Regular Expressions

SI 206 Homework #5 (Fall 2021)

In this homework, you have been given some chapters from **Harry Potter and the Prisoner of Azkaban (my favo!)** in a text file called "Harry-potter-txt.txt". Your work here is to use regular expressions to find important information such as chapter names, capitalized word patterns, dates, and URLs. Take the adventure with Harry now!

To do so, you will complete the following functions in HW5.py:

1. find_chapter_info (string_list)

This function finds and returns a dictionary with the chapter number as the keys and the title of the chapter as the value:

The expected output should be in the format:

{1: "Owl Post", 2: "Aunt Marge's Big Mistak", … }

2. find_capitalized_words (string_list)

This function finds consecutive words (at least two words in succession) that are both/all capitalized. For example, in the text we can find "Diagon Alley" and "Professor Remus Lupin" have matched patterns. The capitalized words should be consecutive, as a result, Mr *and* Mrs Weasley does not qualify. Also, punctuation should not be included, for example, "Harry. When" does not qualify. You need to return a list with all matched consecutive capitalized words.

For example, in the following text:

Harry is also confused and asks why he would go looking for somebody who wanted to kill him. On the train, Harry, Ron, and Hermione share a compartment with Professor Remus Lupin, the new Defence Against the Dark Arts teacher.

The expected output should be in the format:

["Professor Remus Lupin", "Defence Against", "Dark Arts"]

3. find_urls (string_list)

This function finds and returns all the hidden urls which match the following

conditions:

a. The URL should start with http:// or https://.
b. Followed by a www.
c. Followed by any characters except for whitespace.
d. It should contain a .com or .org.

These are examples of valid URLs:

http://www.gutenberg.org/1/11/org
https://www.pythex.com
https://www.youtube.com/watch?v=msvOUUgv6m8

4. find_dates (string_list)
This function finds and returns dates from a text file that match a regular expression. You will write the regular expression. A valid date is any date that follows any of the following formats:

mm/dd/yyyy
mm/dd/yy
mm-dd-yyyy
mm-dd-yy

Any date that does not follow any of the above formats should not be returned from this function. For example, 12162019 is not a valid date and should not be returned. Also, be careful that 08/32/2021 is not a valid date! Day should range from 01-30, and month should range from 01-12. You do not need to worry about whether a month has 31 days or not in this assignment :). Years should range from 1900 - 2021.

5. Make at least 3 test cases for **find_chapter_info**, **find_capitalized_words, find_urls**, and **find_dates**.

Count table

To help you know whether your functions are correctly implemented or not, here are the number of items that SHOULD be returned when you run your functions on the file *'harry-potter-txt.txt'*. To run your functions, you will first have to use the provided function ***read_file*** to convert the contents of *'harry-potter-txt.txt'* into a list of strings.

Note: your functions should return a list/dictionary, not the number of items (except for the extra credit function). This table is for you to verify that you are returning the right list (e.g., maybe you can check the length of the list you return):

Data type	**Number of appearances in the text**
Chapter info	6
Consecutive capitalized words	68
URLs	5
Dates	5
Extra credit: Count "arr" Count "th"	47 18

Grading Rubric (60 points)

This rubric does not show all the ways you can lose points.

- 6 points for creating tests for **find_chapter_info** (2 points per test case, up to a max of 6 points)
- 9 points for correctly implementing **find_chapter_info**
- 6 points for creating tests for **find_capitalized_words** (2 points per test case, up to a max of 6 points)

- 9 points for correctly implementing **find_capitalized_words**
- 6 points for creating tests for **find_urls** (2 points per test case, up to a max of 6 points)
- 9 points for correctly implementing **find_urls**
- 6 points for creating tests for **find_dates** (2 points per test case, up to a max of 6 points)
- 9 points for correctly implementing **find_dates**

Extra Credit (3 points):

Write a function *count_mid_str (string_list, string)* to return a count of the number of times a specified string appears in a file. It should match the string that is in the middle of a word (not the beginning nor the end). For example, if called with "be" it should match "number" but not "vibe". Make sure to account for punctuation (e.g ',' '?') in your regular expression. You MUST use a regular expression to earn credit for this part. (We will not be checking if you make tests for the extra credit, but feel free to write your own tests if it will help you complete this problem!)

Submission:
Make at least 3 git commits and turn in your GitHub repo URL on Canvas by the due date to receive credit.