Regular Expressions
SI 206 Homework #6

In this homework, you have been given the book Alice in Wonderland by Lewis Carroll in a text file called "alice_in_wonderland.txt". However, the Mad Hatter, a fictional character in the book, has corrupted the file and added some clues to the text file. Your job is to use regular expressions to find this information as times, URLs, dates, and words contained in underscores.

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

1. find_times (string_list)
This function finds and returns all the hidden times which match the following conditions:

- **a.** It should start with 1 – 12.
- **b.** It should be followed by a colon (:)
- **c.** It should be followed by two digits between 00 – 59.
- **d.** It should end with am or pm (with one space between the numbers and the letters)

These are examples of valid times:

11:00 pm
12:25 am
5:00 pm

2. 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.
- **e.** Followed by any characters except for whitespace.

These are examples of valid URLs:

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

3. 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
mm.dd.yyyy
mm.dd.yy
dd.mm.yyyy (day and month can be switched like this too)

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.

4. find_underscore(string_list)
The function should find and return a list of all the words contained by an underscore. Return only words that do not have punctuation.

very
this

Example of what not to return; _very!_

5. Make at least 3 test cases for **find_time**, **find_urls**, **find_dates**, and **find_underscore**.

Count table

To aid you in knowing when 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 *'alice_in_wonderland.txt'*. To run your functions, you will first have to use the provided function ***read_file*** to convert the contents of *'alice_in_wonderland.txt'* into a list of strings. **Note: your functions should return a list of items, 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
Times	5
URLs	5
Dates	7
Underscores	177
Extra Credit: Count A Count B	3850 1099

Grading Rubric (60 points)

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

6 points for creating tests for find_time (2 points per test case, up to a max of 6 points)

9 points for correctly implementing find_time

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

6 points for creating tests for find_underscore (2 points per test case, up to a max of 6 points)

9 points for correctly implementing find_underscore

Extra Credit (3 points):

Write a function *count_char(string_list, char)* to return a count of the number of times a specified character appears in a file. It should match the character when it starts or ends a word (It should not match any characters in the middle of a word). For example, if called with “a” it should match “Apple”, “pasta”, "away", "Along", but not “farewell”. 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.