Homework 2

For this homework you will create a python notebook (`.ipynb` file) and output a `.html` file. Both of these files should then be uploaded to wolfware in the assignment link!

The purpose of this homework is to practice with loops, if/then/else logic, string methods, `numpy`, and `pandas`. 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 (7 pts)

For this part, import the `numpy` library and the *new* way to do random number generation using `numpy` given here.

a. Generate 10 values from a standard normal distribution and print those out. Note: each time you run your code you'll get different values unless you set a seed (feel free to if you'd like, not required). (3 pts)

b. To start, use `numpy`'s mean function to find the mean of your 10 values above. (1 pt)

c. Use a list comprehension to generate 1000 data sets of size 10 from a standard normal. In your list comprehension, return the mean of the 10 randomly generated values. (3 pts)

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 999
  - 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 it
- Append that mean value to your list

Part 2: Reading Data with `pandas` Practice (12 pts)

1. Read in the BreastCancer.dat data file available in the assignment link. (Open the file in a program such as notepad or wordpad to determine the delimiter - although a program like notepad++ is a better choice.)
   a. Save the data as an object called `cancer_data`. (2 pts)
   b. Use the `.head()` method to look at the data. (1 pt)
   c. Print out all rows where the `size` is larger than 30. (1 pts)
   d. Print out all rows where the `size` is greater than 30 and the `grade` is 3. (2 pts)
2. There are two files about mosquitos available at:

- https://www4.stat.ncsu.edu/~online/datasets/mosquito.txt
- https://www4.stat.ncsu.edu/~online/datasets/mosquito2.txt

a. Repeat the above process to read in the mosquito.txt file as an object called `mosq_data`. (2 pts)
b. Read in the mosquito2.txt file. Note this file doesn't contain column names! The columns are the same as the other file though. Use an attribute from `mosq_data` to assign the column names as you read in the data. Save this data as an object called `mosq_data2`. (3 pts)
c. Combine the two datasets into one data frame using the `concat()` function from `pandas`. (1 pts)

Part 3: String, loop, and Dictionary Practice (22 pts)

Read in the data (3 pts)

The novel *Pride and Prejudice* by Jane Austen is available in a `.txt` file in the assignment link. You should

- download the `.txt` file
- place the `.txt` file in a directory you can find when you open `JupyterLab` (say the directory your terminal is in when you open `JupyterLab`)
- Create a section header for this section with the following info:

The code below will read in the novel *Pride and Prejudice* by Jane Austen from a local text file. Our file was originally downloaded from this repo and the text itself comes from Project Gutenberg. The first 1000 characters are printed out below to check that it was read in correctly.

- Lastly, read in the text as one long string via:

```
#read in string
with open('jane-austen-pride-prejudice.txt', 'r') as f:
lines = f.read()
print(lines[:1000])
```

Trim the Data Down to the Text (3 pts)

The text has now been read in as one long string. We want to subset the string and basically remove the front matter and end matter. That is, we want to remove everything up to "CHAPTER I" (but leave include "CHAPTER I" in the resulting text) and remove everything after the last "Transcriber's note" (including the text "Transcriber's note").

- Create a new section and use markdown to give some text explaining what we are about to do.
- Use the `.find()` string method to find indices corresponding to each of the quoted text sections above. Note that there are two "Transcriber's note"s and the last one corresponds to the second occurrence - you'll need to use additional arguments on the `.find()` method.
- Create a new object (variable) that corresponds to just the text we want. Do this by slicing the original long string appropriately.
- Print out the first 250 characters. Then print out the last 250 characters of the resulting object (it should start with "Chapter I" and end with (spoiler alert - they get together in the end) "had been the means of uniting them." followed by a few line returns and a row of stars.

Obtain Just the First Chapter (8 pts)

Using the resulting object from the previous part, we'll now obtain just the first chapter.

- Create a new section and some text explaining what we are about to do.
- Use similar string methods to return just the text between "CHAPTER I" (including that text) and "CHAPTER II" (not including that text)
- Submit the resulting object in your code chunk (that is, just type the object name). You should see a quoted string that includes the special characters for line returns `\n`.
- Use the print function to print the object out as well (this should now show with better formatting)

Now we'll count how many words are in the first chapter.

- Replace the `\n` characters with a `' '` using a string method
- Split the data by the `' '` characters using a string method
- Using the resulting object from the above two operations (this should be a list), write a for loop that iterates over the elements of the list and counts the number of words
- Initialize a word count variable at 0 and use augmented assignment to count the words

- Some of the list elements will be empty strings - don't count these! use an `if` statement within the for loop to check if the list element you are on is an empty string (remember this returns `False` if treated as a boolean)

Lastly, we'll print out the following sentence: "There are #### words in the first chapter."

- Print this out using `+` to concatenate strings together with the number of words object you created in your for loop. Remember that you can't concatenate strings with integers so you'll need to cast the word count variable as a string.

Count the Occurrences of Words (8 pts)

Rather than count all the words in chapter 1, let's count the number of times each word appears. We'll 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:

- Import the string module as we'll use it below
- Create an empty dictionary to store the words and the number of occurrences
- When iterating over your list of words:
  - 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 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.
- Print out the sum of the dictionary values (you should get the same value from the previous part where we just counted the words!)

Lastly, write a for loop to print out the first 20 key/value pairs in your dictionary. Recall that dictionaries don't have an ordering, but if you obtain the keys and turn them into a list you can then just use the first 20 that you see there.

Files to Submit

- Make sure all cells are run
- Click on File –> Save and Export Notebook As… –> HTML

That's it! Upload both the `.ipynb` and `.html` files to the assignment link.