Part 3

August 2017

1. Guess the results of the following statements (note: there are a couple of tricks, you may need to import something to avoid receiving an error message):

   a) 1==2
   b) 1=2
   c) 1<2
   d) pi == 3.141592
   e) 1 = True
   f) 1 == True
   g) 0 == False

2. Write a program that allows the user to enter a positive integer. If this number is even, print out its square. If it is odd, print out its cube. Hint: user input should be converted to an int; consider the `int()`method.

3. Write a program that allows the user to enter two positive integers. If one number divides the other, print "yes", otherwise print "no".

4. Similarly to activity 4 and 5, take an inputted number and make a program that will print out whether it is odd/even, bigger/less than 100, zero/non-zero, negative/positive.

5. Create two of your very own infinite loops and explain why they'll be infinite.

6. Create a conversion program that takes two user inputs: a number and a unit of mass ('kg', 'lb') in string form. Tell the user using the input function what they must enter and in what format. Make a program that takes a value and a unit, converts it to the other unit and then prints the result.

   For example, say the user inputs 5 and 'lb'; your program would return '5 lb is xxx kg'

   For some extra work instead of using only units of mass make the input any metric unit or any imperial unit and convert to the other system

   For example, metres to yards, etc...

7. Make a guessing game of your own similar to activity 7, but that guesses two numbers, not just one.

8. Create a program that counts from 0 to 25 and then back down to 0. Each time you add a number have the program print the number it is at. This program should be no more than 10 lines.

9. Write a function sum that takes two integer parameters a and b and returns the sum of squares of all the integers n such that a <= n <= b.

10. Write a function triangle that takes three real parameters which represent the lengths of three sides of a triangle. The function should return True if the triangle is right-angled and False otherwise.

11. a. Write a function factorial that takes a nonnegative integer parameter and returns its factorial; we define 0! = 1.

    b. The number e can be defined as the limit of the infinite sum 1/0! + 1/1! + 1/2! + 1/3! + 1/4! + ... Write a function compute_e which takes a positive integer parameter. This parameter indicates the number of terms that are to be summed (i.e. compute_e(3) should produce the sum 1/0! + 1/1! + 1/2!). Sum the terms and return the approximate value of e. Call/use the function you wrote in part a in your solution.