Web Scraping

Part #2: Web Scraping and Web Crawling Using *Beautiful Soup*

The World Wide Web is a potential treasure trove for a data scientist. But in order to access this data, we need to find it and extract it. In this notebook, you we learn about this process by web scraping and web crawling using a Python module called **Beautiful Soup**.

As an introduction to web scraping we will start by opening a page on Wikipedia, find all links to other Wikipedia pages on that page, then pick one of those links (randomly) to 'crawl' to another page. We will continue this process and see where it leads us.

Importing Useful Modules

First we will need to access modules to assist us - some of them are familiar, others are new. Add some code comments to explain what these imports are used for. You do not need to say exactly what we will use them for, just what their general purpose is. You may want to look this up on the Internet.

In [1]:
```
from urllib.request import urlopen          # Gets a URL, and opens it
from bs4 import BeautifulSoup               # Imports the beautiful soup module
import datetime                             # Imports datetime module, which sho
import random                               # Imports random module, which is us
import re                                   # Fins specific strings in code
import time                                 # Imports current time
from IPython.display import Image, display  # Displays an image
```

Get Links from a Wikipedia Page

Let's see how many links to other Wikipedia pages we can find on the Wikipedia page for Fremd High School. First we will define a general function (method) to get links off of any Wikipedia page. Remember that a big advantage of using functions (methods) is that we can reuse code and perform the same task over and over with different inputs but only one group of code statements. In short, this is an **abstraction that helps manage the complexity of our program**.

In our case here we will be able to use this function to find page links for any Wikipedia page and not just FHS's page. This will make it possible (and much more efficient) to perform our random 'crawl' through Wikipedia pages.

In [2]:
```
def get_links(article_url):                                       # Create functio
    html_page = urlopen("http://en.wikipedia.org"+article_url)    # Opens whatever
    bs_obj = BeautifulSoup(html_page, 'html.parser')              # Saves the html

    # The next line finds specific HTML elements in the page we opened - notice t
    return bs_obj.find("div",{"id":"bodyContent"}).findAll("a",href=re.compile("
```

Notice: The return statement above uses a *regular expression*. The *re.compile()* method puts together an expression for the *findAll()* method to match to a certain *href*. The text below briefly explains what the various parts of the regular expression mean.

What does *^(/wiki/)* look for?

```
- The start of a string followed by the literal "/wiki/"
```

What does *((?!:).)** look for?

```
- As many occurences as exist of anything that is not a ":" followed by
 a "."
```

What does *$* look for?

```
- Anything from the current location to the end of the string
```

Counting the Links on Fremd's Wikipedia Page

Next we will call our function, *get_links(articleUrl)*, by sending in the name of Wikipedia's Fremd High School page. The entire url is https://en.wikipedia.org/wiki/William_Fremd_High_School (https://en.wikipedia.org/wiki/William_Fremd_High_School) but we only need to send in the last part of the url since that is how we wrote our function.

In [3]:
```
links=get_links("/wiki/William_Fremd_High_School")
```

Question 4: How many links are on the Fremd High School Wikipedia page? You will need to write code in the cell below to find out.

HINT: 'links' is a list, how can you find out how long it is?

Your Answer: Use len()

In [4]:
```
# Your code here
len(links)
```

Out[4]:
```
190
```

Question 5: What type of data is stored in links? You will need to write code in the cell below to find out.

Your Answer: ResultSet

In [5]:
```
# Your code below
type(links)
```

Out[5]:
```
bs4.element.ResultSet
```

View the Links Stored in the Variable '*links*'

Now let's look at all of these links:

In [6]:
```
print(links)
```

```
[<a href="/wiki/Palatine,_Illinois" title="Palatine, Illinois">Palatine</a>,
<a href="/wiki/Illinois" title="Illinois">Illinois</a>, <a href="/wiki/Geogra
phic_coordinate_system" title="Geographic coordinate system">Coordinates</a>,
<a href="/wiki/Township_High_School_District_211" title="Township High School
District 211">Township H.S. 211</a>, <a href="/wiki/Forest_green" title="Fore
st green">Forest Green</a>, <a href="/wiki/Gold_(color)" title="Gold (colo
r)">Gold</a>, <a href="/wiki/Fight_song" title="Fight song">Fight song</a>, <
a href="/wiki/Mid-Suburban_League" title="Mid-Suburban League">Mid-Suburban L
eague</a>, <a class="mw-redirect" href="/wiki/High_school" title="High schoo
l">high school</a>, <a href="/wiki/Palatine,_Illinois" title="Palatine, Illin
ois">Palatine, Illinois</a>, <a class="mw-redirect" href="/wiki/Chicago,_Illi
nois" title="Chicago, Illinois">Chicago, Illinois</a>, <a href="/wiki/United_
States" title="United States">United States</a>, <a href="/wiki/Township_High
_School_District_211" title="Township High School District 211">Township High
School District 211</a>, <a href="/wiki/James_B._Conant_High_School" title="J
ames B. Conant High School">James B. Conant High School</a>, <a href="/wiki/H
offman_Estates_High_School" title="Hoffman Estates High School">Hoffman Estat
es High School</a>, <a href="/wiki/Palatine_High_School" title="Palatine High
School">Palatine High School</a>, <a href="/wiki/Schaumburg_High_School" titl
```

Clean the List

Now let's clean the page links up a bit. First we will use a list comprehension to create a list of only the href portion of each tag. Then we will look only at the page names by removing the text that appears at the beginning of each link.

In [7]:
```
link_list=[link['href'] for link in links]   # Creates a list of href's for every
for link in link_list:                       # Iterate over the newly formed lis
    link = link.replace('/wiki/','')         # Remove the repetitive text
    print(link)                              # Display the page names one per li
```

```
Eric_Zorn
Jane_Hamilton
Raymond_Benson
Rosellen_Brown
Harry_Mark_Petrakis
Frederik_Pohl
Nikki_Giovanni
Naomi_Shihab_Nye
Poetry_slam
Marc_Smith_(American_poet)
Bill_Kelly_(writer)
Ted_Nugent
Born_of_Osiris
Tomorrow_We_Die_Alive
Eric_Bradley_(musician)
BrandUn_DeShay
Young_Money
Curren$y
Billboard_(magazine)
Mac_Miller
```

Web Crawling

Enough of just looking at the Fremd HS Wikipedia page. Now let's try to 'crawl' through some pages by randomly picking one of the links on the Fremd High School page and following it to see what page links appear on that page. Then we will pick a random link on the new page and continue this process until we have 20 links from 20 pages we've crawled through.

In [8]:
```
import requests
from IPython.display import Image, display

#This next line "seeds" the random number generator at the current time (gives a
random.seed(datetime.datetime.now())

links=get_links("/wiki/William_Fremd_High_School")                          # Extrac
for i in range(20):                                                         # Loop t
    new_article = links[random.randint(0,len(links)-1)].attrs['href']       # Choose
    print(new_article)                                                      # Prints
    links = get_links(new_article)                                          # Extrac
    time.sleep(1)                                                           # 1-seco
```

```
/wiki/Rick_Bragg
/wiki/ISBN_(identifier)
/wiki/Document_Style_Semantics_and_Specification_Language
/wiki/ISO/TR_11941
/wiki/ISO_657
/wiki/ISO_14031
/wiki/Graphical_Kernel_System
/wiki/American_National_Standards_Institute
/wiki/Washington,_D.C.
/wiki/First_Division_Monument
/wiki/Second_Division_Memorial
/wiki/Constitution_Gardens
/wiki/United_States_National_Arboretum
/wiki/Lincoln_Memorial
/wiki/Rock_Creek_and_Potomac_Parkway
/wiki/MacArthur_Boulevard_(Washington,_D.C.)
/wiki/Tenley_Circle
/wiki/Logan_Circle_(Washington,_D.C.)
/wiki/Capitol_Hill_(Denver)
/wiki/Washington,_D.C.
```

One of the reasons we picked Wikipedia pages to crawl through is that by doing web scraping there is the potential to flood a web server with requests. Wikipedia is used to receiving a lot of requests and can generally handle the volume. It is good practice, though, and considered common courtesy to put code into your web scraping program to pause in between requests. The code is a simple "sleep" request. You will see this above with the link of code that reads *time.sleep(1)*.

Scraping Images

We can scrape many kinds of data from webpages. Instead of just links, let's try scraping images. First, here's a function that looks for the URL of the main image for a Wikipedia article:

In [9]:
```
def get_image_url(article_url):
    html_page = urlopen("http://en.wikipedia.org"+article_url)
    bs_obj = BeautifulSoup(html_page, 'html.parser')
    try:
        image_url = bs_obj.find("meta",{"property":"og:image"}).attrs['content']
    except AttributeError:
        image_url = False
    return image_url
```

Test it out with the Fremd High School Wikipedia page:

In [10]:
```
get_image_url("/wiki/William_Fremd_High_School")  # Call to function created abo
```

Out[10]:
```
'https://upload.wikimedia.org/wikipedia/en/0/02/Viking_emblem_%28William_Fremd_
High_School%29.png'
```

Now a function that takes a Wikipedia article URL as input and then performs the following steps:

- Get the URL of the main image on the page
- Saves the image as *output_image*
- Display the image in the notebook

In [11]:
```
def return_image(wiki_url,image_width):                     # Create function tha

    if get_image_url(wiki_url) != False:                    # If there is an avai
        img=get_image_url(wiki_url)                              # Extract image'
        url_to_file = requests.get(img).content                  # Extract image
        extension = img.split('.')[-1]                           # Extract image'
        name = "output/output_image." + extension                # Create new name
        with open(name, 'wb') as image:                          # Create a new f
            image.write(url_to_file)                             # Write the image
        display(Image(filename=name,width=image_width))          # Open image in
```

Let's test *return_image(wiki_url,image_width)* on the Fremd High School Wikipedia page and set an image width of 200 pixels:

In [12]:
```
return_image("/wiki/William_Fremd_High_School", 200)  # Call to function created
```

Go Vikings! Now Let's repeat our web crawl, but now with images (when available).

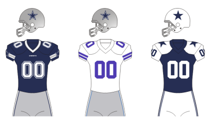
- **NOTES**
  - Remember that this is a random crawl, so it is possible that you could encounter content on Wikipedia that is not school-appropriate. If this happens, simply rerun the cell below:
  - Some links may just display text, while others will also display an image (if present)

In [13]:
```
# This next line "seeds" the random number generator at the current time (gives
random.seed(datetime.datetime.now())

links=get_links("/wiki/William_Fremd_High_School")                          # Extrac
for i in range(20):                                                         # Loop t
    new_article = links[random.randint(0,len(links)-1)].attrs['href']       # Choose
    return_image(new_article,100)                                           # Call f
    print(new_article)                                                      # Prints
    links = get_links(new_article)                                          # Extrac
    time.sleep(1)                                                           # 1-seco
```

```
/wiki/Indianapolis_Colts
```

```
/wiki/2009_NFL_season
/wiki/2003_NFL_season#Major_rule_changes
```



Experiment With a New Wikipedia Page

Now pick a Wikipedia page on a school-appropriate topic you are interested in and then try the following:

Scrape URL links from this page

- Create a 'cleaned up' list of all the links from this wikipage

In [16]:
```
# Your code here
churro = get_links("/wiki/Churro")
```

```
List of fried dough foods</a>,
 <a href="/wiki/List_of_doughnut_varieties" title="List of doughnut varietie
s">List of doughnut varieties</a>,
 <a class="mw-redirect" href="/wiki/Loukoumades" title="Loukoumades">Loukouma
des</a>,
 <a href="/wiki/Puff-puff" title="Puff-puff">Puff-puff</a>,
 <a href="/wiki/Tulumba" title="Tulumba">Tulumba</a>,
 <a href="/wiki/Youtiao" title="Youtiao">Youtiao</a>,
 <a class="mw-redirect" href="/wiki/Zal%C4%81biya" title="Zalābiya">Zalābiya
</a>,
 <a class="mw-redirect" href="/wiki/Zlebia" title="Zlebia">Zlebia</a>,
 <a href="/wiki/Maghreb" title="Maghreb">Maghreb</a>,
 <a href="/wiki/North_Africa" title="North Africa">North Africa</a>,
 <a class="mw-redirect" href="/wiki/The_Huffington_Post" title="The Huffingto
n Post">The Huffington Post</a>,
 <a class="mw-redirect" href="/wiki/ISBN_(identifier)" title="ISBN (identifie
r)">ISBN</a>,
 <a href="/wiki/Doughnut" title="Doughnut">Doughnuts</a>,
 <a href="/wiki/Fritter" title="Fritter">fritters</a>,
 <a href="/wiki/Fried_dough" title="Fried dough">fried-dough</a>,
```

Scrape an image from this page

- Find the main picture from this Wikipedia page and display it in this notebook

In [24]:
```
# Your code here
get_image_url("/wiki/Churro")
```

Out[24]:
```
'https://upload.wikimedia.org/wikipedia/commons/6/6f/Churros_Madrid.jpg'
```

Start your own web crawl

- Run code to crawl through pages based on links that are found on the wiki pages
- Start with your chosen wiki page, then run a loop 20 times to open a random link from each new webpage
- Display all links and images found

  Hints:
- Only one change is needed from the code we used above to crawl from FHS's wiki!
- Change 'links' to be your list of links you created just above

In [29]:
```
# This next line "seeds" the random number generator at the current time (gives d
random.seed(datetime.datetime.now())

lonks=get_links("/wiki/Churro")                           # Extracts and creates a li
for i in range(20):                                                  # Loop to
    new_article = lonks[random.randint(0,len(lonks)-1)].attrs['href']   # Choose
    return_image(new_article,100)                                    # Call fu
    print(new_article)                                               # Prints
    lonks = get_links(new_article)                                   # Extract
    time.sleep(1)                                                    # 1-secon
```

```
/wiki/Wonut
```

```
/wiki/Gulab_jamun
```

```
/wiki/Dahi_chutney
```

Answer the Following Wrap-up Questions:

Question 6: What is web scraping? (Source: https://en.wikipedia.org/wiki/Web_scraping (https://en.wikipedia.org/wiki/Web_scraping))

Your Answer: A way to extract data from webpages on the World Wide Web

Question 7: What is web crawling? (Source: https://en.wikipedia.org/wiki/Web_crawler (https://en.wikipedia.org/wiki/Web_crawler))

Your Answer: Useing a bot, web crawling looks through websites to index them

Question 8: What is the difference between web scraping and web crawling?

Your Answer: Scraping gets data, and crawling assigns data

In [ ]: