ECE 498 - Python **Homework 7** **Fall 2020**

Goals: investigating large array types, generator functions, more programming practice.

For this homework, we'll go back to using Google Colabs. Share your work with me rather than sending the "`.ipynb`" or "`.py`" file. Note that some points may be deducted if your code could obviously be written in a "more Pythonic" way.

1) Write functions with the following prototypes. Each will return a string, list, array, etc. of the given type containing the appropriate number of random values. Each function body should be one line. See notes for "`random`" information.

```
def randstring(length = 10000):     # string of random lower case characters
def randlistchar(length = 10000):   # list of random individual characters
def randlistbytes(length = 10000):  # list of random integers from 0 to 255
def randlistint(length = 10000):    # list of random ints from 0 to 65535
def randarraybyte(length = 10000):  # array of random ints from 0 to 255
def randarrayshort(length = 10000): # array of random shorts from 0 to 65535
def randarraylong(length = 10000):  # array of random longs from 0, 65535**2-1
def randarrayfloat(length = 10000): # array of random floats from 0 to 1
def randarraydouble(length = 10000):# array of random doubles from 0 to 1
def randtuple(length = 10000):      # tuple of random ints from 0 to 255
def randset(num = 10000):           # set of "num" random ints 0 to 65535
```

2) Write test code that will call each of the above functions (using the default value of 10000) and assign it to a variable. For each, print the name of the function called, the size of the variable and the first 5 elements.

3) Write a generator function with prototype "`def genfib():`" that will generate Fibonacci numbers, beginning with 0 (We'll call it `fib0`)

4) Write test code to use `genfib()` to print the first 11 Fibonacci numbers (fib0 through `fib10`), all on the same line, separated by spaces. This code should make use of a "`for`" loop that loops over `genfib()`. (Consider using "`enumerate`")

5) Write second version of the test code that uses `genfib()` to print the first 11 Fibonacci numbers (fib0 through `fib10`), all on the same line. This version should make use of a "`for`" loop that uses "`range`" and grabs values using "`next()`".

6) Write code that uses `genfib()` to compute `fib1000000` (that is 1 million). Print the number of digits in this number, as well as the first and last digits. (Hint: converting to string may help). You can print the number for fun, but leave that out of code you submit.

7) `lookNsayNth()`, given below, is a generator function that generates the digits (as characters) of the "n-th" "look-and-say number" (see class notes). E.g., 1113… generates the characters '1', '1', '1', '3',

etc. Use this generator to print out the 1,000,000-th to 1,000,019-th digits of the 200th "look-and-say" number. Check: your number will start with 211121… (Keep in mind that the 1st digit has index 0 if the digits were put in an array). Note that the 200th "look-and-say" number has somewhere around 10**23 digits. Extra credit for printing them all!

```
def saythis(prev):          # Supporting function
      '''Generate characters of a "look-and-say" number given a generator of
characters  for the previous number.
      '''
      c,n = next(prev), 1
      for t in prev:
            if t==c:
                  n += 1
            else:
                  yield str(n)
                  yield c
                  c,n = t, 1
      yield str(n)
      yield c


def lookNsayNth(n = 1):
      '''Generate characters of the n-th value of the "look-and-say" sequence
recursively
      '''
      if n == 1:
            yield '1'
      else:
            yield from saythis(lookNsayNth(n-1))
```

Notes:

```
import random

random.randint(0,10)        # random int 0 to 10, inclusive

random.random()             # random floating point number 0 to 1
```

Another hint:

```
chr(ord('a') + 1)      # generates ‘b’
```