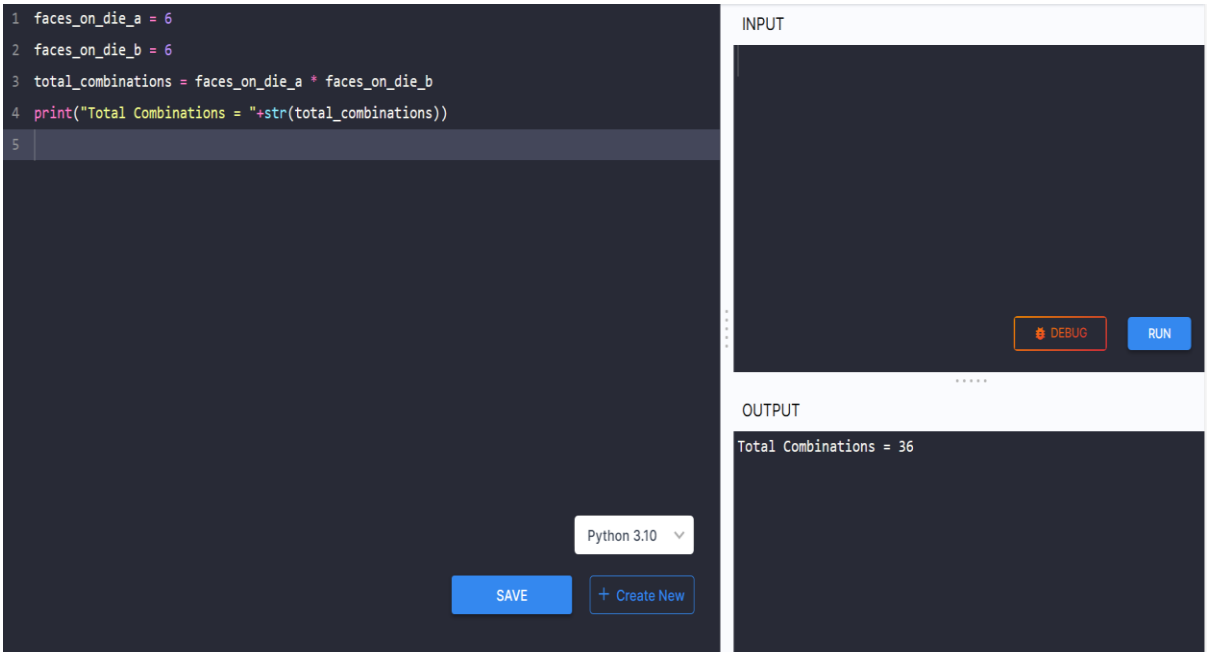
Part A:

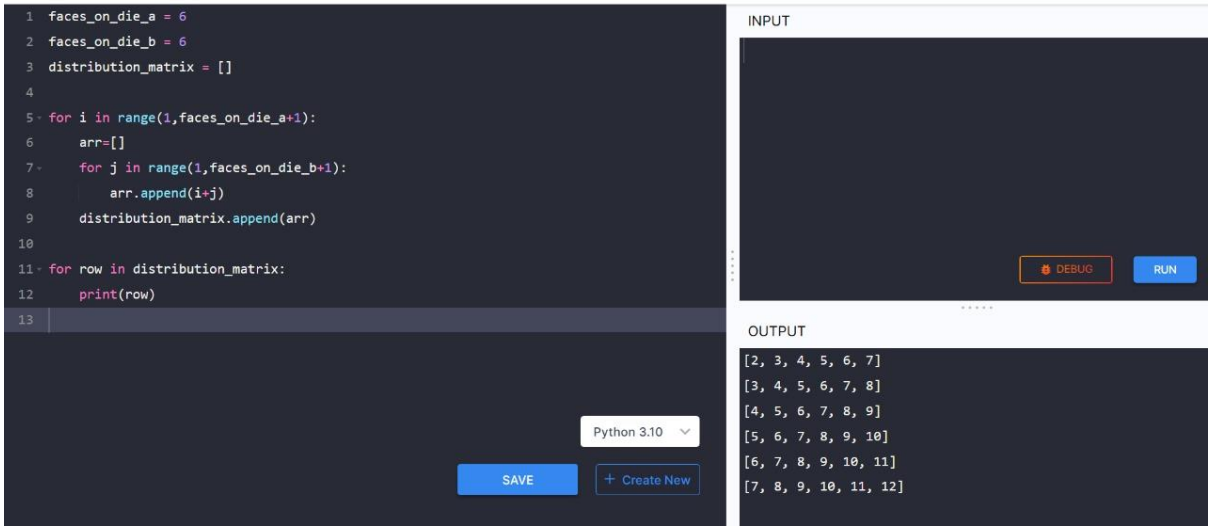
1. How many total combinations are possible? Show the math along with the code!
   Die_A=[1,2,3,4,5,6]
   Die_B=[1,2,3,4,5,6]
   Total possible combinations:
   (1,1), (1,2), (1,3), (1,4), (1,5), (1,6),
   (2,1), (2,2), (2,3), (2,4), (2,5), (2,6),
   (3,1), (3,2), (3,3), (3,4), (3,5), (3,6),
   (4,1), (4,2), (4,3), (4,4), (4,5), (4,6),
   (5,1), (5,2), (5,3), (5,4), (5,5), (5,6),
   (6,1), (6,2), (6,3), (6,4), (6,5), (6,6)

   Therefore, total combinations are 36



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!

   (1,1), (1,2), (1,3), (1,4), (1,5), (1,6) = [2,3,4,5,6,7]
   (2,1), (2,2), (2,3), (2,4), (2,5), (2,6) = [3,4,5,6,7,8]
   (3,1), (3,2), (3,3), (3,4), (3,5), (3,6) = [4,5,6,7,8,9]
   (4,1), (4,2), (4,3), (4,4), (4,5), (4,6) = [5,6,7,8,9,10]
   (5,1), (5,2), (5,3), (5,4), (5,5), (5,6) = [6,7,8,9,10,11]
   (6,1), (6,2), (6,3), (6,4), (6,5), (6,6) = [7,8,9,10,11,12]



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

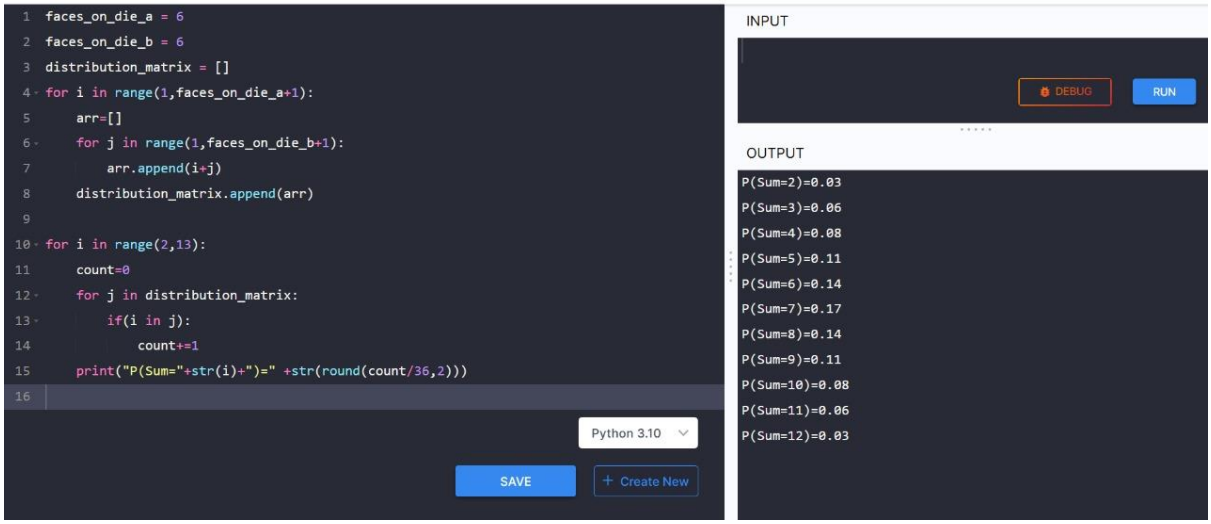
P(event) = favourable chances/ total number of chances

For P(sum=2), (1,1) is the only possibility

Therefore P(sum=2) = 1/36

For P(sum=3), (1,2) & (2,1) are the possibilities

Therefore P(sum=3) = 2/36 like we have to calculate the remaining probabilities up to P(sum=12)



Part B:

- 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.

Based on the above conditions the possible combinations for Die A is :

[1,1,1,1,1,1],

[1,1,1,1,1,2],

.

.

.

[4,4,4,4,4,4]

Possible combinations for Die B is

[1,1,1,1,1,1],

[1,1,1,1,1,2],

.

.

.

[11,11,11,11,11,11]

The maximum value in Die B is 11 , because the minimum value in Die A is 1

```
def generate_combos(faces, n):
    if n==0:
        return [[]]
    combos = []
    for f in faces:
        prefixes = generate_combos(faces, n-1)
        for prefix in prefixes:
            combo = prefix + [f]
            combos.append(combo)
    return combos
```

```
def undoom_dice(die_a,die_b):
    combination_set=set()
    for i in die_a:
        for j in die_b:
            combination_set.add((i,j))
    #print(combination_set)
    length=len(combination_set)
    l=[]
    for i in combination_set:
        s=0
        for j in i:
            s+=j
        l.append(s)
    #print(l)
    probability_list=[]
    for i in range(2,13):
        if(i not in l):
            break
        else:
            probability_list.append(round(l.count(i)/length,2))
    first=probability_list

    die_a_faces = [1,2,3,4]
    die_b_faces = [1,2,3,4,5,6,7,8,9,10,11]



    for i in die_a_faces:
        for j in die_a_faces:
            die_a = [1,2,3,4,i,j]
            die_b_options = generate_combos(die_b_faces, 6)
```

```
        for die_b in die_b_options:
            combination_set=set()
            for i in die_a:
                for j in die_b:
                    combination_set.add((i,j))
            #print(combination_set)
            length=len(combination_set)
            l=[]
            for i in combination_set:
                s=0
                for j in i:
                    s+=j
                l.append(s)
            #print(l)
            probability_list=[]
            for i in range(2,13):
                if(i not in l):
                    break
                else:
                    probability_list.append(round(l.count(i)/length,2))
            if(first==probability_list):
                res_a,res_b=die_a,die_b
                break
            else:
              return None,None
    return res_a,res_b

die_a=[1,2,3,4,5,6]
die_b=[1,2,3,4,5,6]
res_a,res_b=undoom_dice(die_a,die_b)
print(res_a,res_b)
```