MTH 3300
Fall 2024

Programming Problem Set 3: simulation.py

Directions: Download the template file provided on Blackboard. Then open Spyder, load these template files, and write the following program. Submit your source code via Gradescope, in `.py` format; do NOT send any other files. READ THE INSTRUCTIONS on submitting your work in the Course Documents section of Blackboard.

Be sure to read the SPECIFICATIONS carefully! And write comments!

You are playing a game where you roll a fair 6-sided die, and based on the outcome of your roll, you get $1 (if you roll a 1, a 2, or a 3), $5 (if you roll a 4), $10 (if you roll a 5), or $50 (if you roll a 6). **You can roll the die four times**, and if the sum of your earnings strictly greater than $x, you win and can keep your money! But if you lose, you *do NOT get to keep the cash* AND have to pay $100 for playing. (That is, your earnings end up being *negative* 100 dollars.)

For example, if x = 15, suppose we roll a 1, 1, 6, and 2. This means we have total earnings of $1 +$1 + $50 + $1 = $53 for the play, and since this is strictly greater than x = 15, we win and get to keep the money. On the other hand, (with the same value of x), if we roll a 1, 2, 3, and 4, we lose since $1 + $1 + $1+ $5 = $8, which is not strictly greater than $15. In this case, our total earnings are –$100 for the single play (consisting of four rolls of the die).

(For the program, x will be input when you run your code on Gradescope.)

Your job: using a simulation, estimate

1. the probability that you win (meaning your total earnings sum to strictly greater than $x), and
2. the average amount you will win if you play this game repeatedly (where the amount you win in a game could be negative).

Here is a bit more direction: you should simulate 1,000,000 plays of this game. Each simulated play involves rolling a six-sided die four times. The die is equally likely to land on any of its six sides. Your earnings are the sum of the amounts you earn for each roll, provided you make strictly greater than $x total ($1 if you roll a 1, 2, or 3, $5 if you roll a 4, $10 if you roll a 5, and $50 if you roll a 6); these should be summed each time you play. When you do not make strictly greater than $x with your four rolls, your earnings for that play are –100 dollars.

As you proceed through these 1,000,000 simulated plays, you should keep track of the number of simulations where you win: the probability of winning should be approximately equal to $\frac{\text{number of winning simulations}}{\text{total number of simulated games}}$. You should also keep track of the amount of money (in dollars) you've won per game (again, losing a $100 counts as "winning" –100 dollars), which will be used to calculate the average winnings over all 1,000,000 game plays. After completing the simulations, print out your program's estimates for both. (This is set up for you already in the provided template file and specified to print to the desired number of decimal places.)

Do not change anything provided in the template, and insert your code in the provided area (between the hashtag lines). You should use the variables provided in your code.

Specifications: your program must

- NOT ask the user for any additional input. (Ignoring the input lines already provided in the template.)
- print out an estimate of the probability that four rolls of the die result in earnings (as specified according to the rolls above) that sum to a value strictly greater than $x (represented by the variable `x` in the template file).
- also print out an estimate of the average winnings per game, where winnings for a single trial are calculated by either the sum of the earnings or –100, the latter when the sum of the four values is less than $x.
- **use 1,000,000 simulations with random numbers – no credit for theoretical solutions**. Your answers should **not** be exactly the true probability/average. (If you view the answer to more decimal places, you should **not** even get the same answers each time you run it.) If this point isn't obvious, please ask for clarification!

This problem has two invisible and six visible test cases on Gradescope. Each test case is worth one point, and two points will be manually assigned for following all specifications. Additional points may be deducted for incorrect code or code not following specifications, even if test cases pass. (Due to the nature of randomness involved in this program, I had to crudely round the final outputs, which may result in false positives on some test cases, so your code will be reviewed manually for correctness.)

Negative 100 points will be assigned to submissions that fail to fill out the collaborators/resources section.