THE DOOMED DICE CHALLENGE

Google colab link:- [SECURIN ASSESSMENT ANSWERS- SHARON WILSON](https://colab.research.google.com/drive/1aUeoOcX01cTO_LO4kiGp89WbbaNGHf7d?usp=sharing)

PART A-

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

To find the total possible combinations while rolling two dies, die A and B, we can use the cartesian product method to calculate.

The formula to find the total combinations using cartesian product method is:

**Totalcombns = number of die faces in A \* number of die face in B.**

**IN PYTHON :**

**These are some of the approaches to find the total combinations:**

**dieA=[1,2,3,4,5,6]**

**dieB=[1,2,3,4,5,6]**

**totalcombns=len(dieA)\*len(dieB)**

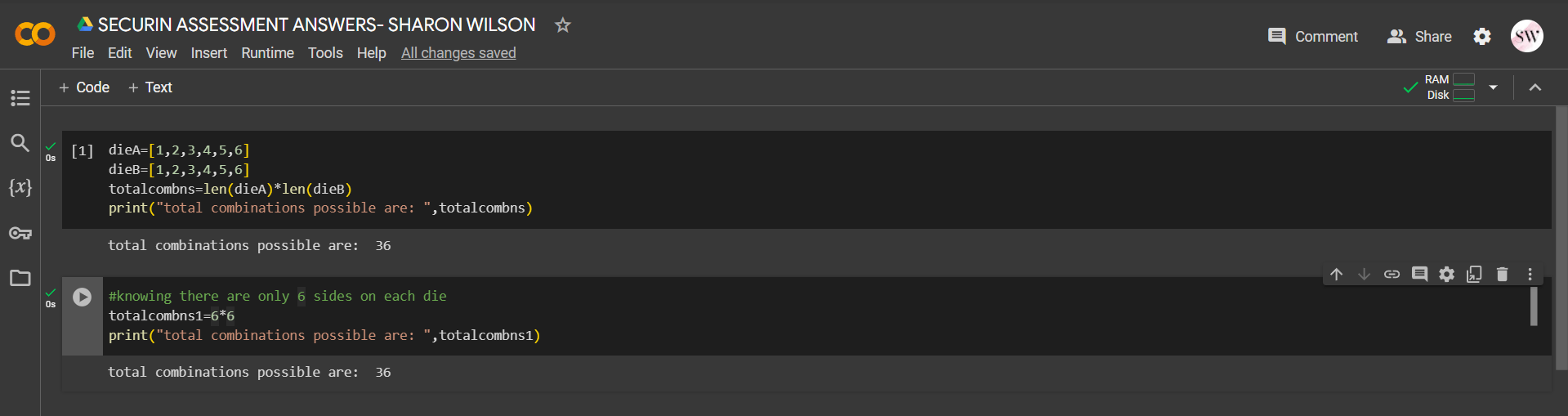
**print(“total combinations possible are:”,totalcombns)**

**—-----------------------------------------------------------------------------------------------------**

**//second approach is taking the given information that each die has only 6 faces**

**totalcombns1=6\*6**

**print(“total combinations possible are:”,totalcombns1)**



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

For the 6 x 6 matrix the distribution of all the possible combinations will be as follows:

[(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)]

**In python, the above matrix can be obtained by**

**// Assuming one set of die faces for calculation first and using list comprehension method**

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

**distribution\_matrix = [[0 for \_ in range(6)] for \_ in range(6)]**

**distribution\_matrix = [[i + j for j in die\_faces] for i in die\_faces]**

**for row in distribution\_matrix:**

**print(row)**

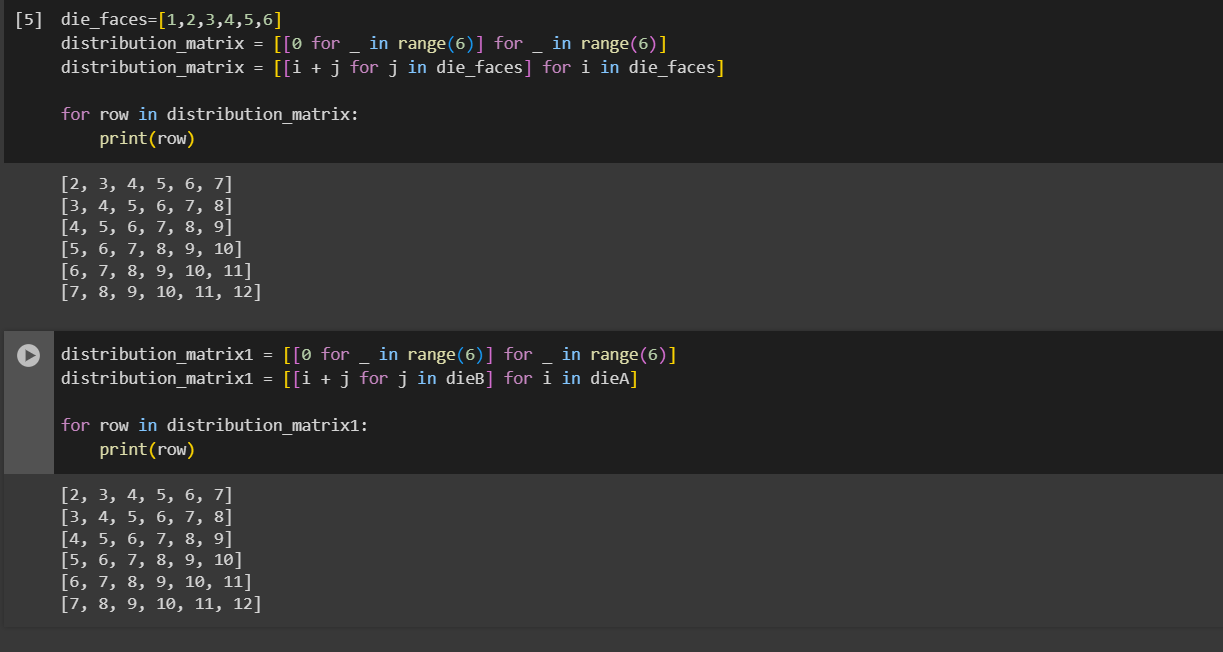
**// taking the same two die faces array of dieA and dieB**

**distribution\_matrix1 = [[0 for \_ in range(6)] for \_ in range(6)]**

**distribution\_matrix1 = [[i + j for j in dieB] for i in dieA]**

**for row in distribution\_matrix1:**

**print(row)**



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.

To calculate the probability of each possible sum, we need to divide the number of combinations that result in a specific sum by the total number of combinations. The formula for calculating the probability is as follows:

totalpossibilities= (number of combinations resulting in sum s) / (total number of combinations)

The python code to solve this is as follows:

**Without using any libraries the probabilities can be calculated as :**

**for i in range(2, 13):**

**combinations = sum(k.count(i) for k in distribution\_matrix1)**

**#probability = combinations / len(die\_faces)\*\*2**

**print(f"P (sum={i}) =({combinations}/{len(die\_faces)\*\*2})")**



**Using the fractions library to further simplify the above probabilities, we get:**

**from fractions import Fraction**

**for i in range(2, 13):**

**num\_combinations = sum(k.count(i) for k in distribution\_matrix1)**

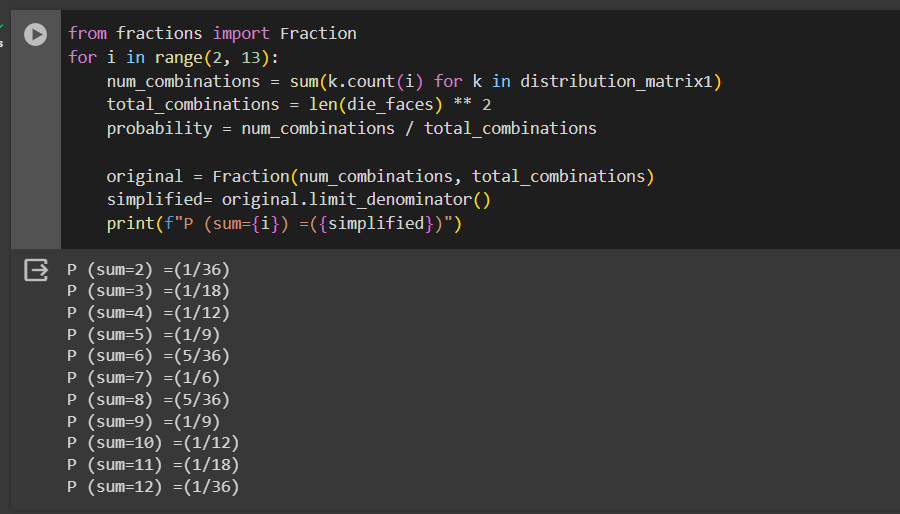
**total\_combinations = len(die\_faces) \*\* 2**

**probability = num\_combinations / total\_combinations**

**original = Fraction(num\_combinations, total\_combinations)**

**simplified= original.limit\_denominator()**

**print(f"P (sum={i}) =({simplified})")**



PART B:

To find the undoomed dice we shall use the following code with given conditions from the question:

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

**new\_die\_a = [min(4, spots) for spots in die\_a]**

**new\_die\_b = die\_b**

**return new\_die\_a, new\_die\_b**

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

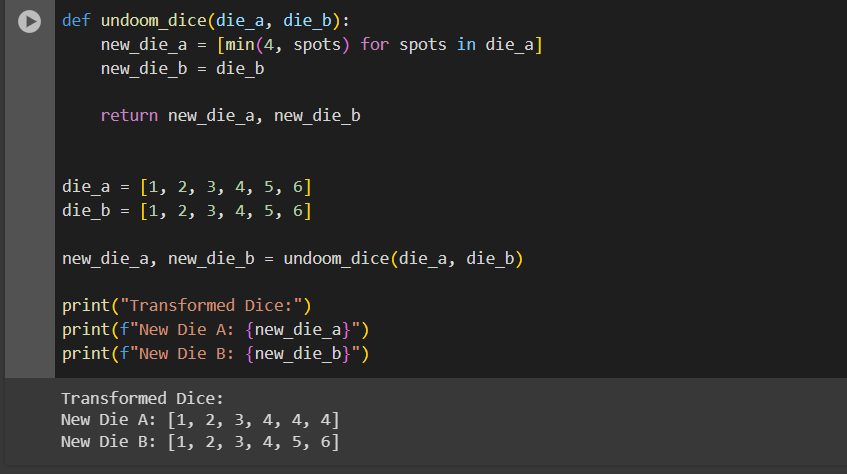
**die\_b = [1, 2, 3, 4, 5, 6]**

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

**print("Transformed Dice:")**

**print(f"New Die A: {new\_die\_a}")**

**print(f"New Die B: {new\_die\_b}")**



COMPLETE CODE FOR THE ABOVE QUESTIONS IS AS FOLLOWS

**print("PART - A")**

**print("--------------------------------------------------------------")**

**print("question 1")**

**dieA=[1,2,3,4,5,6]**

**dieB=[1,2,3,4,5,6]**

**totalcombns=len(dieA)\*len(dieB)**

**print("total combinations possible are: ",totalcombns)**

**print("--------------------------------------------------------------")**

**print("question 2")**

**distribution\_matrix1 = [[0 for \_ in range(6)] for \_ in range(6)]**

**distribution\_matrix1 = [[i + j for j in dieB] for i in dieA]**

**for row in distribution\_matrix1:**

**print(row)**

**print("--------------------------------------------------------------")**

**print("question 3")**

**for i in range(2, 13):**

**combinations = sum(k.count(i) for k in distribution\_matrix1)**

**#probability = combinations / len(die\_faces)\*\*2**

**print(f"P (sum={i}) =({combinations}/{len(die\_faces)\*\*2})")**

**print("--------------------------------------------------------------")**

**print("PART - B")**

**print("--------------------------------------------------------------")**

**print("question 1")**

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

**new\_die\_a = [min(4, spots) for spots in die\_a]**

**new\_die\_b = die\_b**

**return new\_die\_a, new\_die\_b**

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

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

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

**print("Transformed Dice:")**

**print(f"New Die A: {new\_die\_a}")**

**print(f"New Die B: {new\_die\_b}")**

