Problem statement:

Given that there are essentially two dice, dice A and dice B can be taken as [1,2,3,4,5,6] in the grid matrix.

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

So, now we need to find the count of total number of possibilities

Two methods can be used for this solution.

1.By using 2 for loops to count all the possibilities: -

```
def total():
    die_a=[1,2,3,4,5,6]
    die_b=[1,2,3,4,5,6]
    count=0
    for i in die_a:
        for j in die_b:
            count+=1
    return count
```

2.This can also be done by squaring the number of possibilities of a single dice has as both have same number of sides.

Output:

```
def total():
    die_a=[1,2,3,4,5,6]
    die_b=[1,2,3,4,5,6]
    return len(die_a)*len(die_b)
```

```
------ RESTART: C:/Users/Roush
Total Combinations: 36
```

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

We need to find the combinations, which can be achieved by using the nested for loop.

Formula: $\text{Matrix Element}[i][j] = [\text{DieA}[i], \text{DieB}[j]]$

This creates a matrix where each row represents a value of DieA and each column represents a value of DieB

```
def combos():
    die_a=[1,2,3,4,5,6]
    die_b=[1,2,3,4,5,6]
    combo=[]
    for i in die_a:
      k=[]
      for j in die_b:
          k.append([i,j])
      combo.append(k)
    return combo
combos()
```

Output:

```
All Possible combinations:

[[[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]]]
```

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.

Now we need to find the probability of all possible sums where the minimum sum that can be possible is 2 by having 1 on dice A and 1 on dice B and has the maximum of 12 by having 6 on dice A and 6 on dice B.

In order to do so, the mathematical formula is:

- Formula for probability of sum s: $P(s) = \frac{\text{Number of ways to get sum } s}{\text{Total number of combinations}}$

```python
numbers = [1, 2, 3, 4, 5, 6]
total = len(numbers) ** 2
print("Total Combinations:", total)
combos = [(i, j) for i in numbers for j in numbers]
print("\nCombinations Distribution:")
for i in range(0, total, len(numbers)):
    print(combos[i:i + len(numbers)])

print("\nProbability of Sum taken:")
for i in range(2, 13):
    count = sum(1 for j in combos if sum(j) == i)
    probability = count / total
    print(f"P(Sum = {i}): {count}/{total} = {probability:.2f}")
```

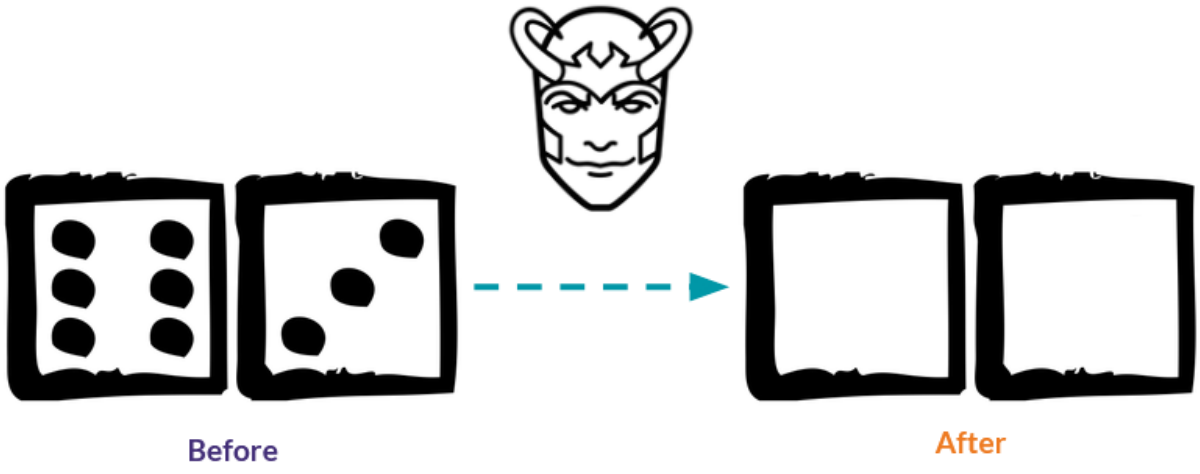
Output:

```
Probability of Sums:
P(Sum = 2): 1/36 = 0.03
P(Sum = 3): 2/36 = 0.06
P(Sum = 4): 3/36 = 0.08
P(Sum = 5): 4/36 = 0.11
P(Sum = 6): 5/36 = 0.14
P(Sum = 7): 6/36 = 0.17
P(Sum = 8): 5/36 = 0.14
P(Sum = 9): 4/36 = 0.11
P(Sum = 10): 3/36 = 0.08
P(Sum = 11): 2/36 = 0.06
P(Sum = 12): 1/36 = 0.03
```

Part B

Everything was running fine but suddenly the cunning Norse god Loki materialised out of nowhere.

For his own amusement, Loki destroys all of the "Spots" on your dice, dooming it.



To reattach the spots, we have some rules

- **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.**

In the process of reattaching spots to the dice faces, specific rules are employed to ensure uniform probabilities. For dice A, it is stipulated that no value should exceed 4. To adhere to this condition, any value greater than 4 undergoes a subtraction of 3.

This adjustment is determined through trial and error, guaranteeing an equal likelihood of repeated numbers on dice A. For instance, if a dice A value is initially 5, subtracting 3 results in 2, effectively introducing a repetition of the value. Conversely, for dice B, the objective is to allow both repeated values and values surpassing 6. To achieve this, even values are selected from the entire collection, and each is incremented by 2.

This modification ensures that the probability distribution remains unchanged, maintaining the desired equilibrium without compromising the inherent probabilities.

```
def undoom_dice(die_a, die_b):
    scaling_factor = sum(die_a) / sum(die_b)
    a = [min(4, spots) for spots in die_a]
    b = [min(6, round(spots * scaling_factor)) for spots in die_b]
    return a, b
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:

```
 Modified Values of the dice

New Die A: [1, 2, 3, 4, 4, 4]
New Die B: [1, 2, 3, 4, 5, 6]
```