Python In-class Programming Assignment
Command Line Arguments

Part 1

The Python programs we've written so far have centered on prompting the user for input, accepting it, processing it, and exiting. That's been effective for our purposes, but in the world of systems programming there's often a premium placed on speed and efficiency.

Let's begin with a discussion of command line arguments. Command line arguments are the names of the programs and data you type in a terminal. For example, when you type: "`cat myFile.txt`", you're running the Linux `cat` command on the file `myFile.txt`, and this will display the file to the screen. Both `cat` and `myFile.txt` are command line arguments.

Python (and other languages) provide a mechanism for your running program to determine what was typed on the command line when the program was run. In Python, everything that was typed will be available to your running program in a list called `argv`, which is short for *argument vector*. You access `argv` like this:

`import sys <-` must import the sys library to access `argv`
`print(sys.argv) <-` here's the syntax for accessing `argv`

Update your repo if necessary (using `git pull`) and copy all the files in the following directory to a location of your choosing: `~/repo201/classwork/cw32/`

Let's start with `argvExample.py`. There are some unique components to this code, so let's discuss them and run through some examples.

Part 2

Let's use the power of `argv`!

Say we wrote a program to prompt the user for the name of a text file. In that same program, we prompt the user for a letter of the alphabet. The program then scans through the file and counts the number of case-insensitive occurrences of the chosen letter, prints that result to the screen, then asks for another letter. It keeps doing this until an empty string (return by itself) is entered. Pretty simple, right? A sample run of such a program might look like this:

```
Enter file name: ulysses.txt
Enter letter: e
The letter e shows up: 143215 times.
Enter letter: r
The letter r shows up 71005 times.
Enter a letter: z
The letter z shows up 1076 times.
Enter a letter:
```

A systems programmer might say: "*I just want to run the program with this file name, and these letters, and get a quick answer.*" She might want to run it like this:

`./letterCount.py ulysses.txt e r z` `<-` This is what she types
`[143215, 71005, 1076]` `<-` This is what gets printed

Using the power of `argv`, write this program. It should take as arguments: the name of a text file and a series of one or more letters, separated by spaces. It will then scan through the file and count the number of case-insensitive occurrences of the chosen letters, saving those results to a list. When complete, it prints the list to the screen.

You may assume that the entered file exists; that valid letters are entered with no duplicates; that at least one letter is entered after the file name; and that letters are properly separated by spaces. You don't need to perform exception handling now, but you can add it later if you wish.

A compiled version of this program is available as:

`~/repo201/classwork/cw32/letterCount`