**The Doomed Dice challenge**

**Problem Statement:**

The doomed dice challenge. Given two dice, Dice A and Dice B each with six sides numbered from 1 to 6. Both the dice are rolled together.

**PART A**

1. **How many total combinations are possible? Show the math along with the code.**

**Explanation:**

Formula to calculate total combinations possible: (Number of faces)^(Number of Dice)

In the given problem Number of faces = 6 and the Number of Dice = 2. so, 6^2 = 36.I have used Python programming to code the given problem. Firstly initialize an empty dictionary sum\_count to keep track of the count of each possible sum of two dice. Then initialize an empty list of combinations to store all possible combinations of two dice rolls. Next, I used nested loops to iterate over all possible outcomes of two dice rolls in which the outer loop represents the result of the first die, and the inner loop represents the result of the second die. For each combination, I have created a list containing the outcomes of both dice and append it to the combinations list. Finally, Update the sum\_count dictionary to keep track of the occurrences of each possible sum. If the sum is not already a key in the dictionary, add it with a count of 1. If it's already a key, increment the count.

**Code:**

faces = [1, 2, 3, 4, 5, 6]

total = len(faces) \*\* 2

print("Total Combinations:", total)

sum\_count={}

combinations = []

for i in range(1,7):

for j in range(1,7):

combinations.append(["Die A:" + str(i) ,"Die B:"+str(j)])

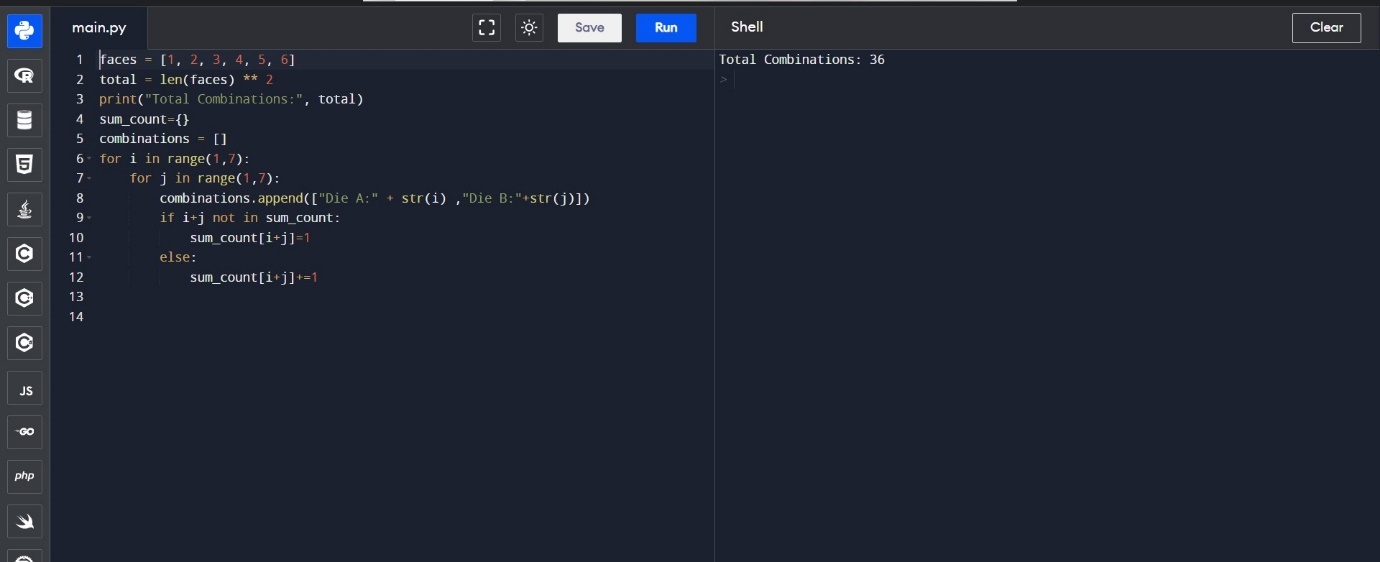
if i+j not in sum\_count:

sum\_count[i+j]=1

else:

sum\_count[i+j]+=1

**Output:**



1. **Calculate and display the distribution of all possible combinations that can be obtained when rolling both Die A and Die B together.**

**Explanation:**

NumberOfFaces=6

TotalCombinations=36

Firstly I have defined a list of faces containing the possible outcomes of a six-sided die. Then I calculated and printed the total number of combinations when rolling two six-sided dice. Since each die has 6 faces, there are 6×6=36 possible combinations. The print statement introduces the next output section. Next, I initialized an empty dictionary sum\_count to keep track of the count of each possible sum of two dice and an empty list of combinations to store all possible combinations of two dice rolls. I have used nested loops to iterate over all possible outcomes of two dice rolls in which the outer loop represents the result of the first die, and the inner loop represents the result of the second die. For each combination, I have created a list containing the outcomes of both dice and append it to the combinations list. Updated the sum\_count dictionary to keep track of the occurrences of each possible sum. If the sum is not already a key in the dictionary, add it with a count of 1. If it's already a key, increment the count. The final loop prints the combinations in a structured way. It uses the range function to iterate over indices in steps of the length of faces. It then prints slices of the combinations list, effectively grouping them by the result of the first die. This gives you a structured output showing all combinations for each value of the first die.

**Code:**

faces = [1, 2, 3, 4, 5, 6]

total = len(faces) \*\* 2

print("\nCombinations Distribution:")

sum\_count={}

combinations = []

for i in range(1,7):

for j in range(1,7):

combinations.append(["Die A:" + str(i) ,"Die B:"+str(j)])

if i+j not in sum\_count:

sum\_count[i+j]=1

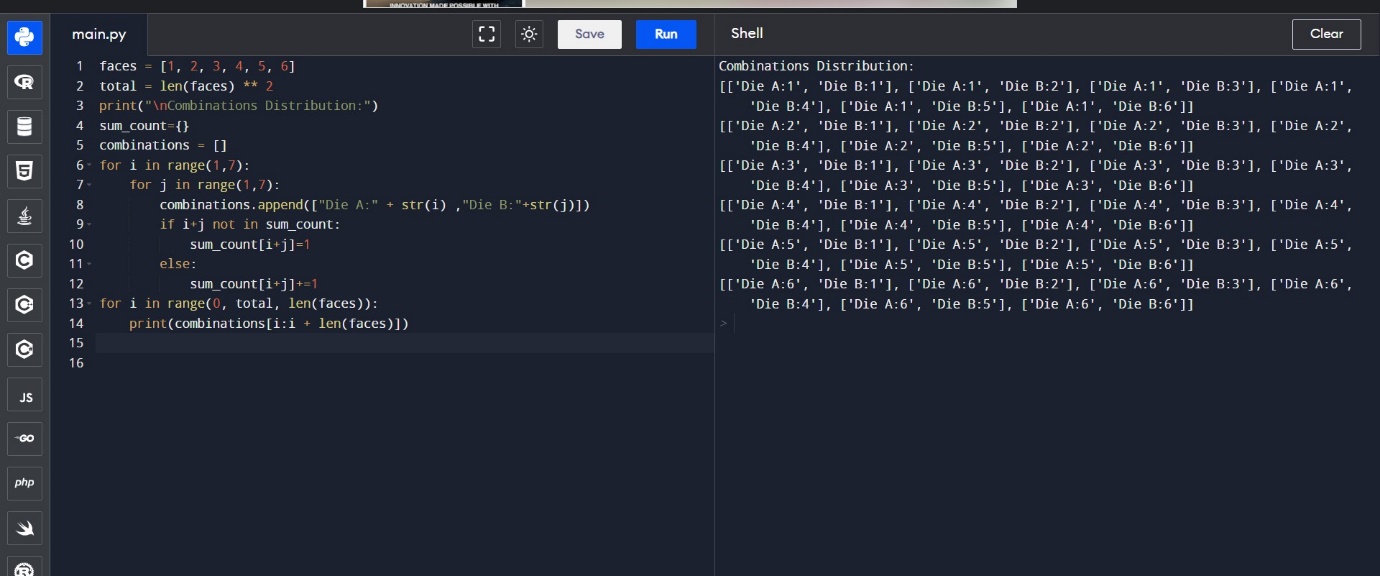
else:

sum\_count[i+j]+=1

for i in range(0, total, len(faces)):

print(combinations[i:i + len(faces)])

**Output:**



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.**

**Explanation:**

The first print statement introduces the next output section. Last for loop calculates and prints the probability of each possible sum from 2 to 12. It uses the counts stored in the sum\_count dictionary and divides them by the total number of combinations to obtain the probability. The results are printed in a formatted way, showing the probability for each sum. The probability is calculated as the count of occurrences of a sum divided by the total number of combinations. The: .2f in the formatting string ensures that the probability is displayed with two decimal places.

**Code:**

faces = [1, 2, 3, 4, 5, 6]

total = len(faces) \*\* 2

print("\nProbability of Sums:")

sum\_count={}

combinations = []

for i in range(1,7):

for j in range(1,7):

combinations.append(["Die A:" + str(i) ,"Die B:"+str(j)])

if i+j not in sum\_count:

sum\_count[i+j]=1

else:

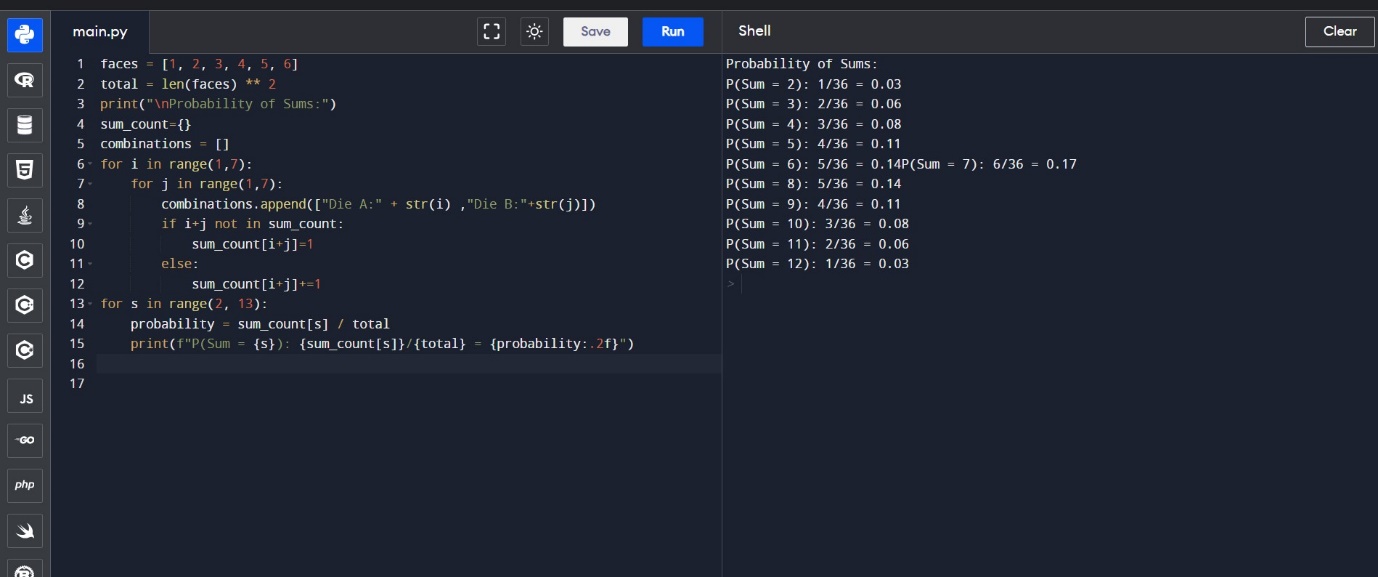
sum\_count[i+j]+=1

for s in range(2, 13):

probability = sum\_count[s] / total

print(f"P(Sum = {s}): {sum\_count[s]}/{total} = {probability:.2f}")

**Output:**



**PART B**

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] & Die B = Die\_A = [1, 2, 3, 4, 5, 6]

Output:

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

● No New\_Die A[x] > 4

**Explanation:(Question)**

The problem statement portrays a situation where Loki has eliminated the spots from your dice, and you want to reattach the spots to both die A and die B. Loki gives some conditions. That includes:

* 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, even more than 6.

The challenge is to reattach the spots so that the likelihood of acquiring the aggregates while throwing the dice continues as before.

**Explanation:(Code)**

scaling\_factor: It is calculated as the sum of spots on die A divided by the sum of spots on die B. This scaling factor is used to adjust the values of die B based on the values of die A.

x: It initializes an empty list and then iterates through each element in die A, appending the minimum of that element and 4 to the list x. This ensures that die A has no more than 4 spots on the face.

y: It initializes another empty list and then iterates through each element in die B. For each element, it calculates a scaled value by multiplying it with the scaling factor, rounding to the nearest integer, and taking the minimum of that value and 6. This ensures that die B can have as many spots as necessary but no more than 6.

The function then returns the modified dice lists x and y. Then I defined two lists die\_a and die\_b representing the faces of two dice. Call the undoom\_dice function with the die\_a and die\_b lists and unpack the returned values into new\_die\_a and new\_die\_b. A. Finally, print the changed dice records new\_die\_a and new\_die\_b.

**Code:**

def undoom\_dice(die\_a, die\_b):

scaling\_factor = sum(die\_a) / sum(die\_b)

x=[]

for spots in die\_a:

x.append(min(4,spots))

y=[]

for spots in die\_b:

y.append(min(6,round(spots\*scaling\_factor)))

return x, y

die\_a = [1, 2, 3, 4, 5, 6]

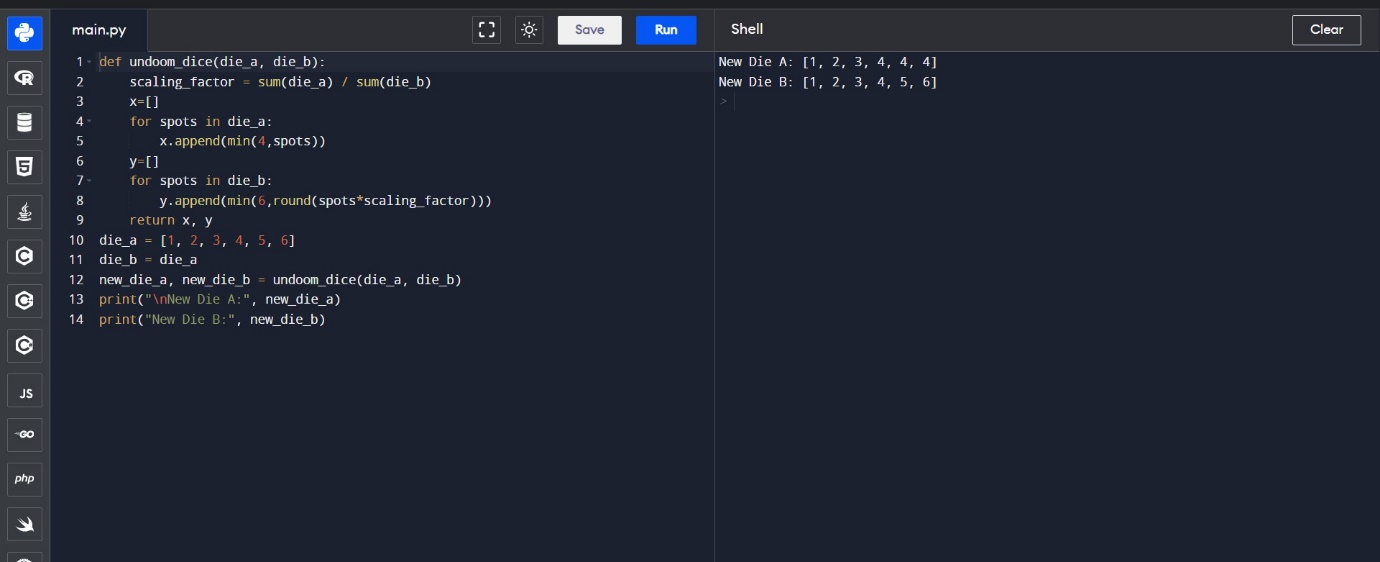
die\_b = die\_a

new\_die\_a, new\_die\_b = undoom\_dice(die\_a, die\_b)

print("\nNew Die A:", new\_die\_a)

print("New Die B:", new\_die\_b)

**Output:**



**GitHub Link**: <https://github.com/mode20230/Securin>