Homework 2: Loops

This homework will be done using the hw2.py file available on the class website. The file contains a series of unwritten function definitions. Below are descriptions of what each of the functions should do. You should complete the function definitions so they do the expected thing. Remember to change the `return` statement in the function.

You should test your functions! Use `print()` commands in your program to make sure that you are getting the correct output from the functions. For this assignment, it is ok to leave these print commands in the program, but make sure there are no `print()` commands in the function bodies themselves – the functions should do nothing other than return the correct value.

It is always ok to use functions defined in one exercise when writing the functions of later exercises. It is also ok to write your own functions to use in writing these functions. That includes copying over functions from previous homeworks.

When you are done defining these functions, upload the modified hw2.py file in a homework2 folder inside your submission folder. Also print the file and bring it to class.

Exercise 1: `is_prime`

This function takes a single positive integer and returns True or False, depending on whether the number is prime. (Remember, "prime" means a number has exactly two factors, 1 and itself.)
`is_prime(1)` returns `False`.
`is_prime(4)` returns `False`.
`is_prime(7)` returns `True`.

Exercise 2: `sum_cubes`

This function takes an integer n and returns the sum of the first n cubes, starting at 1. The cube of n should be included in the sum.
`sum_cubes(1)` returns `1`.
`sum_cubes(2)` returns `9`.
`sum_cubes(6)` returns `441`.

Exercise 3: `fibonacci3`

The Fibonacci sequence begins 1, 1, 2, 3, 5, 8. Each number in the sequence (except for the first two) is gotten by adding together the two numbers before it. You could define another sequence, which we'll call the 3-Fibonnacci sequence, by starting with three 1s and then getting each number by adding the previous three. The function `fibonacci3` should take an integer n and compute the n^{th} number in this sequence.
`fibonacci3(2)` returns `1`.
`fibonacci3(4)` returns `3`.
`fibonacci3(7)` returns `17`.

Exercise 4: `polymax`

Consider the function $z = -x^4 + 3x^2 - y^4 + 5y^2$. We are interested in the value of z when x and y are integers. Your function should take as input minimum and maximum values for x and y and should return the maximum value z can take when x and y are in the specified ranges. (Allow x and y to take values equal to the entered ends of the ranges.)
`polymax(0,5,0,10)` returns `6`.

Exercise 5: `collatz`

Consider the following process. You start with some integer. If the number is odd, multiply it by 3 and add 1. If it is even, divide it by 2. For every number that has ever been tried, this process eventually leads to the number 1. (It is an open problem, the Collatz Conjecture, whether this is true for all integers.) For example, starting with 6 yields the sequence 6, 3, 10, 5, 16, 8, 4, 2, 1. That is a sequence of length 9. The function `collatz` should take as input an integer n, and return the first integer for which the generated generated sequence is at least n in length.
`collatz(10)` returns `7`.