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]:
```python
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: It allows me to send http requests which I think are similiar to me searching on google.

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]:
```python
# 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: That there are 200 request.model.Response on the https://en.wikipedia.org/wiki/William_Fremd_High_School.

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

Out[3]:
```
<Response [200]>
```

In [4]:
```python
# 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: I think that dir gives you the methods of req.

In [5]:
```python
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: This shows the code of the website. Like how we learned that we send a request and then we recieve the webpage code. Then we display it. Instead here we just get the code. I can see things like script tags, links, and classes. If you go the actual site and view the source.

In [6]:
```python
w_page=req.text

# print the value of w_page here
w_page[:1000]
```

Out[6]:
```
'<!DOCTYPE html>\n<html class="client-nojs" lang="en" dir="ltr">\n<head>\n<meta charset="UTF-8"/>\n<title>William Fremd High School - Wikipedia</title>\n<script>document.documentElement.className="client-js";RLCONF={"wgBreakFrames":false,"wgSeparatorTransformTable":["",""],"wgDigitTransformTable":["",""],"wgDefaultDateFormat":"dmy","wgMonthNames":["","January","February","March","April","May","June","July","August","September","October","November","December"],"wgRequestId":"602bf110-342c-4229-b3f1-4d563d83248c","wgCSPNonce":false,"wgCanonicalNamespace":"","wgCanonicalSpecialPageName":false,"wgNamespaceNumber":0,"wgPageName":"William_Fremd_High_School","wgTitle":"William Fremd High School","wgCurRevisionId":1083479842,"wgRevisionId":1083479842,"wgArticleId":1258351,"wgIsArticle":true,"wgIsRedirect":false,"wgAction":"view","wgUserName":null,"wgUserGroups":["*"],"wgCategories":["CS1 errors: generic name","CS1 errors: URL","All articles with dead external links","Articles with dead external link'
```

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

In [7]:
```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 learned the names of the various pieces of the URL 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 [8]:
```python
# /r/n show line breaks I think
# Also if you do ctr + i you get different results than crt + u. The text you get from the request seems to come from ctr + u.
# My only reason I can think of is that js isn't loaded?
req2.text[:1000]
```

Out[8]:
```
'\r\n\r\n<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">\r\n\r\n<html lang="en">\r\n<head>\r\n    <title>Fremd HS / Homepage</title>\r\n    <!--\r\n    <PageMap>\r\n    <DataObject type="document">\r\n    <Attribute name="siteid">9</Attribute>\r\n    </DataObject>\r\n    </PageMap>\r\n    -->\r\n\r\n    \r\n    <meta property="og:type" content="website" />\r\n<meta property="fb:app_id" content="411584262324304" />\r\n<meta property="og:url" content="http%3A%2F%2Fadc.d211.org%2Fsite%2Fdefault.aspx%3FDomainID%3D9" />\r\n<meta property="og:title" content="Fremd HS / Homepage" />\r\n<meta name="twitter:card" value="summary" />\r\n<meta name="twitter:title" content="Fremd HS / Homepage" />\r\n<meta itemprop="name" content="Fremd HS / Homepage" />\r\n\r\n    <!-- Begin swuc.GlobalJS -->\r\n<script type="text/javascript">\r\n staticURL = "https://adc.d211.org/Static/";\r\n SessionTimeout = "50";\r\n BBHelpURL = "";\r\n</script>\r\n<!-- End swuc.GlobalJS -->\r\n\r\n    <script src=\'https://adc.d211.org/S'
```

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

```
{'Content-Type': 'text/html; charset=utf-8', 'Content-Length': '392699', 'Connection': 'keep-alive', 'Date': 'Mon, 23 May 2022 03:43:37 GMT', 'Cache-Control': 'no-cache, no-store', 'Pragma': 'no-cache', 'Expires': '-1', 'Server': 'Microsoft-IIS/8.5', 'Strict-Transport-Security': 'max-age=31536000; includeSubDomains;', 'X-XSS-Protection': '1; mode=block', 'X-AspNet-Version': '4.0.30319', 'Set-Cookie': 'PSN=+IsCigvKEHFzYB1hRU3cjA==; path=/; secure; HttpOnly, PSDB=get+ikDKUSgz5GSCEtbxNpOV6Ak/U0I1RXwREP4/87E=; path=/; secure; HttpOnly, CSAN=46imUsew8wdbkuBsFQu0pQ==; path=/; secure; HttpOnly, AccountID=Xogon24LhVEF1Gfd40nUZQ==; path=/; secure; HttpOnly, APIKey=6bbe9abb-2ca5-42ca-ac9c-a4a52d8c9ccb; path=/; secure; HttpOnly, SWSessionID=96197a67-3996-49d6-9220-bb884131fe18; path=/; secure; HttpOnly, RedirectTo=http%3A%2F%2Fadc.d211.org%2Fsite%2Fdefault.aspx%3FDomainID%3D9; path=/; secure, CancelRedirectTo=; expires=Sun, 22-May-2022 19:43:37 GMT; path=/; secure', 'X-Powered-By': 'ASP.NET', 'Content-Security-Policy': "frame-ancestors 'self' https://*.ally.ac;", 'X-Frame-Options': 'SAMEORIGIN', 'X-Cache': 'Miss from cloudfront', 'Via': '1.1 4cbb89cd343b8f6e6698aa5a9e2ca87e.cloudfront.net (CloudFront)', 'X-Amz-Cf-Pop': 'ORD51-C4', 'X-Amz-Cf-Id': 'HsowRPEdxcXQ2PlvmnfzEdaNCWzueMjtH-ylNgkBY3yiET1mBtQBTg=='}
```

Question \#5: What do you see in the metadata above? What URLs can find in these headers? Can you learn anything about these URLs by searching Google?

Your Answer: I don't really see urls but I do see something called cloudfront. Which when I searched tells me it's a content delivery network by amazon.

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]:
```python
fhs_page = req2.text   # Save the text of the webpage in a variable
# Since I didn't use print before it gave me in raw text form with /r and /n I think.
print(fhs_page[:1000])
```

```


<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">

<html lang="en">
<head>
    <title>Fremd HS / Homepage</title>
    <!--
    <PageMap>
    <DataObject type="document">
    <Attribute name="siteid">9</Attribute>
    </DataObject>
    </PageMap>
    -->


    <meta property="og:type" content="website" />
<meta property="fb:app_id" content="411584262324304" />
<meta property="og:url" content="http%3A%2F%2Fadc.d211.org%2Fsite%2Fdefault.aspx%3FDomainID%3D9" />
<meta property="og:title" content="Fremd HS / Homepage" />
<meta name="twitter:card" value="summary" />
<meta name="twitter:title" content="Fremd HS / Homepage" />
<meta itemprop="name" content="Fremd HS / Homepage" />

    <!-- Begin swuc.GlobalJS -->
<script type="text/javascript">
 staticURL = "https://adc.d211.org/Static/";
 SessionTimeout = "50";
 BBHelpURL = "";
</script>
<!-- End swuc.GlobalJS -->

    <script src='https://adc.d211.org/S
```

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: It return the code of the site as a string.

In [11]:
```python
# Your code here to print the type of data 'fhs_page' holds:
type(fhs_page)
```

Out[11]:
```
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: It matches certain patterns of text. I used it for my pt project.

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 built-in 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. Beautiful Soup provides a few simple methods and Pythonic idioms for navigating, searching, and modifying a parse tree: a toolkit for dissecting a document and extracting what you need. It doesn't take much code to write an application
2. Beautiful Soup automatically converts incoming documents to Unicode and outgoing documents to UTF-8. You don't have to think about encodings, unless the document doesn't specify an encoding and Beautiful Soup can't detect one. Then you just have to specify the original encoding.
3. Beautiful Soup sits on top of popular Python parsers like lxml and html5lib, allowing you to try out different parsing strategies or trade speed for flexibility.

In [12]:
```python
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: It still looks very similar to the previous string.

In [13]:
```python
fhs_soup = BeautifulSoup(fhs_page, 'html.parser')   # Beautiful Soup will allow you to pass in an HTML page as a string

# Print the contents of 'fhs_soup' to see what it looks like
print(fhs_page[:1000])
```

```


<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">

<html lang="en">
<head>
    <title>Fremd HS / Homepage</title>
    <!--
    <PageMap>
    <DataObject type="document">
    <Attribute name="siteid">9</Attribute>
    </DataObject>
    </PageMap>
    -->


    <meta property="og:type" content="website" />
<meta property="fb:app_id" content="411584262324304" />
<meta property="og:url" content="http%3A%2F%2Fadc.d211.org%2Fsite%2Fdefault.aspx%3FDomainID%3D9" />
<meta property="og:title" content="Fremd HS / Homepage" />
<meta name="twitter:card" value="summary" />
<meta name="twitter:title" content="Fremd HS / Homepage" />
<meta itemprop="name" content="Fremd HS / Homepage" />

    <!-- Begin swuc.GlobalJS -->
<script type="text/javascript">
 staticURL = "https://adc.d211.org/Static/";
 SessionTimeout = "50";
 BBHelpURL = "";
</script>
<!-- End swuc.GlobalJS -->

    <script src='https://adc.d211.org/S
```

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: It is a bs4.BeautifulSoup.

In [14]:
```python
# Your code here to print they type of data 'fhs_soup' is holding:
type(fhs_soup)
```

Out[14]:
```
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 [15]:
```python
print(fhs_soup.prettify()[:1000])   # I think it seperates the tags and makes the code easy to read.
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">
<html lang="en">
 <head>
  <title>
   Fremd HS / Homepage
  </title>
  <!--
    <PageMap>
    <DataObject type="document">
    <Attribute name="siteid">9</Attribute>
    </DataObject>
    </PageMap>
    -->
  <meta content="website" property="og:type"/>
  <meta content="411584262324304" property="fb:app_id"/>
  <meta content="http%3A%2F%2Fadc.d211.org%2Fsite%2Fdefault.aspx%3FDomainID%3D9" property="og:url"/>
  <meta content="Fremd HS / Homepage" property="og:title"/>
  <meta name="twitter:card" value="summary"/>
  <meta content="Fremd HS / Homepage" name="twitter:title"/>
  <meta content="Fremd HS / Homepage" itemprop="name"/>
  <!-- Begin swuc.GlobalJS -->
  <script type="text/javascript">
   staticURL = "https://adc.d211.org/Static/";
 SessionTimeout = "50";
 BBHelpURL = "";
  </script>
  <!-- End swuc.GlobalJS -->
  <script src="https://adc.d211.org/Static/GlobalAssets/Scrip
```

In [16]:
```python
fhs_soup.title  # This gets me the title tag
```

Out[16]:
```
<title>Fremd HS / Homepage</title>
```

In [17]:
```python
len(fhs_soup.find_all("p"))  # It has an array of all the paragrpah tags.
print(fhs_soup.find_all("p")[:5])
```

```
[<p>Home of the Vikings</p>, <p class="ui-article-description">
<span class="sw-calendar-block-time">8:30 AM</span>
<span class="sw-calendar-block-title"><a href="
https://adc.d211.org/site/Default.aspx?PageID=12&amp;DomainID=9#calendar8/20220523/event/14277">Senior Breakfast</a></span>
</p>, <p class="ui-article-description">
<span class="sw-calendar-block-time">7:30 PM</span>
<span class="sw-calendar-block-title"><a href="
https://adc.d211.org/site/Default.aspx?PageID=12&amp;DomainID=9#calendar8/20220523/event/13556">Graduation</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#calendar8/20220525/event/14121">Final Exams</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#calendar8/20220526/event/14122">Final Exams</a></span>
</p>]
```

In [18]:
```python
# 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 list of <p> tags) and store
    #       what it returns in a variable. Then loop through each tag in the list and print each one individually
p_tags = fhs_soup.find_all("p")
for p in p_tags:
    print(p)
    break
```

```
<p>Home of the Vikings</p>
```

In [19]:
```python
fhs_soup.p  # This just gets 1 paragraph tag and stops like queryselector. It finds the first paragraph tag it can find.
```

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

In [20]:
```python
fhs_soup.meta["content"]  # It finds the specific tag then it displays what the attribute is.
                          # <meta property="og:type" content="website" />
```

Out[20]:
```
'website'
```

In [21]:
```python
[t["content"] for t in fhs_soup.find_all("meta") if t.get("content")]  # Make a list of 'content' for each <meta> tag

#for t in fhs_soup.find_all("meta"):
#    print(t.get("content"))
#    #print(t["content"])

# Go through all the meta tags and if there is an attribute of 'content' then add t['content'] to the array
```

Out[21]:
```
['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 [22]:
```python
fhs_soup.find_all("a")[:5]  # Gets all link tags
```

Out[22]:
```
[<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" title="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>]
```

In [23]:
```python
import re    # This is the regular expressions module that lets you check if a string matches a general format
```

In [24]:
```python
fhs_soup.find_all(string=re.compile("Calendar"))[1]
#fhs_soup.find_all(string=re.compile('Calendar'))[3]
# It finds the string calendar then adds everythign in between the tags it's between to the array
# <a class='view-calendar-link' href="/Page/12"><span>View Calendar</span></a>
```

Out[24]:
```
'Athletics Calendar'
```

In [25]:
```python
len(fhs_soup.find_all(string=re.compile("Calendar")))
```

Out[25]:
```
7
```

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 [26]:
```python
# Import Required Module
import requests
from bs4 import BeautifulSoup

# Web URL
Web_url = "https://www.reddit.com"

# Get URL Content
r = requests.get(Web_url)
print(r.text[:1000])
# Parse HTML Code
soup = BeautifulSoup(r.content, 'html.parser')
print(soup.prettify()[:1000])
```

```
    <!DOCTYPE html>
    <html lang="en-US">
      <head>
        <script>
    var __SUPPORTS_TIMING_API = typeof performance === 'object' && !!performance.mark && !! performance.measure && !!performance.getEntriesByType;
    function __perfMark(name) { __SUPPORTS_TIMING_API && performance.mark(name); };
    var __firstPostLoaded = false;
    function __markFirstPostVisible() {
      if (__firstPostLoaded) { return; }
      __firstPostLoaded = true;
      __perfMark("first_post_title_image_loaded");
    }
    var __firstCommentLoaded = false;
    function __markFirstCommentVisible() {
      if (__firstCommentLoaded) { return; }
      __firstCommentLoaded = true;
      __perfMark("first_comment_loaded");
    }
  </script>
        <script>__perfMark('head_tag_start');</script>
        <meta charSet="utf-8"/>
        <meta name="viewport" content="width=device-width, initial-scale=1" />
        <meta name="referrer" content="origin-when-cross-origin" />
        <style>
  /* http://meyerwe
<!DOCTYPE html>
<html lang="en-US">
 <head>
  <script>
   var __SUPPORTS_TIMING_API = typeof performance === 'object' && !!performance.mark && !! performance.measure && !!performance.getEntriesByType;
    function __perfMark(name) { __SUPPORTS_TIMING_API && performance.mark(name); };
    var __firstPostLoaded = false;
    function __markFirstPostVisible() {
      if (__firstPostLoaded) { return; }
      __firstPostLoaded = true;
      __perfMark("first_post_title_image_loaded");
    }
    var __firstCommentLoaded = false;
    function __markFirstCommentVisible() {
      if (__firstCommentLoaded) { return; }
      __firstCommentLoaded = true;
      __perfMark("first_comment_loaded");
    }
  </script>
  <script>
   __perfMark('head_tag_start');
  </script>
  <meta charset="utf-8"/>
  <meta content="width=device-width, initial-scale=1" name="viewport"/>
  <meta content="origin-when-cross-origin" name="referrer"/>
  <style>
   /* http://meyerweb.com/eric/tools/css/reset/
    v2.0 | 201101
```

In [27]:
```python
soup.find_all('a')[2]
# The reason it's so long is because the code was all on one line.
```

Out[27]:
```
<a class="_3Wg53T10KuuPmyWOMWsY2F _2iuoyPiKHN3kfOoeIQalDT _2tU8R9NTqhvBrhoNAXWWcP HNozj_dKjQZ59ZsfEegz8 _2nelDm85zKKmuD94NequP0" href="https://www.reddit.com/login/?dest=https%3A%2F%2Fwww.reddit.com%2F" role="button" tabindex="0">Log In</a>
```

In [28]:
```python
soup.find_all('a', text = re.compile('Log In'))
```

Out[28]:
```
[<a class="_3Wg53T10KuuPmyWOMWsY2F _2iuoyPiKHN3kfOoeIQalDT _2tU8R9NTqhvBrhoNAXWWcP HNozj_dKjQZ59ZsfEegz8 _2nelDm85zKKmuD94NequP0" href="https://www.reddit.com/login/?dest=https%3A%2F%2Fwww.reddit.com%2F" role="button" tabindex="0">Log In</a>]
```

In [29]:
```python
soup.find_all('p')
```

Out[29]:
```
[<p class="_1qeIAgB0cPwnLhDF9XSiJM">I'm almost 30 it's getting to the point where I'm getting older and I'm still learning.</p>,
 <p class="_1qeIAgB0cPwnLhDF9XSiJM">For example I'm taking voice lessons from someone who's about 10 years younger than me.</p>,
 <p class="_1qeIAgB0cPwnLhDF9XSiJM">I remember how my parents acted when being taught something, or shown something, they would just write it off as something the "kids" can deal with, and unimportant.</p>,
 <p class="_1qeIAgB0cPwnLhDF9XSiJM">I'm doing my best not to do this to other people as I grow up as it leaves the older person unable to function, and the younger person frustrated enough to want to jump through a window.</p>]
```

If I were to webscrape reddit on my computer, when I send a request would it respond with a reddit page on my account since I'm signed into my account. Or would it be as if I was in incognito.