Lab 4: Working Outside of Jupyter Notebook

This week we'll review some of the different places we can run our Python code and review some of the resources for practicing code.

So far we've been doing most of our projects inside Jupyter Notebook. You may remember in Lab 2 we learned about *three* modes for running code:

Notebook mode

We've done most of the class projects here. Great for interacting with data sets where you want to crunch some data, see the result, and then decide what to do next.

Interactive mode

We've not used this much outside lab 2 yet. It's very useful for quickly checking some lines of Python when you don't want to fire up a whole Jupyter notebook, you're working on a computer with Python but not Jupyter notebooks, or you don't have Internet for using Python Tutor.

Script mode

This mode allows us to type many lines of code into a single file and run it all at once anywhere where python is installed. It also makes it easy to include the code as a module in another project.

This week we're going to do the project in Script Mode! To make sure you don't have any snags when you're doing your project work, let's practice with some of the refactoring exercises covered in Wednesday's lecture.

Practice coding in idle

1. Make a refactor-3-original.py file and enter in the original version of **Refactor Exercise 3** from Wednesday's lecture.
2. Run refactor-3-original.py and see if the result is what you expect. Look for any semantic bugs and fix those.

(If you have trouble remembering how to do steps 1 or 2, review Lab 2.)

3. Make a refactor-3-A.py and run it.
4. Make a refactor-3-B.py and run it.
5. You'll notice that one of the two (refactor A or B) has a semantic error and doesn't follow the flow chart we saw in lecture 9 correctly. Open the buggy code up again and fix it. Save and run again.

Practice coding in Python Tutor

Python tutor not only has great features that let you visualize your code step by step, it let's you share your code with someone else who can chat with you. This can be great for working through something with your partner or studying for the midterm with classmates without making anyone go out into the ice and snow to meet.

Let's try it out!

1. Find **Refactor Exercise 5 - Refactor A** and click 'Edit this code' to open it in Python Tutor.
2. Click 'Start Private Chat' in the upper left hand corner.
3. Copy and paste the URL and email it to one of the people sitting near you in the lab (or, if you or your partner have a laptop in the room, to one of your laptops).
4. Use the chat window to discuss how to fix this refactor to match the behavior of the 'Refactor Exercise 5 - Original Version'

Practice getting the most out of the text book

Think Python is a great book, and it includes some excellent exercises to help cement the concepts in your mind. After reading the assigned chapters, you will find it very helpful to give the exercises at the end of the chapter a whirl. In fact, let's do one now!

1. Complete text book exercise 7-2.

(note: don't run `eval` on input you don't 100% trust! Details of what could go wrong can be read about here (https://nedbatchelder.com/blog/201206/eval_really_is_dangerous.html))

This exercise was chosen because it gives you practice with skills critical for the next project. You'll be glad you did this quick exercise and had a chance to ask the TAs questions before tackling the larger assignment!

New skills for the project

Finally, the project asks you to use two things we've not talked about in lecture or in the slides. They're pretty easy, but we want to give you a chance to try them out in lab first.

In the project, you will be asked to read input from the user and check it against another string for a match. As you have noticed, however.

```
"answer" != "Answer"
```

capitalization counts in string comparison! So, a quick way to solve this problem is to just change both strings to lower case. You can do this with a built in fuction called str.lower() :

```
"answer" == str.lower("Answer")
```

Since we're taking user input, they may also sometimes add spaces before or after their input, but that also will mess up the string comparision:

```
"answer" != "answer "
"answer" != " answer"
```

Use the str.strip() to strip off any spaces before or after the letters:

```
"answer" == str.strip("answer ")
"answer" == str.strip(" answer")
```

Both of these you can use with variables too:

```
case_answer = "AnSwER"
"answer" == str.lower(case_answer)

spaces_answer = " answer     "
"answer" == str.strip(spaces_answer)
```

And that's it! You're done with this lab. Go ahead and start the project, and good luck!