HOMEWORK 9: APIs and WEBSCRAPING

PART 1 (from November 1 lecture): APIs

For the first part of the homework, we will be using the News API. API stands for Application Programming Interface. Many different companies package their code and functionality in APIs to sell to users or other companies. Facebook, Twitter, and many other large companies have APIs that programmers can access.

TASK 1: ACCESSING THE API

Navigate to: https://newsapi.org/

Click on the 'Get API Key' button and fill out the form to access a key. Save this key; you need it to access the API!

TASK 2: PYTHON WRAPPER

The NewsAPI allows us to make requests to a URL and get a response based on the parameters and values we request. This is somewhat messy, so fortunately someone has created a Python wrapper to more use the News API. By running the command:

```
$ pip install newsapi-python
```

We can install the newsapi-python module, which will help us access the API. For more information, you can check the github repository: https://github.com/mattlisiv/newsapi-python

TASK 3: REQUESTING ARTICLES (Terminal)

Let's use this new module to request articles from the News API. Type this code in the terminal, via the interactive shell (type python and press [enter]):

```
>>> from newsapi import NewsApiClient
>>> api = NewsApiClient(api_key='XXXXXXXXXXXXXXXXXXXXXXXX')
```

replacing the X's with the api key that you received. Don't forget the single-quote tick marks.

These commands create a NewsApiClient object. We do not have to worry about what that means now; we can use the api variable to call other functions. We can call these functions like this:

```
>>> api.get_top_headlines(sources='some source')
>>> api.get_everything(q='what you want to search')
```

(* replace 'some source' and 'what you want to search' with your own values)

These two commands use the api object to call the functions 'get_top_headlines()' and 'get_everything()'. The functions return python dictionaries containing the headlines and all information about an article respectively. You can look further at the documentation to better understand the functions.

TASK 4: FIND YOUR OWN ARTICLES

Now, use the functions and api object described above to query your own articles! In a Python script, look up **three** different keywords and print out the 'title' and 'description' for each article. Use the print function and type() function to figure out how to just get the title or description.

PART 2 (from November 6 and 8 lectures): Web Scraping

Beautiful Soup is a Python library for pulling data out of HTML and XML files. In this activity, we'll scrape the text of articles from the New York Times.

TASK 1: BEAUTIFUL SOUP

The Beautiful Soup library is already included in your Anaconda Python installation. We'll begin by trying out Beautiful Soup in the interactive Python interpreter. To access the library, open the interpreter and call:

```
>>> from bs4 import BeautifulSoup
>>> import requests
```

The requests library will allow you open web hyperlinks.

TASK 2: HTML FILES

For this activity, we'll scrape articles from the New York Times. In Google Chrome, open any NYT article. (Navigation is different for other web browsers, so if you don't have Chrome, you may want to ask the TAs for help).

Right click anywhere on the page and navigate to "View Page Source." This is the webpage's HTML format, which structures the layout of text, images, links, etc. Scroll through the HTML to get a general overview.
We want to extract the title and story of the article. They are denoted by the respective tags <title> and <p class="story-body-text story-content">. Find these tags in the HTML.

TASK 3: URL REQUEST

Let's try extracting the article's title in the Python interpreter.

Store your article's webpage as a variable called 'url'. The url must be contained within quotes. For example:

```
>>>url="https://www.nytimes.com/2017/04/16/us/politics/north-korea-missile-crisis-slow-motion.html"
```

Create a web request to access the url:

```
>>> r = requests.get(url)
```

Check that the URL request was successful by calling 'r.status_code'. The code 200 indicates that the request was successful. If you obtain another code, then there may be an error in your URL or request call.

Parse the url's content into Beautiful Soup HTML format:

```
>>> soup = BeautifulSoup(r.content,"html.parser")
```

Then print soup to see the results. (Note: if you had an XML document, you could parse it using Beautiful Soup's "xml.parser.")

Locate the title tag:

```
>>> title = soup.find('title').
```

Print title to see the results. Then, to print the title's text without its tags use:

```
>>> title.get_text()
```

Once you have this working in the Python interpreter, transfer the code over to a new Python program, adding it to a function named get_article_content() that takes in url as a parameter.

TASK 4: EXTRACTING <p> TAGS

Now we'll extract the article's story content. You can do this just in your Python program (not the interpreter). Call:

```
>>> paragraphs = soup.find_all('p')
```

This locates all text encapsulated within the tag <p class="story-body-text story-content">. When you print out paragraphs, you'll see that both the text and tags are returned. **paragraphs** is also a list, for which each index represents a paragraph.

To extract just the text (no tags), iterate through each paragraph in paragraphs and call **paragraph.get_text()**. Join together all the paragraphs into a single string, separated by the newline character '\n'.

TASK 5: STORING TEXT

Now that you've extracted the title and story content, write them to a text file. We want the filename to match the webpage name found in the URL. For this, we can use a regex to get every character of the URL after the last backslash. This part of the URL may contain the extension ".html", which we'll want to replace with the extension ".txt".

```
>>> match = re.search("[^\s\/]+$", url).group(0)
>>> match = match.replace(".html", "")
>>> filename = match + ".txt"
```

TASK 6: FIND OTHER ARTICLES!

Because the HTML format of most NYT articles are consistent, you can use this code to webscrape other NYT articles. Try running your program on a couple more articles. All you need to do is change the value of 'url'!

TASK 7: WEB CRAWLING

In the next phase of the activity, you will create a web crawler that will find the contents of all of the articles on the front page of the New York Times.

To do this, you will need to create a function called get_links() that will return a list of URL links to the articles that are on the front page.

You will first need to create a new requests() object:

```
>>> r = requests.get("https://www.nytimes.com/").
```

Then a new BeautifulSoup() object:

```
>>> soup = BeautifulSoup(r.content,"html.parser")
```

Looking at "View Page Source," you'll see that the titles and links for the articles are in the h2 tag with a class "story-heading."

Use BeautifulSoup to find all of these instances:

```
>>> titles = soup.find_all('h2).
```

This will return a list of all of the titles.

TASK 8: ITERATION

Iterate through the list titles and print out each element. One element should look something like this:

```
<h2 class="story-heading">
<a
href="https://www.nytimes.com/2017/04/20/automobiles/autoreviews/jaguar-xe-review.html">
        Driven: Video Review: Jaguar Figures Out the Compact Sport Sedan        </a>
</h2>
```

We need to extract the link from each element. To do this, call **.find('a')['href']** on each element. Do this and try running the program. You will get an error: TypeError: 'NoneType' object is not subscriptable. This is because not every element in titles contains a link.

TASK 9: ERROR HANDLING

To handle this, as you iterate each title in titles, first check if the element includes a link, denoted by the tag <a>. If there is a link, then extract the url, denoted by the <href> tag. You can use the code below:

```
    >>> link = title.find('a')
    >>> if link != None:
    ...       link_url = link['href']
```

Add all these urls to a list and return that list at the end of the function.

TASK 10: WRITE ALL THE FILES

In main(), call the function get_links() to return the list of links on the front page of the NYT. Iterate through this list and call get_article_content() to write the title and content of each article to its own text file. Because of the large number of files, you may want to make a new directory within your cs0030_workspace and write your files to that directory.

EXTRA CREDIT

- Try scraping information from some of your favorite websites
- Check out this tutorial on parsing XML data into a pandas dataframe. You could try this on the Congressional data that we analyzed at the beginning of the semester.

HOURS WORKED

When you're done, add the number of hours worked, whether you came to TA hours, and the names of any students you worked with at the top of the file in a comment.

HANDIN

Name your submission file 'hw09.py' submit it through Canvas before Nov 15 (Thurs) at 11:59PM.