**Problem Statement: The Doomed Dice Challenge**

Part A:

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

Code......

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

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

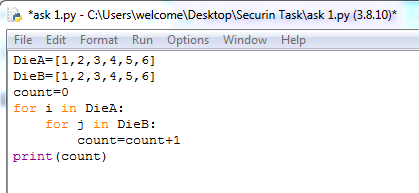
count=0

for i in DieA:

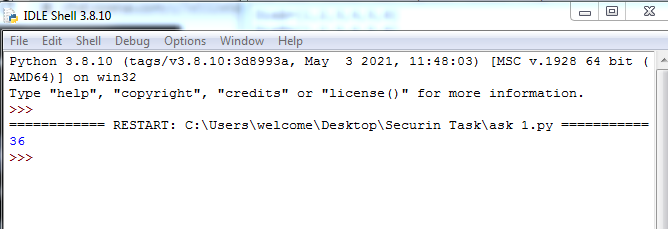
for j in DieB:

count=count+1

print(count)



Output



**Logical explanation**

1. Initialization

`DieA` and `DieB` are two lists representing the faces of two six-sided dice.

`count` is initialized to 0, which will be used to keep track of the total number of combinations.

2. Nested Loops

The outer loop (`for i in DieA:`) iterates over each face of DieA.

The inner loop (`for j in DieB:`) iterates over each face of DieB for each iteration of the outer loop.

This results in a nested loop structure that covers all possible combinations of faces on DieA and DieB.

3. Counting Combinations

For each combination of faces (i, j), the `count` is incremented by 1.

The `count` variable effectively keeps track of the total number of combinations.

4. Print Result

- After both loops complete, the total count of combinations is printed using `print(count)`.

5. Outcome

Since both `DieA` and `DieB` have 6 faces each, the nested loops result in a total of 6 \* 6 = 36 combinations.

The printed result (`36`) represents the total number of possible combinations when rolling two six-sided dice.

In summary, the program systematically considers all possible combinations of faces on the two dice using nested loops and increments a counter for each combination. The final count represents the total number of combinations.

........................................................................................................... 2. 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!

Code...

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

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

Mat=[]

for i in A:

L=[]

for j in B:

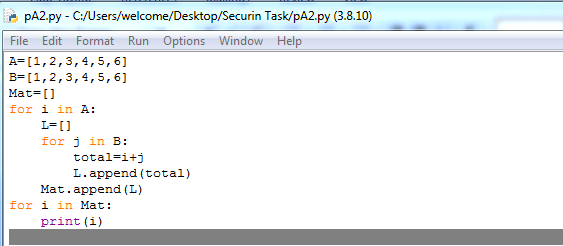
total=i+j

L.append(total)

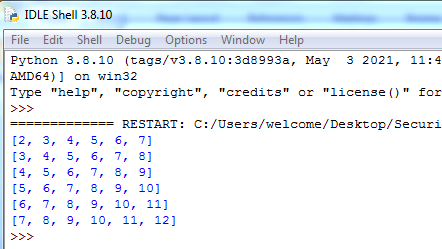
Mat.append(L)

for i in Mat:

print(i)



Output



**Logical explanation**

1. Initialization

Lists `A` and `B` represent the faces of two six-sided dice.

An empty list `Mat` is initialized to store the matrix of sums.

2. Nested Loops:

The outer loop (`for i in A:`) iterates over each face of Die A.

The inner loop (`for j in B:`) iterates over each face of Die B.

For each combination of faces (i, j), the sum is calculated and appended to a row list `L`.

3. Building the Matrix:

The row list `L` is appended to the matrix `Mat` after each inner loop iteration.

4. Displaying the Matrix:

The matrix `Mat` is then printed row by row.

This code effectively creates a matrix where each element represents the sum of a pair of values from Die A and Die B, displaying all possible combinations.

...........................................................................................................

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.

Code...

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

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

Mat=[]

uni=[]

D={}

for i in DieA:

for j in DieB:

total=i+j

Mat.append(total)

if total not in uni:

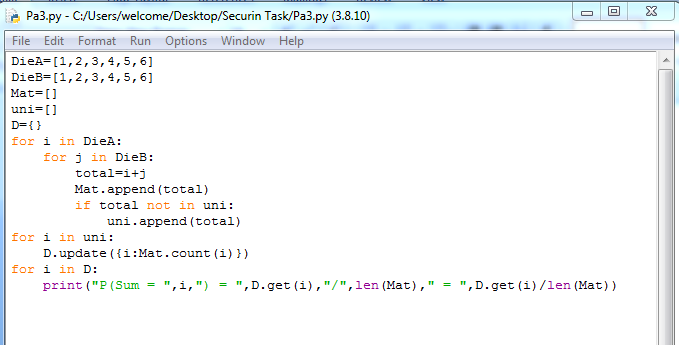
uni.append(total)

for i in uni:

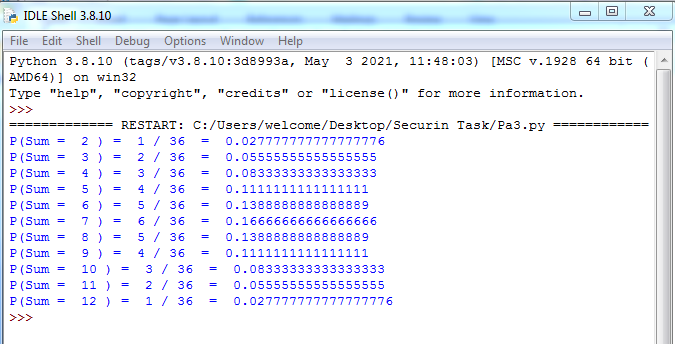
D.update({i:Mat.count(i)})

for i in D:

print("P(Sum = ",i,") = ",D.get(i),"/",len(Mat)," = ",D.get(i)/len(Mat))



Output



**Logical explanation**

1. Nested Loops

The nested loops iterate over each combination of faces from `DieA` and `DieB`.

The sum of each combination (`total`) is calculated and added to both the list `Mat` and the unique sums list `uni`.

2. Counting Occurrences

The loop iterates over the unique sums (`uni`) and counts the occurrences of each sum in the list `Mat`.

The counts are stored in a dictionary `D` where the sum is the key and the count is the value.

3. Displaying Probabilities

The code then prints the probability for each unique sum by dividing the count of each sum by the total number of combinations.

This code calculates the probabilities for each unique sum based on the combinations obtained from rolling both `DieA` and `DieB` together.

...............................................................................................................