CSE 231 Spring 2017

Programming Project 09

This assignment is worth 55 points (5.5% of the course grade) and must be **completed and turned in before 11:59 on Monday, April 10, 2017.**

Assignment Overview

This assignment focuses on the implementation of Python programs to read files and process data using dictionaries and sets.

Assignment Background

What is co-occurrence problem?

You will write a Python program to solve the co-occurrence problem. The co-occurrence problem is stated as follows. We have a file containing English sentences, one sentence per line. Given a list of query words, your program should output the line number of lines that have *all* those words. While there are many ways to do this, the most efficient way is to use sets and dictionaries.

Here is one example. Assume that the following is the content of the file. Line numbers are included for clarity; the actual file doesn't have the line numbers.

1. Try not to become a man of success, but rather try to become a man of value.
2. Look deep into nature, and then you will understand everything better.
3. The true sign of intelligence is not knowledge but imagination.
4. We cannot solve our problems with the same thinking we used when we created them.
5. Weakness of attitude becomes weakness of character.
6. You can't blame gravity for falling in love.
7. The difference between stupidity and genius is that genius has its limits.

(These are quotes from Albert Einstein.)

If we are asked to find all the lines that contain this set of words: {"true", "knowledge", "imagination"} the answer will be line 3 because all three words appeared in line 3. If they appear in more than one line, your program should report all of them. For example, co-occurrence of {"the", "is"} will be lines 3 and 7.

Implementation

You need to implement the following functions:

1) `open_file()`

The `open_file` function will prompt the user for a file-name, and try to open that file. If the file exists, it will return the file object; otherwise it will re-prompt until it can successfully open the file. This feature should be implemented using a `while` loop, and a `try-except` clause.

2) `read_data(fp)`

This function has one parameter: a file object (such as the one returned by the `open_file()` function). This function will read the contents of that file line by line, process them and store them in a dictionary. The dictionary is returned. Consider the following string pre-processing:

1. Make everything lowercase
2. Split the line into words
3. Remove all punctuation, such as ",", ".", "!", etc.
4. Remove apostrophes and hyphens, e.g. transform "can't" into "cant" and "first-born" into "firstborn"
5. Remove the words that are not all alphabetic characters (do not remove "can't" because you have transformed it to "cant", similarly for "firstborn").
6. Remove the words with less than 2 characters, like "a"

Hint for string pre-processings mentioned above:

To find punctuation for removal you can import the `string` module and use `string.punctuation` which has all the punctuation. To check for words with only alphabetic characters, use the `isalpha()` method.

Furthermore, after pre-processing, you add the words into a `dictionary` with the key being the word and the value is a `set` of line numbers where this word has appeared. For example, after processing the first line, your dictionary should look like:

```
Data_dict = {"try":{1}, "not":{1}, "to":{1}, "become":{1},
"man":{1}, "of":{1}, "success":{1}, "but":{1}, "rather":{1},
"value":{1}}
```

This should be repeated for all the lines; the new keys are added to the dictionary, and if a key already exists, its value is updated. At the end of processing all these 7 lines, the value in the dictionary associated with key `"the"` will be the set {3, 4, 7}. (Note: the line numbers start from 1.)

3) `find_cooccurance(D, inp_str)`

The first parameter is the dictionary returned by `read_file`; the second one is a string called `inp_str`. This `inp_str` contains zero or more words separated by white space. You need to split them into a list of words, and find the line numbers for each word. To do that, use the *intersection* or *union* operation on the sets from `D` (you need to figure out which operation is appropriate). Then convert the resulting set to a sorted list, and return the sorted list. (Hint: for the first word simply grab the set from D; for subsequent words you need to use the appropriate set operation: intersection or union.)

4) `main()`

The `main` function of your program should call the three functions above. Loop, prompting the user to enter space-separated words. Use that input to find the co-occurrence and print the results. Continue prompting for input until "q" or "Q" is input.

Call to main required to be:

```
if __name__ == "__main__":
    main()
```

Very important considerations

Every time you want to look up a key in a dictionary, first you need to make sure that the key exists. Otherwise it will result in an error. So, always use an if statement before looking up a key:

```
if key in data_dict:
      ## the key exists in a dictionary, so it is safe to use data_dict[key]
```

After you completed the program, see how it works for the two files we provide: einstein.txt and gettysburg.txt. We also provide function_test.py to test your functions on the pimpernel.txt file.

Optionally, try a larger file. For this, we encourage you to download an ebook from Project Gutenberg (http://www.gutenberg.org) which offers free ebooks in variety of topics, and test your program.

Sample Output

Function Test

```
Testing proj09 functions using pimpernel.txt

Testing read_data.
Dictionary should be:
  {'seek': {1, 2, 3}, 'everywhere': {3}, 'him': {1, 2, 3}, 'those': {3},
'frenchies': {3}, 'they': {1, 2}, 'here': {1}, 'there': {2}}
Dictionary D:
 {'those': {3}, 'here': {1}, 'everywhere': {3}, 'him': {1, 2, 3},
'frenchies': {3}, 'seek': {1, 2, 3}, 'they': {1, 2}, 'there': {2}}

Testing find_cooccurance.
Test 1 should be: [1, 2, 3].
[1, 2, 3]

Test 2 should be: [1, 2].
[1, 2]

Test 3 should be: [1, 2, 3].
```

```
[1, 2, 3]

Test 4 should be: [1].
[1]
```

Test 1

```
Enter a file name: einstein.txt

Enter space-separated words: the
The co-occurance for: the
Lines: 3, 4, 7

Enter space-separated words: the is
The co-occurance for: the, is
Lines: 3, 7

Enter space-separated words: true knowledge imagination
The co-occurance for: true, knowledge, imagination
Lines: 3

Enter space-separated words: q
```

Test 2

```
In [11] runfile

Enter a file name: xxxx

Error -- Enter a file name: einstein.txt

Enter space-separated words: The
The co-occurance for: the
Lines: 3, 4, 7

Enter space-separated words: can't
The co-occurance for: can't
Lines: 6

Enter space-separated words: nature
The co-occurance for: nature
Lines: 2

Enter space-separated words: cat
```

```
The co-occurance for: cat
Lines: None.

Enter space-separated words:
The co-occurance for:
Lines: None.

Enter space-separated words: Q
```

Test 3

```
In [12] runfile

Enter a file name: gettysburg.txt

Enter space-separated words: nation
The co-occurance for: nation
Lines: 2, 6, 9, 23

Enter space-separated words: here dead
The co-occurance for: here, dead
Lines: 14, 22

Enter space-separated words: It is
The co-occurance for: it, is
Lines: 10, 17, 19

Enter space-separated words: q
```

Scoring Rubric

General Requirements

__0__ (5 pts) Coding Standard
(descriptive comments, mnemonic identifiers, format, etc...)

Tests

__0__ (10 pts) Pass Test 1

__0__ (10 pts) Pass Test 2

__0__ (10 pts) Pass Test 3

__0__ (12 pts) Pass Function Test (open_file, read_file, find_cooccurance)

```
__0__ (3 pts) uses dictionary


__0__ (3 pts) find_cooccurance() uses set intersection


__0__ (2 pts) Handles edge cases, such as empty input, no output

TA Comments:
```

Optional Testing

We provide the following frameworks for testing

1. `function_test.py` for testing your functions.
2. `run_file.py` runs the three test cases shown in the sample above and are the three test cases that are collectively worth 30 of the project's 55 points.

Educational Research

When you have completed the project insert the 6-line comment specified below.

For each of the following statements, please respond with how much they apply to your experience completing the programming project, on the following scale:

1 = Strongly disagree / Not true of me at all
2
3
4 = Neither agree nor disagree / Somewhat true of me
5
6
7 = Strongly agree / Extremely true of me

****Please note that your responses to these questions will not affect your project grade, so please answer as honestly as possible.****

Q1: Upon completing the project, I felt proud/accomplished

Q2: While working on the project, I often felt frustrated/annoyed

Q3: While working on the project, I felt inadequate/stupid

Q4: Considering the difficulty of this course, the teacher, and my skills, I think I will do well in this course.

Q5: I ran the optional test cases (choose 7=Yes, 1=No)

Please insert your answers into the bottom of your project program as a comment, formatted exactly as follows (so we can write a program to extract them).

```
# Questions
# Q1: 5
# Q2: 3
# Q3: 4
# Q4: 6
# Q5: 7
```