Regular Expressions

SI 206 Homework #6

Due: February 27th

In this homework, you will help Sherlock Holmes find clues to solve a mystery. You have been given a text file 'The_Adventure_of_Sherlock_Holmes.txt' that contains a lot of text with hidden clues in the form of *coordinates, email addresses,* and *dates* in the text. Your task is to use *regular expressions* to find the clues to help Sherlock in his investigation.

To do so, you will complete the following functions in mystery.py: *find_coordinates*, *find_emails*, and *find_dates*. **For each function, you will do the following:**

1. Write unit tests for these functions in the *TestAllMethods* class we have provided. To get credit, you must have at least 3 tests for each function, and have them be non-trivial (e.g., the tests shouldn't be dummy tests such as **assertEqual(True, True)**). Even though you are only graded on 3 tests, we highly recommend writing more than 3 tests to fully test that your function is working.
2. Implement the function.

1. *find_coordinates (string_list)*

This function finds and returns all coordinates from a list of strings that match a regular expression. What is returned will be a list of strings, each element of the list being a coordinate. Coordinates are written in a very specific format; two numbers with 5 decimal places separated by a space, the first number being latitude and the second being longitude. For the purposes of this assignment, latitudes are numbers that range from -89.99999 to 89.99999, and have 5 decimal places; longitudes range from -179.99999 to 179.99999 and they too have 5 decimal places. Each latitude and longitude is followed by a capital letter; the first capital letter is either N (North) or S (South), the second letter is E (East) or W (West). Some example coordinates are:

- 42.28008N 83.74301W
- -30.55951S 22.00000E
- 46.86245N -103.84677E
- 1.48354S 130.33345W

2. *find_emails (string_list)*

This function finds and returns all email addresses from a list of strings that match a regular expression. Remember, emails have many different kinds of domains (gmail, yahoo) and domain extensions (.com, .uk). They are anything in the form xxxxx@xxxx.xxx, where each x is an alphanumeric character (numbers, letters, underscores). To make matters more difficult, some mischievous character is trying to mislead Sherlock Holmes with fake emails! Oh no! But Sherlock has managed to identify the fake emails. The correct emails are those that have an **odd number of characters before the @ sign.** So a valid email address would be any of the following:

- odd@gmail.com
- five1@yahoo.com
- this_isn_t_even@bb.edu

But the following would not be valid:

- even@weird.com
- n0tt_0dd@gmail.com
- serious1@hotmail.com

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. The days, months, and years can be separated by a forward slash or a dash. A valid date is any date that follows any of the following formats:

dd/mm/yyyy	m/yy	dd-mm-yy
d/mm/yyyy	mm/yy	d-mm-yy
dd/m/yyyy	m/yyyy	dd-m-yy
d/m/yyyy	mm/yyyy	d-m-yy
dd/mm/yy	dd-mm-yyyy	m-yy
d/mm/yy	d-mm-yyyy	mm-yy
dd/m/yy	dd-m-yyyy	m-yyyy
d/m/yy	d-m-yyyy	mm-yyyy

For the purposes of this assignment, assume that days (dd) are numbers that range from 1 to 31, and are optional; months (mm) are numbers that range from 1--12; and year (yy and yyyy)

can be any number. Don't worry about having to filter out 'incorrect' dates (e.g., the 31st of February). Any date that does not follow any of the above formats should not be returned from this function. For example, 02.21.2020 is not a valid date and should not be returned. However, some valid dates are 03-2018, 12/12/7012, 5/9/19, and 06-06.

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 *'The_Adventure_of_Sherlock_Holmes.txt'*. To run your functions, you will first have to use the provided function ***read_file*** to convert the contents of *'The_Adventure_of_Sherlock_Holmes.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
Coordinates	4
Emails	5
Dates	23
EC:counting 'a' in *'The_Adventure_of_Sherlock_Holmes.txt'* Counting 'B' in *'The_Adventure_of_Sherlock_Holmes.txt'*	11988 4712

Grading Rubric (60 points)

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

9 points for creating tests for find_coordinates (3 per test, max of 9 points)

11 points for correctly implementing find_coordinates

9 points for creating tests for find_emails (3 per test, max of 9 points)

11 points for correctly implementing find_email

9 points for creating tests for find_dates (3 per test, max of 9 points)

11 points for correctly implementing find_dates

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.