CISC106 Spring 2013 Lab05

- 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
- Review the code examples from your notes in class.
- 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!
- Create lab05_tests.py (for tests) and lab05.py (for code) files. Use your lab03_tests.py as a reference for how to start lab05_tests.py.

Preparation (do not submit for grading)

1. Run IDLE and at the prompt in the shell window enter the following two commands:

```
>>> print('Two\nLines')
```

   Note what that `\n` in the middle of the string does! This is how you can add multiple lines to a string without using `"""`.

2. Now type the following into the prompt:

```
>>> stuff = 'Stuff'
>>> print(stuff[:-1])
```

   You should see the string `'Stuf'` printed. Putting `[:-1]` at the end of a string causes its last character to be chopped off! (We'll talk more about why this is after Spring Break - for now just feel free to use it to get rid of any pesky `\n`s at the very end of your strings. Even though it's two characters, a `\n` is treated as if it were one character.)

[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.

Problems (to be graded)

1. Download the file *asterisk_diamond.py* from the course website. Open it in Idle and try to run it. At this point, you should just get error messages (about lab05 not existing, or if you already created you lab05.py, about some asterisk functions not existing).

2. Write a *recursive* function `asterisk_triangle` which takes an integer and then returns an *asterisk triangle* consisting of that many lines. An asterisk triangle is perhaps best described graphically:

```
*
**
***
****
```

   The above is a 4-line asterisk triangle

3. Now write an *iterative* (using loops) function `backward_asterisk_triangle` which takes an integer and returns a *backwards* asterisk triangle (one which slopes downward to the left rather than to the right.) Again, a 4-line backwards asterisk triangle looks like:

```
   *
  **
 ***
****
```

4. Finally, you should impement a recursive `upside_down_asterisk_triangle` and an iterative `backward_upside_down_asterisk_triangle` which give an upside down asterisk triangle an an upside down backward asterisk triangle, repsectively. 4-line upside down triangles look like the following (to the backward one is on the right):

```
****                                    ****
***                                      ***
**                                        **
*                                          *
```

   These functions should be relatively easy after you've done the ones from parts 2 and 3.

5. Now you should try running asterisk_diamond.py again. This time, you should see an animated diamond of asterisks expanding and contracting from the center of the popped-up window.

6. From a box full of discs, we would like to know the probability of pulling two blue discs in a row when all the discs in the box are either red or blue. Write a function which can calculate this probability for a box filled with any number of red discs and any number of blue discs. A test case you may want to use: if the box contains 15 blue discs and 6 red discs, you have a 50% chance of drawing two blue discs in a row.

7. Now write a function that calculates the probability of drawing n blue discs in a row for some n between 0 and the number of discs in the box. (You don't have to handle cases where n is larger than the number of discs in the box.)

8. Download the file *discalculator.py* from the course website. Make sure you save it in the same folder as the rest of your lab 05 stuff. Open and run it in IDLE. Note that it tells you you've got a 0% chance no matter what numbers you enter for blue discs, red discs, and draws. Fix it so that it instead uses your function from part 7 to get the probability.

You should submit your lab05.py, lab05_tests.py, asterisk_diamond.py, discalculator.py, and any other docs required by your TA on Sakai.