**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():

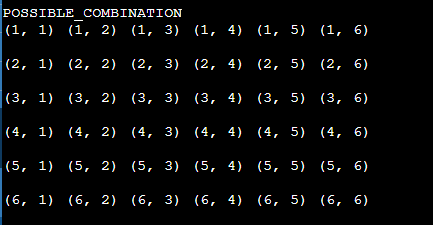
print("| ",i ," || ", round(arr[i]/tot\_comb,3)," |")

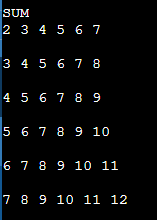
output:

1.

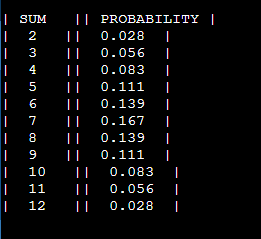


2.





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**

def undoom\_dice(a,b):

arr=[]

d1={}

#finding all possible sum of die a and b

for i in range(6):

arr.append([])

for j in range(6):

s=a[i]+b[j]

arr[i].append(s)

if s in d1.keys():

d1[s]+=1

else:

d1[s]=1

# print(arr)

new\_b=[]

start=[]

# find possible new\_b

c=-1

for i in range(0,-5,-1):

c+=1

if i+5<=4:

start.append(i)

new\_b.append(arr[c])

print(start)

print(new\_b)

#find new\_a

new\_a=[]

for i in start:

new\_a.append([x for x in range(i ,i+6)])

#checking for equal probability between old and new die

for i in range(len(start)):

d2={}

print(new\_a[i])

print(new\_b[i])

for j in range(6):

for k in range(6):

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

# print(s,end=" ")

if s in d2.keys():

d2[s]+=1

else:

d2[s]=1

flag=True

for j in d1.keys():

if (d1[j]/36)!=(d2[j]/36):

flag=False

print(flag)

print("\n\n")

return(new\_a,new\_b)

# old\_die\_A=[1,2,3,4,5,6]

# old\_die\_B=[1,2,3,4,5,6]

print("Die\_A\n")

old\_die\_A=[int(x) for x in input().split(" ")]

print("\nDie\_B\n")

old\_die\_B=[int(x) for x in input().split(" ")]

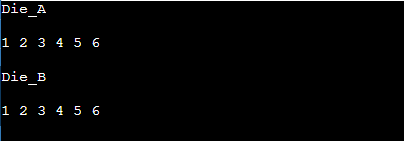
# to print dice

new=undoom\_dice(old\_die\_A,old\_die\_B)

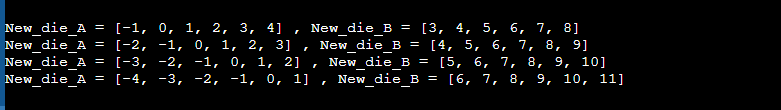
for i in range(len(new[0])):

print(f"New\_die\_A = {new[0][i]} , New\_die\_B = {new[1][i]}")

**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.

* To solve this I used a range for die\_A as -4 to 4 .
* Then for each value of the range , I generated a list containing 6 elements where each element is <4.
* And got 4 list each containing 6 element where each represent a new\_die\_A.
* Using each new\_die\_A , I generated new\_die\_b from the sum of the old\_die\_A and old\_die\_B

i.e.,

c=-1

for i in range(0,-5,-1):

c+=1

if i+5<=4

start.append(i)

new\_b.append(arr[c])

* + here I kept the left end as -4 using ‘ for loop ‘ and right end using ‘if ‘condition .
  + the position of zero in the list generated from ‘i’ was given to the variable c
  + and c was used to find the list from the matrix --- sum of the old\_die A and old\_die\_B
* found the count of possible sum from new\_die\_A and new\_die\_B
* and checked the probability of possible sum from new\_die\_A and new\_die\_B with the probability of possible sum from old\_die\_A and old\_die\_B
* printed the New\_die\_A and New\_die\_B

**LOGIC:**

* + - Found the possible new\_die\_A using the limit -4 to 4
    - Found the new\_die\_B from new\_die\_A and sum 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