CSI Lab 12

Tuesday, April 8th

Introduction

Goals:

- Write some code that uses a list
- Write some code that uses a dictionary.

Today we are going to do some text analysis on an American classic - "Green Eggs and Ham" You can download the full text in the following file:

- greeneggs.txt

I have heard the following fact about "Green Eggs and Ham" several times

Green Eggs and Ham is one of Seuss's "Beginner Books", written in a very simple vocabulary for beginning readers. The vocabulary of the text consists of just fifty different words.[2]

Is that actually true?

In today's lab we will use lists and dictionaries to confirm that this is true and play with some ways to look at how many times each word is used.

While I won't require it, I strongly encourage you to work with a partner.

Before you begin, please create a new Python file called lab12.py and write the traditional comments at the top (file, author(s), and description). In addition, copy the following function into lab12.py where you replace my name with your name(s):

```
#Function printName
#Inputs: None
#Outputs: None
#Description: Prints a name.

def printName():
    #TODO: Modify this function to print YOUR name(s)
    print("Name: Sarah Diesburg")
```

Activity A : Creating a function that will help you with set up (Using Lists)

In this lab you will be reading a variety of files containing text. I have three files you can use for testing purposes:

- The preamble to the constitution (the smallest so a good starting point)
- Green Eggs and Ham (mid-sized and fun to analyze)
- Hamlet (Very large and a good way to see what happens as the problem grows)

In two different activities in this lab you will be analyzing the individual words in these text files. In both of those activities you will need to process the text file into these individual words. Thus, it might be helpful to have a piece of code that does that common prep work.

By the time you are done with this activity, I want you to have a **function** called createWordList() that:

- takes a single string as a parameter which is assumed to be the name of a text file
- opens the file for reading
- reads that file either all at once (using read()) or one line at a time (using readline()). Either way, you will have a **STRING**, and you need to:
    - "clean up" punctuation by removing all of the following characters:

      ```
      .,!?;(){}:
      ```

      (Note: To clean up the punctuation, take a look at the Python string method replace(). Notice how you can replace things you don't want with the empty string.)

    - "clean up" the dash "-" by replacing it with a space " ". (Some of the words, like "sam-I-am", should probably be counted as 3 words.)
    - converts everything to lowercase
    - splits it up based on words (remember, split gives you a **LIST**)
    - add each of these word to a word master list
- return the final word master list.
- and, the method itself is stored in a file called lab12.py

For example,

```
>>> words = createWordList("preamble.txt")
>>> words
['we', 'the', 'people', 'of', 'the', 'united', 'states', 'in', 'order',
orm', 'a', 'more', 'perfect', 'union', 'establish', 'justice', 'insure',
ic', 'tranquility', 'provide', 'for', 'the', 'common', 'defense', 'promo
e', 'general', 'welfare', 'and', 'secure', 'the', 'blessings', 'of', 'li
'to', 'ourselves', 'and', 'our', 'posterity', 'do', 'ordain', 'and', 'es
, 'this', 'constitution', 'for', 'the', 'united', 'states', 'of', 'ameri
>>> len(words)
52
>>> words = createWordList("greeneggs.txt")
>>> len(words)
804
```

Activity B : Figuring out how many *unique* words are in the file (Using Lists)

The results of Activity A are nice, but they don't answer our question about whether or not there are only 50 words in Green Eggs and Ham. Heck, the answer given is that there are 804. But the word "like," for example, is in there many times. We want to figure out how many UNIQUE words are in the file. It turns out this isn't too difficult to do by using a list.

What I will want

By the time you are done with this activity, I want you to have a **function** called countUniqueWords() that:

- takes a single string as a parameter which is assumed to be the name of a text file

- passes that string to the function from Activity A and grabs the resulting list
- creates a second, empty, output list
- for each word in that original word list it:
    - Checks to see if that word is in the output list
    - If it IS NOT, than it adds the word to the output list
    - If it IS then it does nothing
- then returns the number of unique words in that file by returning the size of the output list
- and, the method itself is stored in a file called lab12.py

Think about this problem for a little bit. It might sound easy at first, but how does the actual counting work? How can we take a section of the book like:

```
I do not like them in a box.
I do not like them with a fox.
```

and get the correct count. Notice that while there are 16 words in that section, there are only 10 UNIQUE words. In other words, I need to write some code that does something like:

```
i => 1 word
do => 2 words
...
a = > 7 words
box => 8 words
i => already seen this so 8 words
do => already seen this so 8 words
...
with => 8 words
a => already seen this 9 words
fox => 10 words
```

Well, we could keep a **list** of words that we have seen. Thus, we are really saying:

```
start with an empty list of words
look at the first word => "i"
        Is that in my list? => no
        Add it to my list => ["i"]
look at the next word => "do"
        Is that in my list? =? no
        Add it to my list => ["i","do"]
...
look at the next word => "box"
        Is that in my list? => no
        Add it to my list => ["i", "do", "not", "like", "them", "in", "a",
"box"]
look at the next word => "i"
```

```
        Is that in my list? => yes
        move on => ["i", "do", "not", "like", "them", "in", "a", "box"]
look at the next word => "do"
        Is that in my list? =? yes
        move on => ["i", "do", "not", "like", "them", "in", "a", "box"]
...
look at the next (last) word => "fox"
        Is that in my list? => no
        Add it to my list => ["i", "do", "not", "like", "them", "in", "a",
"box", "with", "fox"]
return how many words are in my list
```

For example,

```
>>> howManyUniqueWords("preamble.txt")
38
>>> howManyUniqueWords("greeneggs.txt")
50
>>> howManyUniqueWords("hamlet.txt")
4996
```

Activity C : How many times does each word appear? (Using dictionaries)

So Activity B produced cool results. We now know that it is true - Green Eggs and Ham does in fact contain only 50 unique words. But it's a reasonably long book. I got thinking about how many times each word occurs - how many times do they use the word "eat" or "sam" ?

This is a perfect place to use a dictionary - the keys in the dictionaries are the words that are found while the value associated with each key is the number of times that word is found.

In this part of the lab I want you to write a method that tallies the occurrences of each unique words:

What I will want

By the time you are done with this activity, I want you to have a **function** called countTimes() that:

- takes a single string as a parameter which is assumed to be the name of a text file

- passes that string to the function from Activity A and grabs the resulting list
- for each word in that list it:
    - Checks to see if that word is in the dictionary yet
    - If it IS, it reads the value, adds one (you just found it again) and sets the key to this new value
    - If it IS NOT, than it adds the word as the key and uses a value of 1 (this is the first time you have seen it)
- At the end you should PRINT each word in the dictionary along with it's frequency somewhat like the screenshot below
- and, the method itself is also stored in a file called lab12.py

For example, my results looked like this (yours MAY be in a different order).

```
and-> 24
sam-> 19
be-> 4
house-> 8
```

...

```
will-> 21
anywhere-> 8
green-> 10
the-> 11
or-> 8
```

You are required to complete Activities A-C for full credit to this lab.

Activity D : Sorting the results? (Bonus option:this one isn't too hard)

Let's figure our how to make the results a little more useful. Figure out how to alphabetize the word list:

```
a         -> 59
am        -> 16
and       -> 24
anywhere-> 8
are       -> 2
```

...

```
try      -> 4
will     -> 21
with     -> 19
would    -> 26
you      -> 34
```

The extra formatting is not required but encouraged.

Activity E : Sorting the results? (Bonus Option - this one is a bit of a challenge)

Figure out how to sort the results by frequency.

```
i        -> 84
not      -> 83
them     -> 61
a        -> 59
like     -> 44
in       -> 40
```

...

```
that     -> 3
are      -> 2
good     -> 2
they     -> 2
thank    -> 2
if       -> 1
```

Getting Credit for this assignment

This week I will again ask you to submit your code for electronic grading, using the eLearning submission system.

Follow the directions on the system to select the appropriate course and assignment and submit

- lab12.py

If you worked with a partner, make sure that both you and your partner's names are in the comment header at the top of the file ***and*** in the printName() function.