate({
                marginTop: "-30px"
            }, 500);
        });
    });
```

**Question #10:** What data type is stored in *fhs_soup*? You will need to write code in the cell below to answer this question.

Your Answer:

In [15]:
```
# Your code here to print they type of data 'fhs_soup' is holding:
type(fhs_soup)
```

Out[15]:
```
bs4.BeautifulSoup
```

**Task #2:** Now we will look at some of the properties and methods of the BeautifulSoup object. For each one, write a code comment explaining what you think the method does (or what the property tells us).

In [16]:
```
print(fhs_soup.prettify())    # Makes it more readable
```

```
  <style type="text/css">
   /* MedaiBegin Standard *//* GroupBegin Font Icons */
@font-face {
  font-family: 'hsd211-icons';
  src: url("data:application/x-font-ttf;charset=utf-8;base64,AAEAAAALAIAAAwAw
T1MvMg8SBiAAAAC8AAAAYGNtYXAXVtKYAAABHAAAAFRnYXNwAAAAEAAAAXAAAAAIZ2x5ZuNV14AAA
AF4AAAUxGhlYWQRiI59AAAWPAAAADZoaGVhB8EDzQAAFnQAAAAkaG10eE4BArsAABaYAAAAWGxvY2
E9XDlkAAAW8AAAAC5tYXhwAB4B6gAAFyAAAAAgbmFtZSDk+iUAABdAAAABwnBvc3QAAwAAAAAZBAA
AACAAAwPlAZAABQAAApkCzAAAAI8CmQLMAAAB6wAzAQkAAAAAAAAAAAAAAAAAAAABEAAAAAAAAAAA
AAAAAAAAAABAAADpEQPA/8AAQAPAAEAAAAABAAAAAAAAAAAAAAAgAAAAAAADAAAAAwAAABwAAQADA
AAAHAADAAEAAAAcAAQAOAAAAAoACAACAAIAAQAg6RH//f//AAAAAAAg6QD//f//AAH/4xcEAAMAAQ
AAAAAAAAAAAAAAAQAB//8ADwABAAAAAAAAAAAAAgAANzkBAAAAAAEAAAAAAAAAAAACAAA3OQEAAAA
AAQAAAAAAAAAAAAIAADc5AQAAAAAHAAkAZgPyAxoAoADsATIBbAGIAc4B5wAAEzwBNTwBNTQ2Nz4B
Nz4BNz4BNz4BNzoBNxYyMzIWMx4BFx4BFx4BFx4BFx4BBw4BBw4BBw4BBw4BBw4BBw4BBw4BB
w4BBxQGMQ4BBzoBMxwBFSIjKgEjIiM0NjU0NjU+ATc+ATc+ATc+ATc+ATc+ATc+ATc+ATc+ATc+AT
c+ATc+ATc2JicuAScuASciJgciBgcOAQcOAQcOAQcqAQcBKgEjPAE1PAE1PAEnLgEnLgEnNCY1PAE
1NDY3PgE3OgEXHgEXFjI3PgE3NjIXHgEXHgEXHgEVHAEVFAYHDgEHBhQVHAEVHAEVHgEVISoBIzwB
NTwBNTQmJy4BJy4BNTwBNTQ2Nz4BNzYWFx4BFxYyNz4BNz4BFx4BFx4BFx4BFRwBFRQGBw4BBw4BF
RwBFRYUFQE+ATceARcOAQcOARUcARUUBicuATU8ATU8ATUOAQcwBhUUBgcOAQcOAScuAScuAScuAT
```

In [17]:
```
fhs_soup.title  # Gives the title code of the HTML file
```

Out[17]:
```
<title>Fremd HS / Homepage</title>
```

In [18]:
```
len(fhs_soup.find_all("p"))  # finds how many the p's in the code
```

Out[18]:
```
30
```

In [22]:
```
# Write the code to see what is contained in all <p> tags
    # HINT: Use the 'find_all' method above to find all <p> tags (this creates a
    #       what it returns in a variable. Then loop through each tag in the lis
p_tags = fhs_soup.find_all("p")
for tags in p_tags:
    print(tags)
```

```
<p>Home of the Vikings</p>
<p class="ui-article-description"><span class='"s1"' small;&quot;="" style
='"font-size:'><em>Updated: April 20, 2020</em></span></p>
<p class="ui-article-description"><span class='"s1"' small;&quot;="" style
='"font-size:'><em>Updated: April 20, 2020</em></span></p>
<p class="ui-article-description">
<span class="sw-calendar-block-title"><a href="
https://adc.d211.org/site/Default.aspx?PageID=12&amp;DomainID=9#calendar8/202
00427/event/7767">Professional (https://adc.d211.org/site/Default.aspx?PageID
=12&amp;DomainID=9#calendar8/20200427/event/7767">Professional) Development D
ay / No e-Learning / No Extra-Curricular Activities</a></span>
</p>
<p class="ui-article-description">
<span class="sw-calendar-block-title"><a href="
https://adc.d211.org/site/Default.aspx?PageID=12&amp;DomainID=9#calendar8/202
00428/event/7751">District (https://adc.d211.org/site/Default.aspx?PageID=12&
amp;DomainID=9#calendar8/20200428/event/7751">District) 211 e-Learning Day /
 No Extra-Curricular Activities</a></span>
</p>
```

In [19]:
```
fhs_soup.p  # Finds the first p tag in the HTML file
```

Out[19]:
```
<p>Home of the Vikings</p>
```

In [23]:
```
fhs_soup.meta["content"]  # This shows the metadat content in the beggining of
                          # HINT: Can you find this result anywhere in the outpu
```

Out[23]:
```
'website'
```

In [24]:
```
[t["content"] for t in fhs_soup.find_all("meta") if t.get("content")]  # Make a
```

Out[24]:
```
['website',
 '411584262324304',
 'http%3A%2F%2Fadc.d211.org%2Fsite%2Fdefault.aspx%3FDomainID%3D9',
 'Fremd HS / Homepage',
 'Fremd HS / Homepage',
 'Fremd HS / Homepage',
 'width=device-width, initial-scale=1.0']
```

In [25]:
```
fhs_soup.find_all("a")  # find all the code between the <a>'s
```

Out[25]:
```
[<a class="sw-skipnav" href="#sw-maincontent" id="skipLink" tabindex="0">Skip
to Main Content</a>,
 <a alt="District Home" href="https://adc.d211.org/Domain/4" tabindex="0" tit
le="Return to the homepage on the district site."><span>District Home<div id
="sw-home-icon"></div>
 </span></a>,
 <a href="/Domain/8">Palatine HS</a>,
 <a href="/Domain/9">Fremd HS</a>,
 <a href="/Domain/10">Conant HS</a>,
 <a href="/Domain/11">Schaumburg HS</a>,
 <a href="/Domain/12">Hoffman Estates HS</a>,
 <a href="/Domain/13">Higgins Education Center</a>,
 <a href="/Domain/14">North Campus</a>,
 <a href="https://adc.d211.org/site/Default.aspx?PageType=7&amp;SiteID=9&amp;
IgnoreRedirect=true"><span>Sign In</span></a>,
 <a aria-label="Submit Site Search" href="javascript:;" id="sw-search-button"
onclick="SWGoToSearchResultsPageswsearchinput();" role="button" tabindex="0"
title="Search"><span><img alt="Search" src="https://adc.d211.org/Static//glob
alassets/images/sw-mystart-search.png"/></span></a>,
```

In [26]:
```
import re    # This is the regular expressions module that lets you check if a s
```

In [27]:
```
fhs_soup.find_all(string=re.compile("Calendar"))     # Finds all the code with t
```

```
{\n                           \'channelCatSectionHeader\': \'\',\n
\'channelCatSections\': \'4\'\n                   }\n                  ]\n
},\n           {\n               \'channelImg\': \'/cms/lib/IL49000007/Cent
ricity/Template/GlobalAssets/images///Faces/default-man.jpg\',\n
\'channelImgBool\': \'false\',\n                \'channelCats\': [\n
{\n                           \'channelCatSectionHeader\': \'\',\n
\'channelCatSections\': \'4\'\n                   },\n
{\n                           \'channelCatSectionHeader\': \'\',\n
\'channelCatSections\': \'4\'\n                   },\n
{\n                           \'channelCatSectionHeader\': \'\',\n

\'channelCatSections\': \'4\'\n                   },\n
{\n                           \'channelCatSectionHeader\': \'\',\n
\'channelCatSections\': \'4\'\n                   }\n                  ]\n
}\n          ],\n        \n         \n         \n        /*****\n        \tMETHOD
S\n        *****/\n        \n// BIG BANG\n        "Init": function() {\n
this.SetTemplateProps();\n            \n           $(\'[data-hide=""], [data
-hide="!false"], [data-hide="true"]\').remove();\n            \n            t
his.RsMenu();\n            this.MyStart();\n            this.Search();\n
this.Header();\n            this.FindItFast();\n            this.GlobalIcons
();\n                this.CustomDropdowns();\n            this.ChannelBar();\n
```

**Task #3:** Now find a school-appropriate webpage that you visit often. Read it in using the Requests library and then use Beautiful Soup to find all instances of two of the following tags: hyperlink, list, paragraph, style, or another of your choice. Be sure to comment your code.

In [28]:
```
# Website used: https://adc.d211.org/Domain/8

req3 = requests.get("https://adc.d211.org/Domain/8")
phs = req3.text

phs_soup = BeautifulSoup(phs, 'html.parser')

style = fhs_soup.find_all("style")

hyperlink = fhs_soup.find_all("hyperlink")

print(hyperlink)
```

```
[]
```

In [29]:
```
print(style)
```

```
#gb-main-footer-top .gb-row.two > .flex-cont p > span[id*="graphic"]{
        font: italic 500 16px/1 'Fira Sans', sans-serif;
        color: #004500;
    padding-right: 11px;
}
#gb-main-footer-top .gb-row.two > .flex-cont p a{
        color: #4D4D4D;
}
#gb-main-footer-top p span[id*="graphic"] + span{
        position: relative;
    top: -1px;
}
#gb-main-footer-top p span[id*="graphic"] i{
        font-size: 48px;
    line-height: 48px;
}
#gb-main-footer-top p span[id*="graphic"] i[class*="phone"]{
        font-size: 36px;
}
#gb-main-footer-top .gb-row.one{
```