Jeff Lee
1/16/17
Final Programming Project

For the final part of this project, I added a webCrawler method that would analyze the links in the initial webpage, being sure not to repeat pages or go past a given distance from the initial page. I also added several new stemming exceptions for more accurate stemming, including adding an ‘e’ to the end of words that end in ‘v’, and reversing stem changes that can happen with suffixes (i.e. carried stems to carry). Finally, I cleaned up style and hardcoded more exceptions that are simply a part of the language.

1. I am most proud of my webCrawler function, as it employs recursion to create a list of words for a given depth of pages without repeating. This could definitely be used in another circumstance that a site needs to be crawled without repeating, possibly to find the lowest price on an item for sale. Another function that works well is the wordSorter function. I decided against using the given sort() function, and found that mine successfully sorts two lists based on the second list in a tuple. There are numerous applications where this could be useful. Finally, my removeBoringWords function is simple yet effective. It is very versatile, and when collecting data it is important to be able to get rid of information that is definitely not needed to maximize efficiency.

2. My program is fully working, however I have found that when I analyze longer text pages such as Moby Dick, the console will simply not show results. No error is shown, and perhaps data will be produced if given enough time, but in the modern day of computers it takes far to long to produce a list of the most occurring words. I feel as though my sophisticated stemming and sorting may slow the program down to a fault.

3. One improvement would be as I mentioned above, a faster program. I think that my methods may be a little slow in that they are too particular and use too many loops, however I am not sure how to analyze the lists otherwise. Another feature I would like to add is the option to allow the user to choose the amount of words they would like to see, and the depth they would like to reach. This would be fairly easy to implement, and would make the program more customizable from IDLE. Finally, I would like to add a dictionary import to remove non-real words. Now, my restraints for non-real words are one letter words (a and I are already removed as boring) and words with no vowels. However, other non-real words will leak through this filter. It is not possible to prevent this without a dictionary to verify the legitimacy of words.