Program 5: Word Frequencies (20 pts)

1. Use a text editor to create a text file (ex: myPaper.txt)

2. Write a Python program that

* asks the user to choose the name of the text file (please use a file open dialog from tkinter)
* reads the data from the text file
* creates a dictionary called **wordDIct** that maps words to frequencies (# of times word occurs).
  + keys are the words, and the values are the frequencies.
  + The first time a word is seen, the frequency is 1.
  + Each time the word is seen again, increment the frequency for that word.
  + Output should be an ALPHABETIZED output table, listing the word and the number of occurrences.
  + Alphabetize the frequency table--extract just the keys and sort them. This sorted sequence of keys can be used to extract the counts from the dict.
* creates a second dictionary **puncNumDict** that maps punctuation and numbers to frequencies
  + keys are the punctuation symbols and numbers; values are the frequencies
  + The first time a punctuation symbol or number is seen, the frequency is 1. Each time it is seen again, increment the frequency.
  + Output should list each punctuation symbol and number that appeared in the text along with the number of times it occurred.

3. You MUST modularize your code – carefully design the program so that it uses functions for each task.

* Be sure to modularize your code, using functions appropriately.
* Your main program should be clean and easy to read, calling the functions with meaningful names.
* Pass parameters – do not use global variables. Keep everything as local as possible!

4. Select your identifier names carefully so that they clearly indicate the purpose.

5. Other considerations:

* The input sequence can be words, numbers or any other immutable Python object, suitable for a dict key.
* Ignore case – apple is the same as Apple is the same as APPLE, etc.
* NOTE: you are REQUIRED to use a dictionary to store the information for this assignment.
* Make sure that your output (the tables) are formatted so that the columns are aligned, and so that there are titles for each column.

Example text file:  
The number 14 is a lucky NumbeR (for some people). So what!  
 Another number, 13, is not a lucky number (for some LUCKY people)!

Dictionary: wordDict  
Word Frequency  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

a 2  
another 1  
for 2  
is 2  
not 1  
number 4  
people 2  
so 1  
some 2  
the 1  
what 1  
  
Dictionary: punctNumDict

Symbol/Number Frequency  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

( 2  
) 2  
. 1  
! 2  
, 2’

BONUS (10 pts) :

* Create a menu-driven front-end to this application.   
   After initially populating the dictionary from the file, provide a menu that allows the user to
  + add words to the wordDict (or cause the count to increase if the word is already there), check if a word is already in the dictionary, delete a word from the dictionary, and print the dictionary entries in table form as described above.
  + Delete an entry from **wordDict** by specifying the key (word)
  + Add punctuation or numbers to the punctNumDict.
  + Delete an entry from punctNumDict by specifiying the key(punctuation symbol or number)
  + Quit
* Allow the user to continue using the menu until they choose to quit.