Random Count Generator

Create a function that prints a random number between 1 and 5 to stdout (or console). The probability distribution of the numbers should be as follows:

1 - 50%

2 - 25%

3 - 15%

4 - 5%

5 - 5%

Or roughly that (no need to worry about rounding errors, etc)

Change the function to a method, add a class member that stores a history of the last 100 numbers, and a function to return the frequency percentages of each number.

Create a method that reads the most recently generated random number and the current time and writes them both to disk on one line.

Modify the writer function so it is launched in a different thread. The random number generator should put the numbers and timestamps into a queue that the writer function uses to write its output to disk.

Modify the program so it has 5 copies of the random number function running concurrently in different threads. They should all feed the single writer thread counts and timestamps, and the writer should ensure they are written in chronological order.

Each step of this should be committed in a version control system such as git. At the end, please submit the entire project and git history to us.

Beam Counter

Imagine an infra-red beam across a sidewalk, used as a sort of open-air convenience store door counter. Each time the beam is interrupted a count is incremented. Suppose that it works, and that an individual walking across the path of the beam triggers exactly 1 count. At the same time, if 2 people simultaneously interrupt the beam, they also only trigger 1 count.

At a rate of one person per hour, the system works perfectly. However, at a rate of 100,000 people per hour, the beam is never un-triggered and has an effective accuracy of 0. At more normal rates (hundreds to low 1000s), some proportion of targets will randomly overlap as they cross the beam. We want to quantify that.

Write a program that first simulates this beam counting effect…

1 -- Write a class or function that, given 1) a sample refresh rate and 2) an expected mean rate of observable events, generates a simulated event streams with the same mean. For example:

class PersonEmitter():

def \_\_init\_\_(self, rate, refresh):

# setup

pass

def next(self):

# this should return values such that if called

# refresh \* rate denominator times,

# it will output the rate

pass

2 -- Write a class or function that uses the above to generate mock events that are sent to a mock door counter. Pick some fixed, realistic window of time during which a target will traverse a beam, and if two events happen to be triggered during that window of overlap, count them as 1 and register the missed count.