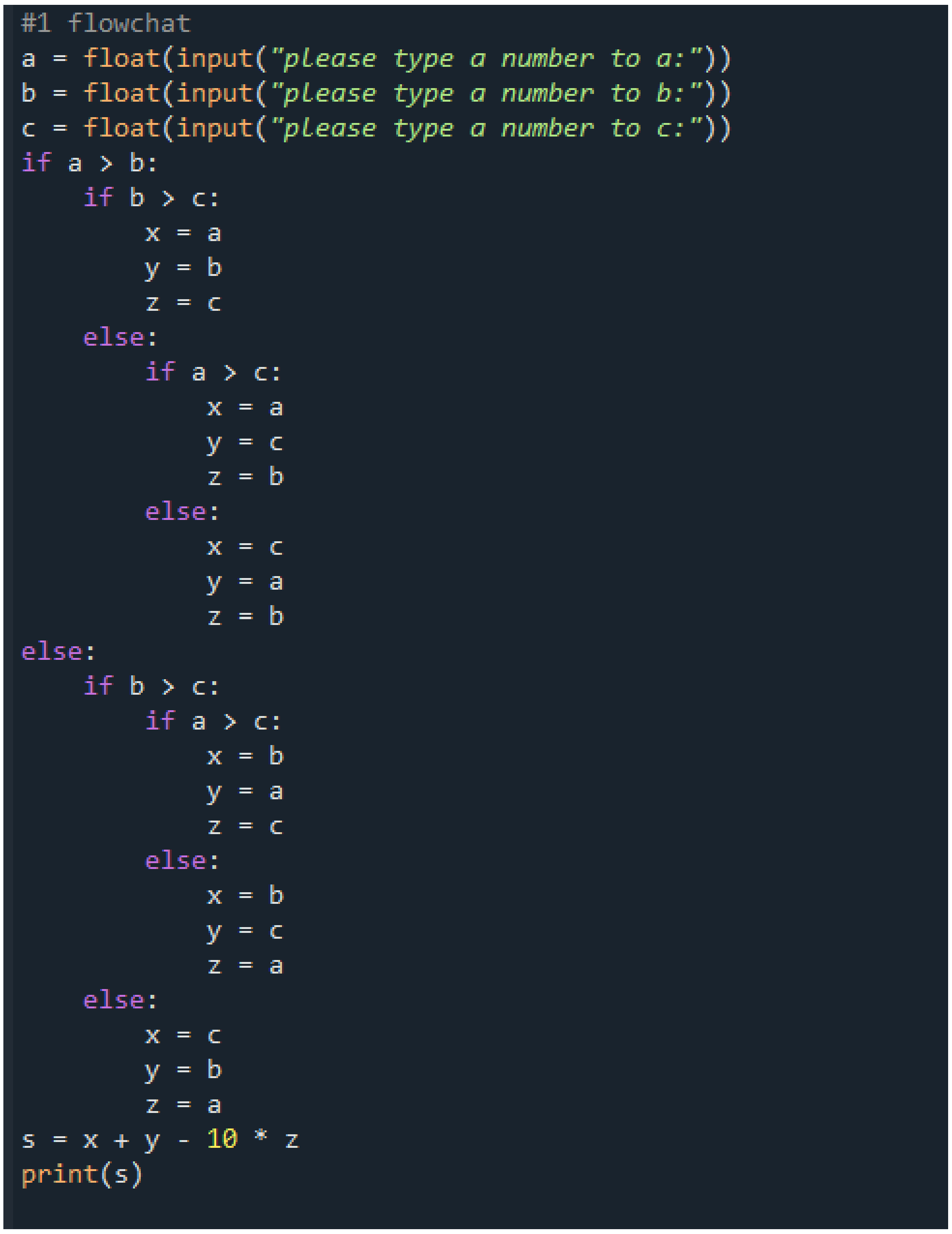
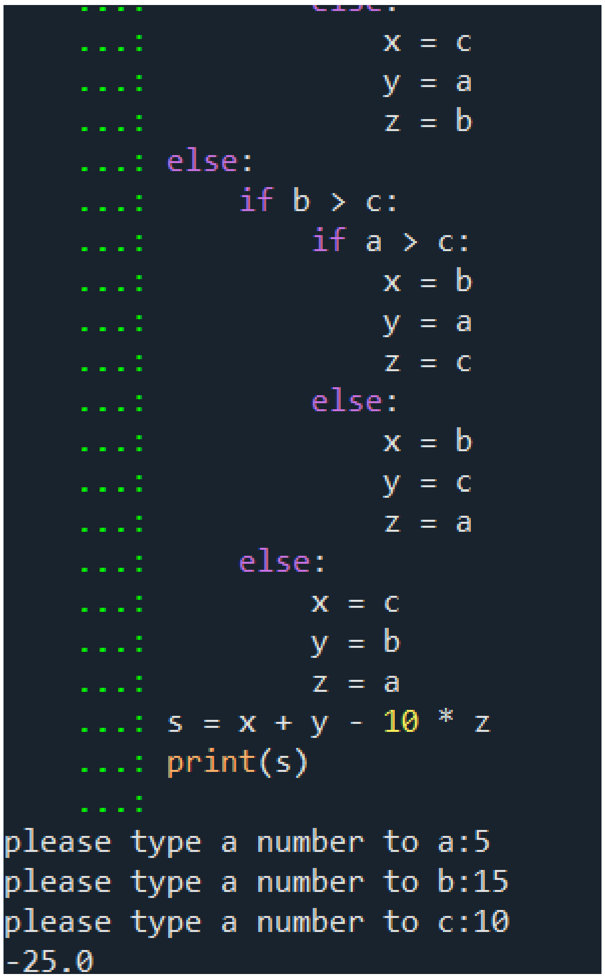
report

1. 在观察第一题流程图后发现这是一道先排大小后赋值计算题目计算结果及代码

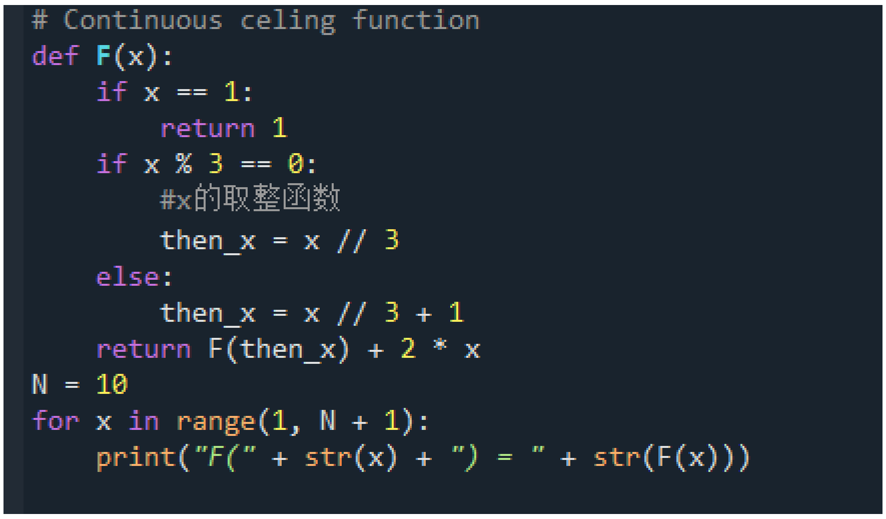


```
#1 flowchat
a = float(input("please type a number to a:"))
b = float(input("please type a number to b:"))
c = float(input("please type a number to c:"))
if a > b:
    if b > c:
        x = a
        y = b
        z = c
    else:
        if a > c:
            x = a
            y = c
            z = b
        else:
            x = c
            y = a
            z = b
else:
    if b > c:
        if a > c:
            x = b
            y = a
            z = c
        else:
            x = b
            y = c
            z = a
    else:
        x = c
        y = b
        z = a
s = x + y - 10 * z
print(s)
```

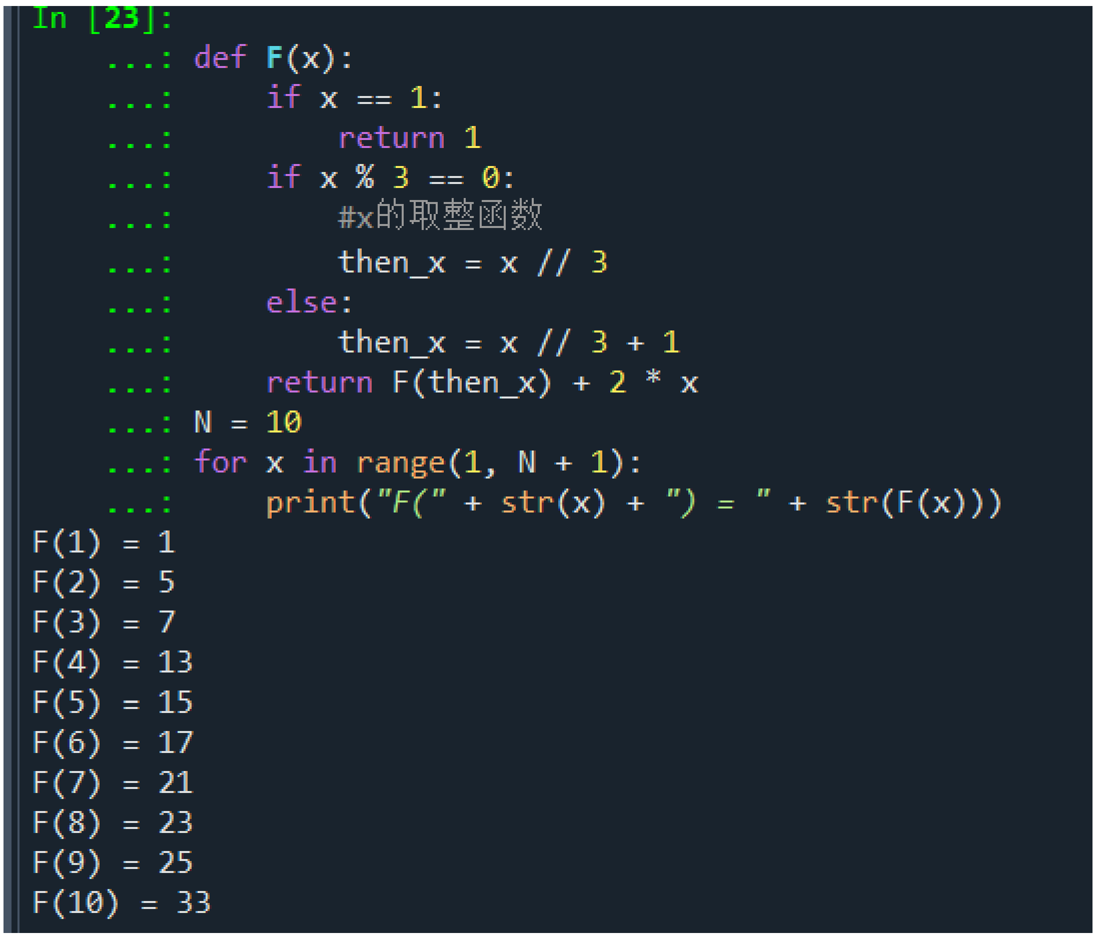


```
    ...:         else:
    ...:             x = c
    ...:             y = a
    ...:             z = b
    ...: else:
    ...:     if b > c:
    ...:         if a > c:
    ...:             x = b
    ...:             y = a
    ...:             z = c
    ...:         else:
    ...:             x = b
    ...:             y = c
    ...:             z = a
    ...:     else:
    ...:         x = c
    ...:         y = b
    ...:         z = a
    ...: s = x + y - 10 * z
    ...: print(s)
    ...:
please type a number to a:5
please type a number to b:15
please type a number to c:10
-25.0
```

2. 第二题递归函数转换为了取整函数，代码及结果如下：

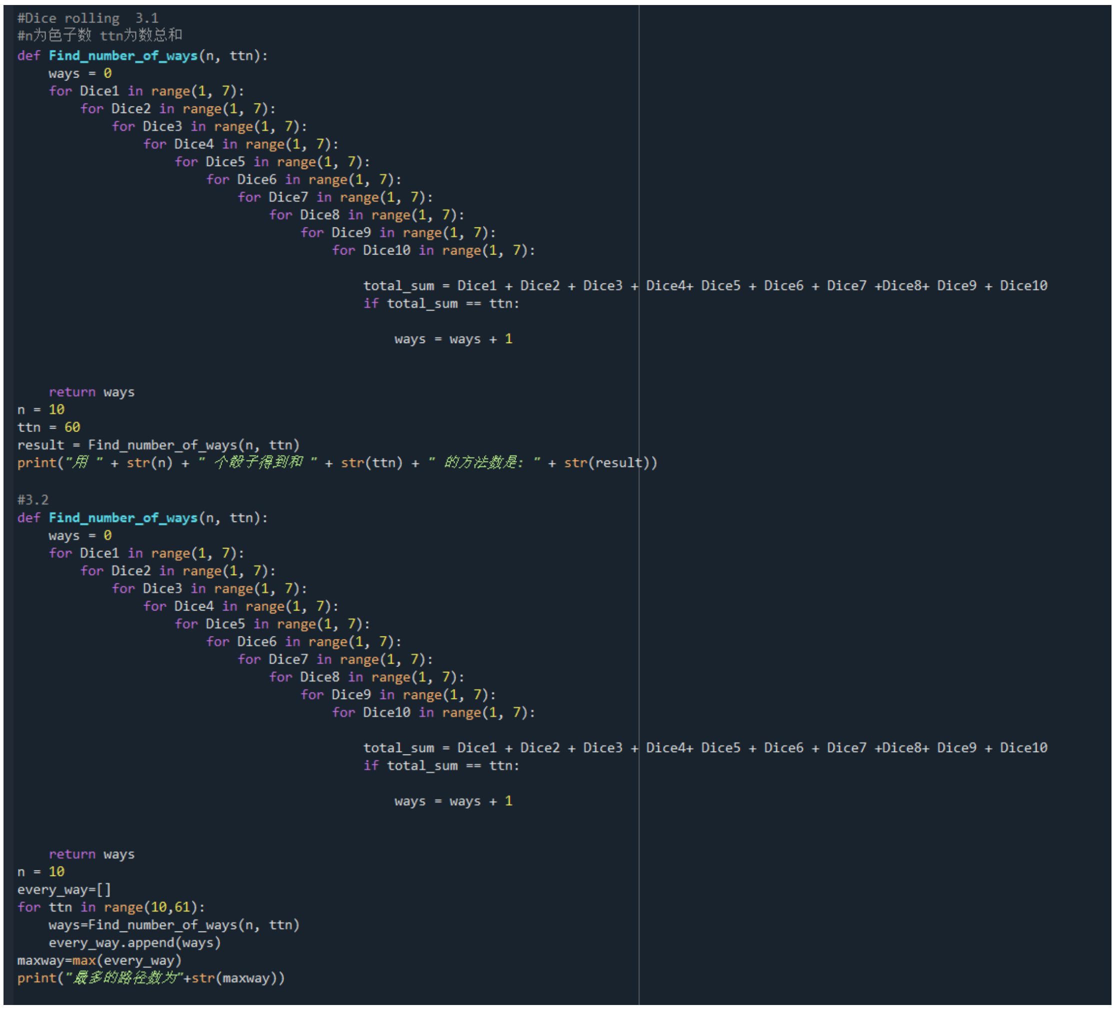


```
# Continuous celing function
def F(x):
    if x == 1:
        return 1
    if x % 3 == 0:
        #x的取整函数
        then_x = x // 3
    else:
        then_x = x // 3 + 1
    return F(then_x) + 2 * x
N = 10
for x in range(1, N + 1):
    print("F(" + str(x) + ") = " + str(F(x)))
```

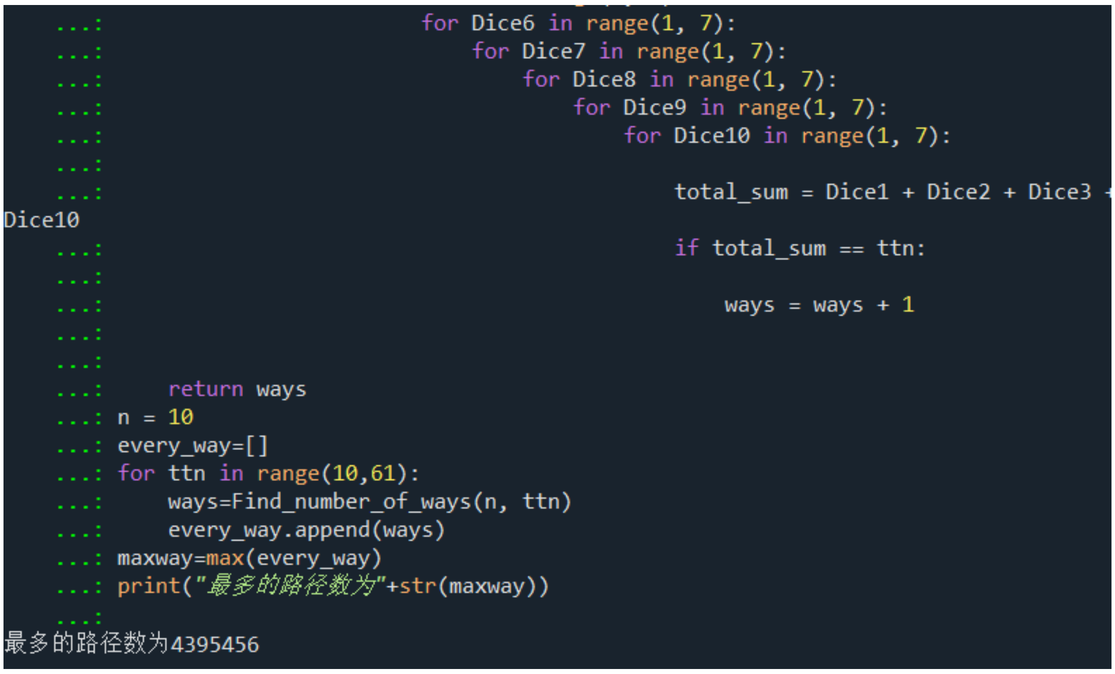


```
In [23]:
    ...: def F(x):
    ...:     if x == 1:
    ...:         return 1
    ...:     if x % 3 == 0:
    ...:         #x的取整函数
    ...:         then_x = x // 3
    ...:     else:
    ...:         then_x = x // 3 + 1
    ...:     return F(then_x) + 2 * x
    ...: N = 10
    ...: for x in range(1, N + 1):
    ...:     print("F(" + str(x) + ") = " + str(F(x)))
F(1) = 1
F(2) = 5
F(3) = 7
F(4) = 13
F(5) = 15
F(6) = 17
F(7) = 21
F(8) = 23
F(9) = 25
F(10) = 33
```

3.第三题使用了六次 for 循环假设得到的数为 60，方法次数为一，最多路径次数为 4395456，代码及结果：

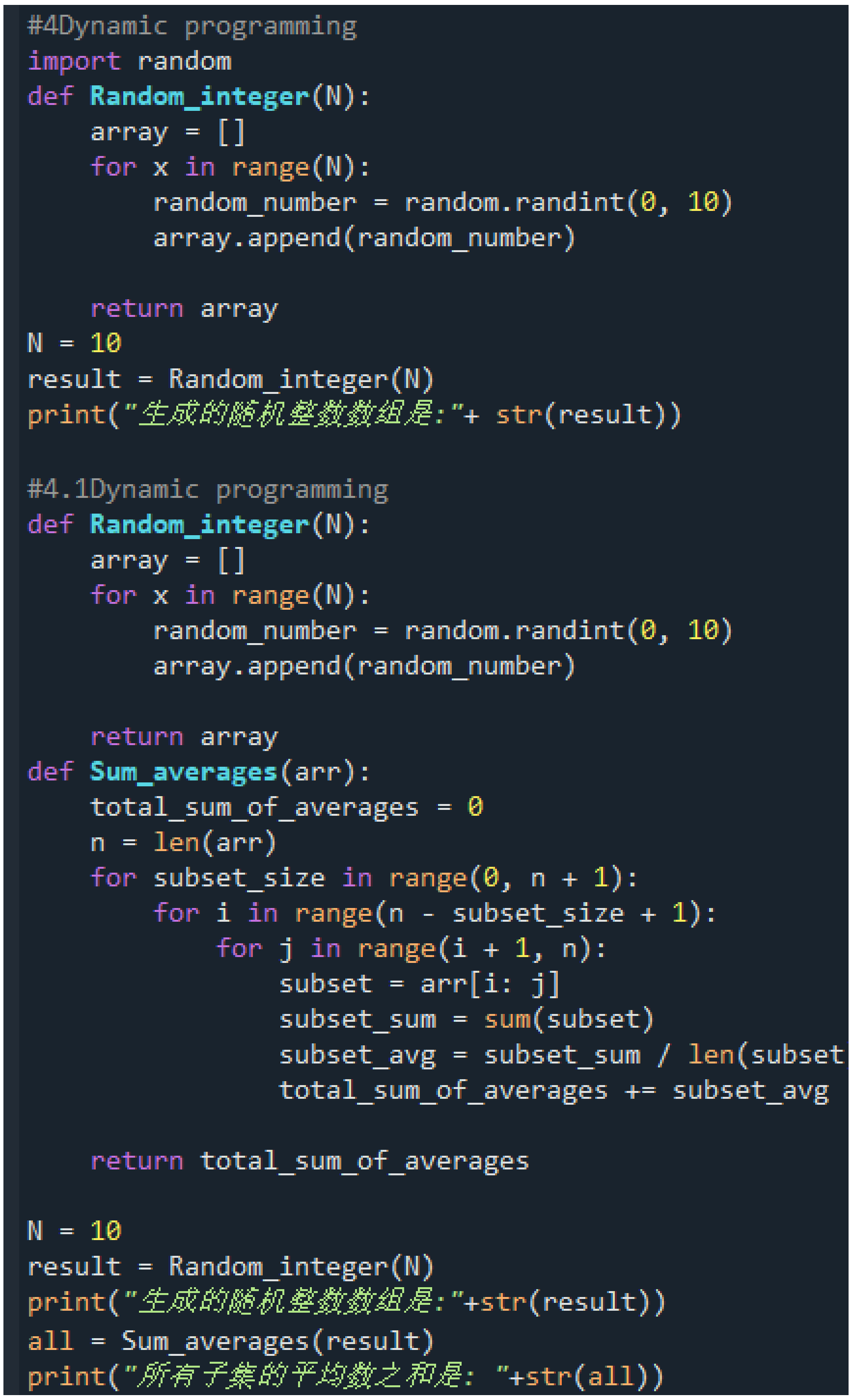


```
#Dice rolling  3.1
#n为色子数 ttn为数总和
def Find_number_of_ways(n, ttn):
    ways = 0
    for Dice1 in range(1, 7):
        for Dice2 in range(1, 7):
            for Dice3 in range(1, 7):
                for Dice4 in range(1, 7):
                    for Dice5 in range(1, 7):
                        for Dice6 in range(1, 7):
                            for Dice7 in range(1, 7):
                                for Dice8 in range(1, 7):
                                    for Dice9 in range(1, 7):
                                        for Dice10 in range(1, 7):

                                            total_sum = Dice1 + Dice2 + Dice3 + Dice4+ Dice5 + Dice6 + Dice7 +Dice8+ Dice9 + Dice10
                                            if total_sum == ttn:

                                                ways = ways + 1


    return ways
n = 10
ttn = 60
result = Find_number_of_ways(n, ttn)
print("用 " + str(n) + " 个骰子得到和 " + str(ttn) + " 的方法数是: " + str(result))

#3.2
def Find_number_of_ways(n, ttn):
    ways = 0
    for Dice1 in range(1, 7):
        for Dice2 in range(1, 7):
            for Dice3 in range(1, 7):
                for Dice4 in range(1, 7):
                    for Dice5 in range(1, 7):
                        for Dice6 in range(1, 7):
                            for Dice7 in range(1, 7):
                                for Dice8 in range(1, 7):
                                    for Dice9 in range(1, 7):
                                        for Dice10 in range(1, 7):

                                            total_sum = Dice1 + Dice2 + Dice3 + Dice4+ Dice5 + Dice6 + Dice7 +Dice8+ Dice9 + Dice10
                                            if total_sum == ttn:

                                                ways = ways + 1


    return ways
n = 10
every_way=[]
for ttn in range(10,61):
    ways=Find_number_of_ways(n, ttn)
    every_way.append(ways)
maxway=max(every_way)
print("最多的路径数为"+str(maxway))
```

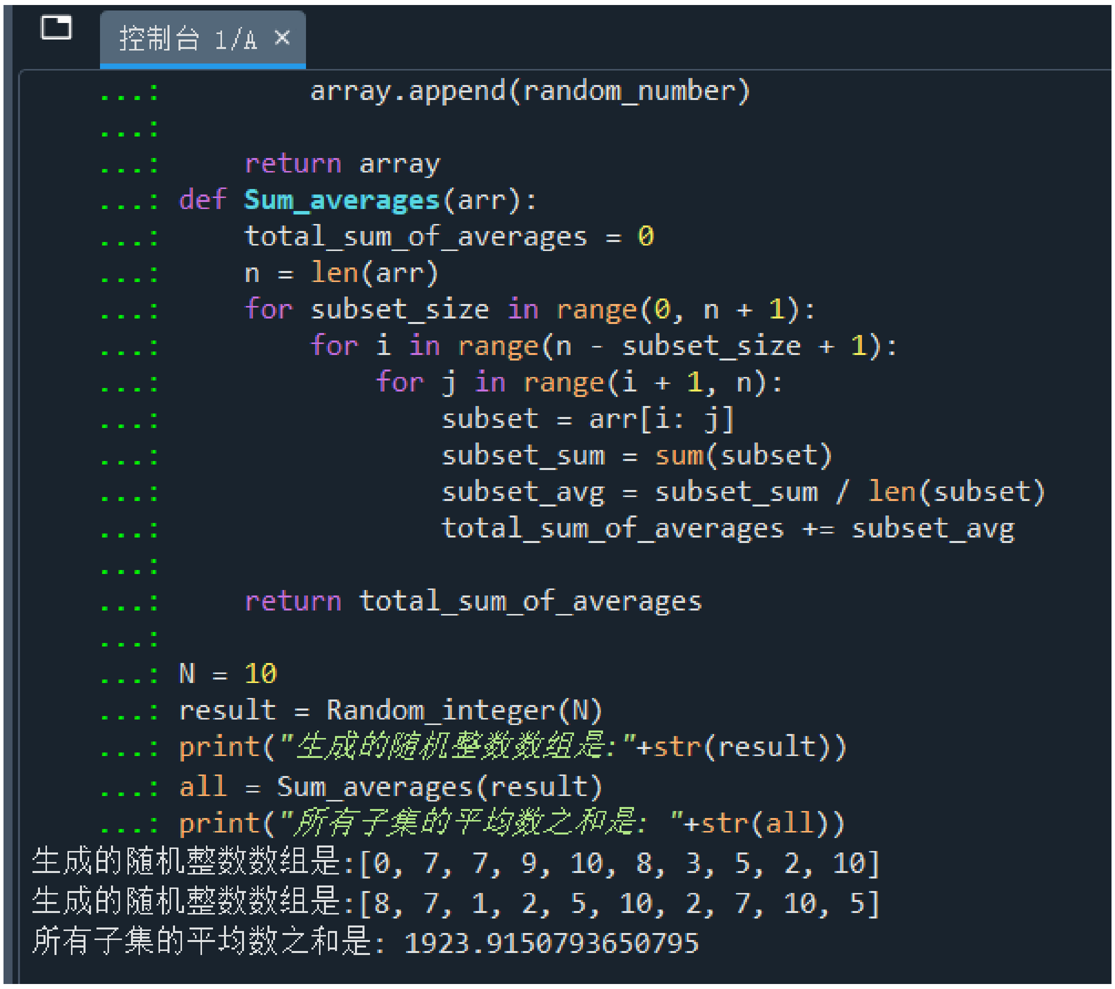


```
    ...:                         for Dice6 in range(1, 7):
    ...:                             for Dice7 in range(1, 7):
    ...:                                 for Dice8 in range(1, 7):
    ...:                                     for Dice9 in range(1, 7):
    ...:                                         for Dice10 in range(1, 7):
    ...:
    ...:                                             total_sum = Dice1 + Dice2 + Dice3 +
Dice10
    ...:                                             if total_sum == ttn:
    ...:
    ...:                                                 ways = ways + 1
    ...:
    ...:
    ...:     return ways
    ...: n = 10
    ...: every_way=[]
    ...: for ttn in range(10,61):
    ...:     ways=Find_number_of_ways(n, ttn)
    ...:     every_way.append(ways)
    ...: maxway=max(every_way)
    ...: print("最多的路径数为"+str(maxway))
    ...:
最多的路径数为4395456
```

4 第四题使用了·求和与取长度公式，取 N=10,得到结果及代码



```
#4Dynamic programming
import random
def Random_integer(N):
    array = []
    for x in range(N):
        random_number = random.randint(0, 10)
        array.append(random_number)

    return array
N = 10
result = Random_integer(N)
print("生成的随机整数数组是:"+ str(result))

#4.1Dynamic programming
def Random_integer(N):
    array = []
    for x in range(N):
        random_number = random.randint(0, 10)
        array.append(random_number)

    return array
def Sum_averages(arr):
    total_sum_of_averages = 0
    n = len(arr)
    for subset_size in range(0, n + 1):
        for i in range(n - subset_size + 1):
            for j in range(i + 1, n):
                subset = arr[i: j]
                subset_sum = sum(subset)
                subset_avg = subset_sum / len(subset
                total_sum_of_averages += subset_avg

    return total_sum_of_averages

N = 10
result = Random_integer(N)
print("生成的随机整数数组是:"+str(result))
all = Sum_averages(result)
print("所有子集的平均数之和是: "+str(all))
```



控制台 1/A

```
   ...:         array.append(random_number)
   ...:
   ...:     return array
   ...: def Sum_averages(arr):
   ...:     total_sum_of_averages = 0
   ...:     n = len(arr)
   ...:     for subset_size in range(0, n + 1):
   ...:         for i in range(n - subset_size + 1):
   ...:             for j in range(i + 1, n):
   ...:                 subset = arr[i: j]
   ...:                 subset_sum = sum(subset)
   ...:                 subset_avg = subset_sum / len(subset)
   ...:                 total_sum_of_averages += subset_avg
   ...:
   ...:     return total_sum_of_averages
   ...:
   ...: N = 10
   ...: result = Random_integer(N)
   ...: print("生成的随机整数数组是:"+str(result))
   ...: all = Sum_averages(result)
   ...: print("所有子集的平均数之和是: "+str(all))
生成的随机整数数组是:[0, 7, 7, 9, 10, 8, 3, 5, 2, 10]
生成的随机整数数组是:[8, 7, 1, 2, 5, 10, 2, 7, 10, 5]
所有子集的平均数之和是: 1923.9150793650795
```

Ps：感谢韩志坚同学提供的思路与函数，并帮忙改正

5.使用 numpy 分别设 mn 为 6/5，4/4，代码及路径结果如下“

[7]:

```python
#5.1
import numpy as np
N = 6
M = 5
matrix = np.random.randint(0, 2, size=(N, M))
matrix[0, 0] = 1
matrix[N-1, M-1] = 1
print(matrix)
```

```
[[1 0 1 1 0]
 [1 1 0 1 0]
 [1 0 1 0 0]
 [1 1 0 1 0]
 [1 0 0 1 1]
 [0 1 1 1 1]]
```

[5]:

```python
#5.2
import numpy as np
N = 4
M = 4
matrix = np.random.randint(0, 2, size=(N, M))
matrix[0, 0] = 1
matrix[N-1, M-1] = 1
def Count_path(matrix):
    matrix[0, 0] = 1
    for i in range(N):
        for j in range(M):
            if matrix[i, j] == 0:
                continue
            else:
                if i > 0:
                    matrix[i, j] = matrix[i, j]+ matrix[i - 1, j]
                if j > 0:
                    matrix[i, j]= matrix[i, j - 1]+ matrix[i, j]
    return matrix[N-1, M-1]
result = Count_path(matrix)
print(matrix)
print("从左上角到右下角的路径数为"+ str(result))
```

```
[[1 2 3 0]
 [2 0 0 1]
 [0 0 0 0]
 [1 0 0 1]]
从左上角到右下角的路径数为1
```