# Task 1: Total Number of Combinations

total\_combinations = 6 \* 6

total\_combinations = 6 \* 6: Calculates the total number of combinations when rolling two six-sided dice.

# Task 2: Distribution of Combinations

from collections import Counter  
# Generate all possible pairs of dice rolls and count occurrences

distribution = Counter([(a, b) for a in range(1, 7) for b in range(1, 7)])

distribution = Counter([(a, b) for a in range(1, 7) for b in range(1, 7)]): Generates all possible pairs of dice rolls and uses Counter to count the occurrences of each pair.

# Task 3: Probability of Sums

# Calculate the probability of each sum

sum\_probabilities = Counter([sum(pair) for pair in distribution])

for sum\_val in sum\_probabilities:

sum\_probabilities[sum\_val] /= total\_combinations

sum\_probabilities = Counter([sum(pair) for pair in distribution]): Calculates the sum for each pair in the distribution and uses Counter to count the occurrences of each sum.

for sum\_val in sum\_probabilities: sum\_probabilities[sum\_val] /= total\_combinations: Converts the counts to probabilities by dividing each count by the total number of combinations.

# Displaying results for Part A

total\_combinations, distribution, sum\_probabilities

This line displays the total number of combinations, the distribution of combinations, and the probabilities of sums.

from collections import Counter

def undoom\_dice():

# Original Dice Configurations

die\_a\_original = list(range(1, 7))

die\_b\_original = list(range(1, 7))

die\_a\_original: Represents the original configuration for Die A with spots ranging from 1 to 6.

die\_b\_original: Represents the original configuration for Die B with spots ranging from 1 to 6.

# New Dice Configurations

die\_a\_new = [1, 2, 3, 4, 3, 2] # Restrict Die A to no more than 4 spots

die\_b\_new = [1, 2, 3, 4, 5, 8] # Adjust Die B accordingly

die\_a\_new = [1, 2, 3, 4, 3, 2]: New configuration for Die A, limiting it to no more than 4 spots on any face.

die\_b\_new = [1, 2, 3, 4, 5, 8]: New configuration for Die B, adjusting it accordingly.

# Validate that the sum probabilities are maintained

original\_sums = Counter(a + b for a in die\_a\_original for b in die\_b\_original)

new\_sums = Counter(a + b for a in die\_a\_new for b in die\_b\_new)

original\_sums: Counter for the sum probabilities using the original configurations.

new\_sums: Counter for the sum probabilities using the new configurations.

# Check if the distributions are the same

if original\_sums == new\_sums:

return die\_a\_new, die\_b\_new

else:

return None, None # In case the distributions do not match

Compares the original and new sum probability distributions.

If they match, the function returns the new configurations; otherwise, it returns None

undoomed\_die\_a, undoomed\_die\_b = undoom\_dice()

undoomed\_die\_a, undoomed\_die\_b

Calls the undoom\_dice function and prints the undoomed configurations for Die A and Die B.

# The result (None, None) indicates that the function undoom\_dice was unable to find new configurations for Die A and Die B that maintain the same sum probabilities as the original dice within the specified constraints and search range.