Homework 2

Instructions

For this homework you will create a python notebook (`.ipynb` file) using Google Colab and submit a link to your notebook. This can be done by clicking on the 'share' icon in the top right. No edits should be made after the due date!

The purpose of this homework is to practice with control flow and some of our common data objects. Most homework assignments will have a part that pushes you beyond what was in the lectures! Learning to search for the right questions and browsing stackoverflow are really life skills that we should hone :)

Part 1 - List and `numpy` practice

For this part, import the `numpy` library. We'll use the *new* way to do random number generation using `numpy` given here to create some data and plot it.

a. Generate 30 values from a Poisson distribution with $\lambda = 3$ and print those out. Note: each time you run your code you'll get different values. To avoid this, first **set a seed** for your random number generator.

   Run the code a few times and make sure you are getting the same values generated!

b. Use `numpy`'s mean **function** to find the mean of your 30 values.

c. Use a list comprehension to generate 5000 data sets from the Poisson with mean 3 and return the mean (within the list comprehension - no need to return the values themselves).

   If it helps, write this via a for loop first and then translate it to a list comprehension! Pseudo code for the loop:

   - Initialize an empty list (not needed for the comprehension)
   - Have your for loop iterate from 0 to 4999
     - Notice you don't really use the index anywhere! Commonly you'd use _ as the index in a case like this.
   - Generate the data using code from above and find the mean of the values
   - Append that mean value to your list

d. Now run the code below to create a histogram of the means. You should see a histogram that looks roughly like a Normal distribution with mean 3 and standard deviation $\sqrt{1/10}$.

```
import matplotlib.pyplot as plt
plt.hist(your_list_of_means)
```

e. Lastly, we'll wrap the above process into a function.

   - The function should take in
     - a value for the mean of the poisson - with a default of 3

  - a value for the sample size - with a default value of 30
  - a value for the number bins for the histogram - with a default value of `None`. The bins can be set in the `plt.hist()` function via `plt.hist(your_list_of_means, bins = 15)`

- The function should check if the `bins` argument is `None`.
  - If `True` then use the `plt.hist()` code without specifying the `bins` in order to use the default bin number.
  - If `False` then specify the `bins` user the value specified by the user.
- The function should output a graph. Test it on a few combinations of your inputs!

Part 2 - Dictionary Practice

In this part, we'll read in a raw text file. Then we'll do the first steps of a basic sentiment analysis.

After reading the data in, we'll use string methods to split the data into words. We'll store the words as keys in dictionaries with the value being the count of the number of times the words occurs. Create sections using markdown (along with text explaining what we are doing) to organize this part of the homework!

a. First, we need to read the data into python. No need to focus too hard on the code below (modified from here). Run it to read the text of *Oliver Twist* into our python session (downloaded from Project Gutenburg).

```
import urllib.request
url = "https://www4.stat.ncsu.edu/~online/datasets/charles-dickens-oliver-twist.txt"
urllib.request.urlretrieve(url, 'charles-dickens-oliver-twist.txt')
#now that we have a reference to the url, read the data into a string
with open('charles-dickens-oliver-twist.txt', 'r') as f:
lines = f.read()
```

b. The text has now been read in as one long string called `lines`. Print out the first 1000 characters of the string. Then print out the last 1000 characters.

c. You should see that there is a lot of 'front' and 'end' matter. We want to subset the string first to remove the front and end matter. That is, we want to remove everything up to "CHAPTER I" (but leave "CHAPTER I" in the resulting text) and remove everything that isn't part of the actual book at the end ("End of the Project Gutenberg EBook").

Hints:

- Use the `.find()` string method to find indices corresponding to each of the quoted text bits above. Then subset the string using slicing.
- Print out the first 250 characters and the last 250 characters of the resulting object

d. Our goal is to iterate through the words in the book and count the number of times each word occurs. We can use a dictionary to store this type of data! For instance, you'll end up with a key of "the" with a corresponding value being the number of times "the" occurred.

Some hints:

- Replace all the `\n` values with spaces as we did previously
- Split the string by spaces and save that resulting list
- Initialize a dictionary to save your words (keys) and counts in (values)
- Import the `string` module as you'll need some things from that
- Use a for loop, iterating over the list elements. When iterating:
  - If you get a space, *ignore* that iteration of the loop (that is move to the next iteration)

  - Convert each ‘word’ to lower case
  - We’ll need to remove any punctuation (for instance, “end.” would need the ‘.’ removed).
    * The object `string.punctuation` has a list of punctuation marks
    * You can write a loop inside your loop (a nested for loop) that iterates over the values in `string.punctuation`. If the punctuation value exists in the ‘word’ then you can use the `.replace()` method to replace it with an empty string “ “. This loop should remove any punctuation! (Note: this will make words like didn’t into didnt, that’s fine - ignore that as it will still count appropriately for the most part.)
  - Now the ‘word’ is processed and we are ready to count. Use `if/else` logic to check if the the word already exists in the dictionary. If it does, add one to the associated value. If not, add that key to the dictionary with a value of 1.

e. We want to print out the 40 most prevalent words and how prevalent they were (i.e. the 40 most prevalent key/value pairs).

   - Use the `sorted()` function with appropriate key (this may take a quick google) to obtain a list of keys in descending order (largest to smallest).
   - Grab the first 40 of those keys and use a for loop to print out the keys with their values.

f. You probably note a bunch of words that aren’t that important (like `the`). Remove the following words from our dictionary and then print the 40 most common words:

   ‘the’, ‘and’, ‘to’, ‘of’, ‘a’, ‘in’, ‘that’, ‘with’, ‘for’, ‘at’, ‘on’

Submission

Now you can grab the ‘share’ link and submit that! Good luck and let us know if you run into issues.