Paragraph Scramble:

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

Background:

Read the following paragraph as quickly as you can and see if you encounter any difficulties.

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.

This was published as an example of a principle of human reading comprehension. If you keep the first letter and the last letter of a word in their correct positions, then scramble the letters in between, the word is still quite readable in the context of an accompanying paragraph. However, it seems that this is a bit of a myth and not truly based on solid research. See http://www.balancedreading.com/cambridge.html for some more details.

In short, for longer words the task is much more difficult. Nonetheless, we are going to imitate the process on some English text.

Program specifications:

Your program will reproduce this behavior on a file of text.

Some requirements:

1. You must use at least one function in your code. That function must be an integral part
   of the code's processing (not just a throw away)
2. Handling punctuation is tricky. You are required to deal with punctuation that comes at the end of a word (period, question mark, exclamation etc.). It is optional, and worth 5 points of extra credit, to deal with all punctuation (apostrophe for possessives, dashes in the middle of a word).
3. You must take care not to shuffle the first and last letters of each word. Furthermore, punctuation at the end or beginning of a word must be preserved.

The code outline is:

1. Open a file for reading. Prompt the user for the file name (no error checking)
2. Read that file, a line at a time.
3. Use the string split method to extract a list of words, all the words in a line
4. Process each word such that

    a. Each word maintains its first and last letter in the correct position
    b. For every word larger than three letters long, the middle letters (between the first and last letter) should be randomly scrambled (re-ordered).
    c. Capitalization and punctuation must be preserved.
5. Output the revised text file to a new file. Prompt the user for the new file name.

Turn-In: mixup.py -- your source code solution

There is a convenient way to scramble the letters of a word, but it involves some conversion work. The relevant steps are:

a. convert the string to a list, using the list function.
b. in the random package use the shuffle function to scramble the list.
c. use the join method to recreated the list as a string.