<u>Part-A (15-20 Minutes):</u>

1. How many total combinations are possible?
    - ❖ 36 combinations are possible when throwing two dice.

2. Calculate and display the distribution of all possible combinations that can be obtained when rolling both Die A and Die B together.

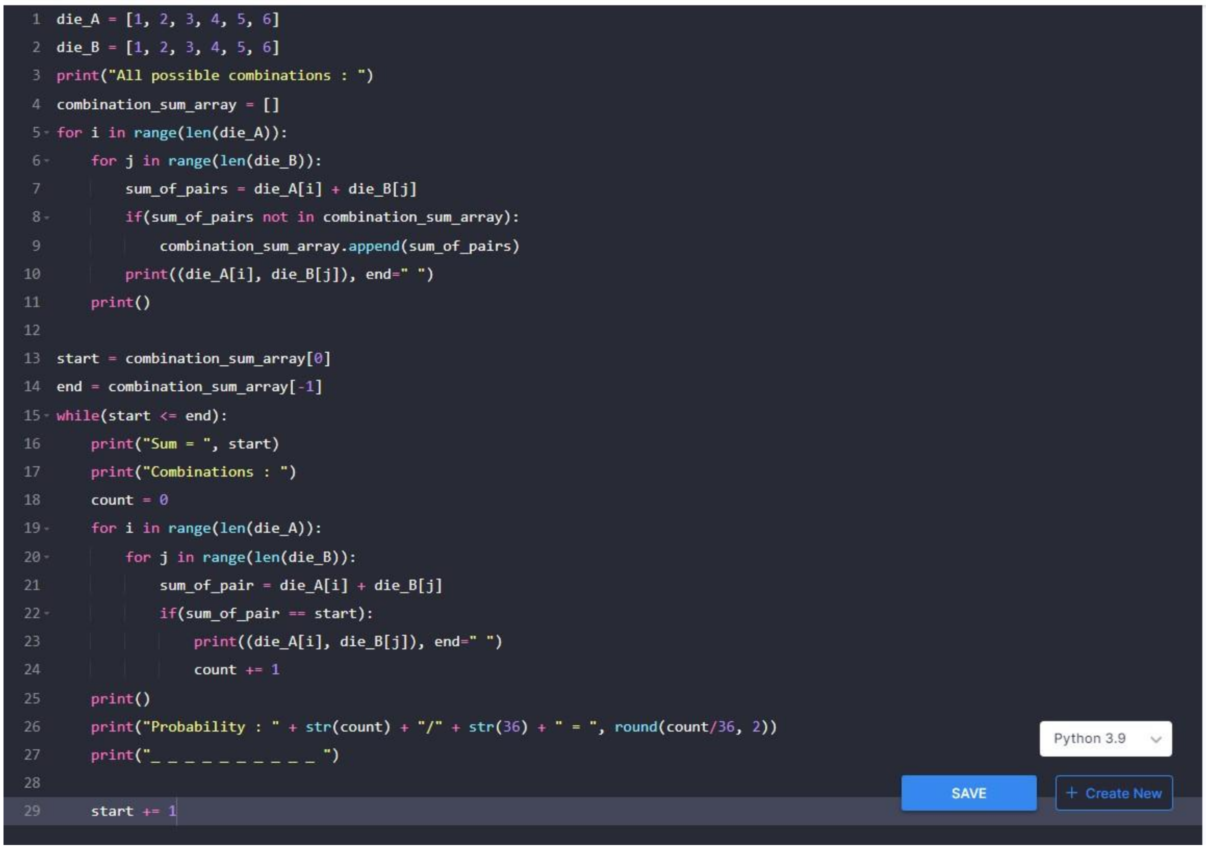
3. Calculate the Probability of all Possible Sums occurring among the number of combinations from (2).

Code :

```
die_A = [1, 2, 3, 4, 5, 6]
die_B = [1, 2, 3, 4, 5, 6]
print("All possible combinations : ")
combination_sum_array = []
for i in range(len(die_A)):
   for j in range(len(die_B)):
      sum_of_pairs = die_A[i] + die_B[j]
      if(sum_of_pairs not in combination_sum_array):
         combination_sum_array.append(sum_of_pairs)
      print((die_A[i], die_B[j]), end=" ")
   print()

start = combination_sum_array[0]
end = combination_sum_array[-1]

while(start <= end):
   print("Sum = ", start)
   print("Combinations : ")
   count = 0
   for i in range(len(die_A)):
      for j in range(len(die_B)):
         sum_of_pair = die_A[i] + die_B[j]
```

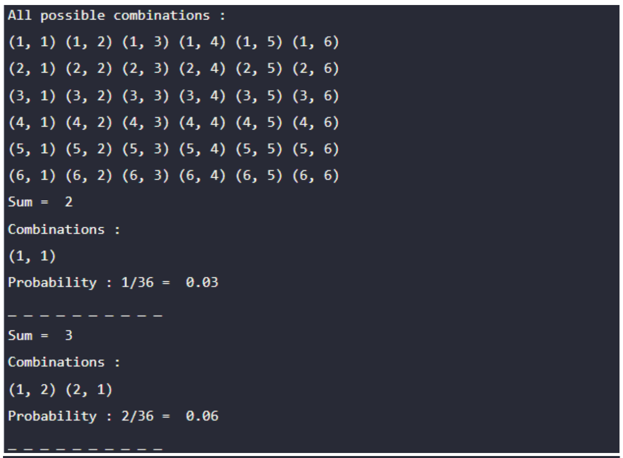
```
            if(sum_of_pair == start):
                print((die_A[i], die_B[j]), end=" ")
                count += 1
    print()
    print("Probability : " + str(count) + "/" + str(36) + " = ", round(count/36, 2))
    print("_ _ _ _ _ _ _ _ _ _ ")

    start += 1
```

Screenshot of code:

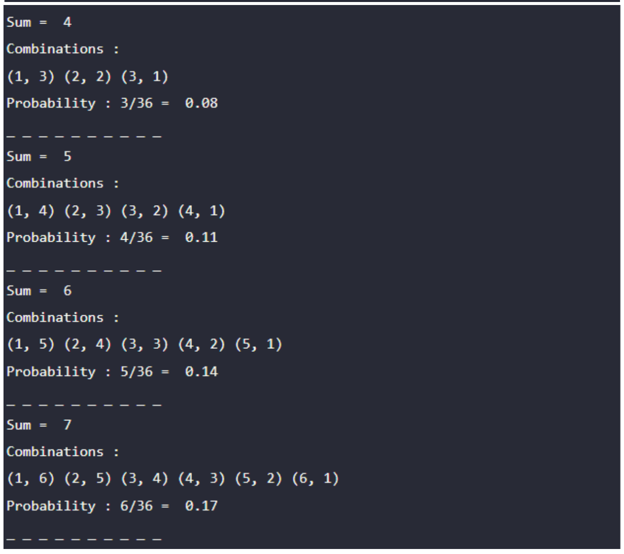


```
die_A = [1, 2, 3, 4, 5, 6]
die_B = [1, 2, 3, 4, 5, 6]
print("All possible combinations : ")
combination_sum_array = []
for i in range(len(die_A)):
    for j in range(len(die_B)):
        sum_of_pairs = die_A[i] + die_B[j]
        if(sum_of_pairs not in combination_sum_array):
            combination_sum_array.append(sum_of_pairs)
        print((die_A[i], die_B[j]), end=" ")
    print()

start = combination_sum_array[0]
end = combination_sum_array[-1]
while(start <= end):
    print("Sum = ", start)
    print("Combinations : ")
    count = 0
    for i in range(len(die_A)):
        for j in range(len(die_B)):
            sum_of_pair = die_A[i] + die_B[j]
            if(sum_of_pair == start):
                print((die_A[i], die_B[j]), end=" ")
                count += 1
    print()
    print("Probability : " + str(count) + "/" + str(36) + " = ", round(count/36, 2))
    print("_ _ _ _ _ _ _ _ _ _ ")

    start += 1
```

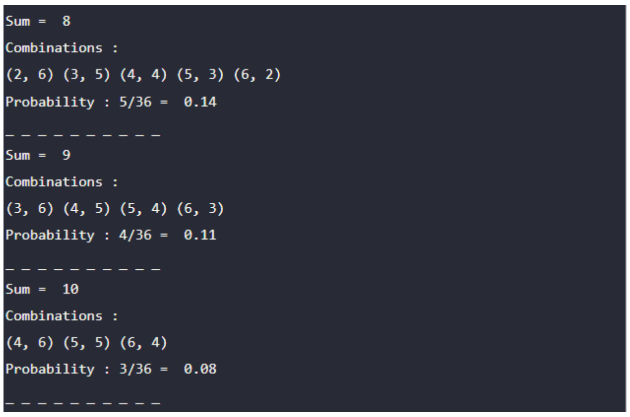
Output Screenshot:



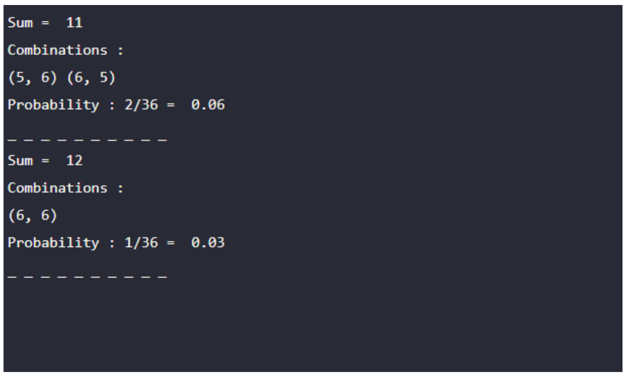
```
All possible combinations :
(1, 1) (1, 2) (1, 3) (1, 4) (1, 5) (1, 6)
(2, 1) (2, 2) (2, 3) (2, 4) (2, 5) (2, 6)
(3, 1) (3, 2) (3, 3) (3, 4) (3, 5) (3, 6)
(4, 1) (4, 2) (4, 3) (4, 4) (4, 5) (4, 6)
(5, 1) (5, 2) (5, 3) (5, 4) (5, 5) (5, 6)
(6, 1) (6, 2) (6, 3) (6, 4) (6, 5) (6, 6)
Sum =  2
Combinations :
(1, 1)
Probability : 1/36 =  0.03
_ _ _ _ _ _ _ _ _ _
Sum =  3
Combinations :
(1, 2) (2, 1)
Probability : 2/36 =  0.06
_ _ _ _ _ _ _ _ _ _
```



```
Sum =  4
Combinations :
(1, 3) (2, 2) (3, 1)
Probability : 3/36 =  0.08
_ _ _ _ _ _ _ _ _ _
Sum =  5
Combinations :
(1, 4) (2, 3) (3, 2) (4, 1)
Probability : 4/36 =  0.11
_ _ _ _ _ _ _ _ _ _
Sum =  6
Combinations :
(1, 5) (2, 4) (3, 3) (4, 2) (5, 1)
Probability : 5/36 =  0.14
_ _ _ _ _ _ _ _ _ _
Sum =  7
Combinations :
(1, 6) (2, 5) (3, 4) (4, 3) (5, 2) (6, 1)
Probability : 6/36 =  0.17
_ _ _ _ _ _ _ _ _ _
```



```
Sum =  8
Combinations :
(2, 6) (3, 5) (4, 4) (5, 3) (6, 2)
Probability : 5/36 =  0.14
_ _ _ _ _ _ _ _ _ _
Sum =  9
Combinations :
(3, 6) (4, 5) (5, 4) (6, 3)
Probability : 4/36 =  0.11
_ _ _ _ _ _ _ _ _ _
Sum =  10
Combinations :
(4, 6) (5, 5) (6, 4)
Probability : 3/36 =  0.08
_ _ _ _ _ _ _ _ _ _
```



```
Sum =  11
Combinations :
(5, 6) (6, 5)
Probability : 2/36 =  0.06
_ _ _ _ _ _ _ _ _ _
Sum =  12
Combinations :
(6, 6)
Probability : 1/36 =  0.03
_ _ _ _ _ _ _ _ _ _
```

Explanation:

- Declare die_A and die_B as list and drop both the value [1, 2, 3, 4, 5, 6].
- Declare empty list to save sum of all combinations (combination_sum_array= []).
- Find all combinations, I took two for loop.

  for i in range length of the die_A list and for j in range length of the die_B list.

  and added a conditional statement to check whether sum_of_pair is not in **combination_sum_array** . If condition is True, allowed within the conditional statement.

- Print all combinations when throwing two dice.
- Now, Value of combination_sum_array = [] is [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12].
- Assigned first index value to **Start**, last index value to **end**. i.e, (Start = 2, end = 12).
- I have Used while loop range from (2 to 12).
- Print Sum of Combinations, Possible combinations and Probability of all Possible Sums.

  Example :

Start = 2, end = 12

count = 0

While loop **(2 <= 12)** is True. Print Sum = 2 inside

Enter into the for loop, here calculate the sum_of_pair (1, 1) is the only possible pair of Sum = 2.

Assuming sum_of_pair == start, i.e., (2 == 2) allowed in conditional statement, and print possible pairs and count value increases by 1, now count = 1.

Print the Probability count value divided by 36 and round it up to two decimal places. **(1 / 36) => 0.03.**

Start value increases by 1.

Output:

Sum = **2**

Combinations :

(1, 1)

Probability **: 1/36 = 0.03**

_ _ _ _ _ _ _ _ _ _

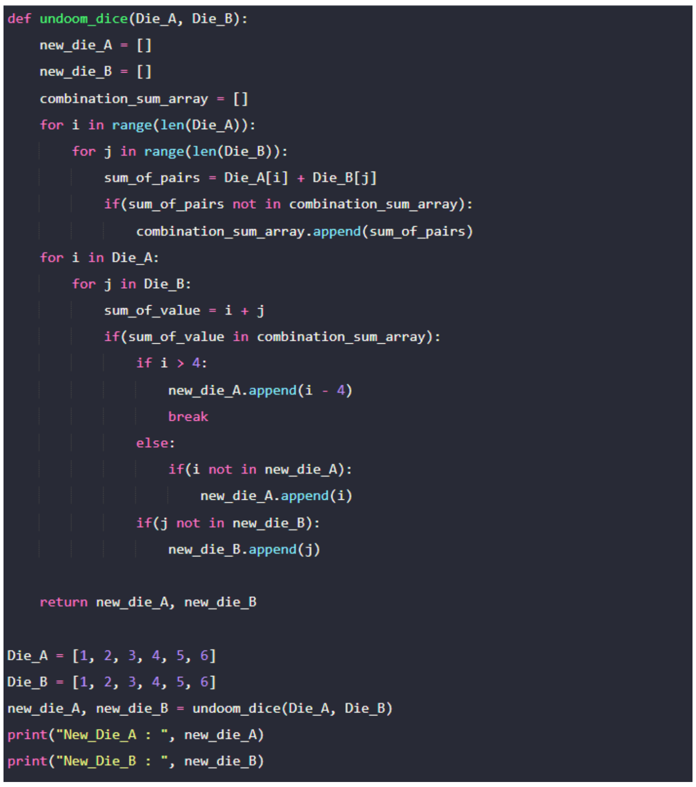
Part – B:

Code :

```
def undoom_dice(Die_A, Die_B):
  new_die_A = []
  new_die_B = []
  combination_sum_array = []
  for i in range(len(Die_A)):
    for j in range(len(Die_B)):
      sum_of_pairs = Die_A[i] + Die_B[j]
      if(sum_of_pairs not in combination_sum_array):
        combination_sum_array.append(sum_of_pairs)

  for i in Die_A:
    for j in Die_B:
      sum_of_value = i + j
      if(sum_of_value in combination_sum_array):
        if i > 4:
          new_die_A.append(i - 4)
          break
        else:
          if(i not in new_die_A):
            new_die_A.append(i)
        if(j not in new_die_B):
          new_die_B.append(j)
  return new_die_A, new_die_B

Die_A = [1, 2, 3, 4, 5, 6]
Die_B = [1, 2, 3, 4, 5, 6]

new_die_A, new_die_B = undoom_dice(Die_A, Die_B)
print("New_Die_A : ", new_die_A)
```

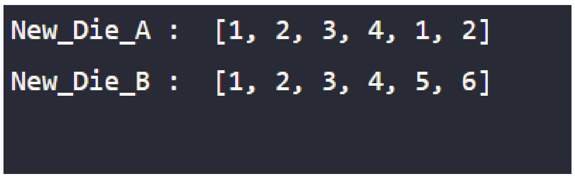
print("New_Die_B : ", new_die_B)

Code Screenshot :



```
def undoom_dice(Die_A, Die_B):
    new_die_A = []
    new_die_B = []
    combination_sum_array = []
    for i in range(len(Die_A)):
        for j in range(len(Die_B)):
            sum_of_pairs = Die_A[i] + Die_B[j]
            if(sum_of_pairs not in combination_sum_array):
                combination_sum_array.append(sum_of_pairs)
    for i in Die_A:
        for j in Die_B:
            sum_of_value = i + j
            if(sum_of_value in combination_sum_array):
                if i > 4:
                    new_die_A.append(i - 4)
                    break
                else:
                    if(i not in new_die_A):
                        new_die_A.append(i)
                if(j not in new_die_B):
                    new_die_B.append(j)

    return new_die_A, new_die_B

Die_A = [1, 2, 3, 4, 5, 6]
Die_B = [1, 2, 3, 4, 5, 6]
new_die_A, new_die_B = undoom_dice(Die_A, Die_B)
print("New_Die_A : ", new_die_A)
print("New_Die_B : ", new_die_B)
```

Output:



```
New_Die_A :  [1, 2, 3, 4, 1, 2]
New_Die_B :  [1, 2, 3, 4, 5, 6]
```

Explanation:

- Define a function undoom_dice and pass it as arguments Die_A, Die_B with an initial value.
- Declare new-die_A and new_Die_B with an empty list. Similarly, combination_sum_array.
- for i in range length of the Die_A list and for j in range length of the Die_B list.

and added a conditional statement to check whether sum_of_pair is not in combination_sum_array . If condition is True, allowed within the conditional statement.

- Two for loops were used, one for Die_A and one for Die_B.
- Once the sum_of_value is calculated, I added an If condition that the sum_of_value is present in the combination_sum_array allowed inside the statement, Die_A value > 4 is subtracted from 4 and added to the new_die_A.
- If not, block added a value to new_die_A that did not exist previously.
- The same process was followed for new_die_B.

- As a final step, return the new_die_A, new_die_B from the function. After that, print the new_die_A and new_die_B.