COS 125 Fall 2016 Lab #3
Due Friday Sept. 23 EoD

This lab asks you to create two short functions that work with lists or sequences. For each problem, write the function indicated, as well as a main program ("driver") that tests and demonstrates each of your functions. To submit your lab, create a document that has your source code as well as some output demonstrating your functions. You may submit not-quite-working programs for partial credit. You can submit a text document or a pdf. Please put "Lab 3", your name, and the date submitted at the top of the document.

Note: We did not have short coding exams in this class last semester so there are no old exams that you can review. However, both of these problems are typical of what you might expect to see on a short coding exam. List and sequence algorithms are core material for this course.

1. (50 points) Design a function that returns a list of random character strings containing digits only. This is almost equivalent to a list of random integers, except that (a) each item in the list has the same number of digits; and (b) leading 0's can be part of a sequence of random digits.

Inputs:

1. How many random digits to display in a sequence;
2. How many sequences to display.

Output:

It will return a list containing the number of strings requested, each having the length requested

Notes:

1. For incremental development, consider using one function that generates a single sequence of the specified length and another function that generates a list of sequences
2. You might want to refer to http://docs.python.org/2/library/random.html; there are a number of different functions you might be able to use
3. A random sequence can include leading 0's. This might affect your choice of functions to use.
4. Take a look at random.seed(). If you do not call this your program will produce the same sequences every time.

For example, if the user entered 4 for the number of digits and 3 for the number of sequences, the function might return ['6723', '0288', 9541']. Note that there are quite a few possible ways to approach this problem. You might for example randomly choose from a set of digit characters, or you might create random integers in range and then prepend leading 0's where appropriate.

2. (50 points) Design a function called **seqzip** that has the following specifications.

Inputs: Two sequences (lists, strings, or tuples)

Output:

A list of two-element lists containing corresponding elements from each input sequence (elements in the same index position).

If the lists are of different length, return a list of lists with length equal to shortest list. If either list is empty, return an empty list.

Examples:

```
>>> print seqzip ([1,2,3]['a','b','c'])
 [[1,'a'],[2,'b'],[3,'c']]
>>> print seqzip ('5432',['spam','and','eggs'])
 [[5,'spam'],[4,'and'],[3,'eggs']]
```

Notes:

1. Because the indexing operation is common to any sequence type (e.g., x[0] refers to the first element of x), the function will work correctly even if the inputs differ in type, such as a list and a string.
2. While you can use an **`if`** statement to test for the shortest of the two input sequences, try a web search for "Python min function." The seqzip function can be written in as few as four lines of code, but as noted in class, clear and well-written code that is somewhat longer is just as good as the shortest possible program.

3. **Extra Credit** (up to 20 points) Write a list of test cases for each problem, with a brief note explaining why the test case is considered. For example:

```
seqzip([1,2,9],"abcd")        test inputs of different types and lengths
```

The goal of this extra credit exercise is to systematically consider possible test cases. Some points to consider are:

1. Look at boundary cases (for example empty lists, 0 length sequences, etc.).
2. From the principles of mathematical induction, if a program works for a case with N items, it can be expected to work with the case of N+1 items, unless N+1 crosses some sort of boundary. In other words, if it works for 3 items, it can be expected to work with 4 items.
3. Look at the range of acceptable input types as well as the range of acceptable input values.