CSE 231 Spring 2013

COMPUTER PROJECT #7

Assignment Overview

This program focuses on the use of strings, lists, files and their various operators. It also incorporates a number of things we have learned about loops, if statements and functions.

This assignment is worth 50 points (5% of course grade), and must be submitted before 11:59 PM on Monday, March 11th .

Background

An Internet meme made the rounds in the early 2000's concerning some quirks about how human beings read a sentence. Below was the paragraph that was used in the meme. Can you read it? Can you tell what is "wrong" with the words?

```
Aoccdrnig to rscheearch at an Elingsh uinervtisy, it deosn’t
mttaer in waht oredr the ltteers in a wrod are, the olny iprmoetnt
tihng is taht the frist and lsat ltteer is at the rghit pclae. The
rset can be a toatl mses and you can sitll raed it wouthit a
porbelm. Tihs is bcuseae we do not raed ervey lteter by itslef but
the wrod as a wlohe.
```

If you keep the first letter and the last letter of a word in their correct positions, then scramble the letters in between, the claim was that people could still read the words. It was, however, not a legitimate research claim as the meme might lead you to believe. In fact, the process of scrambling as indicated above does not work for words that have a longer length. The web site http://www.balancedreading.com/cambridge.html shows some counter examples (as well as debunking the research claims) for this kind of reading comprehension.

We are going to imitate the scrambling process on some files of English text: some with long words and some without, to test the claim.

Program overview:

You will write a program that:

- prompts for the name of a file of English text to read in and a file to write out your results
    - constant prompting required for the file to be read.
- reads each line of the file
- performs the scrambling operation on each word of the line, preserving the non-word features (see details below)
- outputs the scrambled line to the indicated output file.

Program specifications

- `scramble_word(word_str)` returns `string`. This function takes in a single word from the file and returns a scrambled word according to the following rules:
    - The first and last letter are preserved in the original positions
    - Seems like a waste to shuffle something of length 3 or less
    - Furthermore, if there are punctuation marks associated with the word, then those positions are preserved as well. For example, a scrambling of the word "finished." could be "fnsieihd." but not "fsndi.hd". The period would have to be in the same place. Same for commas and other punctuation marks. Look at the examples
- `scramble_line(line_str)` returns `string`. This function takes in an entire line of text and returns a new line with each word scrambled using the above function `scramble_word`.
- `open_read_file(file_name_str)` returns `file_object`. This function takes a file string, the name of a file to be opened for reading. It contiguously prompts until it can open a file. It then returns the opened `file_obj` .
- main():
    - prompts for a file to write
    - prompts for a file to read, calls `open_read_file` with that string.
    - using the `file_obj` returned from `open_read_file` :
        - for each line in the file:
            - gets a scrambled line by calling `scramble_line`.
            - writes the scrambled line to the output `file_obj`.

Deliverables:

`proj07.py` -- your source code solution

1. Please be sure to use the specified file name, and to submit your file for grading via the "handin" program (http://www.cse.msu.edu/handin/webclient).

2. Save your file to your CSE disk (H drive on CSE computers) -- archiving files on the CSE disk is the only evidence that you have completed the project, if something has gone wrong. Get a TA to help you do this, if you do not know how.

Assignment Notes:

0. Be smart!!! Implement the first function `scramble_word` and test it on individual words. Then write `scramble_line` and test it with a single line of text. Remember, it uses `scramble_word` to accomplish its goal. Do the `main` function last, as this ties everything together.
1. We need a way to scramble some letters. Conveniently there is a method in the `random` module called `shuffle`. It takes a list as an argument and shuffles (re-arranges) the positions of the elements of the list ***in-place***. It doesn't have a return value. Here are some examples:

```
import random
my_list = [1,2,3,4,5]
```

```
random.shuffle(my_list)
print(my_list) → [2, 4, 3, 1, 5]
```

2. The shuffle method only works on a mutable, so you cannot shuffle a string. Thus we have to move back and forth between a string and a list. Here are some relevant methods.
    a. you can convert a string to a list, using the `list` function. This function requires an iterable argument and adds each element of the iterable (each character of a string) as an element to a list

    ```
    my_str = "hi mom"
    my_list = list(my_str)
    print(my_list) → ['h', 'i', ' ', 'm', 'o', 'm']
    ```

    b. the `join` method is called on a string and takes a list of letters (or strings) as an argument. It returns the string produced by inserting a copy of the calling string between elements of the argument list and concatenating them all together. For example:

    ```
    my_list = ['h', 'i', ' ', 'm', 'o', 'm']
    my_str = ','.join(my_list) # use comma to join
    my_str → 'h,i, ,m,o,m'
    my_str = ''.join(my_list)  # use empty str to join
    print(my_str) → 'hi mom'
    ```

    Notice that the calling string is not added at the end of a join. That can be very useful. Look at listStr.py in the project directory