CSE 231 Fall 2008

Programming Project #5

Assignment Overview

This project focuses again on strings as well as introducing lists. It also introduces file input. It is worth 40 points (4% of your overall grade). It is due Monday, Sept 29th before midnight.

The Problem

Many of the words in a variety of languages involve roots, prefixes and suffixes from Latin. In this project we will get a feeling for how common some of these word parts are in our language.

For an overview of what roots, prefixes and suffixes are, see:

- http://en.wikipedia.org/wiki/Prefix
- http://en.wikipedia.org/wiki/Suffix
- http://en.wikipedia.org/wiki/Root_%28linguistics%29

For example, the word ‘prescribable’ can be broken down as

- prefix: pre, meaning ‘before’
- root: scrib, meaning ‘to write’
- suffix: able, which forms adjectives and means ‘capable or worthy of’

You are going to write a program that reads from a word-list file (a file of words) and tries to counts how often a handful of common Latin word parts show up in those words. The Latin word parts you will be looking for will be read from a word-part-file.

Program Specifications

1. Your program will open the word-part-file rootsPrefixesSuffixes.txt. You will need to ignore commented lines (i.e. those that begin with a ‘#’) and blank lines. You will need to extract the word part (the first word on each line) in each section (roots, prefixes and suffixes) and store them.

2. Search through all of the words in the word list file and keep a tally of the number of times each of the roots, prefixes and suffixes were encountered. Recall that a prefix must begin the word, a root can be anywhere in the word, and a suffix must finish the word. For example, the prefix ‘co’ is not found in the word ‘incorporated.’ Your program must work for any properly formatted (i.e. one word per line) word list. For your convenience, we have provided three files for you to work with: shortWordList.txt, mediumWordList.txt, and longWordList.txt. NOTE: When you turn in your code, it should be using **longWordList.txt**. The file to be used is encoded in the program (i.e. you do not prompt the user for which word-file to use).

For the purposes of this experiment, if a word starts with a prefix or ends with a suffix, we will consider that prefix or suffix present. This is a rough approximation, however. In reality, just because a word starts with a prefix does not mean it uses that prefix. For example, coordinate uses the prefix "co" but contradict does not.

3. Output:
a. Your target output can be found in the following files: shortWordListSolution.txt, mediumWordListSolution.txt, and LongWordListSolution.txt. Obviously, you need to arrive at these results by searching through the word-list file. Programs that simply output the correct results because they have been typed are considered cheating and will receive a zero or worse.
b. Your output must follow what is in the solution **exactly**. You can check your progress by running the gradeIt.py file. Note: to use gradeIt you need to have the proper solution file, your project file (called proj05.py), and the gradeIt.py file all in the same directory.

Deliverables
proj05.py -- your source code solution (remember to include your section, the date, project number and comments).

1. Please be sure to use the specified file name, i.e. “proj05.py”
2. Save a copy of your file in your CSE account disk space (H drive on CSE computers).
3. You will electronically submit a copy of the file using the "handin" program: http://www.cse.msu.edu/handin/webclient

Assignment Notes:
There are a couple of problems here. Think about each one before you start to program.

1. How can I input text from a file? We haven’t done file I/O yet, but here is the short version.

   First, use the open function. It takes a single string argument, the name of the file you want to open, and returns a file descriptor. Make sure the file you want to open is in the same directory as the program you are running.
   Once you have the file descriptor, you can iterate over it using a `for` statement.
   Each iteration through the for loop gets another line from the file
   After you are done, use the close method to close the file.
   ```
   fd = open(‘wordList.txt’)
   for ln in fd:
       print ln  # prints each line of the file
   fd.close()
   ```

2. How can I input roots, prefixes and suffixes and store them such that I know which category (e.g. root) they are in? Hint: each type is under a commented line (e.g. ‘#Roots’) that bears the category name.
3. How can I keep a counter associated with each word part? Additional hint: you may have lists of lists.
4. How can I determine if a word starts with, contains, or ends with a string of letters?

```
word = ‘prescribable’
word.startswith(‘pre’) # returns true
word.startswith(‘foo’) # returns false
word.endswith(‘bable’) # returns true
‘scri’ in word         # returns true
```

5. How do I make sure that all words are lowercase, so that I don’t miss the fact that ‘Babelfish’ is the same word as ‘bablefish’?

```
word = ‘Babelfish’
word.lower() # returns ‘babelfish’
```

6. How can I output in ordered columns? Basically, we want to set up text boxes of a certain width and then put strings into them. You can do this like so:

```
string1 = ‘foo’
string2 = 42
print "%-6s %10i"%(string1, string2)
 #1st text box is left justified (because of the negative
 sign), 6 characters wide, and expects a string (‘s’)
 #2nd text box is right justified (no negative sign), 10
 characters wide, and expects an int (‘i’)
```

It is usually a good idea to start small (e.g. with the smallWordList) and work your way up. That way, while debugging, each test is faster and more manageable.

Getting Started

1. Do the normal startup stuff (create proj05.py, back it up to your H: drive, etc)
2. Write a high-level outline of what you want to do. Think before you code.
3. Start by inputting the word parts.
4. Make sure you can read the word-list file.
5. See if you can find a single word part in the shortWordList.txt file. Then search for all word parts.
6. Massage your output to match the target output (use gradeIt to check).

7. Try it on bigger wordlist files.

Remember to turn in code that is searching on the longWordList.txt file!

Your goal is go have gradeIt say the following for each word-list file

```
CONGRATULATIONS!!! Your code is producing the right output. Now make
sure your coding style is up to snuff!
```