In [2]:

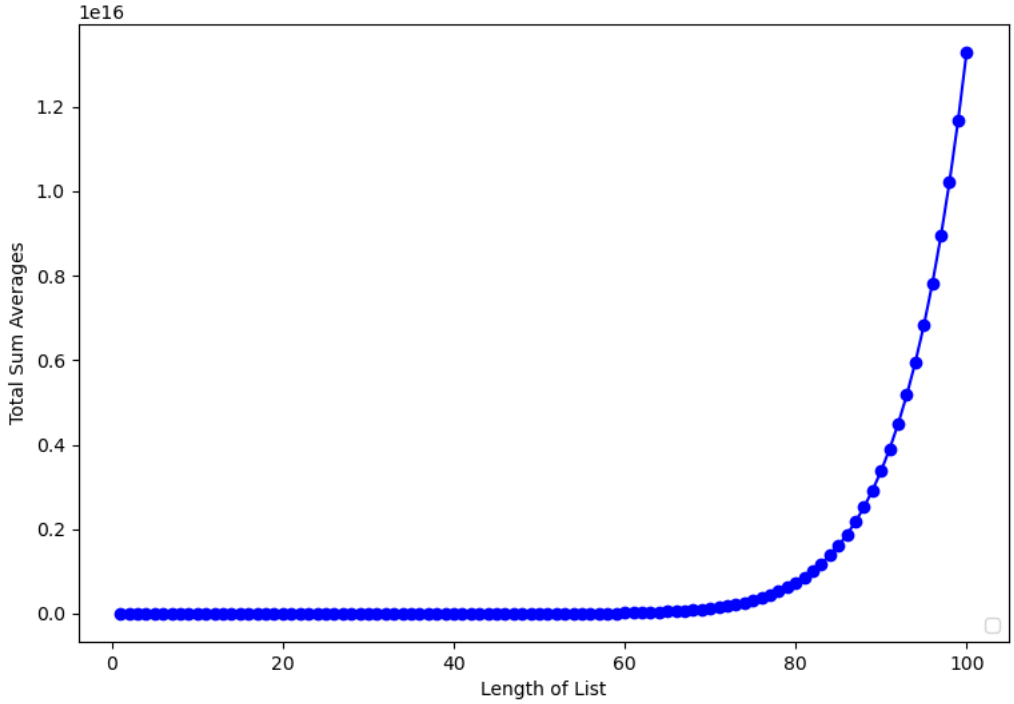
```
#coding=utf8

import math
import numpy as np
import matplotlib.pyplot as plt

#Q1 Flowchart
def Ascending(listofNum):
    '''

    :param listofNum: 需要比大小的数组
    :return: 参数输入错误，返回Error
    '''
    if not isinstance(listofNum, list) or len(listofNum) < 3:
        return "Error"
    else:
        #取输入数组的前三位，避免处理大于三个数的数组
        tempList = listofNum[:3]
        #对数组进行两次遍历，若前一位小于后一位则将其位置互换
        for i in range(2):
            for j in range(2 - i):
                if tempList[j] < tempList[j + 1]:
                    tempList[j], tempList[j + 1] = tempList[j + 1], tempList[j]
    #计算x+y-10
    resultList = np.multiply(tempList, [1, 1, -10])
    result = np.sum(resultList, axis=0)
    print("Q1: After comparing, x+y-10z = ", result)

#Q2 Continuous celing function
def CCF(listofNum):
    '''

    :param listofNum: 需要取整的数组
    :return: 参数输入错误，返回Error
    '''
    if not isinstance(listofNum, list):
        return "Error"
    yList = []
    for x in listofNum:
        if not isinstance(x, int) or x < 0:
            return "Error"
        #初始化
        y = x*2
        #循环至ceil(x/3)=1结束
        while math.ceil(x / 3) != 1:
            x = math.ceil(x / 3)
            y += math.ceil(x)*2
        #F(x) = F(1) + y
        yList.append(y + 1)
    print("Q2: Result of Continuous Ceiling Function:", yList)

#Q3 Dice rolling
def Find_number_of_ways(dices):
    '''

    :param dices: 骰子个数
    :return:
    '''
    listNum = []
```

```
    #初始化，6个骰子可以掷出的和为(10,60)，即51种
    Number_of_ways = [0 for i in range(51)]
    listNum.append(list(range(1,7)))
    #对n个骰子进行遍历，sumofNum(n) = sum(sumofNum(n-1) + (1, 6))
    for n in range(dices - 1):
        temp = []
        for i in listNum[-1]:
            for j in listNum[0]:
                temp.append(i + j)
        listNum.append(temp)
    #对sumofNum(10)进行遍历
    for i in listNum[-1]:
        Number_of_ways[i-10] += 1
    #Q3-1
    print("Q3:")
    print("(1): Sum of ways:", len(listNum[-1]))
    maxValue = max(Number_of_ways)
    max_x = Number_of_ways.index(maxValue) + dices
    #Q3-2
    print("(2): Number_of_ways:", Number_of_ways)
    print("Number of maximum ways:", max_x)

#Q4 Dynamic programming
#Q4-1
def Random_integer(N):
    '''

    :param N: 随机数组的长度
    :return: 返回生成的随机数组
    '''
    random_list = np.random.randint(1, 10, N)
    return random_list.tolist()
#Q4-2
def Sum_averages(listofNum, n, sets):
    '''

    :param listofNum: 需要进行组合计算的数组
    :param n: 取n个数进行组合
    :param sets: 所有n的组合数集合
    :return:
    '''
    if not isinstance(listofNum, list):
        return "Error"
    num = len(listofNum)
    #递归出口，当n递归至数组长度时停止递归
    if n == num:
        sets.append(np.mean(listofNum))
        sumAverage = np.sum(sets)
        return sumAverage
    #初始化，并求得按正序组合的组合数
    indexofNum = list(range(n))
    sets.append(np.mean([listofNum[i] for i in indexofNum]))
    while True:
        #将上述初始化的正序组合的组合数从后往前逐个移动，当其全部移动至其对应的最后一位时跳出循环
        for i in reversed(range(n)):
            if indexofNum[i] != i + num - n:
                break
        else:
            #递归入口，当第n个组合数全部组合完成后，进行n+1个组合计算
            return Sum_averages(listofNum, n + 1, sets)
            break
```

```
        indexofNum[i] += 1
        start = i + 1
        #将中间数进行平移组合
        for j in range(start, n):
            indexofNum[j] = indexofNum[j - 1] + 1
        sets.append(np.mean([listofNum[i] for i in indexofNum]))
#Q4-3
def plot(N):
    '''

    :param N: 长度为(1, N)之间N个数的Sum_averages
    :return:
    '''
    Length_of_list, Total_sum_averages, sets = [i for i in range(1, N + 1)], [], []
    #遍历(1,N)，并调用Sum_averages方法，得到N个数组的组合数平均和组成的数组Total_sum_averages
    for i in range(1, N + 1):
        listofNum = Random_integer(i)
        Total_sum_averages.append(Sum_averages(listofNum, 1, sets))
    plt.figure(figsize=(9, 6))
    plt.xlabel("Length of List")
    plt.ylabel("Total Sum Averages")
    plt.plot(Length_of_list, Total_sum_averages, linestyle='-', marker='o', color='b', label="direc
    print("Q4: Fig. Plot of total sum averages")
    plt.show()
    plt.close()

#Q5 Path counting
#Q5-1
def Create_matrix(N, M):
    '''

    :param N: 行数
    :param M: 列数
    :return: 返回生成的矩阵
    '''
    data = np.mat(np.random.randint(2, size = (N, M)))
    data[0, 0] = 1
    data[N - 1, M - 1] = 1
    return data
#Q5-2
def Count_path(n, N, M):
    '''

    :param n: 计算次数
    :param N: 行数
    :param M: 列数
    :return:
    '''
    total_number = []
    #最外层循环即计算次数
    for _ in range(n):
        matrix = Create_matrix(N, M)
        rows, cols = matrix.shape
        #找到最后一行/列最右/下的0，并将其左/上的数全部赋值为0
        for row in reversed(range(rows)):
            if matrix[row, cols-1] == 0:
                matrix[:row, cols-1] == 0
        for col in reversed(range(cols)):
            if matrix[row-1, col] == 0:
                matrix[row-1, :col] == 0
        #第i行第j列的path数即:path(i, j) = path(i-1, j-1)，并从倒数第二行/列开始计算
```

```
        for i in reversed(range(rows-1)):
            for j in reversed(range(cols-1)):
                if matrix[i, j] == 0:
                    continue
                else:
                    matrix[i, j] = matrix[i+1, j] + matrix[i, j+1]
        #计算结束后，取第一行第一列的数即从左上向右下移动的path数
        total_number.append(matrix[0, 0])
    # Q5-3
    Mean_of_total_number = np.mean(total_number)
    print("Q5:")
    print("Mean of total number:", Mean_of_total_number)

if __name__=="__main__":
    #Q1 Flowchart
    listofNum = [10, 5, 1]
    Ascending(listofNum)

    #Q2 Continuous celing function
    listofNum = [1, 4, 8, 9, 15]    #list for test
    CCF(listofNum)

    #Q3 Dice rolling
    Find_number_of_ways(10)

    #Q4 Dynamic programming
    plot(100)

    #Q5 Path counting
    Count_path(1000, 10, 8)
```

```
Q1: After comparing, x+y-10z =  5
Q2: Result of Continuous Ceiling Function: [3, 13, 23, 25, 45]
Q3:
(1): Sum of ways: 60466176
(2): Number_of_ways: [1, 10, 55, 220, 715, 2002, 4995, 11340, 23760, 46420, 85228, 1
47940, 243925, 383470, 576565, 831204, 1151370, 1535040, 1972630, 2446300, 2930455,
3393610, 3801535, 4121260, 4325310, 4395456, 4325310, 4121260, 3801535, 3393610, 293
0455, 2446300, 1972630, 1535040, 1151370, 831204, 576565, 383470, 243925, 147940, 85
228, 46420, 23760, 11340, 4995, 2002, 715, 220, 55, 10, 1]
Number of maximum ways: 35
Q4: Fig. Plot of total sum averages
```



Q5:
Mean of total number: 0.883