# **The Doomed Dice Challenge Solution**

# **PART A:**

1. **How many total combinations are possible? Show the math along with the code!**   
     
   There are six possibilities for each die (1 to 6). Since we're rolling them together, each die roll contributes to the final sum independently. So, the total number of combinations is the product of the number of possibilities for each die.

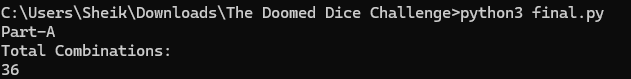
**Logic:** Total combinations = len (Die A) \* len (Die B) = 6\*6 =36.

# using the formula here

def combination Finder (length A, length\_B):

combination Count = length A \* length B

return combination Count

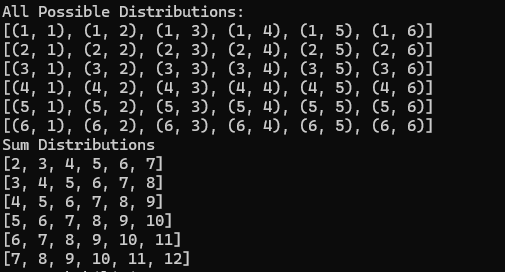


1. **Calculate and display the distribution of all possible combinations that can be obtained when rolling both Die A and Die B together. Show the math along with the code! Hint: A 6 x 6 Matrix.**

Distribution of Combinations:

We have 6 X 6 matrix.

**Logic:** Simply iterate through both dice and add their corresponding values to populate the matrix.



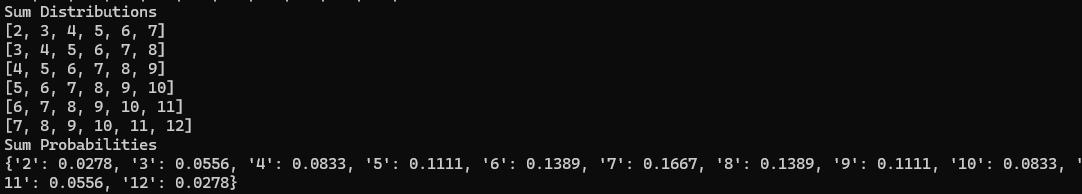
1. **Calculate the Probability of all Possible Sums occurring among the number of combinations from (2). Example: P(Sum = 2) = 1/X as there is only one combination possible to obtain Sum = 2. Die A = Die B = 1**

To calculate the probability of all possible sums occurring among the number of combinations:

**Logic:**

* Iterate through all combinations of dice rolls.
* For each combination, calculate the sum of the values on the dice.
* Count how many times each sum occurs.
* Calculate the total number of combinations by multiplying the number of outcomes for each die.
* For each sum, divide the count of occurrences by the total number of combinations to get the probability.

OUTPUT: Probabilities of all possible sums



# **PART B:**

1. **Now comes the real challenge. You were happily spending a lazy afternoon playing your board game with your dice when suddenly the mischievous Norse God Loki ( You love Thor too much & Loki didn’t like that much ) appeared.**

**Loki dooms your dice for his fun removing all the “Spots” off the dice.**

**No problem! 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!**

**So if you could only roll P(Sum = 2) = 1/X, the new dice must have the spots reattached such that those probabilities are not changed.**

**Input:**

**● Die\_A=[1,2,3,4,5,6]&DieB=Die\_A=[1,2,3,4,5,6]**

**Output:**

**● A Transform Function undoom\_dice that takes (Die\_A, Die\_B) as input & outputsNew\_Die\_A = [?, ?, ?, ?, ?, ?],New\_Die\_B = [?, ?, ?, ?, ?, ?]where,**

**● No New\_Die A[x] > 4**

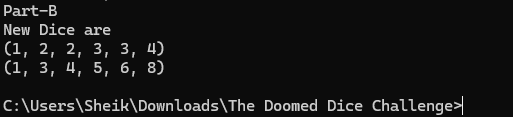
**Logic:**

The heart of the problem lies in ensuring that the total sum adds up to 42, all while adhering to Loki's restrictions. Initially, the natural inclination is to try out every possible combination of Dice A and Dice B, each with six faces ranging from 1 to 6 and 1 to 11, respectively. This brute-force approach gives us multiple lists with overlapping values, yet still manages to provide the correct answers using nested for loops.

To make things more efficient, we turn to recursion and memorization, using concepts from Dynamic Programming. This helps us store combinations and get rid of any duplicates, making our solution more streamlined and reducing the amount of time and space needed.

Once we have all the possible combinations for both dice, we calculate the probabilities of their sums and compare them to the original probabilities.

To simplify things further, we impose a condition: the total sum of all the numbers on both dice must be 42. After some careful analysis, we realize that in order to achieve this, Dice A must contain 1 and 4, while Dice B must contain 1 and 8. This reduces the number of free spots on both dice to 4, and decreases the input values needed for Dice B from 12 to 6. This significant reduction in complexity allows us to solve the problem much more efficiently.

****