**Problem Statement: The Doomed Dice Challenge**

**Part-A**

**Solution:**

1. Objective:

* Address three tasks related to two six-sided dice, Die\_A and Die\_B.

1. Task 1: Total Combinations

* Calculate the total combinations possible when rolling both dice.
* Formula: Total Combinations = Number of faces on Die\_A \* Number of faces on Die\_B.

1. Task 2: Combinations Distribution

* Create a 6x6 matrix representing the distribution of combinations for all possible sums.
* Utilize nested loops to iterate through all face combinations of Die\_A and Die\_B.
* Increment the corresponding matrix element based on the sum of the faces.

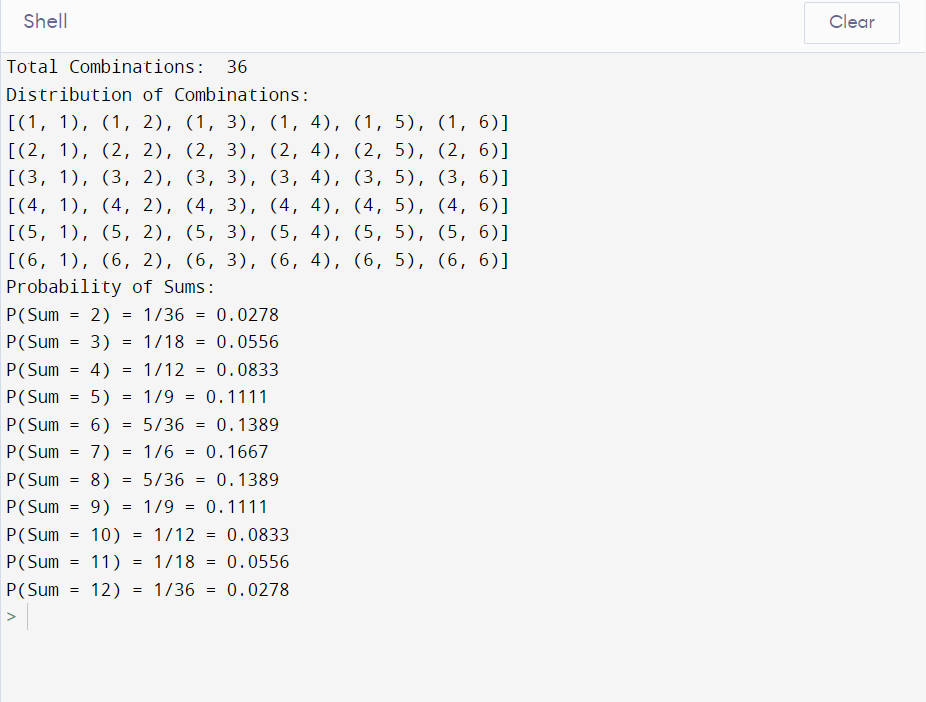
1. Task 3: Probability of Sums

* Calculate the probability of each possible sum occurring.
* Utilize the total combinations and the distribution matrix.
* Iterate through the matrix and compute the probability for each sum.
* Display the probabilities for each sum, indicating the likelihood of obtaining that specific sum.

**CODE :**

****

**OUTPUT :**

****

**Explanation :**

1. In the problem statement they have given two dice, and the goal is to calculate and analyze the initial probabilities when rolling both dice together.
2. the key components and requirements of the problem is need to calculate the total combinations, create a distribution matrix, count combinations for each sum, calculate probabilities, and analyze the distribution of sums.
3. Recall and apply basic principles of probability. For example, when rolling dice, each face on each die has an equal probability of landing face up. This knowledge forms the basis for your solution.
4. Calculate the total number of possible combinations when rolling two dice together. In this case, it's the product of the number of faces on each die. For standard six-sided dice, there are 6 x 6 = 36 total combinations.
5. Create a matrix or data structure to represent all possible combinations of Die A and Die B. Each cell in the matrix represents the sum of the faces of Die A and Die B for a specific combination.
6. Iterate through the matrix to count how many different combinations result in each possible sum (ranging from 2 to 12). by using nested loops to go through the faces of both dice and tally the counts for each sum.
7. Calculating the probability of each sum occurring. This involves dividing the number of combinations leading to a specific sum by the total number of combinations (36 in this case).
8. Analysing the distribution of sums to understand the behavior of the original dice. Determining which sums are more likely and which are less likely. This analysis provides valuable insights into the characteristics of the dice.

**Part-B**

**Solution :**

1. Start by understanding the constraints set by Loki. Die A is limited to having no more than 4 spots on a single face, and Die B can have any number of spots. The primary goal is to adjust both dice while ensuring that the probabilities for all possible sums remain unchanged.
2. Calculating the initial probabilities for Die A and Die B.These initial probabilities are essential for maintaining consistent probabilities for the sums during the adjustment.
3. Adjust Die A (New\_Die\_A):

* Begin the adjustment process for Die A:
  + Start with a copy of Die A's current configuration.
  + Sort the faces of Die A in descending order based on the number of spots on each face. This step allows you to prioritize adjustments to faces with more spots.
* For each face of Die A:
  + While a face has more than 4 spots and does not meet the target probability for its value, redistribute the excess spots to other faces. The redistribution should aim to maintain the target probabilities for the sums.
  + Continue this process until no face on Die A has more than 4 spots.

1. Adjust Die B (New\_Die\_B):

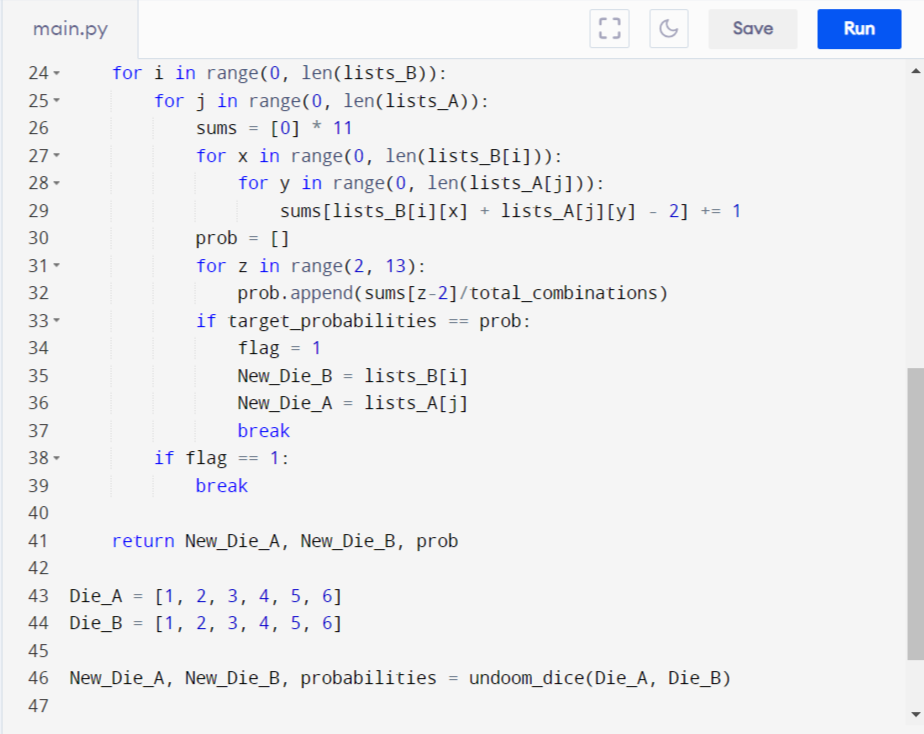
* Adjusting Die B can be challenging due to its lack of constraints on the number of spots. There are multiple approaches to adjust Die B:
  + One approach is to redistribute spots while attempting to match the target probabilities for the sums. However, this can be complex and may not always be possible to achieve, especially when Die B can have more than 6 spots on a face.
  + An alternative approach, if the redistribution is not feasible, is to keep Die B as it is (unchanged). However, this should be considered a last resort as it may not fully satisfy Loki's constraints.

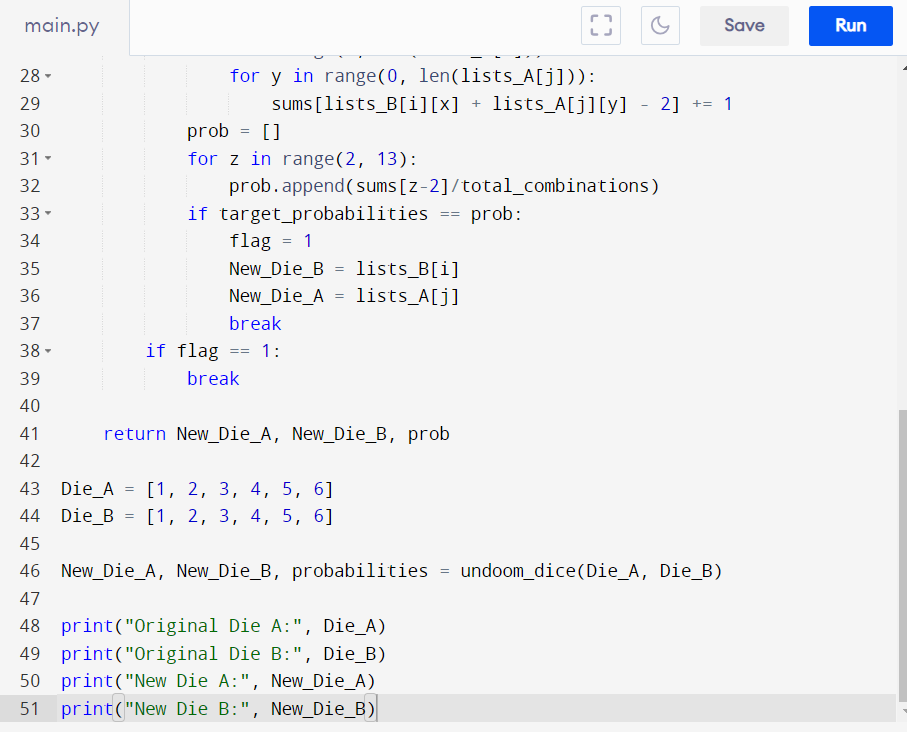
1. Throughout the adjustment process for both Die A and Die B, it is essential to ensure that the probabilities of all possible sums when rolling the adjusted dice remain consistent with the initial probabilities calculated in step 2. The key challenge is to make adjustments while preserving the same probabilities for the sums.
2. Output the Adjusted Dice (New\_Die\_A and New\_Die\_B):

* Once the adjustment process is complete, you will have the adjusted configurations for Die A (New\_Die\_A) and Die B (New\_Die\_B).
* These adjusted dice should adhere to Loki's constraints while also maintaining the same probabilities for all possible sums when rolling them together. The adjusted dice can be considered the solution to Part B.

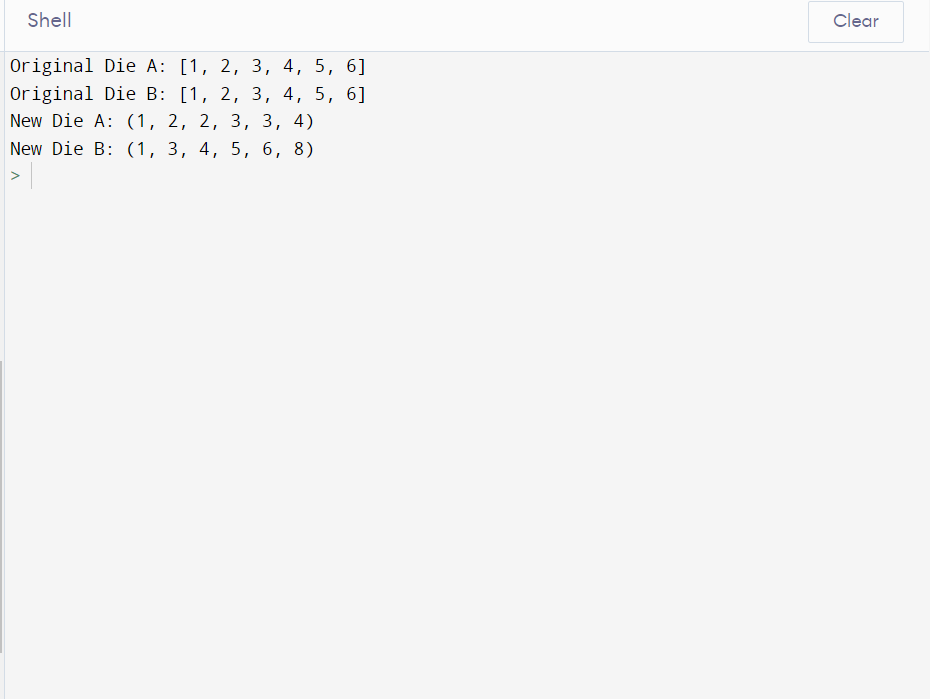
**CODE :**

****

****

****

**OUTPUT :**

****

**Explanation :**

1. Import the product and combinations functions from the itertools module.

2. Calculate the total number of combinations for two six-sided dice, storing the result in the variable total\_combinations (36).

3. Initialize a list sums with 11 elements, representing the possible sums of two six-sided dice (from 2 to 12). Each element is initialized to 0.

4. Use nested loops to iterate through all possible combinations of two six-sided dice, updating the `sums` list based on the sum of each combination.

5. Calculate the target probabilities for each possible sum (from 2 to 12) and store them in the list target\_probabilities.

6. Define a function named undoom\_dice that takes two dice (Die\_A and Die\_B) and attempts to find new dice combinations (New\_Die\_A and New\_Die\_B) that produce the same target probabilities as the original dice.

7. Inside the function, initialize a flag variable to 0 and create two empty lists, New\_Die\_A and New\_Die\_B, to store the new dice combinations.

8. Generate all possible combinations of values for Die\_A using the product function, ensuring that the values are sorted.

9. Generate all possible combinations of values for Die\_B using the combinations function.

10. Use nested loops to iterate through all combinations of Die\_A and Die\_B, updating the `sums` list based on the sum of corresponding values.

11. Calculate the probabilities for each possible sum and compare them with the target probabilities. If a match is found, set the flag to 1 and store the new dice combinations.

12. Return the new dice combinations (New\_Die\_A and New\_Die\_B) and the probabilities if a match is found; otherwise, the lists remain empty.

13. Define original dice values for Die\_A and Die\_B.

14. Call the undoom\_dice function with the original dice values and store the results in variables New\_Die\_A, New\_Die\_B, and probabilities.

15. Print the original and new dice values for both Die\_A and Die\_B.