CS 101
Fall 2013
Program 4
Algorithm due Sunday, October 6
Program due Sunday, October 13

A recent study claims that English text is still readable, with almost no loss of comprehension, even if the letters of a word are scrambled, as long as the first and last letters of each word are kept in the same position, along with sentence punctuation and any apostrophes in the word.

For example, the above sentence, scrambled while keeping the first and last letters in place, may look like this:
A recent sudty calmis that Ensgilh text is stlil rlbaeade, with amsolt no lsos of cpenoiromsehn, even if the leertts of a wrod are sacmblred, as long as the fsirt and lsat lrttees of ecah wrod are kpet in the smae pistooin, anlog with sncenete pottcuiuann and any asoherppots in the word.

(Note: Later research didn't really bear out this idea; comprehension is much harder for longer words, and this really only works well for short, simple texts. Nonetheless, we're going to write a program to produce such scrambled text.)

You **must** use at least 2 functions in your program, and they **must** be integral to the program's functioning; that is, they must do something important to the program, not a throwaway tacked on at the end.

Handling punctuation is tricky; you're required to deal with punctuation at the end of a word (periods, commas, quotation marks, etc). For **5 points extra credit**, you can also deal with internal punctuation (apostrophes in contractions, dashes in the middle of a word, etc.).

Program specification:

- Ask the user for a file name, and open the file for reading; read the file into a list of strings (see below for instructions on how to do this).
- Use the string split() method to break each line into a list of words.
- Process each word such that each word maintains its first and last letter in place; for every word longer than 3 letters, everything other than the first and last letter should be randomly scrambled. Capitalization and punctuation must be preserved.
- Print output to the screen.

Development Notes:
We'll deal with file handling in much more detail later, but for right now, here's what you need to get started:

```
InputFile = open('input.txt')  # or open(filename) where filename is a string w/ name of file
Lines = InputFile.readlines()
InputFile.close()
```

This reads the entire file into a list of strings, where each line in the file (everything up to a \n character) is a string. You can then iterate through this list to process one line at a time.

Shuffling the characters in a string is straightforward, but involves some conversions:

- The list() method can be used to get a list with the characters of a string:

    - `list('abc') == ['a', 'b', 'c']`
- In the random module, there's a function called shuffle() that shuffles a list in place.
- The .join() string method can then be used to reassemble the shuffled list back into a string.

Be careful not to shuffle the first and last character in a string; punctuation at the beginning or end of a string must be preserved as well.

I provide you with a short text file you can use for testing. Note that because words are being randomly shuffled, it is very unlikely your output will match this exactly, or that your output will be exactly the same from one run to the next.

```
>>> ================================ RESTART ================================
>>>
What's the name of the file? holmes.txt
Poecuseritn for the exiessrpon of opinoins smees to me pcrletefy lcogial. If you have no
dobut of yuor pieremss or your power, and wnat a ctaeirn result with all yuor haert, you
nrtalluay exsreps your weshis in law, and sewep aawy all otiooipspn. To allow opoitosipn
by spceeh seems to itiadcne taht you tihnk the seceph imeotpnt, as wehn a man syas taht he
has suqared the cilcre, or that you do not crae wotahdeerelhly for the rleust, or that you
dobut ehietr your pewor or yuor pireemss.

But when men hvae reaelzid taht time has upest many fgiinthg fihtas, they may cmoe to
beeivle eevn mroe tahn they bivleee the very fnuaoditons of teihr own cncduot that the
utmaltie good dseeird is betetr rechead by fere tarde in iaeds -- that the bset test of
trtuh is the peowr of the thhogut to get itlsef aetepccd in the ciomeittpon of the mkaert,
and taht ttruh is the olny gurond uopn wihch tiher wieshs sleafy can be carierd out.

Taht, at any rtae, is the thorey of our Ctositionutn. It is an eeenixrpmt, as all life is
an epixnermet. Eevry yaer, if not eervy day, we have to wegar our svtoalain uopn some
phrocpey besad upon iecrfmpet kdgnweloe. Wilhe that eiprmexent is part of our sytesm, I
tinhk taht we shluod be enaelrlty vliagnit asgniat amttepts to cchek the erpxiesosn of
oniinpos taht we loahte and beviele to be fghaurt wtih daeth, uselns they so imennitmly
tteraehn immeatide irnfecnrteee with the lfuwal and prsiseng prpusoes of the law that an
iadimteme check is rirequed to svae the cutnroy.

-- Oevilr Wneeldl Hmloes, Armbas v. United Seatts, 1199
>>>
```