def calculate\_probabilities(die):

# Calculate probabilities for each sum using the given die

probabilities = {}

for i in range(1, len(die) + 1):

for j in range(1, len(die) + 1):

total = i + j

probabilities[total] = probabilities.get(total, 0) + 1

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

for key in probabilities:

probabilities[key] /= total\_combinations

return probabilities

def undoom\_dice(Die\_A, Die\_B, target\_probabilities):

# Calculate original probabilities for each sum

original\_probabilities = calculate\_probabilities(Die\_A)

# Transform Die\_B to maintain sum probabilities

# (Assuming Die\_B is allowed to have more than 6 spots)

# Here, I'm just copying Die\_B since no restrictions were imposed on it

New\_Die\_B = Die\_B

# Adjust Die\_A to maintain the provided sum probabilities

New\_Die\_A = []

for face in Die\_A:

# Ensure no face has more than 4 spots

if face > 4:

New\_Die\_A.append(4) # If face > 4, set it to 4

else:

New\_Die\_A.append(face)

# Check if the transformed dice maintain the provided probabilities

new\_probabilities = calculate\_probabilities(New\_Die\_A)

for key in target\_probabilities:

if abs(new\_probabilities[key] - target\_probabilities[key]) > 0.0001:

return None # Unable to find a transformation meeting the conditions

return New\_Die\_A, New\_Die\_B

# Input

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

Die\_B = [1, 2, 3, 4, 5, 6]

# Provided target probabilities

target\_probabilities = {

2: 0.0278,

3: 0.0556,

4: 0.0833,

5: 0.1111,

6: 0.1389,

7: 0.1667,

8: 0.1389,

9: 0.1111,

10: 0.0833,

11: 0.0556,

12: 0.0278

}

# Output

result = undoom\_dice(Die\_A, Die\_B, target\_probabilities)

if result:

New\_Die\_A, New\_Die\_B = result

print("New Die A:", New\_Die\_A)

print("New Die B:", New\_Die\_B)

else:

print("Unable to transform the dice while maintaining the provided sum probabilities.")