**SECUREIN ASSESSMENT**

**Problem Statement: The Doomed Dice Challenge**

**PART A:**

**CODE:PYTHON**

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

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

#1.TOTAL COMBINATION

tot\_comb=len(die\_A)\*len(die\_B)

print(f"TOTAL\_COMBINATION={tot\_comb}\n")

arr={}

#2.possible combination

print("POSSIBLE\_COMBINATION")

for i in range(len(die\_A)):

for j in range(len(die\_B)):

print((die\_A[i],die\_B[j]),end=" ")

print("\n")

#2.sum

print("SUM")

for i in range(len(die\_A)):

for j in range(len(die\_B)):

print(die\_A[i]+die\_B[j],end=" ")

s=die\_A[i]+die\_B[j]

if s in arr.keys():

arr[s]+=1

else:

arr[s]=1

print("\n")

#print(arr)

#3.PROBABILITY OF SUM

print("| SUM || PROBABILITY |")

for i in arr.keys():

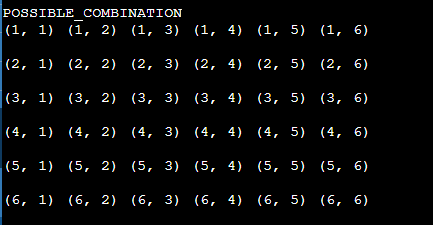
print("| ",i ," || ", round(arr[i]/tot\_comb,3)," |")

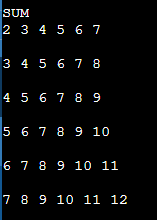
output:

1.

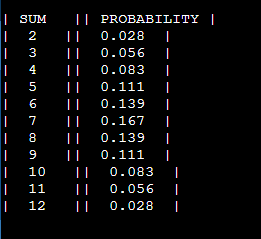


2.





3.



**HOW I CAME UP WITH THE SOLUTION?:**

The problem focused on three parts

1.total combination

2.possible combinations and its sum

3. probability of the sum

Total combination= faces of die\_A \* faces of die\_B

i.e, len(die\_A) \*len(die\_B)

* I employed the method of **matrix addition** in my approach.
* Where I used nested for loops to find the combination of the die\_A and die\_B
* printed the combination
* calculated the sum of each combination within the for loop and
* inserted it in a dictionary with key as the sum and value as the count
* then printed the sum(which is the key from the dictionary ) and
* the probability(count of the sum /total combination)

**LOGIC:**

* + Total combination= faces of die\_A \* faces of die\_B i.e, len(die\_A) \*len(die\_B)
  + employed the method of **matrix addition** in my approach.
  + Nested for loop gave the possible combination
  + Calculated the sum using the combination
  + Probability = Count of the sum / total combination

**PART B:**

**CODE:PYTHON**

c={}

def freq(a,b):

d={}

for i in range(1,7):

for j in range(1,7):

s=i+j

if s in d.keys():

d[s]=d[s]+1

else:

d[s]=1

c[s]=0

return d

def check\_prob(d,c):

for i in d.keys():

if (d[i]/36) != (c[i]/36):

return False

return True

def undoom\_dice(a,b):

#sum=2 has only one possibility (1,1)-first element of new\_a and new\_b

# as 4 is the most spots on a face -last element of new\_a

#the rest elemets are initalized to 1

new\_die\_a=[1,1,1,1,1,4]

new\_die\_b=[1,1,1,1,1,1]

d=freq(a,b)

#finding new\_die\_a

for i in range(1):

for j in range(6):

s=new\_die\_a[j]+new\_die\_b[i]

while((c[s]+1)>d[s]):

new\_die\_a[j]=new\_die\_a[j]+1

s=new\_die\_a[j]+new\_die\_b[i]

c[s]=c[s]+1

#finding new\_die\_b

for i in range(1,6):

for j in range(6):

s=new\_die\_a[j]+new\_die\_b[i]

while((c[s]+1)>d[s]):

new\_die\_b[i]=new\_die\_b[i]+1

s=new\_die\_a[j]+new\_die\_b[i]

c[s]=c[s]+1

# checking for equal probability between old and new die

if(check\_prob(d,c)):

print("NEW\_DIE\_A = ",new\_die\_a," NEW\_DIE\_B = ",new\_die\_b)

else:

print("incorrect die")

print("DIE\_A:")

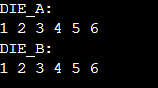
A=[int(x) for x in input().split(" ")]

print("DIE\_B:")

B=[int(x) for x in input().split(" ")]

undoom\_dice(A,B)

**INPUT:**



**Output:**

****

**HOW I CAME UP WITH THE SOLUTION?:**

This problem focused on finding the new\_die\_A and New\_die\_B with certain

Constrains:

● 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.

* I created a list new\_die\_A where the first and the last element were initialized to 1 and 4 – as (1,1) is the only combination in which sum=2 is possible(so the first element of new\_die\_A and new\_die\_B) and 4 is the most spots possible in a face(so the last element is initialized as 4 in new\_die\_a )

new\_die\_a=[1,1,1,1,1,4]

new\_die\_b=[1,1,1,1,1,1]

* The other elements are initialized to 1 which will be changed by comparing the frequency of the sum of the possible combination form the old\_die\_A and old\_die\_B
* the new\_die\_A is found ,here c is the dictionary to have the count of the frequency of the sum of the possible combination from new\_die\_a and new\_die\_b

#finding new\_die\_a

for i in range(1):

for j in range(6):

s=new\_die\_a[j]+new\_die\_b[i]

while((c[s]+1)>d[s]):

new\_die\_a[j]=new\_die\_a[j]+1

s=new\_die\_a[j]+new\_die\_b[i]

c[s]=c[s]+1

* the new\_die\_b is found

#finding new\_die\_b

for i in range(1,6):

for j in range(6):

s=new\_die\_a[j]+new\_die\_b[i]

while((c[s]+1)>d[s]):

new\_die\_b[i]=new\_die\_b[i]+1

s=new\_die\_a[j]+new\_die\_b[i]

c[s]=c[s]+1

* Then the old\_dice and new\_dice are checked for equal probability
* And the new\_die\_a and new\_die\_b is printed

**LOGIC:**

* + - Found the new\_die\_a by comparing the frequency of sum of the possible combination and by initializing the first and last element of the new\_die\_a as 1 and 4 – as (1,1) combination is the only possible way to get the sum=2 and the most number of sports that can be added in a face is 4- so the last element is initialized as 4
    - Found the new\_die\_b from new\_die\_a and the frequency of the sum of possible combination of the old\_die\_A and old\_die\_B
    - Checked the probability of old and new
    - Printed the new\_die\_a and new\_die\_b