CISC106H - Fall 2017
Lab 1, due Sunday 9/17

Welcome to your first lab! Please read directions carefully if and only if you want a good grade on the assignment.

1. Find zero, one or two partners. Go to the TA and let the TA know your names and that you will be working together for this week-long lab. You will work with someone different next lab. It is good to practice working with other people, and good to network with other students and start to build working relationships and learn to communicate about work.

2. Go to the class website and download cisc106.py. You may have to use "save link as". You will need to have this file in the same directory as any python files you write that want to import it. Once you have downloaded it, upload it to replit so you can use it there.

Problems for submission later:

3. Create a Python file, **sum.py**. Write a function header (the header is the line of code that starts with "def") for sum, which will compute the sum of only two numbers.

   Give the function a body that says simply "pass" (this is how we make an empty function body). This way it will run, but won't give correct answers.

   At the top of your sum.py file, import from cisc106.py by saying:

   ```
   from cisc106 import assertEqual
   ```

   at the top of the file with the sum function.

   Now write three assertEqual tests below your function:

   ```
   assertEqual( sum(2,3), 5)
   assertEqual( sum(0,3), 3)
   assertEqual( sum(-2,0), -2)
   ```

   Run the module and you should get fail messages for all three tests. Cheer when this happens, you have correctly used assertEqual from cisc106.py. Show your TA that you have successfully failed.

   Now fill in the correct body of the function in your sum.py file and demonstrate your tests passing.

4. Code the function discussed in the second lecture, close_enough(num1, num2, tolerance). Then write the function circle_area(radius)[1], and test it using composition of functions assertEqual and close_enough.

5. Use composition to write the function cylinder_volume(radius, height) using your circle_area function. Test as above.

[1] Be sure to use the built-in pi constant discussed in class!

6. Design tests for an absolute value function named **myAbs**. (Python has a built-in function for computing absolute value, but **do not** use it here. As a rule, when asked to write a function you should never use the built-in version[2].) Code the tests and then write the function **myAbs** using an **if** statement[3].

7. Observe the menu function below. Run this program and call menu().

```
def menu():
    """
    What does this program do?
    """

    choice = input("Please enter \n\
1) to calculate your taxes;\n\
2) to achieve world peace:\n")

    print(choice)
```

   Modify this code to make a fun program that uses simple input to tell the user facts about your hobby. Pairs must submit two programs (i.e. one program per student). Nested input testing is extra cool.

8. Using only conditionals, composition, and recursion (all covered in class so far), write four functions that take a size parameter and use line_of_stars and line_of_spaces to draw an isoceles triangle[4]. You will also write print_space() and line_of_space(). Your triangle functions may not call print_stars or print_spaces. You may write a helper function if you need another parameter, but the functions shown below must operate with a single parameter as shown.

   For example:

```
def print_star():
    print('*', end="")

def line_of_stars(n):
    """Does not return anything, just prints a single line"""
    if n>0:
        print_star()
        line_of_stars(n-1)
```

[2]Why do you think we ask you to write functions that Python already has?

[3]Could you code it without an if? What would it look like? Would it be a good idea?

[4]In other words, these four functions will not directly print asterisks or spaces - they get the previous two functions to do that work

```
>>> upLeft(3)
* * *
* *
*
>>> downLeft(4)
*
* *
* * *
* * * *
>>> downRight(3)
    *
  * *
* * *
>>> upRight(2)
* *
  *
```

Since your functions only print, they won't have results testable with assertEqual[5].

Assignment Submission:

Make sure both names of your programming pair are in each file.

Have one or more python functions in a .py file, with the relevant assertEqual tests in the same file. You need functions sum, close_enough, circle_area, cylinder_volume, myAbs, all with tests; and menu, line, space, and triangle functions without tests.

Submit on Canvas: One or more python files containing code and tests. Follow the TA's instructions if they vary from this document.

[5]Think: How could you rewrite the line and triangle functions so that assertEqual could test them? Don't do this yet.