CISC106 Summer 2012 Lab03

- This lab an all subsequent labs will be due Sunday at 11:55 PM EDT on Sakai.
- The preparation problems below are to develop your understanding without creating extra work for you or the TA; these problems will not be graded. Be sure to read and understand them - they will help with the problems you must submit for grading
- You may work in pairs on your lab. If you do, **one** of you should be designated to submit the assignment on Sakai. **Both of your names** should appear on code that you develop together[1].
- Whom do you think deducts more points: a happy TA, or a frustrated TA? Make your work easy to read! It isn't just good software engineering, it is good for your grade!
- EVERY python program/function must include header, doc string that contains a human-readable desciption of what the function does, and must be followed by a good series of tests, as discussed in class. Always test boundaries. Do not test erroneous input (e.g. a factorial function does not need to correctly handle strings).
- EVERY .py file must have a comment line at the very top containing your name(s), lab section, and a brief description of what the file is.
- Write the tests first! Real software engineers do this for very good reasons - so should you!

Preparation (do not submit for grading)

1. Fire up IDLE so that you have a shell window. At the prompt type in:

   ```
   >>> min(5, max(1, 7))
   ```

   This is a simple example of function composition: taking the result of one function and using it as input to another function.

2. Type the following into the shell window (you need to press enter twice at the end of the function to get the prompt back):

   ```
   >>> def plus1(x):  return x + 1
   ```

   Figure out how to compose multiple calls to the `plus1` function to get the number 3 if you start by calling `plus1(0)`.

3. This following exercise walks you through the process of writing a a function which converts from Celsius to Farenheit. It will not be graded, but you *must submit it as part of the lab.* Download *lab03.py* and *lab03_tests.py* from the Labs folder on the course website (be sure to save them both in the same directory,) as you will be putting the following code in those two files. You can open a file in IDLE by selecting File→Open from any IDLE window.

   (a) The first thing you want to do is think about what a function which converts Celsius to Farenheit should look like. What parameters does it take and what does it return? Write a test which reflects your decisions. This test should be one you expect to pass

[1]If you would like to work with someone but don't know whom, your TA may be able to help connect you to other students looking for lab partners.

when the function is working - even though it won't pass right now because you have yet to write the function:

```
def test_temperature_convert(self):
    self.assertEqual(celsius_to_far(0), 32)
```

Add the above test to lab03_tests.py, directly beneath the sample test, indented exactly as that test.
Run your tests. (by going to Run→Run Module in the lab03_tests.py window.) You should notice that this test now fails.

(b) Now add to the bottom of lab03.py a *signature* and *header* for your function:

```
def celsius_to_far(temp):
    """
    Takes temperature 'temp' in Celsius and computes and
    returns the equivalent Farenheit temperature.
    """
```

The function signature is the def statement line and the header is everything between the `"""`. The point of a header is to describe in English what the purpose of the function is. Describe each parameter and why it is necessary, as well as what the function returns. This way other programmers reading your code (or even yourself after a couple of months) will have an idea of what you were thinking when you wrote it.
If you run your tests now, your test for celsius_to_far will still fail but now the error message will be different. Why?

(c) Now you'll want to implement the function such that your test passes. Add the code for the function underneath the header:

```
def celsius_to_far(temp):
    """
    Takes temperature 'temp' in Celsius and computes and
    returns the equivalent Farenheit temperature.
    """
    return 9/5 * temp + 32
```

Run your tests again and note that `test_temperature_convert` now passes.

(d) Modify `test_temperature_convert` with a new assertion that you expect should hold for a properly implemented celsius_to_far function:

```
def test_temperature_convert(self):
    self.assertEqual(celsius_to_far(0), 32)
    self.assertEqual(celsius_to_far(100), 212)
```

Run your tests again. In this case, `test_temperature_convert` still passes. If it didn't, you would need to go back and modify your implementation of area such that the test again passed.

Programs (to be graded)

1. Write a function that takes in values representing whether some thing is made of wood and whether (the same) something is a duck. The function will return true if the something floats (both wood and ducks float; nothing else floats). However, if something is both wood and a duck, the function will return false.[2]
2. For this part, you'll be writing some code you'll use in the next parts.
   (a) Write a function which calculates the area of a rectangle. Be sure to use the process outlined in the preparation section of this lab.
   (b) Now write a function which calculates the circumference of a circle. Again, write the tests first like in the preparation!
   (c) Finally you'll need a function which calculates the area of a circle. Feel free to "borrow" the code we wrote in class for this one. Either way, make sure it's properly tested.
3. Implement a function to calculate the surface area of a cylinder, as always writing the tests first. In your implementation you can (and should!) use the functions you wrote in part 2. Note how two of the calculations you did in 2 are needed for this function, and how you don't have to re-do them since they're each a callable function.
4. Now you'll write a function to calculate the volume of a cylinder. You should notice that this function can again make use of some of the functions from part 2.
5. Download the file *cylculator.py* from the course website. Make sure you save it in the same folder as the rest of your lab 03 stuff. Open and run it in IDLE.[3] Note that it tells you that the area and volume for your cylinder are both 0 no matter what numbers you enter for its height or radius. Fix it so that it instead uses your functions from parts 3 and 4 to get the area and volume, respectively

You should submit your lab03.py, lab03_tests.py, cylculator.py and any other docs required by your TA on Sakai.

[2]Your job is to code what the customer asks for, not question his or her competence or sanity.

[3]Geez there's a lot of code in this file! Most of it probably makes no sense to you. Fortunately, you need to make sense of almost none of it to make the necessary changes to plug in your code. This is a thing which happens often in software development.