Problem Statement: The Doomed Dice Challenge

Part-A:

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

- **LOGIC TO THE PROBLEM STATEMENT**

  Since there are two six-faced dice (die A and die B) numbered from 1-6, every number from die A can occur with every number from die B. This would mean "1" from die A can occur with numbers 1-6 from die B leading to 6 combinations with the number "1" on die A. Thus, applying the same logic, there will be six occurrences of each of the six numbers in die A.

  This will lead to having:
  6*6=36 possibilities
  (6 occurrences of die A) *(6 occurrences of die B for each occurrence of A)

- **HOW I CAME UP WITH THE SOLUTION**

  Since it was given that each of the dice was six faced, every number can be paired up with every other number. Using this logic, I used a function to return the product of the number of faces on the dice. If the dice have different number of faces, the variables can be changed accordingly, or taken as input.

2. 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.

- **LOGIC TO THE PROBLEM STATEMENT**

  From the previous problem, we know that every side of die A occurs with every side of die B, the combinations will include:

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

  The above table consists of all the numbers in the form [dieA, dieB]. Every number of die A can occur with every number of die B.

- **HOW I CAME UP WITH THE SOLUTION**

  Since the basic idea behind generating this matrix is to ensure that all the numbers of die A occur at least once with all the elements of die B, I iterated over the range from 1

to 6 (both inclusive) for die A and die B, and appended the combinations to a 2-D matrix.

3. 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

- **LOGIC TO THE PROBLEM STATEMENT**

  There is one major observation that came from the matrix ie., the minimum possible sum of the dice is 2, which comes from [1,1] and the maximum possible sum of the dice is 12, which comes from [6,6].
  Thus, by iterating over all the elements in the array, and counting the sum of every element, we will get the count of elements that sum up to the numbers from 2-12.

  Additionally, since we already know the total number of combinations, the probability of getting that sum in the matrix will be:

  Number of occurrences of that sum/total number of combinations

- **HOW I CAME UP WITH THE SOLUTION**

  I realized that the best way to calculate the sum probabilities was to first count the number of occurrences of each of the sums and divide them with the total combinations and store them in an array. Based on previous observations, the sums range from 2 to 12. I used a dictionary to store the probabilities of all possible sums that occur in the 2-D array. I obtained the count by iterating over the array and adding up the numbers in each combination.

Part-B:

- **LOGIC TO THE PROBLEM STATEMENT**

  The high-level understanding of the logic is that all the possible combinations of the two dice are generated, followed by calculating their probabilities and comparing them with the original probabilities. When a combination was found where the probabilities were exactly the same, the function returned that as the new pair of dice.

  Few things to keep in mind are:
  - Sums only range from 2 to 12.
  - If I fix the minimum and the maximum values in die A as 1 and 4, we must have 1 and 8 as minimum and maximum possible values in the second die.
  - This stems from the fact that the sums 2 and 12 have probabilities (1/36) and we cannot have a value higher than 12 as the sum.
  - By using this logic, I generated combinations:

    First die: [1]+ list(itertools.product([2,3], repeat=4)) + [4]
    Second die: list(itertools.combinations([1,2,3,4,5,6,7,8], 6))

To understand the logic of limiting the dice in the way described above, better, we can start with the extremes:

- We must have 1 in both the arrays in order to achieve the sum 2. Thus, 1 must exist once in both the dice. (as 0+2 or 2+0 is not possible)
- We can have the following combinations that add up to 12:
  0+12 (Having 12 on die B would restrict the first array to only have 0 in it as the range of the sum is 2-12. The probability of 12 becomes 6/36, so not possible)
  1+11 (Having 11 on die B would restrict the first array to 1 being maximum so at least a 1 or a 0 must repeat. This would violate either sum 2's probability or sum 12's probability)
  2+10 (Having 10 on die B would restrict the first array to 2 being maximum so at least a 1 or a 2 must repeat. This would violate either sum 2's probability or sum 12's probability)
  3+9 (Having a 9 on die B would restrict the first array to 3 being maximum. If 3 occurs more than once, 12's probability gets violated. If 1 occurs more than once, 2's probability gets violated and if 2 occurs more than twice, 3's probability gets violated so not possible)
  4+8 (Having 4 and 8 as the maximum values on the two dice doesn't violate the sums at the extremes. Thus, we start generating all the combinations of these range of numbers)

Functions:

- calculate_probabilities(dieA, dieB) – Used the same function as part A to calculate the sum probabilities. Returns the sum probability distributions of the given dice.
- is_probability_preserved(newA, newB, total_outcomes, possible_sums) – Compared all the probabilities of the original with the new probabilities. Returns if the probabilities of the new dice are the same as the original dice.
- undoom_dice(dieA, dieB) – Generates all the permutations of die A (range 1 to 4 inclusive) and die B (range 1 to 8 inclusive) and calls the above functions to calculate probabilities and check if they are the same. Returns the new dice.

- **HOW I CAME UP WITH THE SOLUTION**

  I first ran a brute force program where I tried to generate all the combinations of the two dice with ranges [1,4] for die A and [1,11] for die B. This yielded a time taking output. I then realized that the extremes i.e., 1 and 12 have probabilities 1/36. I then started checking if that would help me add constraints to the output.
  Thus, I figured on pen and paper that the combinations can be limited to just [2,3] and [1,8]. Reducing the range of die B from 11 to 8 significantly changed the time complexity because the time complexity is O (n choose k) and in this case n changes from 11 to 8.