Securin

Assessment

(the Doomed Dice challenge)

# Problem Statement ( PART A )

You have two dice - dice\_a and dice\_b.You need to find the total number of combinations possible.You also need to find the distribution of all combinations.Also you need to find the probabilities of each sum out of all combinations.

## **SOLUTION**

1. The total\_number\_of\_combinations function calculates the total outcomes when two standard dice are rolled. It multiplies the number of sides on each die (6) to find all possible combinations.
2. The distribution\_matrix function generates a matrix illustrating all possible outcomes and their frequencies when two standard dice are rolled. It uses two lists representing the dice, iterating over each possible combination, and records the sum and its frequency. This matrix visually represents the distribution of sums resulting from rolling two dice.
3. The calculate\_probabilities function computes and displays the probabilities of each possible sum when rolling two standard dice. Using two lists representing the dice, it calculates probabilities by dividing the frequency of each sum by the total possible outcomes and prints out the results.

# Problem Statement ( PART B )

Loki dooms your dice for his fun removing all the “Spots” off the dice.You have the tools to re-attach the “Spots” back on the Dice. However, Loki has doomed your dice with the following conditions: Die A cannot have more than 4 Spots on a face. Die A may have multiple faces with the same number of spots. Die B can have as many spots on a face as necessary i.e. even more than 6. But in order to play your game, the probability of obtaining the Sums must remain the same!

## **SOLUTION**

1. The problem begins by defining a function undo\_dice that takes two dice as input. These dice are represented as lists of numbers, die\_A and die\_B.
2. Inside this function, the frequency of sums of the numbers on the original dice is calculated using the frequency\_of\_sums function and stored in prob\_original.
3. A list of possible elements for the new die\_A is generated. This list contains numbers from 1 to 4, each repeated 6 times, to allow for repetition of numbers on the die.
4. All combinations of these possible elements of length equal to the length of the original die\_A are generated and stored in combinationsOfdieA\_elements.
5. A similar process is followed for die\_B, but the possible elements are numbers from 1 to 8, without repetition.
6. All combinations of these possible elements of length equal to the length of the original die\_B are generated and stored in combinationsOfdieB\_elements.
7. The function then iterates over each combination of elements for die\_A and die\_B. For each pair of combinations, it calculates the frequency of their sums using the frequency\_of\_sums function and stores this in prob\_pos.
8. If prob\_pos matches prob\_original, the function returns the current combinations for die\_A and die\_B as the new dice.
9. The function undo\_dice is then used with two dice, represented as lists of numbers from 1 to 6. The returned new dice are stored in New\_Die\_A and New\_Die\_B.

