About Lab 6

In this lab you will write a program that asks the user for the name of a file, and then prints a concordance for that file.

A concordance is a kind of index -- it lists each word in the file and gives the location (in our case the line number) of each use of that word in the file.

Let's see what you remember from the prelab:

Which is the correct concordance for the following file?

This is a test,

This is only a test.

A

a 1 2
is 1 2
only 2
test. 2
test, 1
This 1 2

B

a 1 2
is 1 2
only 2
test 1 2
this 1 2

C

a 1 3
is 1 3
only 3
test 1 3
this 1 3

D

a 1 3
is 1 3
only 3
test 1 3
This 1 3

We will store the concordance in a Python dictionary -- with words of the file for keys, and lists of line numbers as values.

Suppose C is such a dictionary. Which is the correct code for adding the fact that the word "bob" appears on line 23?

A

```
C["bob"] = 23
```

B

```
C["bob"] =[23]
```

C

```
C["bob"].append(23)
```

D

None of these

Here is an outline of the program:

A. Get the name of the text file.
B. Create the concordance as a dictionary.
C. Open the file and use a for-loop to read it line-by-line
D. You need to get rid of the end-of-line marker. If *line* is the variable holding your line, say

```
line = line.strip( )
```

E. If *line* is not the empty string, increment your line counter.
F. Break the line into a list of words. line.split( ) does this.

G. For each word on the line
    - Remove any punctuation from the word.
    - If the word isn't an empty string, translate it to lower case and add it to the concordance along with its line number.

H. After all of the words have been added, have a loop that prints all of the words and their line numbers, with the words in alphabetical order.

I. At the end print the number of lines in the file and the number of different words you found.

You are welcome to organize this any way you can make it work, but here are the functions I used:

main(): This opens the file, creates the dictonary, and has the primary loop that reads lines, breaks them into words and adds the words to the dictionary

AddWord(w, n, C): This adds the fact that word *w* appeared on line *n* to the dictionary *C*. This is called by main( ).

RemovePunctuation(word): This takes a word, removes the punctuation from its beginning and end, and returns the result. Called from main( ).

PrintEntry(w, C): This prints one line of the output: word *w* and all of its line numbers from the C[w] list.

Here are a few helpful tricks

A. Reading a file.

Suppose the name of our file is "bob.txt". Here is how we open and read it:

```
F = open( "bob.txt", "r")
for line in F:
        line = line.strip()
        # now do stuff with the line
```

B. Dividing a string into words.

Suppose string s is "It was a dark and stormy night."
Then s.split( ) is the list

["It", "was", "a", "dark", "and", "stormy", "night."]

Of course, you can process this list with a for-loop:

```
for word in s.split():
        # do stuff with word
```

C. Removing punctuation

Python is your friend here. The strip( ) method for strings is exactly what you need. If you give it an argument string it will remove all of the characters of this string, in any order, from either end of the word. For example, if *word* is "!#?!bob!!" and *punc* is "#?!" then word.strip(punc) is just "bob". To write the RemovePunctuation(w) function, create a variable *punc* that is a string with every punctuation symbol you can find, then return

```
w.strip(punc)
```

D. **To add a (word, line) combination to the concordance**

Before you can append the line number onto the word's list, you have to determine if the word is already in the concordance. You need code such as

```
if word in C.keys():
        # append the line onto C[word]
else:
        C[word] = <a new list containing
                                the line number>
```

E. **To print the concordance so the words are in alphabetical order**

You need to get a list of the keys:

```
keyList = list( C.keys() )
keyList.sort()
for word in keyList:
    # etc.
```