Homework 7

Files and Web Scraping

Out 11/23 – Due 12/8

Exercise 1. Renaming Files with American-Style Dates to Asian-Style Dates (35 pts)
Your boss emailed you thousands of files with American-style dates (MM-DD-YYYY) in their names and need them renamed to Asian-style dates (YYYY-MM-DD). Note that filenames cannot contain slashes (/), otherwise your file system would confuse it with the path separation.[1] This boring task could take all day to do by hand. Let's write a program to do it instead.

Here's what the program should do:

- Search all filenames in the current working directory for American-style dates in the names.
- When one is found, rename the file to make the date portion Asian style (the rest of the filename doesn't change).

This means the code will need to do the following:

1. Create a regex that can identify the text pattern of American-style dates in a filename. Please note that the MM and DD parts of a date have different ranges of numbers. You can manually create a few files for testing.
2. Call os.listdir() or os.walk(path) to find all the files in the working directory.
3. Loop over each filename and use the regex to check whether it contains an American-style date.
4. If it has a date, rename the file with shutil.move().
5. Save your code as hw7_firstname_lastname_ex_1.py.

Exercise 2. Selective Copy (25 pts)
Write a program that

1. Walks through a directory tree (os.walk()) and searches for files with an extension of .pdf.
2. Print these files with their absolute path and file size to the screen.
3. Copy these files from whatever location they are into a new folder.
4. Save your code as hw7_firstname_lastname_ex_2.py.

Exercise 3. I'm feeling lucky Google (40 pts + 15 bonus pts)
Whenever I search a topic on Google, I don't look at just one search result at a time – I open the first several links in a bunch of new tabs to read later. This workflow— opening my browser, searching for a topic, and opening several links one by one in new tabs — is repetitive. It would be nice if I could simply type a search term on the command line and have my code automatically open a browser with all the top search results in new tabs. Let's write a script to do this.

Your code will need to do the following:

[1] https://en.wikipedia.org/wiki/Filename#Reserved_characters_and_words "Many file system utilities prohibit control characters from appearing in filenames. In Unix-like file systems, the null character and the path separator / are prohibited."

1. Read the command line arguments as the search term that you'll use in Google. Be creative and use the terms you'll actually search, like 'tech jobs', 'Beautiful Soup tutorial', 'how to make tortilla soup'.
2. Fetch the search results page with the requests module (tips below), then open the results to inspect. You may either open the search results directly in the browser, or save the results to an html file locally then open it.
3. (Bonus step) Find the links to each search result. Then call webbrowser.open() to open the top three hits in new browser tabs. You will receive 15 bonus points if you do this correctly. Since this is a bonus, I will not provide hints beyond what's already here in this document.
4. To make running code on the command line easier, save your program as lucky.py.

Tips:

- By looking at the browser's address bar after doing a Google search, you can see that the result page has a URL like https://www.google.com/search?q=SEARCH+TERM Anything after the search term is optional.
- Sometimes the html response we get is different from the browser (well, because our code is not a browser). One thing you could do is provide a User-Agent string in your request as follows (to pretend we are searching from a browser).[2] However, you could still get slightly different html pages compared to a real browser visit, but the slight difference doesn't matter for the purpose of this exercise.

```
userAgent = "Mozilla/5.0 (X11; CrOS i686 2268.111.0) AppleWebKit/536.11 (KHTML, like Gecko) Chrome/20.0.1132.57 Safari/536.11"
headers = {'User-Agent': userAgent}
res = requests.get(url, headers=headers)
```

- If you are still getting different html pages, you could save your html response to a file. Then, open that file with your browser and inspect that.

- You can open the browser's developer tools and inspect some of the link elements on the page. They look incredibly complicated. It doesn't matter. You just need to find the pattern that all the search result links have.

Submit your code files in **a zipped archive named hw7_firstname_lastname.zip**. Comment everything so we know you wrote the code! On top of your files write this multiline comment with your information:

"""
Homework 7, Exercise 1 (or 2…)
Name
Date
Description of your program.
"""

[2] This page also provides the User-Agent strings for various browsers: https://stackoverflow.com/questions/27652543/how-to-use-python-requests-to-fake-a-browser-visit