Lab 2

Learning Objectives

I. **What characters do I type next?** *How do I use Python data types and builtins efficiently?* This lab will provide an opportunity to practice constructing and modifying built-in data structures using both loops and built-in functions.

II. **How do I plan a project and execute my plan?** *How do I work in IPython?* This lab will provide the opportunity to practice using the interface and basic features of the IPython notebook.

IV. **How do I communicate science and Python with others?** *How do I work with starter code?* This lab will provide an opportunity to practice figuring out what simple, documented functions do, and to practice filling in the gaps in pre-written code.

A. **Python is a physical system. Experiment!** *If you don't know a bit of syntax, see if you can guess it.* This lab will continue providing an opportunity to practice one method for learning how to learn new syntax.

Today we'll leverage Python's powerful built-in data types, particularly strings, lists, and dicts, and Python's built-in functions, which allow common operations on these types to be expressed in a single line, towards the second major application of scientific computing: **data analysis**.

While You Work will give you structure and motivation to practice experimentation in Python.

In **part 1**, you will have the opportunity to practice loading data into a list. This will require manipulating strings and lists using builtins. You'll also get practice defining a function with optional arguments (keyword arguments).

Part 2 will ask you to analyze that data using a dict. You'll have the opportunity to implement some common statistics functions on data (i.e. a list of numbers), and to use Python builtins to write one-liners.

Part 3 is about using string formatting to create readable output.

Part 4, if you have time before the end of class, gives you an opportunity to explore some data structures and features that are tangential to the course but potentially useful.

The order in which you do this lab is pretty flexible. Take advantage of that to jump around, so that you can spend the most time on the things you find most valuable.

While You Work: Look for New Syntax

Any programming language has a lot of little syntactical details. We've covered the most important ones in lecture, but there will probably be times while you're working that you stray beyond the material in lecture. This is your opportunity to practice guessing that syntax.

If you figure out a bit of new syntax, share it with the rest of the class! Call over one of the instructors and we'll put it on the board.

Part 0: Get Set Up and Get the Starter Code

This quarter we'll be storing the starter code on GitHub and using it for lab submissions. If you haven't already, get a GitHub username at https://github.com and remember it. The instructors will need to know it too.

First you'll need to set up an SSH method, if you haven't already. Navigate to
http://web.stanford.edu/class/physics91SI/cgi-bin/?page_id=947,
open up the appropriate Getting Started file, and follow the instructions.

Once you're logged in, you're ready to get the starter code from GitHub. The following will make sense next week, when you learn how to use UNIX and the relevant command called Git to get the starter code and keep track of your work. But for now just copy the following commands verbatim. Type the following into the terminal (plugging in your GitHub username) and then press enter.

```
git clone https://github.com/physics91si/github-username-lab2.git lab2
```

(If you know how to make a new directory for this course, feel free to do that first. If not, we'll teach that next time.) Then navigate into the newly created directory (folder) by typing the following and then pressing enter.

```
cd lab2
```

Finally, open the IPython notebook containing today's lab by entering the following command and then pressing enter,

```
ipython notebook
```

waiting for the X-forwarded browser to open, and then opening `lab2`.

Part 1: Parsing Textual Data

Scientists don't only deal with numbers; they deal with words, too. Some scientists only care about words when they're writing papers, but for others the words are their primary interest. Either way, there's a lot that you can automate. In this part you'll have an opportunity to practice using Python's string operations to deal with text.

We've given you a sample text written in English. **Your task is to parse that sample into a list of English words.**

You should write this in the function `parse_sample`. This function should:

- accept two arguments:
  - the text to be parsed
  - two Boolean arguments that are optional, `words_only` and `sort_list`
- separate the sample at each whitespace character to get a list of words, in order of appearance (don't get rid of repeats)
- if `words_only` is `True`, get rid of any punctuation or capitalization of each word (e.g. `"test123"` should be left as is, but `"Test123!"` should be turned into `"test123"`)
- if `sort_list` is `True`, sort the list
- return the list of words

Change anything about `parse_sample` that you need to change to get it to work!

As you write code, you should check at convenient intervals that your function works. We have provided you with the function `load_sample`, which you should figure out how to use. When you're testing your functions, we suggest you create another cell and work inside it.

Finally, two hints for `parse_sample`:

- Sometimes the character `'-'` separates words, instead of whitespace. If you replace it with whitespace, it'll be easier to identify as a word break.
- The punctuation marks `. , : ; " ' ? !` are pretty much all you'll find.

Part 2: Analyzing a Data Set for Statistical Information

Now you can get all the individual words from the sample text. We can analyze this text numerically by examining how often each word appears. **Your task is to extract these word frequencies and compute some common statistics for this data set.**

In the part 2 cell, you'll find some starter code and a bunch of empty functions:

- `count_freq`
- `mean`
- `stdev`
- `median`

A docstring is provided for each of the functions. Figure out what they're supposed to do, and then implement them! Suggestion: do them in the order listed. Once you've written `count_freq`, try looking at the frequencies of some common words, like `'the'` and `'hamlet'`. (Try comparing `'honour'` and `'happiness'`.) **If there's a piece of terminology or math that you're not sure about, feel free to ask a classmate or an instructor.**

While in the last lab you probably wrote code conventionally, using procedures familiar from C and Java (like for loops), now you have the tools to implement some similar methods the Pythonic way. You can do each of these the "brute force" way, but it's much more efficient to use Python's builtins. With these, you can write some of these functions in a single line! A question to consider: what tasks can and can't you do with builtins?

As you work, make sure to save the notebook occasionally. You'll submit today's lab next Tuesday in class.

Part 3: Human-Readable Output

Now that you've calculated the statistics, your task is to print them nicely to the IPython notebook. Play around with string formatting and see what you can do!

Part 4: (If You Have Time) Extensions

Here are some useful tangents you can take:

- Create a function that returns the words in the sample, without repeats. Suggestion: use a set.
- Create a function to check if a phrase is a palindrome. A palindrome is a word or phrase that is the same backwards and forwards. For example, `"race car"` is a palindrome.
- Create a function that analyzes letter frequencies, instead of word frequencies.
- Create a function to find the mode of a list of numbers, or equivalently the most common word in a list of words.

Part 5: Submit Your Lab

We'll be using GitHub to submit labs as well. Just like part 0, this will make more sense next week, but for now just follow along verbatim.

1. Save your work in the IPython notebook
2. Close the tab containing the notebook
3. In the remaining tab, press "Shutdown" next to `lab2`
4. Close the browser window
5. In the terminal, enter the command `git add -A`
6. Then enter the command `git commit -m "I finished my lab!"`
7. Finally, enter the command `git push`

And you're done! You can check with an instructor to make sure it's all submitted.