Note: this has the flavor of a Track B "Less-Mathematical Applications" assignment, but you are welcome to do this for credit whether you intend to complete Track A or Track B. You do not need to complete this assignment if you complete 1F instead.

PIC 16, Winter 2018 – Assignment 2W

Assigned 1/17/2018. Code (a single `.py` file) due by the end of class 1/22/2018 on CCLE. Hand in a printout of this document with the self-assessment portion filled out by the end of class on 1/22/2018.

In this assignment, you will write a Python program to find the unique words in a body of text and the number of times each appears. First, you will write a very concise but inefficient program, then you will modify the algorithm so it can feasibly be used on larger works.

Task

Copy the example text of lorem ipsum into a string. See the official documentation on strings to find a method that will roughly split the text into a list of words (at spaces). Don't worry about the commas, periods, capitalization, etc… for now; we'll take care of those in a future assignment.

1. Now, in a single line, use dict comprehension to produce a dictionary in the format where the key is a (unique) word and the value is the number of times that word appears in the list. For example, `{'laborum.': 1, 'irure': 1, 'in': 3, …, 'dolor': 2, 'voluptate': 1, 'eiusmod': 1}` (Notice that `laborum.` rather than `laborum` appears as a "word" above; don't worry about it today. Also, `dolor` and `dolore` appear as separate words, although they have the same root, and `ut` and `Ut` appear separately even though they're just differently capitalized. We'll take care of all this later.)

2. Calculate how many *unique* words appear in the text.

3. While compact, the dict comprehension method of producing the dictionary quite inefficient (computationally. Can you see why?) A more efficient algorithm is again to step through each word in the list, but rather than *count* the number of appearances of each word, *increment* a counter for the word each time it is encountered. Implement this. (I'm not being 100% explicit about the algorithm on purpose; please think about it a bit and then ask me for help if you are having trouble.)

4. Download the plain text of Agatha Christie's "The Mysterious Affair at Styles" from Project Gutenberg and save it in the same folder as your Python code. Skip ahead to 7.2 of the tutorial to see how to load the text from the file into a string. How long does it take for your algorithm from 1 to produce the dictionary (interrupt the kernel if you get bored…)? How long does it take for your more efficient algorithm? (It is optional to have your program calculate execution time. Search online if you're interested….)

Self-Assessment

1. If you have been able to produce a dictionary for lorem ipsum, what is the count of the word "dolore" reported in your dictionary?

2. How many unique words did your program count in the text of lorem ipsum?

3. If you have been able to produce a dictionary for "The Mysterious Affair at Styles" using a more efficient algorithm, what is the count of the word "found" reported in the dictionary?

If your dictionary counted "dolore" twice, you get 1/3 credit. If you count 64 unique words in lorem ipsum, that's another 1/3. And if you found "found" about 47-50 times, that's the final 1/3.