Lab #4

Physics 91SI Spring 2014

Objective: This lab will introduce you to Python's powerful **built-in data types**, particularly strings, lists, and dictionaries, and Python's **built-in functions**, which allow common operations on these types to be expressed in a single line.

In **Part 1**, you will implement some common statistics functions on a data (i.e. a list of numbers). You'll also get used to sorting lists.

Part 2 gives you practice working with dictionaries and strings, as well as string formatting.

In **Part 3**, you will write a few Python functions to perform common UNIX operations within the UNIX shell (using a pipe, your Python code will take `stdin` as its input).

For this lab and the rest of the quarter, make liberal use of Python's `help(`*`name`*`)` function, as well as the documentation at http://docs.python.org/library/.

As always, `cd` to your `physics91si` folder and clone the starter code for this lab with

```
hg clone /afs/ir.stanford.edu/class/physics91si/src/labN labN
```

Remember to `hg commit` often! When you're done, submit in the usual way:

```
hg push /afs/ir.stanford.edu/class/physics91si/submissions/yourname/labN
```

Part 1: Statistics

While the last lab showed you how to write programs in the "conventional" way, using procedures familiar from C and Java, now we're going to implement some similar methods the "Pythonic" way. In the stats.py file, you'll find some starter code and a bunch of empty functions:

- `loadcsv`
- `mean`
- `stdev`
- `median`
- `mode`

Your first task is to implement the `loadcsv()` function, which loads a "comma-separated-value" (.csv) file and returns a list of numbers. A csv file is just a text file with fields separated by commas and line breaks, and is a common format for instrument output and basic spreadsheet data - look at `sample.csv` to get an idea of what you're dealing with. In this case, the datafiles also have some comments, which are lines that start with the # character (just like in Python), and your function should `print` these out as it reads the file.

With that working, now implement the mean, stdev, median, and mode methods - see the docstrings (in `"""triple quotes"""`) for implementation details. While you can do each of these the "brute force" way, it's much more efficient to use Python's builtins, particularly `len()`,

`sum()`, and `sorted()`: use `help(function)` in the interpreter or check http://docs.python.org/library/functions.html for documentation. With these, you can write `mean` and `median` in a single line!
Mode is a bit harder — you might want to come back to this after working on Part 2 (if you do this, be sure to `commit`!).

With all these functions written, use the included main method to print out stats on each .csv file (`sample.csv` and `shots.csv`). While testing, you can comment out lines that you haven't yet implemented. Also, if you want to explore the data visually, uncomment the `histogram(data)` line.

Bonus: If you have time, implement the `find_files()` function based on the docstring in the starter code. You'll need to use the function `os.listdir` - read about it in the documentation. Then extend your main method so that it reads all the `.csv` files in the current directory and performs your analysis on each.

Part 2: Language

The file language.py contains the skeleton of a program for exploring language models, in the same vein as the CS107 spellcheck assignment. You'll need to implement the following functionality:

- `load_model`
- `spellcheck`
- `find_palindromes`

`load_model()` should take the name of a text file and parse it, generating a dictionary that contains all unique words and the frequency with which they appear. In addition to file reading, you'll need to implement some string processing to convert words to lowercase, remove punctuation, and skip words that have non-alphabetic characters in them (i.e. `"test123"` should be ignored, but `"Hello!"` will be registered as `"hello"`). Test this on the `.txt` files in the starter repo.

`spellcheck()` will check a word and print a message depending on whether it is in the dictionary you provide. This is little more than an if statement, but we encourage you to play with Python's string formatting (`%`) operator to make your output pretty. (If you have time, you can try to create an `editdistance()` function (Google "Levenshtein distance") and use it to check if a given word is a misspelling of a different word.)

Finally, `find_palindromes()` should search the dictionary for all palindromes and print the top 5, by frequency. A palindrome is a word that is the same backwards - for example, "bob" is a palindrome, but "palindrome" is not. Don't forget Python's builtins here! `sorted()` and array slicing (i.e. `a[:n]`) may be useful here. Also note that `sorted()` works on a list of tuples, sorting

by the first element.

You're free to put whatever you like in `main()`, and run your code as a module (`import language`) or from the command line.

You're done! The last part is optional, but fun - take a look at it if you have time. In the meantime, submit your code with the usual `hg push`.

Part 3 (Challenge): Command-Line Tools

Remember pipes? Well, it turns out that Python makes it very easy to write command-line utilities that work just like any other UNIX program. A pipe connects to the stdin of the program on the right, and you can access this stream from `sys.stdin` in Python. Starting with the skeleton code in `filter.py`, write a useful program that parses the output from some other UNIX utility and manipulates it in some way. Some ideas, in rough order of difficulty:

- Use the output of `ps -af` to find heavy CPU users, and print what program they are using
- Parse the output of `who` to find people you know (match a user list)
- Manipulate the output of `ls -alR` to find a file buried in a subdirectory, and extract the relative path

Once your program is set up to read stdin, you can use it as follows:

```
unixcommand -flags | python filter.py
or
python filter.py < inputfile
```

If you come up with something clever, `cp` it over to the `physics91si/submissions/dropbox` folder to share it with the class!