**Securein**

**Github Link :** [**Link**](https://github.com/vlokesh08/Securein-The-Doomed-Dice-Challenge)

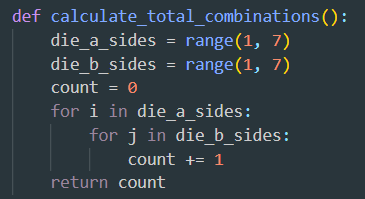
**Problem statement:**

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 exist for locating it.

We can quickly determine it by manual calculation of each possibility by taking out both the dices and this can be done by using 2 nested loops.

The another way of doing this is squaring the number of possibilities a single dice has.

Output:

2. Calculate and display the distribution of all possible combinations that can be

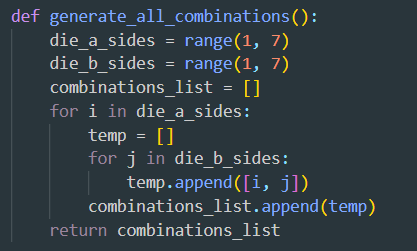
obtained when rolling both Die A and Die B together. Show the math along with

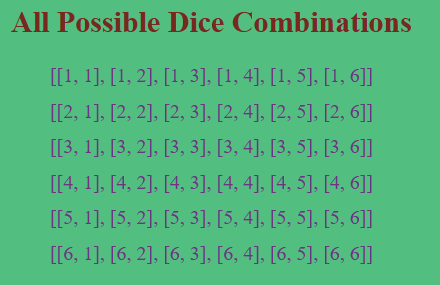
the code!

Hint: A 6 x 6 Matrix

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



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

Output:

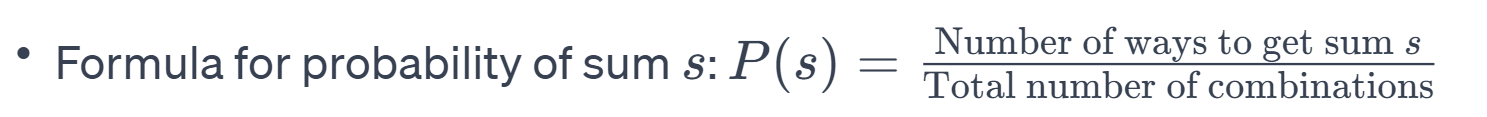
3) Calculate the Probability of all Possible Sums occurring among the number of combinations from (2).

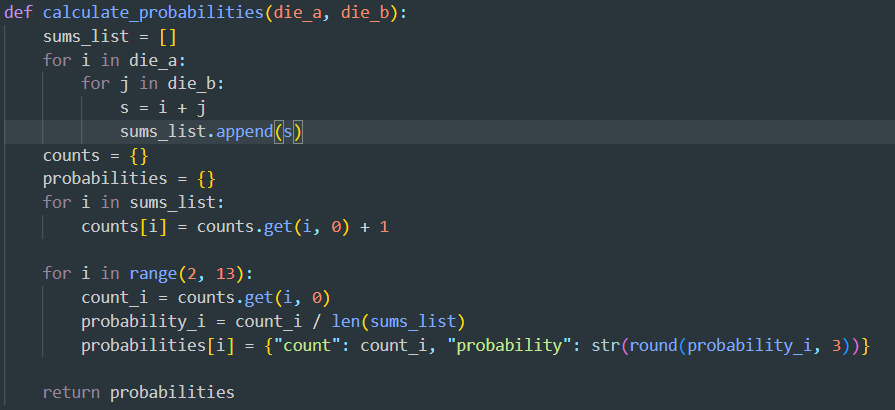
Example: P(Sum = 2) = 1/X as there is only one combination possible to obtain

Sum = 2. Die A = Die B = 1.

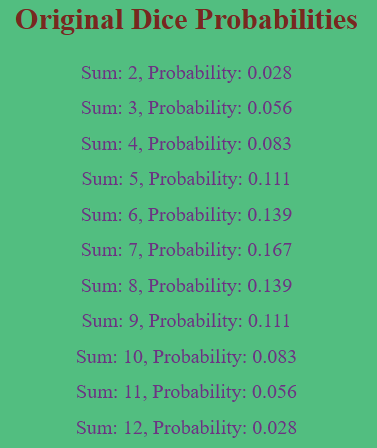
Now we need to find the probability of all possible sums where the minimum sum that can be possible is 2 by having 1 on dice A and 1 on dice B and has the maximum of 12 by having 6 on dice A and 6 on dice B.

In order to do so, the mathematical formula is:

****

****

**Output:**

****

**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.**

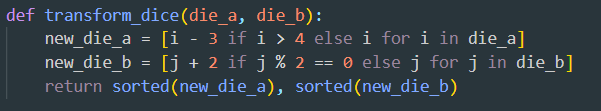
**● Die B can have as many spots on a face as necessary i.e. even more than 6.**

So we are going reattach all the spots by using these rules

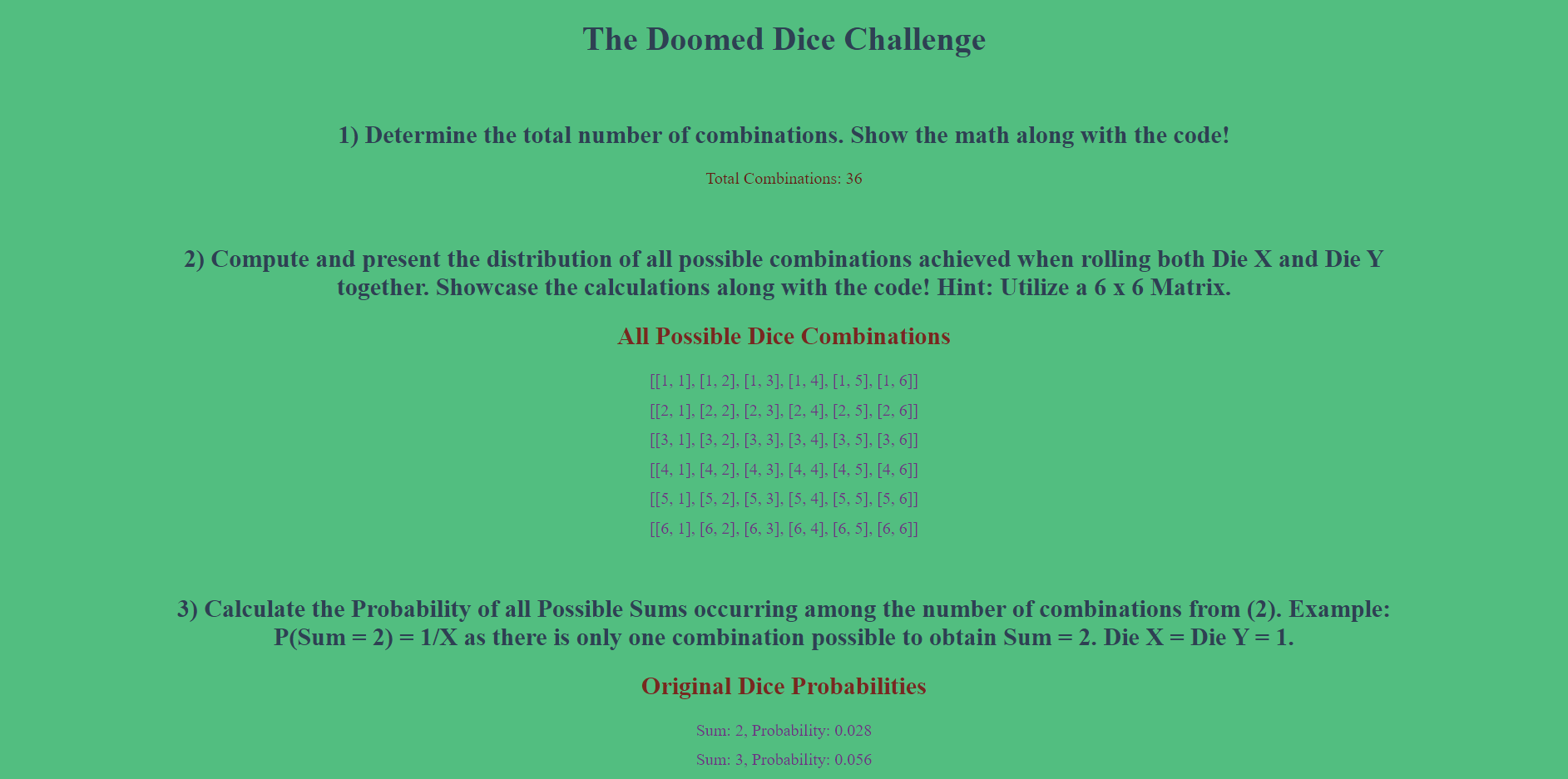
According to the rule dice A should not be greater than 4. If the value in dice A is > 4 then we will subtract the value by 3 (we are using 3 by trail and error method so that all the probabilities will be same) by doing like this we can satisfy 2 rules of dice A i.e. same number of spots can be repeated.

For example if the dice value is 5 we will subtract it with 3 then the value becomes 2 that means 2 came twice hence the values are repeated.

Similarly for dice B it can have values repeated and also can exceed 6 so taking the even values from all the collection and changing them by adding 2 to them so that the probability will be maintained same without any alteration.



These were made as Full Stack Application and the Outputs for this are





Flask Framework is been used for this application and the code for this were

from flask import Flask, render\_template

app = Flask(\_\_name\_\_)

def calculate\_total\_combinations():

    die\_a\_sides = range(1, 7)

    die\_b\_sides = range(1, 7)

    count = 0

    for i in die\_a\_sides:

        for j in die\_b\_sides:

            count += 1

    return count

def generate\_all\_combinations():

    die\_a\_sides = range(1, 7)

    die\_b\_sides = range(1, 7)

    combinations\_list = []

    for i in die\_a\_sides:

        temp = []

        for j in die\_b\_sides:

            temp.append([i, j])

        combinations\_list.append(temp)

    return combinations\_list

def calculate\_probabilities(die\_a, die\_b):

    sums\_list = []

    for i in die\_a:

        for j in die\_b:

            s = i + j

            sums\_list.append(s)

    counts = {}

    probabilities = {}

    for i in sums\_list:

        counts[i] = counts.get(i, 0) + 1

    for i in range(2, 13):

        count\_i = counts.get(i, 0)

        probability\_i = count\_i / len(sums\_list)

        probabilities[i] = {"count": count\_i, "probability": str(round(probability\_i, 3))}

    return probabilities

def transform\_dice(die\_a, die\_b):

    new\_die\_a = [i - 3 if i > 4 else i for i in die\_a]

    new\_die\_b = [j + 2 if j % 2 == 0 else j for j in die\_b]

    return sorted(new\_die\_a), sorted(new\_die\_b)

@app.route('/')

def index():

    total\_combinations = calculate\_total\_combinations()

    die\_a\_sides = range(1, 7)

    die\_b\_sides = range(1, 7)

    original\_probabilities = calculate\_probabilities(die\_a\_sides, die\_b\_sides)

    all\_combinations = generate\_all\_combinations()

    new\_die\_a, new\_die\_b = transform\_dice(list(die\_a\_sides), list(die\_b\_sides))

    modified\_probabilities = calculate\_probabilities(new\_die\_a, new\_die\_b)

    return render\_template('index.html',

                           total\_combinations=total\_combinations,

                           all\_combinations=all\_combinations,

                           original\_probabilities=original\_probabilities,

                           modified\_probabilities=modified\_probabilities)

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True)