CSE 231 Spring 2008

Programming Project #5

Assignment Overview

This project focuses again on strings as well as introducing lists. It also introduces a little file input/output. It is worth 40 points (4% of your overall grade). It is due Monday, February 18th before midnight. Notice, this is a *two week project*.

The Problem

An anagram, according to the wikipedia, is:

> a type of word play, the result of rearranging the letters of a word or phrase to produce a new word or phrase, using all the original letters exactly once (http://en.wikipedia.org/wiki/Anagram)

For example, the words canter, nectar, recant and trance are all anagrams of each other. That is, exactly the same letters are in each word but in a different order, and each ordering of the letters is a word in the English language.

You are going to write a program that reads from a word-list file (a file of words) and tries to find all the anagrams of that word. Your goal is to find the largest list of words that are anagrams of each other from a provided word-list.

Program Specifications

1. Your program will prompt for the name of a file from which you can read the words. The word-list is formatted to have one word on each line.
2. For each word read, find all anagrams (some words have more than one, as the above example showed) of that word.
3. Output:
    a. counts of the pairs, triples, etc up to but not including the longest anagrams.
    b. output the list of words that form the most anagrams(all of them if there are multiples).
4. An example proj05.pyc and test.py file are provided for testing, as previously

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 CS account disk space (H drive on CS computers).
3. Electronically submit a copy of the file.

Assignment Notes:

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

1. How can I easily determine if two words are an anagram of each other? That is, what simple process can I apply to the words and, as a result, know if those words are

anagrams? Note that the only characteristic that matters is what letters are in each word, irrespective of order. Is there an arrangement of the letters that would make the (Also, see the hint in note #4 below!)

2. Once I have such a process, how can I apply the process to all the words in the word-list and then find all the words with the same anagram?
3. We haven't done file I/O yet, but here is the short version.
    a. 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.
    b. 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
    c. 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()
```

4. It will be useful to convert a string to a list and a list to a string. Here are some idioms for that:

```
myStr = 'abcd'
myList = list(myStr)     # produces list ['a','b','c','d']
myList.append('z')
newStr = ''.join(myList) # produces string 'abcdz'
```

The string method `join` takes each element from the argument (in this case, `myList`) and joins them together using the calling string. If the calling string is the empty string (which it is in this case), it just makes the string. Calling `join` with a different string, such as ':' puts that character between each element

```
':'.join(myList)    # produces 'a:b:c:d:z'
```

So here is the big clue! Ready? If I take a word (a string) and turn that word into a list, what operation can I apply with that list that I can't to the word as a string, and which would be most helpful for anagram representation? Think!

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. Solve the anagram part first. We provide a short word-list for testing a big word-list (from http://wordlist.sourceforge.net/12dicts-readme.html the 3esl word-list, 21,877 words) and the standard Unix dictionary. The unix dictionary includes plurals and will produce longer lists of anagrams.
4. Apply the process first to the small then the large word-list
5. Find the largest anagram (the anagram that works for the largest number of words).