Lab 7 – Dictionaries and Sets

Part 1: Rework the last lab to use *sets* in certain functions.

Here's what you'll do:

1. Change *find_common_unique_elements* to use sets. **The resulting function using sets may be coded in one line** (but take as many lines of code you like)
2. Change *gen_return_random_number_list* to ***gen_return_random_unique_number_list*** and have the function return a *list of unique random numbers.*

Bottom line – you'll remove about 4 lines in one function and change one line in another.

Part 2: Create a table of word and letter counts

Write a program that parses out the G*ettysburg Address and produces a* ***sorted*** *table of word and letter counts that resemble:*

Word	*Word count*
But	*1*
For	*1*
…..	
world	*1*
years	*1*
Letter	*Letter count*
a	*102*
…	
z	*0*

The file **skel.py** contains close to *100 lines of hints disguised as comments*.

Here's a list of tasks for this lab; this list is described in the above-mentioned file as well:

1. Search and use an *import* from the Python standard library that is a *string of lower-case characters* (you'll need them later)
2. Import the Gettysburg Address from the *included Python module* **getty.py**
3. Remove the punctuation characters in the text. The list of characters as well as guidance for this (and all) task is in skel.py. Look at *string functions* – I bet there's one that will remove *a single character*.
4. Your program will count word occurrences. You *don't want to count the same word more than once;* good idea to create a structure containing unique words. AND...while you're doing that, *convert the words to lower-case and sort them* (your tables will be sorted)
5. Create a dictionary containing the word as the key and the count of that word as the value. Can do that in one line but a one-liner is not required. The structure of *unique words* will hold your keys BUT you need the count these words in the string that has been stripped of punctuation in step 3.
6. Print the table as shown in skel.py; words left-justified in a 20-column field, word counts right-justified in a 30-column field.
7. Use the string of lower-case characters imported from some standard library that you located earlier and *count each letter in the string created in step 3.* The code will be almost identical to that used to create the word table.
8. Print the letter/letter count table using the same code from step 6, with the changes to the header (letter versus word, I mean)

The program may be coded in **less that 20 lines (not much less, but...)**