```
#flowchat
str_a=input('please input a:')
str_b=input('please input b:')
str_c=input('please input c:')
a=float(str_a)
b=float(str_b)
c=float(str_c)
if a>b:
    if b>c:
        list=[a,b,c]
    else:
        if a>c:
            list=[a,c,b]
        else:
            list=[c,a,b]
else:

        list=[c,b,a]
x=list[0]
y=list[1]
z=list[2]
rs=x+y-10*z
print(rs)
#Continuous ceiling function
import numpy as np
#获取输入的元素
list=[]
print("please input number,untill input 'done':")
while True:
    user_input=input()
    if user_input=='done':
        break
    list.append(user_input)
int_list=[int(x) for x in list]
list2=[]#储存结果的列表
#循环
for i in int_list:
    rs=0
#每迭代一次，结果加 2
    while np.ceil(i/3)!=1:
        rs=rs+2
        i=np.ceil(i/3)
#因为迭代的次数少一次，而且需要加上 F（1），所以最后结果还要加 3
    rs=rs+3
```

```
        list2.append(rs)
print(list2)



#Dice rolling
def find_number_of_ways(x):
        import copy
        x=int(x)
        #通过 10 次循环，把所有的结果储存在列表 list 中
        list=[]
        for i1 in range(1,7):
                for i2 in range(1,7):
                        for i3 in range(1,7):
                                for i4 in range(1,7):
                                        for i5 in range(1,7):
                                                for i6 in range(1,7):
                                                        for i7 in range(1,7):
                                                                for i8 in range(1,7):
                                                                        for i9 in range(1,7):
                                                                                for i10 in range(1,7):
                                                                                        rs=i1+i2+i3+i4+i5+i6+i7+i8+i9+i10
                                                                                        list.append(rs)
        #得到列表长度
        l1=len(list)
        list2=copy.copy(list)
        #删除列表中所有与选中数字相同的元素，前后列表长度差即为得到该数字的途径数
        #list=list(filter((x)._ne_,list))
        try:
                while True:
                        list.remove(x)
        except ValueError:
                        pass
        l2=len(list)
        rs2=l1-l2
        return rs2,list2
out_rs2,out_list2=find_number_of_ways(10)
print('there are',out_rs2,'ways to get',10)
#3.2 由于数据太大，需要较长的运行时间
number_of_ways=[]
for i in range(10,61):
        l1=len(out_list2)
        try:
                while True:
                        out_list2.remove(i)
```

```
    except ValueError:
            pass
    l2=len(out_list2)
    rs2=l1-l2
    number_of_ways.append(rs2)

    #Dynamic programming
    #4.1    random_integer
    import numpy as np
    import random
    N=input('please input a number:')
    N=int(N)

    list=[]
    i=1
    while i<=N:
            list.append(random.randint(0,10))
            i+=1



    print(list)
    #4.2     sum average
    #对于原数列，需要分别求含有 1、2、...N 个元素的子集的平均值
    #由数学知识可知，含有 n 个元素的子集的取法为 N！/（n!*(N-n)!）
    #则所有含有 n 个元素的子集的元素数量和=子集个数*每个子集中的元素数=N！/（n!*(N-n)!）*n
    #对于 n 元素子集，由于取法不重不漏，所以每个元素出现的次数相同，都是 N！/（n!*(N-n)!）*n/N
    #所以 n 元素子集平均值=（元素和）*N！/（n!*(N-n)!）/N
    import math
    sum_ave=[]
    sum=0
    int_list=[int(x) for x in list]
    for i in range(N):
        sum=sum+int_list[i]
    for i in range(1,N+1):
        sum_ave.append(sum*math.factorial(N)/math.factorial(i)/math.factorial(N-i)/N)
    sum_average=0
    for i in range(N):
        sum_average=sum_average+sum_ave[i]
    print(sum_average)
    #4.3
    total_sum_average=[]
    for j in range(1,101):
```

```
        list=[]
        i=1
        while i<=j:
                list.append(random.randint(0,10))
                i+=1

        sum_ave=[]
        sum=0
        int_list=[int(x) for x in list]
        for i in range(j):
            sum=sum+int_list[i]
        for i in range(1,j+1):
            sum_ave.append(sum*math.factorial(j)/math.factorial(i)/math.factorial(j-i)/j)
        sum_average=0
        for i in range(j):
            sum_average=sum_average+sum_ave[i]

        total_sum_average.append(sum_average)
    print(total_sum_average)

    #Path counting
    #5.1
    import numpy as np
    M=input('please input the number of rows:')
    N=input('please input the number of columns:')
    M=int(M)
    N=int(N)
    matrix=np.random.random((M,N))
    matrix=np.round(matrix)
    matrix[0][0]=1
    matrix[-1][-1]=1
    print(matrix)
    #5.2
    #对于该问题，由于每次前进一步。即序列数之和每次增加 1，如果我们将矩阵以副对角线方向分层，则每次跨越一层
    #假如原矩阵是全 1 矩阵，则由数学知识可知，抵达某层的方法数可以安装杨辉三角的计算规则得出
    #再考虑矩阵中出现 0 的影响：出现 0 说明此路不通，意味着之前到某 0 位置的方法全部无效，并且该位置不对此后的位置产生影响
    #在杨辉三角中，把相应位置变成 0，可以得到相同的效果
    def count_path(M,N,matrix):
        matrix1=np.zeros((M,N))
        matrix1[1][0]=matrix[1][0]
        matrix1[0][1]=matrix[0][1]
```

```
        matrix1[-1][-1]=1
        #此时原矩阵 matrix 相当于布尔矩阵，只用于条件判断，计算在 matrix1 中进行，在对第 2 层赋值后，matrix1 设置为全 0 矩阵可以避免一些 bug
        #计算矩阵左边缘和上边缘
        a1=[]
        a2=[]
        for i in range(N):
            a1.append(matrix[0][i])
        for i in range(M):
            a2.append(matrix[i][0])
        index1=next((i for i,x in enumerate(a1) if x==0),False)
        index2=next((i for i,x in enumerate(a2) if x==0),False)
        if index1:
            for i in range(index1):
                matrix1[0][i]=1
        if index2:
            for i in range(index2):
                matrix1[i][0]=1
    #计算非方阵
        if M!=N:
            T=min(M,N)
            T2=abs(N-M)
            for i in range(2,T):
                for j in range(i):
                    if j*(i-j)!=0:
                        if matrix[j][i-j]!=0:
                            matrix1[j][i-j]=matrix1[j-1][i-j]+matrix1[j][i-j-1]
                        else:
                            matrix1[j][i-j]=0
            for i in range(T,T+T2):
                if M>N:
                    for j in range(i-N+1,i):
                        if i-j==0:
                            matrix1[j][i-j]=matrix1[j-1][i-j]
                        else:
                            if matrix[j][i-j]!=0:
                                matrix1[j][i-j]=matrix1[j-1][i-j]+matrix1[j][i-j-1]
                            else:
                                matrix1[j][i-j]=0
                if M<N:
                    for j in range(i-M+1,i):
                        if i-j==0:
                            matrix1[i-j][j]=matrix1[i-j][j-1]
                        else:
```

```
                    if matrix[i-j][j]!=0:
                        matrix1[i-j][j]=matrix1[i-j-1][j]+matrix1[i-j][j-1]
                    else:
                        matrix1[i-j][j]=0


        for i in range(T+T2,M+N-1):
            if M>N:
                for j in range(i-N+1,M):
                    if matrix[j][i-j]!=0:
                        matrix1[j][i-j]=matrix1[j-1][i-j]+matrix1[j][i-j-1]
                    else:
                        matrix1[j][i-j]=0
            if M<N:
                for j in range(i-M+1,N):
                    if matrix[i-j][j]!=0:
                        matrix1[i-j][j]=matrix1[i-j-1][j]+matrix1[i-j][j-1]
                    else:
                        matrix1[i-j][j]=0
#计算方阵
    else:
        for i in range(2,M):
            for j in range(i):
                if j*(i-j)!=0:
                    if matrix[j][i-j]!=0:
                        matrix1[j][i-j]=matrix1[j-1][i-j]+matrix1[j][i-j-1]
                    else:
                        matrix1[j][i-j]=0
        for i in range(M,M+N-1):
            for j in range(i-N+1,M):
                if matrix[j][i-j]!=0:
                    matrix1[j][i-j]=matrix1[j-1][i-j]+matrix1[j][i-j-1]
                else:
                    matrix1[j][i-j]=0
    pa=np.copy(matrix1[-1,-1])
    return pa
pa=count_path(M,N,matrix)
print(pa)

#5.3
import numpy as np
M=input('please input the number of rows:')
N=input('please input the number of columns:')
M=int(M)
```

```
N=int(N)
mean_p=[]
sum=0
for t in range(1000):
    matrix=np.random.random((M,N))
    matrix=np.round(matrix)
    matrix[0][0]=1
    matrix[-1][-1]=1
    pa=count_path(M,N,matrix)
    pa=int(pa)
    sum=sum+pa
    mean_p.append(pa)
mean_path=sum/1000
print(mean_path)
```