SI 206 – Data-Oriented Programming Project 2

OBJECTIVES:

This assignment will utilize the information we covered in Regex (Chapter 11) and Networked Programs (Chapter 12). However, you will go much deeper into the topic by researching Beautiful Soup and the source code for different web sites.

WHAT YOU SHOULD DOWNLOAD TO WORK ON:

- The `SI206project2.py` file, which contains some provided code
- opinion.html – a content grab for Part Two of the homework you can use for testing purposes.

WHAT YOU SHOULD SUBMIT TO CANVAS:

- Your edited `SI206W17_project2.py` file
- A link to your github account to this project
- A text file with anything you need to tell us about the project / your submission, if applicable

Your .py file must run successfully using Python3. We do not grade projects that do not run. Make sure you check that your program runs before submitting it!

TOTAL POSSIBLE POINTS: 200

The project is in multiple parts but each part builds on both the homework you have done previously.

We have included starter import statements and comments indicating where you may want to place your code in the file.

The file also includes tests, which are the primary determining factor as to how you get points (but we do look at your code, etc). Testing will be done using subsets of other code, so hardcoding your answers won't work – but we will not change the structure of the HTML files (class names, etc) just some of the content.

Make sure your file runs without syntax errors!

You should start this early! Break it up into pieces, write out a plan, do a little bit at a time, and keep running and testing it, adding and committing and pushing to GitHub.

Part 1 – Regular expressions + function definition

- Define an additional function using Python regular expressions called **find_urls**.
- **INPUT:** any string
- **RETURN VALUE:** a list of strings that represents all of the **URLs** in the input string
- See code file for example.
- There are tests for this. **50 points.**

Useful notes for Part 1:

- You can define what a URL is in a simple way in this assignment, though sometimes this can be quite complicated. For us, a URL is:
    - Anything that begins with **http://** or **https://** AND
    - Includes at least one "**.**"within it, and each "**.**" character must be followed by at least 2 characters
        - E.g. **http://bbc.co.uk** is a valid url
        - And **https://www.gmail.com** and **https://gmail.com** are also both valid urls
        - As is **http://nationalparkservice.gov/pictures/badlands**
        - But **gmail.gov** is not a valid url for our purposes
        - And **http://bbc.c** is not a valid url for our purposes either
    - Includes no spaces or any other whitespace characters
    - (OK to be more specific than our requirements above if you want to challenge yourself to get absolutely all URLs even if they're quite unusual. You must at least handle all of all of those listed requirements.)

Part 2 – BeautifulSoup

- Define an additional function using called **grab_headlines**.
- **INPUT:** NONE
- **RETURN VALUE:** a list of strings that represents of titles of all of the Most Read articles in the Michigan Daily Opinion Page
- There are tests for this. **50 points.**
- Since the data returned by this text may change daily, you may use the data file opinon.html to test your data. (You can also change the values of the unit tests, but we will change it back later.)

Part 3 a – BeautifulSoup and complex Python mechanics

- Define an additional function called **get_umsi_data**.
- **INPUT:** No input.
- **BEHAVIOR:** The function should access each page of the directory, get the HTML associated with it, and create a dictionary named **`umsi_titles`** whose keys are the names of each person in the UMSI directory, each of which's associated value is that person's title, e.g. "PhD Student" or "Associate Dean of Research and Arthur F. Thurnau Professor of Information, School of Information" … . e.g. **"Lindsay Blackwell":"PhD Student"** should be one of many key-value pairs inside this dictionary. You should NOT worry about cases where there is no title / the title is an empty string. Include the empty strings in your keys or values of the dictionary.

- **RETURN VALUE:** umsi_titles

There are tests for this whole function. **90pts**

Useful notes for Part 3(a):
You should get ALL the HTML data in the directory, *starting* with the page at this URL: https://www.si.umich.edu/directory?field_person_firstname_value=&field_person_lastname_value=&rid=All
And ending at the page at this URL:
https://www.si.umich.edu/directory?field_person_firstname_value=&field_person_lastname_value=&rid=All&page=11
and including each of the pages in between in between, each of which can be represented by a big HTML string

HINT: What do those two page links have in common? What's different? That's the first and the 12th page, respectively. What do you think the URL of the 7th page of the directory looks like?

Remember that to get data from the UMSI site, you must have the headers parameter like so in your request, so that you will not get blocked by the site: `requests.get(base_url, headers={'User-Agent': 'SI_CLASS'})`

Part 3 b - Python mechanics

- Define an additional function called **get_umsi_data**.
- **INPUT:** value returned from get_umsi_data.
- **BEHAVIOR:** The function should search the dictionary to find the number of PhD students in the directory.
- **RETURN VALUE:** integer

Notes – this function will be graded on efficiency
There are tests for this whole function. **10pts**