You only have to do one of the following: 2F or 3M (this assignment)

PIC 16, Winter 2018 – Assignment 3M

Assigned 1/22/2018. Code (a single `.py` file) due by the end of class 1/26/2018 on CCLE. Hand in a printout of this document with the self-assessment portion filled out by the end of class on 1/26/2018.

In this assignment, you will write a Python program that illustrates the utility and relative efficiency of exception handling over `if` – `else` statements.

Task

1. Write a function `my_divide1` that accepts two arguments, `a` and `b`. Assuming both arguments are lists of numbers, the function is intended to divide each element of `a` by the corresponding (same index) element of `b`. Write this function assuming that the arguments will be lists of numbers and that `b` contains no zeros. If the lists contain different numbers of elements, perform the operations for all the elements that you can, and simply ignore the extra elements of the longer list. This should all be accomplished in a single line using list comprehension and one of the handy built-in functions we learned last time.
2. Now consider:
   - What b contains a zero?
   - What if any of the elements are strings or other objects for which division cannot be performed?

   Without using `try-except` (using `if-else` instead), write a function `my_divide2` that performs the intended list division when it can, but returns an empty list [] and prints a message like “`Somethings’s wrong with the inputs to my_divide2`” otherwise. This is the LBYL (Look Before You Leap) approach. *Note: You’ll probably need to look online for how to detect whether division is going to work (before your code attempts it).*
3. For long lists, it takes a while to check your `if` conditions on each element before even starting to do the division. It might be more efficient to just try to do the division operation, and then print the message and return [] if any exception occurs. Write a function `my_divide3` that uses this EAFP (Easier to Ask Forgiveness than Permission) approach.
4. Which is faster, `my_divide2` or `my_divide3`? Make sure the functions produce the same output for the inputs:
   ```
   a = range(0,1000000); b = range(1,1000001)
   a = range(0,1000000); b = range(1,1000000)+ [0]
   a = range(0,1000000); b = range(1,1000000)+ ['a']
   ```
5. Often you’ll want to be able to print a more detailed message depending on what went wrong, e.g. “`There is a zero in b`” or “`Non-numeric data detected`”? Create `my_divide4` (by modifying `my_divide3`, still without using `if` statements) to accomplish this.

Self-Assessment

1. Were you able to perform task 1 with a single line in the function?
2. Were you able to perform task 2?
3. Were you able to perform task 3?
4. According to your tests, which is faster: `my_divide2` or `my_divide3`?
5. Were you able to perform task 5?