CPSC 3400 Languages and Computation
Winter 2018

Homework 1
Due: Monday, January 22 at 10:55am

Summary

In this assignment, you will be processing two letter pairs. For example, the word "Banana" has the pairs "ba", "an", "na", "an", and "na". This means that "ba" occurs once and the pairs "an" and "na" appear twice.

Your program will read in a text file and count the number of times each pair occurs in the file plus some additional processing. One of the learning objectives is to get familiar with Python data structures so be sure to implement the functions as described.

Directions

You must create the following functions in the file `hw1.py`.

`countPairs`
Parameters: Name of the input file (string)
Returns: A dictionary that has two letter strings as keys and the frequency as the value
Description: Reads a text file and returns a dictionary that has two letter pairs as keys and the frequency as the value. The dictionary only contains pairs that appear in the file.
Assumptions: The input file exists and contains at least one pair.
Additional Notes:

- The pairs are not case-sensitive. A capital 'A' and lowercase 'a' are the same letter.
- Each key in the dictionary should be two lowercase letters.
- Anything that is not a letter (such as digits, symbols, and spaces) is considered a delimiter. Pairs do not span across delimiters. For instance, in "`E-mail: r2d2@seattleu.edu`", these pairs do not exist as there are one or more delimiters separating them: "em", "lr", "rd", "ds", "ue".

`getTopFivePairs`
Parameters: Dictionary returned from `countPairs`
Returns: A list of tuples. Each tuple in the list has two entries: the first entry is a two letter string and the second entry is an integer.
Description: Creates and returns a list of that contains the top five most frequent pairs in order from the most frequent to the least frequent. The pairs must be tuples like this: `('an', 4)`. If there is a tie, place the pair that is alphabetically earlier first. If there is a tie such that multiple pairs could occupy the last (5th) position, include all such pairs (this means the list could contain more than five pairs). If there are fewer than five pairs, simply return the appropriate list with fewer than five pairs.

`createFollowsDict`
Parameters: Dictionary returned from `countPairs`, a single letter
Returns: A dictionary where individual letters are the keys and integers are the values.
Description: Creates and returns a dictionary that has each letter in the alphabet as a key and the frequency of how often that letter follows the given letter (parameter) in the dictionary. This dictionary must have 26 entries in it – one for each letter. If a letter never follows the given letter, a dictionary entry of zero must be present.

In addition to these functions, you must do the following steps to create a simple test driver. This code should reside outside the function:

1. Get the name of an input file from the command line (using `sys.argv`). WARNING: Do not prompt the user for a file name.
2. Call `countPairs` with the input file from the command line storing its result in `pairs`.
3. Print out the number of keys in `pairs`.
4. Compute and print out the total number of pairs (sum of the values in `pairs`).
5. Call `getTopFivePairs` with `pairs`.
6. Print the result from step 3 to the screen.
7. *This step must be done using a loop.* For each vowel (`a`, `e`, `i`, `o`, and `u`):
    a. Print the vowel used during this iteration (a single character).
    b. Call `createFollowsDict` with `pairs` and the vowel for this iteration.
    c. *Using a list comprehension* on the dictionary returned from part b, create a list of 26 integers such that the first element of the list is the frequency of 'a', the second element is the frequency of 'b', etc.
    d. Print the list from step c.

Sample Input File and Output

A sample input file is provided on `cs1` at:

`/home/fac/elarson/cpsc3400/hw1/sample.txt`

Here are the contents of that file:

```
These words should never appear in computer error messages:
exception
socket
listener
code
receiver
thread
process
unknown
trap
protocol
null
```

If your program is working correctly, it should have the following output:

```
75
101
[('er', 5), ('es', 4), ('ce', 3), ('co', 3), ('oc', 3), ('ro', 3)]
a
[0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
e
[2, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 5, 4, 1, 0, 1, 0, 1, 0, 0]
i
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0]
o
[0, 0, 3, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 2, 0, 1, 1, 0, 1, 0, 0, 0]
u
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]
```

A few comments about the output:

- The first line is the number of keys (Step 3).
- The second line is the number of pairs (Step 4).
- The third line is the result of `getTopFivePairs` (Step 6). Note it has six pairs due to a tie in this instance.
- The remaining lines are from the loop that calls `createFollowsDict` for each vowel (Step 7).
- Keep the output in this format. Do not try to "beautify" the output in any fashion.

IMPORTANT! Do not assume that testing is complete if your program produces the correct output for this provided input file. During grading, your assignment will be tested with other input files.

Grading

The assignment will be graded in accordance with the programming assignment expectation handout.

Grading is broken down as follows:

`countPairs`	12 points
`getTopFivePairs`	12 points
`createFollowsDict`	12 points
test driver	14 points
TOTAL	50 points

For full credit for a function, it must: pass all tests and adhere to the type requirements specified in the descriptions of the functions.

Functionality is graded for each of the three functions as follows:

- all tests pass (no penalty)
- all tests pass except one -4
- all tests pass except 2-3 tests (sample input passes) -6
- all tests pass except 2-3 tests (sample input fails) -8
- at least one test is correct but more than 3 fail -10
- no tests are correct -12

Type errors:

`countPairs`

- 9 point deduction if the output is not a dictionary (no further type penalties)
- 2 point deduction if the dictionary keys are not two character strings
- 2 point deduction if the dictionary values are not integers

`getTopFivePairs`

- 4 point deduction if the output is not a list
- 4 point deduction if the items in the list are not tuples in the form (string, integer)

`createFollowsDict`

- 8 point deduction if the output is not a dictionary
- 2 point deduction if dictionary keys are not single letters
- 2 point deduction if dictionary values are not integers

The test driver is graded (14 points) as follows:

Step 3 (number of keys): *2 points*

- Works in all tests (no penalty)
- Fails in one or two tests -1
- Fails in three or more tests -2

Step 4 (total number of pairs): *3 points*

- Works in all tests (no penalty)
- Fails in one or two tests -2
- Fails in three or more tests -3

Step 7 (`createFollowsDict` for each vowel): *8 points*

- Works in all tests, properly formatted (no penalty)
- Works but formatting issues -1 (or more if severe)
- Fails in one or two tests -2
- Fails in three or more tests -7
- Did not use a loop -4
- Did not use a list comprehension -4

Formatting:	*1 point*
• No extra output	(no penalty)
• Extra output	-1

In addition, programs may lose additional points as follows:

- Up to 10 points may be deducted if programs if they exhibit poor readability or style. See the programming assignment expectation handout.
- 6 point deduction if the file name is not obtained from the command line.
- Programs that contain syntax errors will receive a zero.

Submitting your Assignment

On `cs1`, run the following script in the directory with your program:

```
/home/fac/elarson/submit/cpsc3400/hw1_submit
```

This will copy the file `hw1.py` to a directory that can be accessed by the instructor. Please be sure to keep the same file names or the submission program will not work. Only the last assignment submitted before the due date and time will be graded. ***Late submissions are not accepted and result in a zero.***