Homework 9: APIs and Webscraping

November 2018

1 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.

1.1 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!

1.2 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:

```bash
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

1.3 TASK 3: REQUESTING ARTICLES (Terminal)

Lets 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]):

```python
from newsapi import NewsApiClient
api = NewsApiClient(api_key='XXXXXXXXXXXXXXXXXXXXXXXXX')
```

replacing the Xs 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:

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

1.4 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 returned from the API. Use the print function and `type()` function to help figure out how to just get the title or description.

2 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, well scrape the text of articles from the New York Times.

2.1 TASK 1: BEAUTIFUL SOUP

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

```python
from bs4 import BeautifulSoup
import requests
```

The requests library will allow you open web hyperlinks.

2.2 TASK 2: HTML FILES

For this activity, well scrape articles from the New York Times. In Google Chrome, open any NYT article. (Navigation is different for other web browsers, so if you dont 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 webpages 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.
2.3 TASK 3: URL REQUEST

Let’s try extracting the articles title in the Python interpreter. Store your articles webpage as a variable called url. The url must be contained within quotes. For example:

```python
```

Create a web request to access the url:

```python
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 urls content into Beautiful Soup HTML format:

```python
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 Soups xml.parser.)

Locate the title tag:

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

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

```python
title.get_text()
```

TODO: 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.

2.4 TASK 4: EXTRACTING <p> TAGS

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

```python
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`. 
2.5 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 want to replace with the extension .txt.

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

Add to your `get_article_content()` function so that it correctly opens and writes to a file with the webpage name in the inputted URL. Your `get_article_content()` function should contain Tasks 3-5.

2.6 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`!

2.7 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:

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

Then a new `BeautifulSoup()` object:

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

If you click on "View Page Source," you’ll see that the titles and links for the articles are all contained within `<a>` `(</a>)` tags.

Use `BeautifulSoup` to find all of these instances:

```python
links = soup.find_all('a')
```

This will return a list of all of the `<a>` tags, which contain the links and the corresponding titles for those links.
2.8 TASK 8: ITERATION

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

```html
<a href="/2018/11/08/us/recount-runoff-florida-georgia.html">
<div class="css-1j836f9 esl82me3">
<h2 class="css-78b01r esl82me2">
<span>Voters in Florida and Georgia Still Wonder Who Won</span></h2>
</div>
<ul class="css-1rrs2s3 e1n8kpyg1">
<li>The two states races for governor and Florida s Senate race hung in the balance, and the stress was palpable for politicians and voters alike.</li>
<li>Officials were still counting absentee, provisional and overseas ballots, and the partisan camps were disputing how to go about it.</li>
</ul>
</a>
```

If the tag is stored as the variable `a_tag`, write the following command to get the link of the tag:

```python
>>> a_tag['href']
```

However, many links on the NYTimes homepage might be ads or other links, so to make sure that the links of all the `<a>` tags are to NYTimes articles, we must check if there is a title for each tag. These titles are contained within `<h2>` tags, so we must check to see if one exists.

To get the `<h2>` tags (which contain the title), call:

```python
>>> a_tag.find("h2")
```

Try running these commands and print out the result of finding the `<h2>` tags. Some of the results will be `None`, because some links are not to articles.

To handle this, as you iterate through each `<a>` tag, first check if the element includes a header, denoted by the tag `<h2>`. If there is a header, then you want to add that link to a list to return. You should write something similar to the code below:

```python
header = a_tag.find('h2')
if header != None:
    # add the link in a_tag to some list to return
```

**TODO:** Create a function called `get_links()` that completes Tasks 7 and 8. The function should add all of these urls to a list and return that list at the end of the function.
2.9 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.

(***HINT: When you are passing these links as parameters into your `get_links()` function, make sure to add ”http://www.nytimes.com/” to the beginning of the link ***)

Because of the large number of files, you may want to make a new directory/folder within your cs0030_workspace and write your files to that directory.

3 EXTRA CREDIT

- Try scraping information from some of your favorite websites

4 HANDIN

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