CISC106 Honors - Fall 2017
Lab 4, due Sunday 10/8

Every program below must have at least three good tests unless otherwise noted.

1. Download version 3.4 or higher to use for writing this assignment. Python comes with an IDE called Idle, which is faster and more reliable than using repl.it across a network connection. It saves files nicely on your machine, which can then be uploaded to Canvas.
2. https://docs.python.org/3/tutorial/index.html
3. Do labs for Understanding! You are paying for this education in some way(s), make the best of it! Use your prof and TA to your advantage!
4. Go to TA office hours!!! They are valuable. They know things. They have been there and done that.
5. For this lab, you may work alone or with a partner of your choosing. If you plan to work with a partner, work with the same partner the entire lab. Do not share your work with other pairs or individuals.
6. Working with a partner has many benefits, but a major one is that you can be scouting for a partner to work with on a project. The best teams are people of similar experience/skill levels (why?). Friends often make really bad project partners (why?).

Problems for submission:

7. Write a function **sumN** that uses a for loop and a range (not a formula) to compute and return the sum of all numbers 1-N, inclusive.
8. Write a function **productN** that uses a for loop and a range to compute and return the product of all numbers 1-N (not inclusive).
9. Write a function **evensN** that uses for and mod (%) to make a list of all the even numbers from range 0-N, inclusive.
10. Write a function **evensN_B** that makes a list of the same numbers as the previous problem, only without using mod; use only range to generate the even numbers.
11. Write a function **squaresN** that makes an empty list and binds it to a name. Write a for loop that uses assignment and slicing with an index to add elements to the list. Do not use +, append, extend, etc. Put the squares of the numbers from 0-N in the list.
12. Write a function **contains(key, alist)** that uses a for loop to return True if key is in the list, and False otherwise. Show it working with strings and numbers.
13. Read about while loops and multiple assignment (two different things!) at

    `http://docs.python.org/release/3.2.3/tutorial/introduction.html#first-steps-tow`

Write a function get_two_nums that uses returns **two** specified elements (as shown in the article) from a sequence:

get_two_nums(2, 5, range(7,21)) $->$ (9, 12)

14. Read about while loops and multiple assignment (two different things!) at

    `http://docs.python.org/release/3.2.3/tutorial/introduction.html#first-steps-tow`

    Write a function **guessN(target, max)** that takes a target and a max (the upper limit of the range). The function then loops using **while** , taking input from a user until the user guesses the target within the range(look back at your menu function if you do not remember how to do this). You may provide hints to the user if you like[1]. This function does not require tests.

15. Code the procedure make_combos that returns all combinations of a list. Order is not important[2]. Hint1: This may be easier using recursion than a loop. Hint2: think about how you would go about changing the result shown if you wanted to add 4 to the list? 5? Write it out to see the pattern.

```
> make_combos([1 2 3])
[[1] [1 2] [1 2 3] [1 3] [2] [2 3] [3]]
```

16. Consider this function. Write three good tests for the code according to the comments. Run the code, then fix the code. Be certain you understand the problem.

```
data = [2,2,2,2,2,4,2,4,2,4,2,4]

def myRemoveSmallNums(data):
        """ Removes numbers smaller than four from the data """
        index = 0
        for elt in data:
                if elt < 4:
                        data[index:index+1] = []
                        index += 1

myRemoveSmallNums(data)
```

[1]If you tell the user the possible range of numbers and then provide hints of greater or less than, how long will it take the user to find the target in the general case, assuming the user has an optimal strategy?

[2]Challenge: *after* you pass your tests, can you shorten your code using the Python functions map, apply, or reduce?