Elizabeth Sundsmo
Data Mining Project
Due 22.25.16

Project Overview:

My data sources were from the Gutenberg Project-- *The Book of Nature Myths* by Florence Holdbrook and *The Pheonix and the Carpet* by E. Nesbit. I used python tools including dictionaries, string methods, and multiple functions to sort all the words within the texts and isolate repeated words and long words in common to enable a linguistic understanding of similarity between pieces of literature.

Implementation:

At a high level, the architecture of this system is pretty simple. After obtaining the text, the huge string is trimmed of spaces and punctuation, and tallied into a dictionary. That dictionary is then sent two two other places. In the first place is is sorted by its values-- how often each word it contains occurs-- then a copy is made of the words that recur the most. In the second place it is compared to another dictionary, and long words both dictionaries contain are copied onto a list. These lists are then collected and printed to the terminal for the reader.

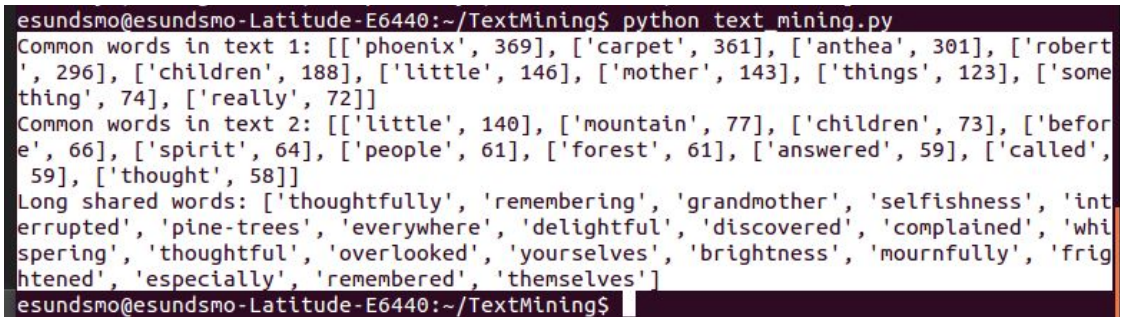
When i was writing create_dict() i was initially going to structure it like the DNA code for the first mini-project-- but i was dissuaded by the amount of typing. I started looking into different sorting abilities and string methods, and stumbled upon a gem of an algorithm-- a single line that would cut up the string and place all of the cut words into a list:

```
word_list = [s.strip(string.punctuation) for s in text.split()]
```

In short, it breaks apart the input (text) wherever there is a space, and removes any punctuation from the newly cut string before storing it in a list. Initially i wasn't going to use this line, but because it was so succinct and i understood it i decided it was a good line to use.

Results:

The chosen words were all incredibly visual and gives a window into what is happening in the text without actually reading it page for page. By changing the word length requirement, a variation in words commonly recurring within their respective texts can be explored, giving more insight into book content. The long words in common for this set of books seems to have a lot of juxtaposition-- brightness/mournfully, thoughtfully/overlooked, delightful, frightened. Perhaps poems are in order in a following iteration?




```
82
83  def analyze_text():
84      """ Uses above functions to compare two texts: returns dictionaries of
85          the most commonly occuring words, as well as the longest words in
86          common.
87      """
88
89      text1 = create_dict(reloaded_copy_of_phoenix_texts[1700:len(reloaded_copy
90      text2 = create_dict(reloaded_copy_of_nature_texts[6222:len(reloaded_copy_c
91
92      common_words_text1 = most_common(text1, 7, 10)
93      common_words_text2 = most_common(text2, 7, 10)
94
95      shared=shared_words(text1, text2)
96
97      print 'Common words in text 1: ' +str(common_words_text1)
98      print 'Common words in text 2: ' + str(common_words_text2)
99      print 'Long shared words: '+ str(shared[0:])
100
101
102 analyze_text()
```

Line 70, Column 23 Spaces: 4

Reflection:

From a process point of view, I was able to do an incredible amount of this project in not very much time. I definitely should have tried to start earlier and ask for help sooner. This project was appropriately scoped for the amount of time I had to do it, but I probably could have learned more if I had budgeted more time for it amongst my other classes. I'm still a bit confused about pulling text from non-plaintext sources (since i didn't cover that in the scope of this project) and it would have been cool to do some graphics analysis with my word output.