Programming Project 9

This assignment is worth 55 points (5.5% of the course grade) and must be **completed and turned in before 11:59 on Monday, November 20, 2017.**

Assignment Overview
(learning objectives)
This assignment will give you more experience solving complete problems with:

1. dictionaries
2. sets

The goal of this project is to investigate the most common Twitter hashtags used by a few MSU-related Twitter accounts, and the similarity of hashtag-use between accounts.

Given Twitter data in a comma separated values (CSV) file, extract and store the necessary data you need to answer the questions about hashtags. Use dictionaries to help count frequency of hashtags, and sets to easily answer questions about shared hashtags between users.

Before you begin, you should play with the following concepts in the Python shell or programs to better understand how they work so when you attempt the project you will have an idea of the kinds of problems each tool can solve for you:

1. dictionaries: creating, modifying, print all keys, values, printing all keys and their values.
2. sets: creating, modifying, how is it different from a list?, intersections, unions, length.
3. String `find()` method: what does it do? If you had to, how would you use it to find all positions of `'#'` in a string that had 0 or more `'#'`s in it? (try it out – make up a string)

Assignment Background

The psychometrics company Cantabrigia Analytica (C.A.) uses data from many diverse datasets to understand and predict how people would respond to various marketing strategies. C.A. is very interested in how MSU and Michigan Twitter accounts use hashtags, which hashtags are the most common (and thus possibly the most important for Michigan), and which hashtags are all used by groups of important accounts. C.A. has hired you to develop a method to answer these questions and provide them with the resulting detailed datasets as preliminary answers which they will integrate into their prediction software. You have decided to begin with only a small subset of Twitter data for the Michigan accounts, and have downloaded only up to 500 tweets from each account (the data provided to you) just to see if your algorithms will work before progressing to the entire Twitter history for each account (you won't need to do this).

Project Description

Read through the project .py file provided to you. That file defines stub functions for the functions you are required to have, though you may have more. These stub functions will be individually tested in grading to ensure their proper function. Remember, function parameters can be named anything you want – so feel free to change their names so they make sense to you, but do not change the name of the function itself. Also, the order of the parameters is important and should not change.

Function skeletons are provided for you, but you must fill them in to make them work correctly (only the plotting function is provided in its entirety). They are listed in the following format `function_name() -> returned_types`

We begin by building some functions that we can then use to answer questions:

`open_file() -> fp`
This function returns a file pointer. You likely have a copy from a previous project. It repeatedly prompts for a file until one is successfully opened. It should have a try-except statement.

`validate_hashtag(str) -> Boolean`
This function has one parameter, a string which is a hashtag. Assume that the string starts with # because you shouldn't call this function with a string that doesn't start with #. According to Twitter a valid hashtag has no punctuation (other than starting with #) so if a hashtag contains punctuation, return `False`. For the purposes of this project we do not want to consider #1 such as you might find in a tweet string "We are #1!" as a valid hashtag so any hashtag that is a single digit (after the #) will return `False.` All other hashtags return `True` . Hint: import `string` and use `string.punctuation` to check for punctuation.

`get_hashtag(str) -> list[str,…]`
This function has one parameter, a string which is a tweet, and returns a list of valid hashtags in the tweet string. Hashtags start with #.

`read_data(fp) -> list[[str,int,list],…]`
This function has one parameter, the file pointer for a csv file, and reads in the data collecting 3 things from each row: username as string, month as integer, and a list of all hashtags found in the tweet message. See the Notes section for hints on how to find all hashtags in the tweet. Hashtags are words which begin with the character `'#'`, so `'#pythonforever'` is a hashtag in the string `'Having a great time learning python #pythonforever . '` According to the way Twitter defines a hashtag a hashtag will always have a space after it, so you know hashtags always begin with a `'#'` and end with a `' '`. This function should return a list of 3-entry lists, where there is a 3-entry list for each line of the csv file: `[username, month, list_of_hashtags].`

`get_histogram_tag_count_for_users(data,usernames) -> dict`
Once you've stored all relevant information into one big list, you can pass that list to functions like this one to extract meaningful information. This function creates a histogram (See Notes for 'What is a Histogram') of hashtags for how often they occur. The key is the hashtag; the value is the count of occurrences of the hashtag. This function has two parameters. The first, `data`, is the list of lists returned by the `get_data_from_file` function. The second parameter is a list of one or more `usernames` that allows the building of the histogram for one or multiple users. That is, this function can build a histogram of all hashtags used by a single user `'Charles'` or it could build a histogram of all hashtags used by either `'Charles'` or `'Sonja'` or `'Jehar'`. This function should return the histogram as a dictionary. Hint: CodeListing 9.2 in the text illustrates how to build this kind of dictionary.

`get_tags_by_month_for_users(data,usernames) -> list`
Instead of a histogram, this function builds a *set* of unique hashtags grouped by the month in which they are used. That is, this function returns a lists of tuples: specifically a sorted list of `(key,value)` tuples where each key is a month (int), and each associated value is a *set* of hashtags used by `usernames` in that month. The list is sorted by month, i.e. January first. Instead of a count of hashtags, start with an empty set and keep adding to it. Similar to `get_histogram_tag_count_for_users`, this should be able to be done for a single user or across multiple users. The first, `data`, is the list of lists returned by the `get_data_from_file` function. The second parameter is a list of one or more `usernames` that allows the building of the histogram for one or multiple users. Hint: start with a list of 12 (key,value) tuples where each value is an empty set.

`get_user_names (data) -> list`
Return a sorted list of user names that are in the Twitter data (sort alphabetically—the default sorting order. Sorting allows consistent testing in Mimir).

With these functions in hand, let's answer some questions:

- "What are the most common hashtags used by users *collectively*?"
- "What are the most common hashtags used by users as *individuals*?"
- "How does hashtag *similarity* between two users vary over time?"

`three_most_common_hashtags_combined(data,usernames) -> list`
Answers the question "What are the most common hashtags used by users *collectively*?" Return an ordered list of three `(count, hashtag)` tuples where count is the count of all occurrences of hashtag across all users in `usernames`. The three tuples in the list represent the three highest counts (and the hashtag) with highest first. Hint: call `get_histogram_tag_count_for_users` and convert that dictionary to a sorted list of tuples, largest first, and return the first three (slicing!).

`three_most_common_hashtags_individuals(data,usernames) -> list`
Answers the question "What are the most common hashtags used by users as *individuals*?" Return an ordered list of three `(count, hashtag, username)` tuples where count is the count of occurrences of hashtag only for a user in `usernames`. The three tuples in the list represent the three highest counts (and the hashtag and user) with highest first. Hint: call `get_histogram_tag_count_for_users` for each user in `usernames`, create a master list of tuples, then sort the tuples largest first, and return the first three (slicing!).

`similarity(data,user1,user2) -> list`
Answers the question "How does hashtag similarity between the users `user1` and `user2` vary over time?" Compares the hashtags used by each user for each month, and returns the numbers of hashtags which were used by both accounts in that month. To do this you will need to organize hashtags by month for each of the 2 users. We are going to plot this data so it must be ordered by month: January through December, but since our months are numbers, the data is ordered from month 1 to month 12. That is, you return an ordered list of `(month, tag_set)` tuples. (Hint: use `get_tags_by_month_for_users`), then compare those sets of hashtags (Hint: set intersection).

`plot_similarity(x_list,y_list,user1,user2)`
This function is provided for you. Its purpose is to plot the data returned by the `similarity` function. However, you need to set up the data to pass to the function. Most important is that `x_list` and `y_list` are *both lists of numbers of the same length*. In this case, the two lists are lists of ints of length 12. The `x_list` is simply the number of the months [1,2,…,12]. The `y_list` is the count of common hashtags by month that was returned by the `similarity` function (DO NOT call the similarity function within this plotting function.).

Assignment Deliverable

The deliverable for this assignment is the following file:

`proj09.py` – the source code for your Python program

Be sure to use the specified file name and to submit it for grading via the **Mimir system** before the project deadline.

Assignment Notes

What is a histogram?

While technically a graphical representation, a histogram may be thought of as a collection of things, and a record of their frequency (See https://www.wikiwand.com/en/Histogram). As an example, the following is a dictionary in Python that represents a histogram of programming languages and how many people in a class know them:

```
languages_hist = {'c++':4, 'java':7, 'python':6, 'nim':1, 'matlab':3}
```

How to find all occurrences of a substring

Let's say you want to find all words in a sentence which begin with the letter `'b'` and let's assume any such word will always end with a `' '` (space). You could try using the `split()` function but that will quickly become cumbersome. Instead, use the `find()` function. Because the `find()` function only ever returns the location of the first occurrence it finds (or -1 if none) then you need to keep calling it until it returns -1 because it can't find anymore. Each new time `find()` is called, it must be told to begin looking at the location immediately after the last location it found something, otherwise it'll keep finding and reporting the same location. An example of how this might work looks like this. Study the example and adapt it to your own needs.

```
string = "finding bugs will become easier when viewing boring code ."
beginning = string.find('b')
while(beginning != -1):
    end = string.find(' ',beginning) #look for end of word since start
    print( string[beginning:end] )
    beginning = string.find('b',end) #look for next word since last end
```

Output

```
bugs
become
boring
```

How to sort a dictionary

Use the `itemgetter()` function from the `operator` module. This function allows you to tell a sorting mechanism like the `sorted()` function which field to use for sorting, as a field number starting with 0. Consider the following example. Because the `items()` method returns a list of tuples, then each item in the tuple has the values `key` at position 0 and `value` at position 1. Try these out:

```
counting = {'jacks':2, 'aces':3, 'fives':1}
sortedByName = sorted(counting.items(), key=itemgetter(0))
sortedByCount = sorted(counting.items(), key=itemgetter(1))
sortedByCountR = sorted(counting.items(), key=itemgetter(1), reverse=True)
```

And this can be directly used in a for loop like this:

```
for key,value in sorted(counting.items(), key=itemgetter(1)):
    print(key,value)
```

(List of useful instructions to look up or examples of instruction use)
dictionaries – book page 425
sets – book page 445
string.find() – book page 205
sorted() – book page 327

Mimir Tests
Test 1: no error checking

```
Input a filename: twitterdata.csv

Top Three Hashtags Combined
 Count Hashtag
   370 #MSU
   212 #MSUPride
   180 #SpartansWill

Top Three Hashtags by Individual
 Count Hashtag              User
   193 #MSUPride            MSUnews
   190 #MSU                 michiganstateu
   149 #MSU                 MSUnews

Usernames:  MSUnews, WKAR, WKARnewsroom, michiganstateu

Input two user names from the list, comma separated: MSUnews,michiganstateu

Similarities for MSUnews and michiganstateu
Month       Count
January     4
February    3
March       5
April       4
May         5
June        0
July        6
August      6
September   6
October     8
November    3
December    4


Do you want to plot (yes/no)?: no
```

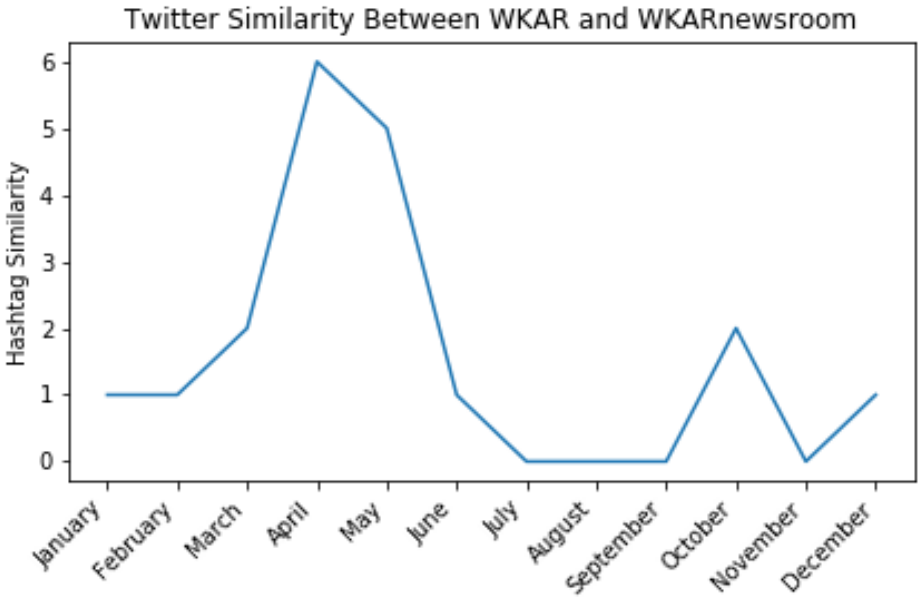
Test 2: error checking (note the spaces after the comma when user names were input)

```
Input a filename: xxx
Error in input filename. Please try again.

Input a filename: twitterdata.csv

Top Three Hashtags Combined
 Count Hashtag
   370 #MSU
   212 #MSUPride
   180 #SpartansWill

Top Three Hashtags by Individual
 Count Hashtag              User
   193 #MSUPride            MSUnews
   190 #MSU                 michiganstateu
```

```
   149 #MSU                 MSUnews

Usernames:  MSUnews, WKAR, WKARnewsroom, michiganstateu

Input two user names from the list, comma separated: xxx
Error in user names.  Please try again

Input two user names from the list, comma separated: xxx, yyy
Error in user names.  Please try again

Input two user names from the list, comma separated: WKARnewsroom,             michiganstateu

Similarities for WKARnewsroom and michiganstateu
Month       Count
January     0
February    0
March       0
April       1
May         1
June        0
July        0
August      0
September   0
October     1
November    0
December    0


Do you want to plot (yes/no)?: no
```

Test 2 : plotting (not tested on Mimir)

```
Input a filename: twitterdata.csv

Top Three Hashtags Combined
 Count Hashtag
   370 #MSU
   212 #MSUPride
   180 #SpartansWill

Top Three Hashtags by Individual
 Count Hashtag              User
   193 #MSUPride            MSUnews
   190 #MSU                 michiganstateu
   149 #MSU                 MSUnews

Usernames:  MSUnews, WKAR, WKARnewsroom, michiganstateu

Input two user names from the list, comma separated: WKAR, WKARnewsroom

Similarities for WKAR and WKARnewsroom
Month       Count
January     1
February    1
March       2
April       6
May         5
June        1
July        0
August      0
September   0
October     2
November    0
December    1


Do you want to plot (yes/no)?: yes
```



Function Tests: for each there is an input and an "instructor value" that the instructor's version of the function returned.

Function Test validate_hashtag

```
Instructor values for validate_hashtag #abcd, #4, #Great!job: True False False
```

Function Test get_hashtag

```
Testing string: s = "#tag1 some words, etc. #2, another #tag2"
Instructor value for get_hastag(s): ['#tag1', '#tag2']
```

Function Test read_data

```
Reading smalldata.csv

Instructor value for read_data(fp): [['michiganstateu', 5, []], ['michiganstateu', 5, ['#MSU']],
['michiganstateu', 5, ['#MothersDay', '#Spartans']], ['MSUnews', 4, ['#ArborDay']], ['MSUnews',
4, []], ['MSUnews', 5, ['#Spartans']], ['MSUnews', 4, ['#MSUPride']], ['MSUnews', 1, ['#MSU',
'#MSUPride']], ['MSUnews', 1, ['#MSUPride']], ['MSUnews', 1, ['#MSU', '#MSUPride']]]
```

Function Test get_user_names

```
Data: [['michiganstateu', 5, []], ['michiganstateu', 5, ['#MSU']], ['michiganstateu', 5,
['#MothersDay', '#Spartans']], ['MSUnews', 4, ['#ArborDay']], ['MSUnews', 4, []], ['MSUnews', 5,
['#Spartans']], ['MSUnews', 4, ['#MSUPride']], ['MSUnews', 1, ['#MSU', '#MSUPride']], ['MSUnews',
1, ['#MSUPride']], ['MSUnews', 1, ['#MSU', '#MSUPride']]]

Instructor value: ['MSUnews', 'michiganstateu']
```

Function Test get_ histogram_tag_count_for_users

```
Data: [['michiganstateu', 5, []], ['michiganstateu', 5, ['#MSU']], ['michiganstateu', 5,
['#MothersDay', '#Spartans']], ['MSUnews', 4, ['#ArborDay']], ['MSUnews', 4, []], ['MSUnews', 5,
```

```
['#Spartans']], ['MSUnews', 4, ['#MSUPride']], ['MSUnews', 1, ['#MSU', '#MSUPride']], ['MSUnews',
1, ['#MSUPride']], ['MSUnews', 1, ['#MSU', '#MSUPride']]]

Instructor value : {'#MSU': 3, '#MothersDay': 1, '#Spartans': 2, '#ArborDay': 1, '#MSUPride': 4}
```

Function Test get_tags_by_month_for_users

```
Data: [['michiganstateu', 5, []], ['michiganstateu', 5, ['#MSU']], ['michiganstateu', 5,
['#MothersDay', '#Spartans']], ['MSUnews', 4, ['#ArborDay']], ['MSUnews', 4, []], ['MSUnews', 5,
['#Spartans']], ['MSUnews', 4, ['#MSUPride']], ['MSUnews', 1, ['#MSU', '#MSUPride']], ['MSUnews',
1, ['#MSUPride']], ['MSUnews', 1, ['#MSU', '#MSUPride']]]

Instructor value: [(1, {'#MSU', '#MSUPride'}), (2, set()), (3, set()), (4, {'#MSUPride',
'#ArborDay'}), (5, {'#MothersDay', '#MSU', '#Spartans'}), (6, set()), (7, set()), (8, set()), (9,
set()), (10, set()), (11, set()), (12, set())]
```

Function Test three_most_common_hashtags_combined

```
Data: [['michiganstateu', 5, []], ['michiganstateu', 5, ['#MSU']], ['michiganstateu', 5,
['#MothersDay', '#Spartans']], ['MSUnews', 4, ['#ArborDay']], ['MSUnews', 4, []], ['MSUnews', 5,
['#Spartans']], ['MSUnews', 4, ['#MSUPride']], ['MSUnews', 1, ['#MSU', '#MSUPride']], ['MSUnews',
1, ['#MSUPride']], ['MSUnews', 1, ['#MSU', '#MSUPride']]]

Instructor value: [(4, '#MSUPride'), (3, '#MSU'), (2, '#Spartans')]
```

Function Test three_most_common_hashtags_individuals

```
Data: [['michiganstateu', 5, ['#Spartans']], ['michiganstateu', 5, ['#MSU']], ['michiganstateu',
5, ['#MothersDay', '#Spartans']], ['MSUnews', 4, ['#ArborDay']], ['MSUnews', 4, ['#MSU']],
['MSUnews', 5, ['#Spartans']], ['MSUnews', 4, ['#MSUPride']], ['MSUnews', 1, ['#MSU',
'#MSUPride']], ['MSUnews', 1, ['#MSUPride']], ['MSUnews', 1, ['#MSU', '#MSUPride']]]

Instructor value: [(4, '#MSUPride', 'MSUnews'), (3, '#MSU', 'MSUnews'), (2, '#Spartans',
'michiganstateu')]
```

Function Test similarity

```
Data: [['michiganstateu', 1, ['#MSUPride', '#MSU']], ['michiganstateu', 5, ['#Spartans']],
['michiganstateu', 5, ['#MSU']], ['michiganstateu', 5, ['#MothersDay', '#Spartans']], ['MSUnews',
4, ['#ArborDay']], ['MSUnews', 4, ['#MSU']], ['MSUnews', 5, ['#Spartans']], ['MSUnews', 4,
['#MSUPride']], ['MSUnews', 1, ['#MSU', '#MSUPride']], ['MSUnews', 1, ['#MSUPride']], ['MSUnews',
1, ['#MSU', '#MSUPride']]]
[(1, {'#MSU', '#MSUPride'}), (2, set()), (3, set()), (4, set()), (5, {'#Spartans'}), (6, set()),
(7, set()), (8, set()), (9, set()), (10, set()), (11, set()), (12, set())]

Instructor value: [(1, {'#MSU', '#MSUPride'}), (2, set()), (3, set()), (4, set()), (5,
{'#Spartans'}), (6, set()), (7, set()), (8, set()), (9, set()), (10, set()), (11, set()), (12,
set())]
```

Grading Rubric

```
Project #9                                          Scoring Summary

General Requirements:

4 pts   Coding Standard 1-9
                    (descriptive comments, function headers, etc…)

Function Tests:
```

```
 2 pts   open_file (no Mimir test)
 2 pts   validate_hashtag
 3 pts   get_hashtags
 3 pts   read_data
 3 pts   get_user_names
 4 pts   get_histogram_tag_count_for_users
 4 pts   get_tags_by_month_for_users
 4 pts   three_most_common_hashtags_combined
 4 pts   three_most_common_hashtags_individuals
 4 pts   similarity

Program Tests
 8 pts   Test1

 6 pts   Test2

 4 pts   Test3 (no Mimir test)
```