**PART – A**

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

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

**QUESTION – 1 logic**

We need to find total number of combinations when both the dice are rolled together, since the number of faces are 6 on each of the dice. Therefore 6x6 (or) length(dieA)xlength(dieB) gives no of combinations as 36.

**QUESTION – 2 logic**

We need to find the distribution of all possible combinations when both dice are rolled together. So by using two for loops we can find all the possible combinations, and storing each combination as a key in a dictinary and frequency of each combination as a value to the corresponding key (frequency of all combinations is 1, since no repetition takes place in the combinations)

**QUESTION – 3 logic**

We need to find the probability of all combination sums , possible from the list of all combinations. We know the formula for finding the probability of any value is

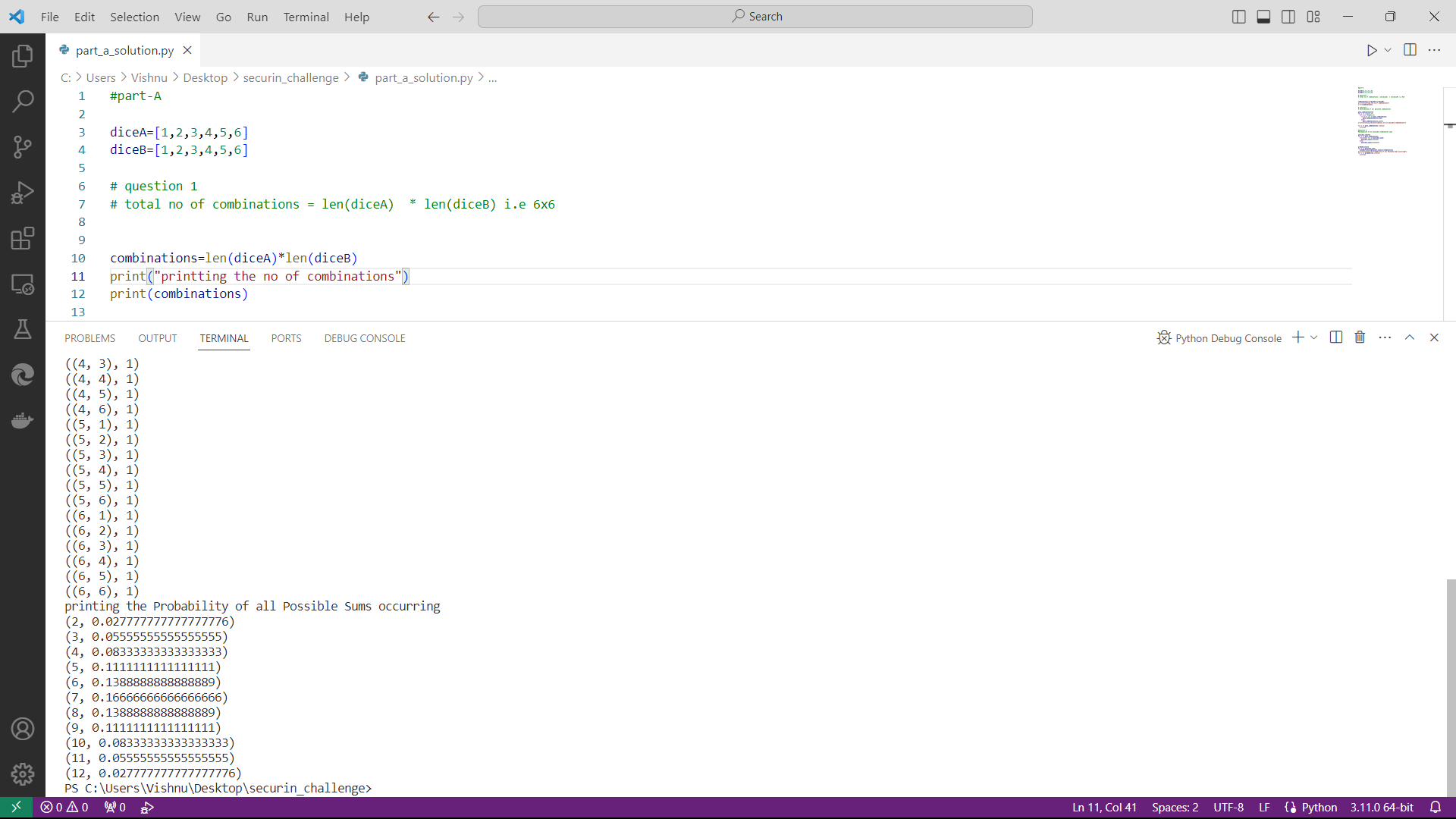
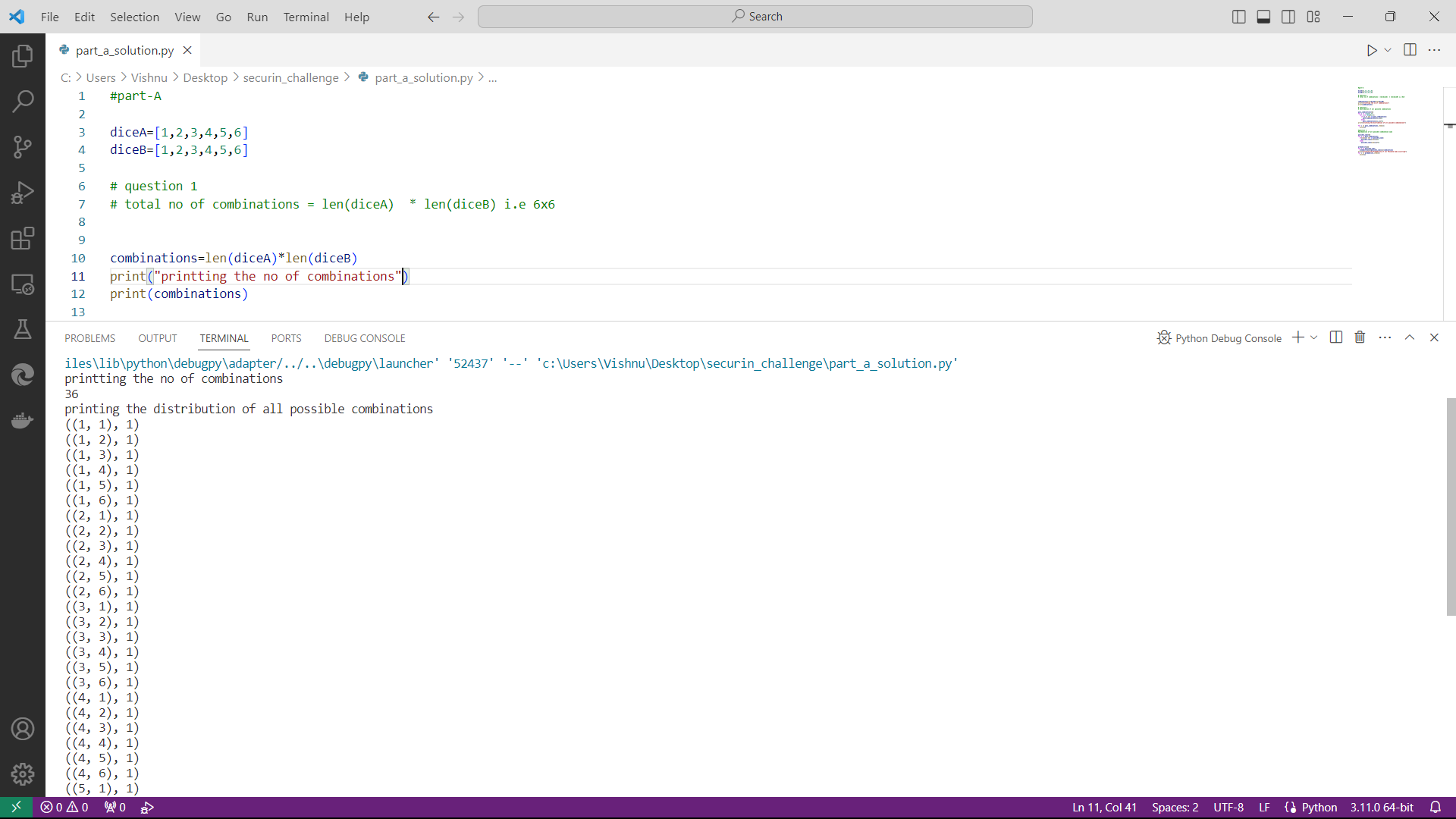
Probability(X) = No.of possible values/total no.of values

So now we find all the possible sums from the list of combinations, now store the sum as the key and frequency of sum as the value to the corresponding key in a dictinary.

Example dict[12]=1

12 is the sum and 1 represents that no of combinations has the sum value as 12.

Therefore **P(sum==12)** = **dict[12] / total no.of combinations(36).**

****

**PART – B**

Now comes the real challenge. You were happily spending a lazy afternoon playing your board game with your dice when suddenly the mischievous Norse God Loki ( You love Thor too much & Loki didn’t like that much ) appeared. Loki dooms your dice for his fun removing all the “Spots” off the dice.

No problem! We have the tools to re-attach the “Spots” back on the Dice. However, Loki has doomed your dice with the following conditions:

● 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.

But in order to play your game, the probability of obtaining the Sums must remain the same! So if you could only roll P(Sum = 2) = 1/X, the new dice must have the spots reattached such that those probabilities are not changed.

**Input:**

● DieA = [1, 2, 3, 4, 5, 6] & Die B = DieA = [1, 2, 3, 4, 5, 6]

**Output:**

● A Transform Function undoom\_dice that takes (DieA, DieB) as input & outputs New\_DieA = [?, ?, ?, ?, ?, ?] ,New\_DieB = [?, ?, ?, ?, ?, ?] where, ● No New\_Die A[x] > 4

**SOLUTION :**

The only way to re-attach the spots back into the dice based on the conditions is to use bruteforce approach. we can have a maximum value 4 in diceA and diceB can have the value greater than 6 based on the conditions.

So possible values for diceA is [1,2,3,4] we have to arrange the six faces of diceA with the mentioned four values.

The base for solving this problem is to realize that the possible ways to get the sum value 2 is only by (1,1) from both dice, so its clear that 1 sholud present on both of the dice, it’s mandatory and the same logic applies for the number 12 also so we need to have 4 on dieA in order to get the sum 12 to maintain the same probability

So here we are trying to get the list of all possible combinations from the list of [1,2,3,4,5,6,7,8], the list of combinations contains the combinations that is suitable for both new diceA and new diceB ,the combination have the property of repetation of numbers so its also suitable for diceA also which has to satisfy the condition **New\_die A[x]<=4.**

Now we have to make all possible pairs from the list of all possible combinations i.e we have to make a pair with every combination to every other combination and these pairs should satisfy the condition that “combination1 < combination2 “ this condition is mandatory which makes pairs that are suitable for diceA having the condition that it should have a maximum value of 4 since we are following the condition combination1<combination2 which means that combination1 in a pair is always less than combination2 satisfying the condition for dieA.

So in this code the **undoom\_dice()**  function will repatch the numbers to the dice by using the functions **poss\_combinations()** and **combination\_sums()**

**Poss\_combinations() -** this function takes no parameters and it returns all possible combinations of arrays from the elements[1,2,3,4,5,6,7,8] which includes the combinations suitable for both dieA and dieB i.e it generates an list **all\_combi[]** which contains all possible combinations. The list all\_combi[] is later used to create the all possible pairs from the list of combination by satisfying the condition combination1<combination2 i.e it finally returns the list of tuples **all\_pairs[]** that contains the all the possible pairs form the list all\_combi[].

**Combination\_sums() –** this function is used to return a list sums\_val[] which contains all possible combination sum values when both dice are rolled together.

We maintain a original\_sums[] list which contains all the possible combination sum values, then we pass each tuple from the list all\_pairs[] to verify which tuple of lists have the same sums\_val[] as that of original\_sums[] then we can consider that pair as the transformed values for New\_dieA and New\_dieB.

Finally the undoom\_dice() function will return the new\_pair[] list that contains a single tuple of two lists for both New diceA and New diceB.

