CS417 Lab #7

Getting Started

Begin the lab by creating a folder for your files. Then, go to `mycourses.unh.edu`, find Lab07, and download these files into that folder:

- `numbers.txt`
- `histogram.py`
- `roll_die.py`

Exercises

1. The file `numbers.txt` contains 1000 numbers. Modify `histogram.py` to compute a *histogram* of these numbers. A histogram consists of several counts, corresponding to several bins. Each count says how many values fall into one bin.

   To run the program, type this on the command line:

   ```
   python histogram.py numbers.txt 10
   ```

   You should get this output:

   ```
   22
   41
   112
   152
   192
   189
   148
   82
   52
   10
   ```

   I've already written code which reads the numbers from the file in `sys.arvg[1]`, and gets `n_bins`, the number of histogram bins, from `sys.argv[2]`.

   Find the min (`lo`) and max (`hi`) of the numbers. Then find the `span = hi - lo` of the numbers. Finally, increase `span` by a little bit:

   `span *= 1.0001`, to make it bigger than `lo - hi`.

   Now, split the `span` into equal-width intervals, or "bins". There should be `n_bins` of them. Given a number x, which bin does it go into? Here's the arithmetic you'll need:

- x is between lo and hi, thus
- x - lo is between 0 and hi - lo, thus
- (x - lo) / span is between 0.0 and 1.0, thus
- (x - lo) / span * n_bins is between 0.0 and n_bins, so write this:
- bin = int((x - lo) / span * n_bins)

Because span is a bit larger than hi - lo, the bin will never be n_bins. It will range from 0 upto n_bins - 1. This covers exactly n_bins possible different values, which is what we want.

---

2. Open roll_die.py, and implement the function cummulative_probabilities(probs).

   To do this, build the cummulative probabilities. This is a list of values. Each value is the sum of values in probs, thus:

```
cumm_probs[0] = probs(0)
cumm_probs[1] = probs(0) + probs(1)
cumm_probs[2] = probs(0) + probs(1) + probs(2)
    ...
```

   However, you can't type that, because you don't know the length of probs; you need a more general approach. Do this:

   - start with an empty list: cumm_probs = [].
   - the first cumm_probs[0] is probs[0] (just append that to cumm_probs)
   - all the other cumm_probs[i] are cumm_probs[i - 1] + probs[i] (append that). You'll need for i in range(1, len(probs)):

---

3. In roll_die.py, implement the generator biased_generator(). It is passed a list of cummulative probabilities, and generates a sequence of biased random values.

   To do this, first call x = random.random() to generate a random number x from 0.0 to 1.0.

   Visit all the cummulative probabilities, and stop when x exceeds it the first time:

```
for k, prob in enumerate(cumm_probs):
    if prob > x:
```

   You want the k where this first occurs: break the for loop, and then yield k.

   Notice that biased_generator() doesn't return; it yields. This works with the for x loop in the main() function, to produce a sequence of values.

---

4. Test your programs: the `main()` function runs the generator 10000 times, and prints the die values. Save these in a file, and then use `histogram.py` to count them. You can do it from the command line thus:

```
python roll_die.py > rolls.txt
python histogram.py rolls.txt 6
```

The biased die has these probabilities:

```
0 0.1
1 0.2
2 0.1
3 0.2
4 0.15
5 0.25
```

So, if your program works, it should generate 10,000 counts distributed *approximately* like this:

```
1000
2000
1000
2000
1500
2500
```

Turning in your work

At the end of the lab session, turn in any work you have completed. You can continue to work today, and turn in all the work by midnight, with no lateness penalty. Go to `mycourses.unh.edu`, find lab 9, click "Submit", and upload `histogram.py` and `roll_die.py`.