Laboratory Exercise 0: NumberCube

Assigned: Wednesday 26 August
Due: Sunday 30 August at 11:59 PM

The purpose of this exercise is to (re-)familiarize yourself with our UNIX computing environment including emacs and the shell, learn about how Prof. Campbell's assignments work, demonstrate some programming proficiency, and get comfortable submitting your work early and often.

Collaboration Policy

This assignment is meant to be a strictly individual effort. You may get help from other people only with respect to the mechanics of using the computing facility or general questions about the programming language. Concerning the algorithmic content of the problem, **you may not use any outside source** including the Internet,[1] written materials, teaching assistants, or any other people. If you have any conceptual questions, please direct them only to the Professor.

Introduction

A statistician is studying sequences of numbers obtained by repeatedly tossing a six-sided number cube. On each side of the number cube is a single number in the range of 1 to 6, inclusive, and no number is repeated on the cube. The statistician is particularly interested in runs of numbers. A run occurs when two or more consecutive tosses of the cube produce the same value. For example, in the following sequence of cube tosses, there are runs starting at positions 1, 6, 12, and 14.

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Result	1	**5**	**5**	4	3	1	**2**	**2**	**2**	**2**	6	1	**3**	**3**	**5**	**5**	**5**	**5**

The number cube is represented by the following class:

```
class NumberCube:

    def toss(self):
        """ return a random integer between 1 and 6, inclusive """
        return random.randint(1,6)

    # There may be instance variables, constructors and other
    # code not shown.
```

[1] Searches involving conceptual keywords of this assignment are not allowed.

You will implement a function that collects the results of several tosses of a number cube and another function that calculates the longest run found in a sequence of tosses.

What to do

Provide implementations of the following functions. Here are some details about the procedure.

1. Log in to the computer and connect to gemini to get a shell window with a `gemini` prompt. That's where you'll enter commands. In our lab, you can do this by double-clicking on the large X icon labeled `GEMINI-CS`.

2. Create the directory for this assignment. You only need to do this step once for each assignment.

   ```
   mkdir -p ~/111/numbercube
   ```

3. Each time you log in to work, change to the correct directory

   ```
   cd ~/111/numbercube
   ```

4. Write your functions in the file `numbercube.py`. You can use any text editor you like (nano, vim, emacs), but we can only help you with the editor if you use emacs.

   ```
   emacs numbercube.py &
   ```

   It's a very good idea to write just a small portion (outline) and them proceed to try the next steps. Then you can come back and refine things. You are allowed to write auxiliary functions.

5. Use good style at all times. Be consistent with use of white space. Choose appropriate variable names. Never exceed 80 characters per line. Carefully, succinctly and completely comment functions. At the top of the file, provide a header block like this one:

6. When you think you're ready to submit your work, at the shell window type

   ```
   submit
   ```

   This will collect `numbercube.py` and subject it to a battery of tests. Results will be shown. You can always `submit` as many times as you want. We retain all copies, but we grade only the last one before the deadline.

Functions to write

(A) Write the function `getCubeTosses` that takes a number cube and a number of tosses as parameters. The function will return a list of the values produced by tossing the number cube the given number of times.

```
def getCubeTosses(cube, numTosses)
    '''Returns a list of the values obtained by tossing a number cube numTosses times.
       @param cube : NumberCube, the cube to be tossed
       @param numTosses : int, the number of times to toss the cube
              Precondition: numTosses > 0
       @return int list, having length numTosses'''
```

(B) Write the function `getLongestRun` that takes as its parameter a list of integer values representing a series of number cube tosses. The function returns the starting index in the list of a run of maximum length. A run is defined as the repeated occurrence of the same value in two or more consecutive positions in the list.

For example, the following list contains two runs of length 4, one starting at index 6 and the other starting at index 14. The function may return either of those starting indexes.

If there are no runs of any value, the method returns -1.

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Result	1	**5**	**5**	4	3	1	**2**	**2**	**2**	**2**	6	1	**3**	**3**	**5**	**5**	**5**	**5**

Complete the function:

```
def getLongestRun(values):
    '''Return the starting index of a longest run of two or more consecutive repeated
       values in the list values
       @param  values : int list, a list of integer values representing a series of
                                  number cube tosses
               Precondition: len(values) > 0
       @return int, the starting index of a run of maximum length;
                    -1 if there is no run'''
```

References

This assignment is adapted from a redacted source.