ISTA 350
Spring 2014
Programming Assignment 1 – Binary Madness!
Due Monday, February 3, at midnight (10% penalty until Tuesday at midnight, then 0)
Turn in to your dropbox on D2L.

Instructions and suggestions:

- **When implementing your methods, do not convert your binary numbers to decimal, do the math, and convert back. You will not get any points for this. Your methods may use any of your other methods except __int__. One of the two main points of this assignment is to practice working with lists.**
- Do your own work.
- File names and function signatures must be exactly as specified.
- The data attribute for you class must be named num_list.
- The web is your best friend.
- **Start now!**

Background. All data is stored in computers as sequences of 0's and 1's. Signed integer arithmetic is typically done using a "two's complement" binary representation of numbers. Other schemes, such as leftmost sign bit and one's complement, turn out to have significant drawbacks (namely two 0's and more complicated arithmetic). In two's complement, you find the negative of a binary number by first making all 0's into 1's and vice versa, and then adding 1 to the result. In computers, bytes are the smallest addressable unit of memory and are typically composed of 8 bits (binary digits). Thus, a byte representation of 5 is 00000101. Its negative, -5, in twos complement is 11111010 + 1 = 11111011. This works both ways: if you flip the bits of -5 and add 1, you get 5 back.

In this assignment, you will implement a two's complement class, containing 10 methods, each worth 10 points, plus any helper methods/function and test methods (see below). Each instance (object) of the class represents some number in two's complement. The digits of the number will be stored in a Python list of 0's and 1's (integers, not strings), the class's only data attribute (instance variable), and will not have any fixed length. Positive numbers will begin with 01 and negative numbers with 10, except for -1 (1) and 0 (0). The numbers should be stored with no extraneous leading 0's or 1's. You may temporarily add some for convenience while manipulating them in your methods, but they should be removed at the end of any such method. Here are -8 to 7:

1	-1
10	-2
101	-3
100	-4
1011	-5

0	0
01	1
010	2
011	3
0100	4

1010	-6
1001	-7
1000	-8

0101	5
0110	6
0111	7

You are required to unit test all of your code. For this assignment that just means you need to have a test method for every method/function that you write. This includes any additional methods/functions that you choose to add yourself. You should end up with at least 10 test methods. The test_interface.py file on D2L contains a basic starter point for your unit tests. Rename that file to test_binary.py and complete the tests. You might find it useful to write the unit tests before working on binary.py. This file also falls under the style guidelines so don't forget to add documentation and follow best practices.

Download interface.py from D2L, rename it binary.py, and implement your class. Test it thoroughly! It is easy for the arithmetic methods to go awry in some cases while others appear to work fine.