Laboratory Exercise #1

This Lab Exercise assumes:

1. You have read Chapter 1
2. The Pre-Lab Assignment has been completed before attending your lab session.

This assignment focuses on the mechanics of installing and using Python. Complete the steps given below to download and install Python. Finally, complete the exercise in step 7.

1. Get Python

Download and install Python 3.5 from https://www.continuum.io/downloads
(do **not** get Python 2).

2. Getting started with Python

Microsoft Windows

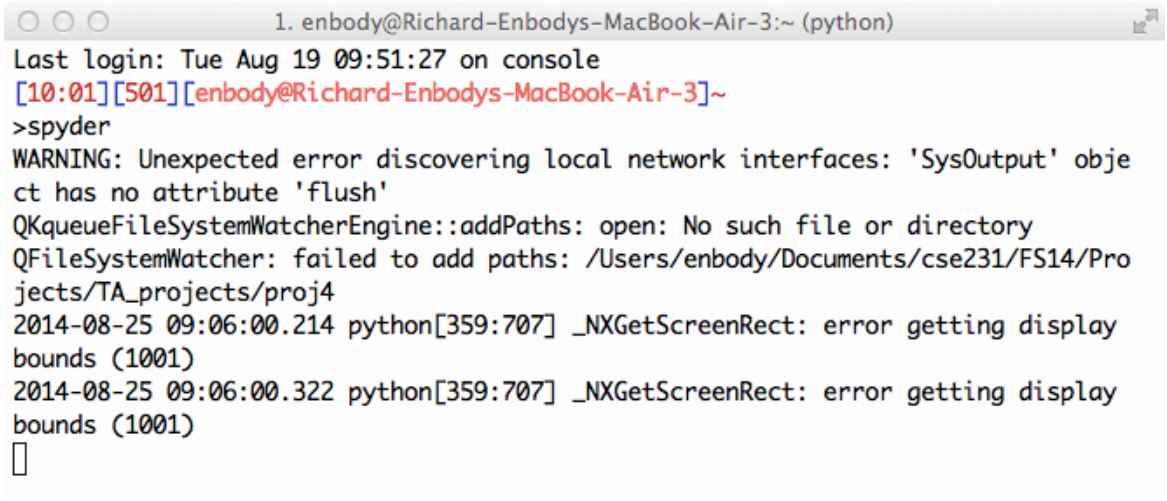
To start the Python combination under Microsoft Windows, go to Start Menu ==> All Programs ==> Anaconda ==> Spyder.

Then wait because it takes minutes to load while you look at this.

Mac (OS X)

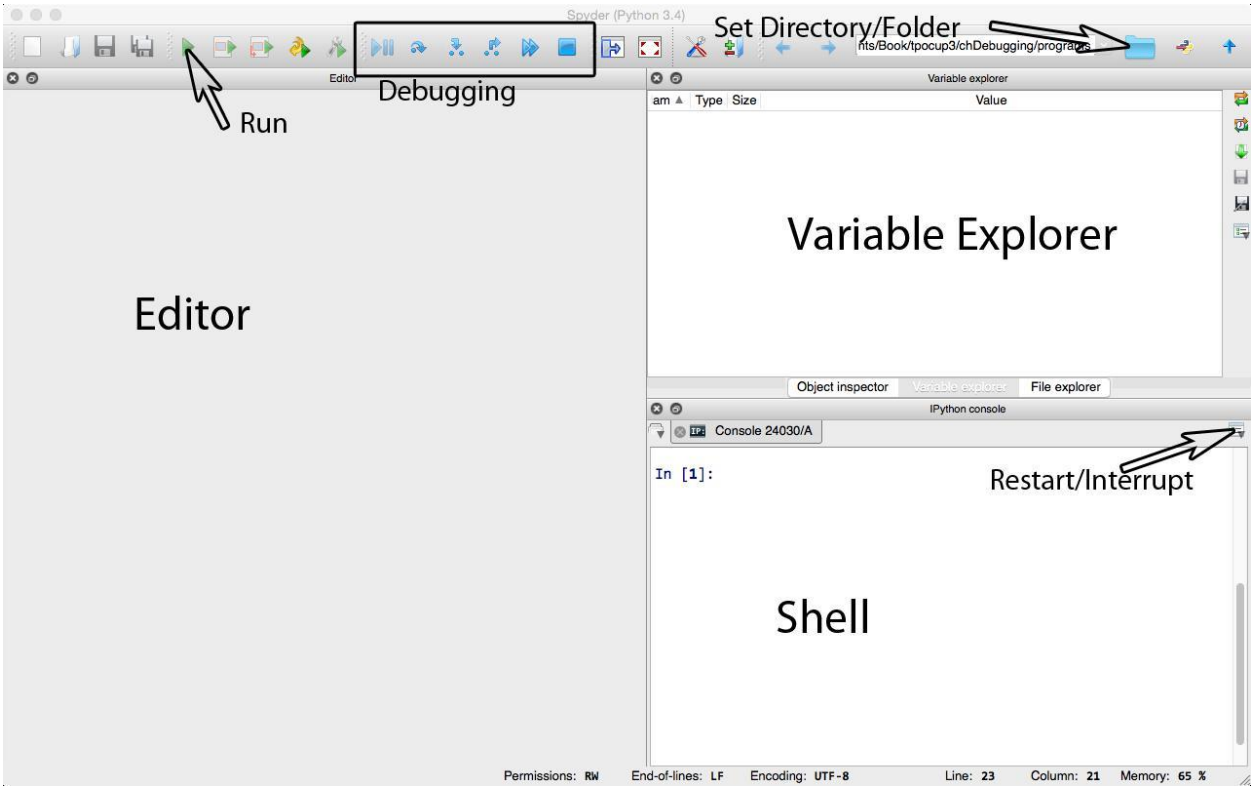
On a Mac (OS X), type command-space for the Mac Spotlight prompt and enter in spyder.

Or go to the Finder and Select Applications ==> Utilities ==> Terminal and at the prompt type **spyder** (some error messages may show as can be seen below, but that is fine.)



Then wait—it takes minutes to load while you look at this…

3. Using Spyder



There are three parts to the window:

- The left half is where you write your programs.
- The bottom right region is the *Python shell* where you can both enter code and where output from your program appears.
- The upper right region is where you can observe the values of variables that you create—essentially the current *namespace*, a term you will learn about soon (you have to click on Variable Explorer).

Finally, at the right end of the top bar is an *important* region where you can specify the folder/directory where your programs will be stored.

An important capability of Spyder is that it has a built-in debugger—something that means nothing to you now, but will be useful later.

4. Working with the shell

The window in the lower right corner is the Python shell (the shell in Spyder is the IPython shell). The shell is interactive: you can type Python commands in the shell and Python will execute them, generating a result.

In the Python shell (lower right window pane) try typing:

```
1 + 1 <Enter Key>

print( "Hi Mom" ) <Enter Key>
```

What output do you get? It should look like this:

```
In[1]: 1 + 1
Out[1]: 2

In[2]: print( "Hi Mom" )
Hi Mom

In[3]:
```

What you type shows up after the [1] prompt (the number between brackets increments each time). When you type something and hit the Enter key, the result shows up on the next line(s). Sometimes you can get some surprising results, and sometimes an error. For example, try entering:

```
1 + 1.2
```

or perhaps

```
print( hello )
```

The results would look like the following:

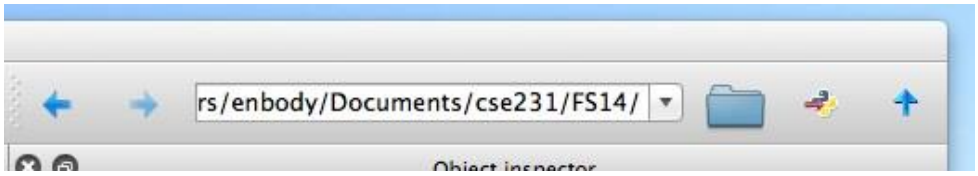
```
In[3]: 1 + 1.2
Out[3]: 2.2

In[4]: print( hello )
Traceback (most recent call last):
    File "<ipython-input-4-43a14fcd4265>", line 1, in <module>
        print( hello )

NameError: name 'hello' is not defined
```

The last lines show that an error occurred. Since *hello* was neither a variable nor had quotes around it, the print operation failed (we'll learn more about this later).

5. Developing a program

Typing into the shell is useful, but the commands that you write there are not saved as a file so they cannot be reused. We need to save our commands in a file so we can run the program again and again, and more importantly turn it in!

Before you create a file you need to select the folder/directory where you will be working, i.e. saving your program. In the top bar on the upper right is a one-line window indicating which folder/directory you will be working in.



Here I have specified that I will be working in the folder/directory named Documents/cse231/FS14 (I'm on a Mac, but Windows people will see something similar). The file folder allows you to browse for your desired folder/directory.

To open a file, select File ==> New file (or figure out the shortcut).

It is always a good idea to Save the file and name it "hello.py" (without quotes and all lowercase). You can do that by selecting File => Save or select the Disk-Save icon in the Spyder window. Always put a ".py" at the end of the filename and notice the "dot" (a.k.a. period) – it is very important. (Actually, Spyder will use .py as the file extension automatically if you forget, but it is still a good habit to put it there.) Hint: always check the directory/folder that you are saving into is the one you want to use because your operating system settings may default to somewhere unexpected.

The big window on the left is an editor window into which you can type your first program.

There is a tradition in computer science: the first program you create is the HelloWorld program. This program does nothing but display "Hello, World!" on the screen. It is a useful tradition because it does very little except focus on the mechanics of writing your first program and running it.

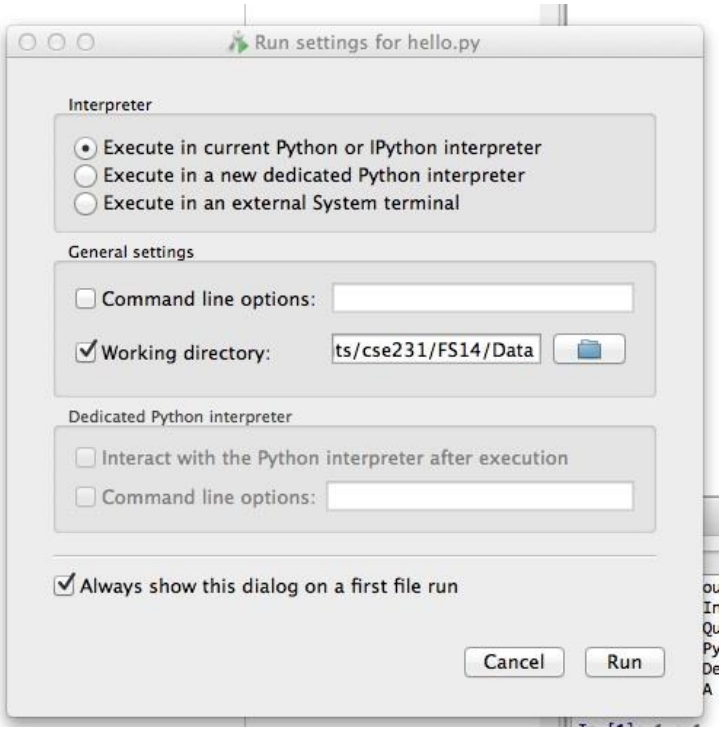
In Python, the HelloWorld program is easy. Type the following in the editing window (i.e. the big window on the left half of Spyder.) There may be some text already in the program (often in green) so you want to enter the following on the first blank line below that text:

```
print( "Hello, World!" )
```

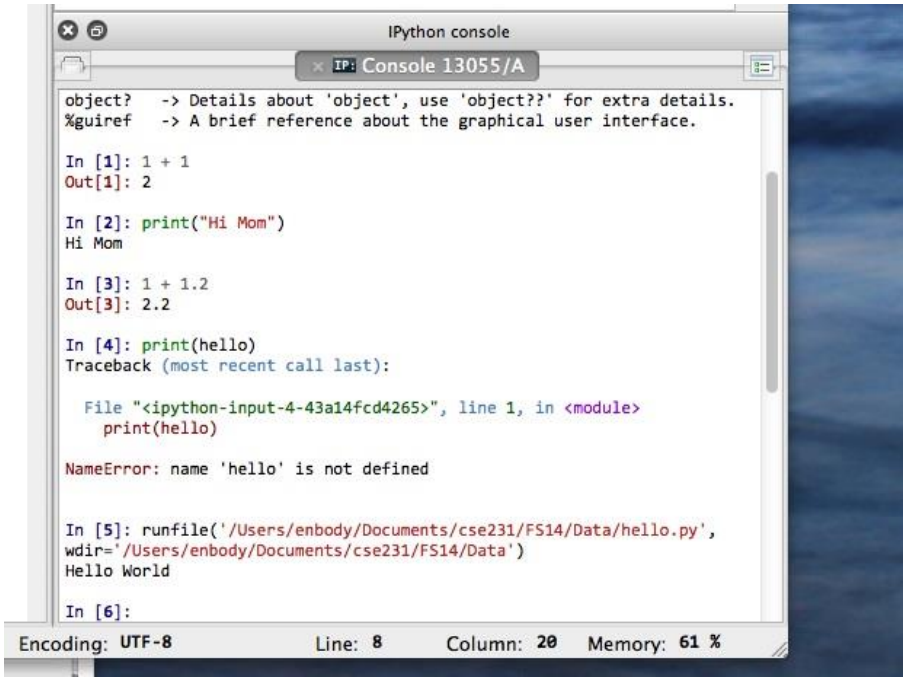
The phrase inside the parentheses should be in quotes (either single quotes or double quotes).

Save the program (as above).

You can then run your module to see what your program produces. To run the program, select Run ==> Run (or hit the big green arrow or hit the F5 function key on your keyboard). The first time you will get the following dialog box—simply select the default choice: Run.



The result is new output displayed in your Python shell (bottom right window), as shown below.



You can see “Hello World” printed near the bottom.

6. Running an Existing Program

Copy the following Python program to your computer:

http://www.cse.msu.edu/~cse231/Labs/Lab01/lab00.py

Put it in the same folder/directory that your HelloWorld program is (so you can find it easily).

In the Spyder window select menu File ==> Open. Find the program that you copied and open it.

As before, when you open a program the code of the example appears in the editing window. You can look at that code, and when ready you can run the code by selecting the editing window menu Run ==> Run.

Take a look at the output (in the Python shell) and compare output lines to lines in the program. After reading Chapter 1 you should be able to understand the program. Ask your TA or on Piazza about parts you don't understand.

(Note that after you run a program that is 'big enough' or 'complicated enough', a new file is created automatically. It will have the same name as your original file, but with the extension .pyc . For example, when you run numberInput.py, the file lab00.pyc is automatically created. It is a file that is the "compiled" version of your file. You can completely ignore that file—we are pointing it out only because you may notice it.)

7. Setting Up Your CSE Account

Before you can hand an assignment in you must set up your CSE account. Follow the instructions here:
http://www.cse.msu.edu/Facility/Account/FirstLogin.php

Important: your initial password is your PID (including the capital A). You must change it for a new, and better one. A good password is one that your best friend or sibling cannot guess.

If you added the class late, it takes about 24 hours for your account to appear.

8. Final exercise

Assignment Overview

This exercise provides practice with numeric computations in Python.

You will work with a partner on this exercise during your lab session. Two people should work at one computer. Occasionally switch the person who is typing. Talk to each other about what you are doing and why so that both of you understand each step.

Roots of a Quadratic Equation

Consider the quadratic equation:

```
A * x**2 + B * x + C = 0
```

where x is the unknown and A, B and C are constants (with A not equal to 0). A quadratic equation has two solutions (called roots), which may not be distinct values and which may not be real values.

The two roots of a quadratic equation may be calculated using the quadratic formula. See the brief article at Wolfram MathWorld if you don't recall the formula:

http://mathworld.wolfram.com/QuadraticFormula.html

Develop a program to compute the two roots of a quadratic equation.

1. Copy the file named "lab01.py" into your account.

2. Modify that program to compute the two roots of a quadratic equation, as described in the program comments. Note that the program does not perform any error checking, so the results displayed by the program may not be correct in all cases. For example, when A is zero, the equation is not quadratic (e.g. if you get a `ValueError: math domain error`, you have tried to take the square root of a negative value.)

3. Test your completed program using the following values:

```
A = 1, B = 0, C = -4

  Root #1 = _____________  (should be 2.0)

  Root #2 = _____________  (should be -2.0)

A = 1, B = 5, C = -36

  Root #1 = _____________

  Root #2 = _____________

A = 2, B = 7.5, C = 6

  Root #1 = _____________

  Root #2 = _____________

A = 0, B = 3.5, C = 8

  Root #1 = _____________

  Root #2 = _____________

A = 5, B = 0, C = 6.5

  Root #1 = _____________

  Root #2 = _____________
```

★ **Demonstrate your completed program to your TA. On-line students should submit the completed program (named "lab01.py") for grading via the CSE handin system.**