**Computer Science CS134C (Fall 2018)**
*Duane A. Bailey*
Laboratory 3
*Building a Python Toolbox (Thursday/Friday at noon)*

**Objective.** To construct a toolbox of functions for manipulating words.

This week we'll construct a small module of tools for manipulating words from word lists or dictionaries. When finished, we'll be able to answer some trivia questions.

**Motivation: The NPR Puzzle.**
Will Shortz is the editor of the New York Times Crossword and the Puzzlemaster at National Public Radio. Each Sunday morning he challenges listeners with a puzzle to solve by the following Thursday. Typically these are language-based challenges, but, as we'll see, their solutions can frequently be computed.
Here are some interesting problems:

**p1:** (Proposed February 11, 2018.) Name part of the human body in six letters. Add an 'r' and rearrange the result to name a part of the body in seven letters. What is it?

**p2:** (Proposed February 12, 2017 by Michael Shteyman of Odenton, Md.) Name some things commonly seen on a kitchen table. Switch the positions of the fifth and sixth letters of the word, and read the result backward. You'll name two things commonly seen in a kitchen. What are they?

**p3:** (Proposed September 24, 2017.) Think of a familiar 6-letter boy's name starting with a vowel. Change the first letter to a consonant to get another familiar boy's name. Then change the first letter to another consonant to get another familiar boy's name. What names are these?

**p4:** (Proposed April 2, 2017 by David Edelheit of Oyster Bay, N.Y.) Think of four 4-letter proper names that are all anagrams of each other. Two of them are first names–one male and one female. The other two are well-known geographical names. What names are these?

**p5:** (Proposed last Sunday, September 23, 2018 by Jim Levering of San Antonio) Think of an affliction in five letters. Shift each letter three spaces later in the alphabet—for example, 'a' would become 'd', 'b' would become 'e', etc. The result will be a prominent name in the Bible. Who is it? (If you solve this problem, you can submit the answer to `https://help.npr.org/customer/portal/emails/new?i=7`. You might possibly be called to compete on-the-air this Thursday!)

Are you up for solving one or more of these challenges?

**Getting Started.** Before you begin, you need to check out a repository that contains a few starter files. This is best done when you're sitting in your `cs134` folder:

```
cd ~/cs134
```

then, check out the repository (replace 22xyz9 with your CS username):

```
git clone ssh://22xyz9@davey.cs.williams.edu/~cs134/22xyz9/lab3.git lab3
```

When the repository has been checked out, you'll find this handout, some starter code, and some word lists. Now, you're ready to begin.

**Required Tasks.** Please complete these steps by your due date.

1. Build a small toolbox of string-related functions, `wordTools.py`:

   - ◇ Write a function, `words(wlfile)`, that returns a `list` of words found one per line in a file whose name is specified by `wlfile`. Strip off any unnecessary whitespace from the words as you read them in. You might use it this way:

     ```
     >>> len(words('wordlists/firstNames'))
     5166
     >>> words('wordlists/kitchenThings')[124]
     'oven'
     ```

   - ◇ Write a function, `sized(n,l)`, that takes a word list `l` and a word length, `n`. It returns the words in the list that are exactly length `n`. For example:

     ```
     >>> sized(6,words('wordlists/firstNames'))[0:3]
     ['Adelia', 'Adella', 'Adelle']
     ```

   - ◇ Write a function, `rev(s)`, that returns string `s`, reversed. For example:

     ```
     >>> rev('devil')
     'lived'
     ```

   - ◇ Write a function, `canon(s)`, that returns a lower-cased, space-free, and sorted rearrangement of the letters of `s`. For example:

     ```
     >>> canon('Lot A')
     'alot'
     >>> canon('aim')
     'aim'
     >>> canon('Mia')
     'aim'
     ```

   You may develop other useful functions, as well. If you do, collect them in `wordTools.py`.

2. Make sure your `wordTools` toolkit is a solidly built module:

   (a) The module starts with appropriate comments.

   (b) It define `__all__` to be a list of strings of the names that should be imported when you write

   ```
   from wordTools import *
   ```

   (c) There is a triple-quoted docstring that appears at the top of the file that helps the user understand the purpose of this module. You can check this out with: `pydoc3 wordTools`

   (d) Make sure that every function is documented with a docstring.

   (e) Thoroughly test each of the functions.

3. Now, solve at least one of the puzzles. For example: to solve puzzle 3, write a new python script, `p3.py` that prints either the solution directly or at least no more than a screen of possible solutions to be considered. The word lists found in the `wordlists` folder will be useful.

*Good luck!* Do not forget to add, commit, and push your work as it progresses!

⋆