Assignment 1
Due Date: XYZ

The purpose of this assignment is to get familiarity with the basic structure of programming we will do through the semester. You will write two programs, one which outputs a text file of data and another which plots it. You will be graded as much on the structure as the correctness.

For this assignment, you will be investigating one of the most difficult open problems in mathematics, the Collatz conjecture. This problem was first posed in the 1930s, and while there was a major breakthrough last year, it is still unsolved. Many mathematicians believe that the current state of math is such that it can't handle problems like this. But, it is easy to explore with a computer program.

The Collatz conjecture deals with the following sequence:

$$f(n) = \begin{cases} \frac{n}{2} & \text{if } n \text{ is even} \\ 3n+1 & \text{if } n \text{ is odd} \end{cases}$$

The sequence is formed by repeatedly applying this expression to a starting positive integer. For example, starting with 3, we get (10, 5, 16, 8, 4, 2, 1, ~~4, 2, 1~~, etc). Starting with 8, you get (4, 2, 1, ~~4, 2, 1~~, etc). Consider a sequence "ended" when it hits 1 for the first time. The conjecture is *"every Collatz sequence ends at 1"*. No one knows if this is true. For example, it may be possible that some sequences run away forever, or some may get into a repetitive cycle involving numbers other than 4, 2, 1.

1. Write a program to calculate the length of each sequence *L(n)* for positive integers 1 to 100,000. In the examples above, *L(*3) = 7 and *L*(8) = 3. Your program should accept no inputs and output a text file with two columns, *n* and *L*(*n*). You can use the program *collatz_skeleton* to get started.
2. Write a separate program that reads in the text file and plots *n* vs *L(n)* for ranges n=(1, 100); n=(1, 1000), and n=(1, 10000). You can use the program *plot_collatz_skeleton* to get started. You can check your results against the Wikipedia page.
3. (Bonus) The program you wrote in 1) will likely be able to handle *n* from 1 to 10**6 in a reasonable amount of time. But what about *n* = 1 to 10**9? Can you think of a way to modify your program to efficiently handle extremely large ranges? Hint: what happens if a sequence arrives at a number that is in a previous sequence? How are the lengths *L*(*n1*) and *L*(*n2*) related? Also, warning—if you try to store your output from 1 to 10**9 as a text file, that will likely take about 10 GB of space.