CISC106 Spring 2013 Lab04

- This lab an all subsequent labs will be due Thursday 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 (should, even) 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!

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] You should make a `lab04_test.py` file for the tests for this function. You'll probably want to use the lab03_tests.py in the labs folder on the course site as a template for your lab04_tests.py. You should also make a `lab04.py` file and implement this function there.
2. Download zelda_demo.py and Frames.py from the course website. Put them in the same folder as your lab04.py and lab04_tests.py. Run zelda_demo.py and you should get an error about not being able to import get_direction or some such.
3. Implement a function called `get_direction` which, on a particular character[3], gives the direction corresponding to that character. The correspondences are as follows:
   - The character `'w'` corresponds to the direction `'North'`
   - The character `'a'` corresponds to the direction `'West'`
   - The character `'s'` corresponds to the direction `'South'`
   - The character `'d'` corresponds to the direction `'East'`

[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.

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

[3] by 'character' I of course mean a string of one letter

On any other input, the behavior of the function is *undefined*[4] As always *write the tests before you write the code!* There are four cases you need to test for.

4. Now try running zelda_demo.py again. You should get a window with Link from *The Legend of Zelda* somewhere near the middle. Try moving him around on the screen:

   - `w` should cause him to move *up*
   - `a` should cause him to move *to the left*
   - `s` should cause him to move *down*
   - `d` should cause him to move *to the right*

   If he reaches the edge of the window, he'll wrap around to the other side. **Now**, to prove that what you just did wasn't magic, change the name of `get_direction` to simply `direction`. Of course, you should update the tests to reflect this new name as well. You should also update zelda_demo.py so that it works with this change.[5]

You should submit lab04.py, lab04_tests.py, and zelda_demo.py along with any other docs required by your TA on Sakai.

[4] In programming, when behavior is documented as undefined, this means that the behavior could technically be anything. For example, your get_direction could deliver a box filled with angry cats to the caller if the caller passed in a 'u' and this would be correct behavior.

[5] **Hint:** Look for the occurences of get_direction in zelda_demo.py and replace them with direction. Find and replace them individually - if you just do a "replace all" you're ripping yourself off!