Testing Guidelines for CS16

This document defines some fundamental rules to help you write good tests for your code in cs16 and beyond! While the suggestions covered her are by no means exhaustive, they'll help you start thinking about how to test your code well, and therefore ensure that your code works as expected.

- Try to cover every line of code (coverage)
    - As a simple example, if you have "if and else" statements, write your tests so that each case is tested
    - In the following example function, it would be good to test both positive and negative numbers

      ```
      def isGreaterThanZero(num):
              if num > 0:
                      return True
              else:
                      return False
      ```
    - In other words, try to cover every input/output case. So you could test the following:
        - self.assertFalse(isGreaterThanZero(0)) → should be false (0 is not greater than 0)
        - self.assertTrue(isGreaterThanZero(5)) → should be true (5 > 0)
        - self.assertFalse(isGreaterThanZero(-5)) → should be false (-5 not greater than 0)
- Try to test edge cases
    - For instance, if you have code that sums up all the numbers in a list, make sure to test the empty array
        - self.assertEqual(sumList([ ]), 0)
    - You might also want to see that certain errors are raised (on bad inputs)
        - For instance, if a function should take in an integer, but someone passed it a string, you should make sure the the correct errors are raised
    - Other edge cases include: testing code behavior with empty string, empty array
- Think about different kinds of inputs
    - For instance, if you were trying to perform an function on a list, would it work with duplicate elements?
    - What about odd or even length inputs?
    - Or positive or negative inputs?
- Sometimes randomization helps
    - For instance, if you wanted to make sure a function always return the max value in the list, you can create a random list and make sure the function returns the correct value by checking the value returned by our function with python's "max()" method. (Assume the function we wrote was ourMaxValue())

```
■ l = [ ]
  for int in range(0,100) //created a list of 100 numbers
          l.append(random.randint(-1000, 1000)) //append a random integer
from -1000 to 100
  maxNum = max(l) //use python’s max
  self.assertEqual(ourMaxValue(l), maxNum)
```