Assignment 01

1. Flowchart

最初思路是把每条能输出结果的路径分别单独写出来,后请教吴星沂助教认为思路太过于繁琐,可直接采用 if else 来进行判断。

```
#Define a function(10.15. Discuss with teaching assistant W def Print_values(x,y,z):
    return x+y-10+2

#Define a function to determine abc def values(a,b,c):

#a>b>c
    if a>b:
        result=Print_values(a,b,c)

#a>b,bcc,a>c
    else:
        result=Print_values(a,c,b)
    else:
        result=Print_values(c,a,b)

#a<b,b>c,a>c
    else:
        result=Print_values(c,b,a)
    print(result)

values(10,5,1)
```

2. Continuous ceiling function

思路: 先定义函数再确定 x 的取值范围

定义公式时 ceil 函数无法运行,通过 Google 发现需要先 import math 才能调用 ceil 函数。

写完后发现 x=1 时函数等于 3, 搜索 CSDN 案例发现应单独定义 x=1 的情况。

最后用 result=F(x)输出时,发现只能输出一个结果,请教吴星沂助教给出解决办法将 result 改为 list 输出列表。

3. Dice rolling

3.1

错误思路:将3.1理解为十个骰子掷出后计算朝上面数字的总和。将骰子掷出抽象为随机生成一个1到6之间的数字,然后循环十次,将结果放入一个列表里再进行求和。

正确思路:和彭文乐同学讨论后,并请教吴星沂助教后发现上述思路错误,应当计算值为x时,产生x值的组合方式有多少种。通过讨论后,采用最直接暴力的穷举法(6^{10})。

```
def Find_number_of_ways(x):
#设置一个计数器counts,用于记录总和为x的骰子组合的数量
     counts=0
                                                                                                         帮助 变量浏览器 绘图
     for a in range(1,7):
    for b in range(1,7):
        for c in range(1,7):
                                                                                                 □ 控制台 1/A X
                     for d in range(1,7):
for e in range(1,7):
                                                                                                       ...: x=20
                                                                                                 H=Find_number_of_ways(x)
                                          g in range(1,7):
                                                                                                         .: print(H)
                                               h in range(1,7)
                                                                                                 85228
                                                for i in range(1,7):
for j in range(1,7):
sum=a+b+c+d+e+f+g+h+i+j
                                                              sum==x:
                                                                counts += 1
     return(counts)
H=Find_number_of_ways(x)
print(H)
```

因为写法有点 ugly 所以通过 Google 和 AI 的案例选择导入 itertools 模块,将写法变得简洁一些。

```
#尝试2(想办法简化)
#來自google、导入itertools模块,能高效地处理循环和迭代任务
import itertools

def Find_number_of_ways(x):

    counts = 0
    #用itertools.product生成所有可能的散子组合(1-6,重复10次)
    for outcome in itertools.product(range(1, 7), repeat=10):
        #如果总和等于下,计数器加一
        if sum(outcome)==x:
            counts+=1

    return(counts)

x=20
H=Find_number_of_ways(x)
print(H)
```

思路:调用3.1的函数,再将x在10到60之间依次取值。

如何取最值通过 Google 找到案例模仿写。顺手搜了一下算平均值等写法。

```
#3.2
import itertools
                                                                                                                                             帮助 变量浏览器 绘图 文件
                                                                                                                                  □ 控制台 1/A X
                                                                                                                                                                         î
def Find_number_of_ways(x):
        #用itertools.product生成所有可能的骰子组合(1-6,重复10次)
                                                                                                                                  min_ways=min(Number_of_ways)
        for outcome in itertools.product(range(1, 7), repeat=10):
                                                                                                                                print(max_ways,min_ways)
[1, 10, 55, 220, 715, 2002,
4995, 11340, 23760, 46420,
85228, 147940, 243925, 383470,
576565, 831204, 1151370,
1535040, 1972630, 2446300,
2930455, 3393610, 3801535,
4121260, 4325310, 4395456,
4325310, 4121260, 3801535,
3393610, 2930455, 2446300,
1972630, 1535040, 1151370,
831204, 576565, 383470, 243925,
147940, 85228, 46420, 23760,
11340, 4995, 2002, 715, 220, 55,
10, 1]
4395456 1
              if sum(outcome)==x:
                     counts+=1
       return(counts)
Number_of_ways = []
for x in range(10, 61):
       {\tt Number\_of\_ways.append(Find\_number\_of\_ways(x))}
print(Number_of_ways)
max ways=max(Number of ways)
min_ways=min(Number_of_ways)
print(max_ways,min_ways)
```

4. Dynamic programming

4.1

思路: 先能使用随机命令生成 N 个随机数,放入列表中,再通过 Google 搜索限制数取值的范围的方法。(通过 3.1 的错误思路在 Google 上学会 import random)

```
import random

def Random_integer(N):
    random_int=[]

for _ in range(N):
    #N个元素从0到10的范围内随机补充
    random_number = random.randint(0, 10)
    random_int.append(random_number)

return(random_int)

N=20
H=Random_integer(N)
print(H)

N=20
H=Random_integer(N)
print(H)
```

4.2

思路: 先给出一个简单数组,定义一个函数使其能生成该数组的每一个子集。再将生成的子集放入列表中,对列表元素求和再求平均。

通过 Google 案例学会 from itertools import combinations, combinations 函数能生成数组子集。

因为平均值的输出结果很奇怪,请教吴星沂助教后发现之前写法只是单纯将所有结果求和再算平均,并不是每个子集的平均值再求和。结合助教意见和 Google 案例,

在生成每个子集的后面添加计算平均值的代码,再将结果放入列表中求和。

```
#来自Google调用函数生成一个数组的每个子集
from itertools import combinations
                                                                                                          帮助 变量浏览器 绘图 文件
def Sum averages(arr):
                                                                                                  □ 控制台 1/A X
     subsets=[]
                                                                                                             arr=(1,2,3)
num=Sum_averages(arr)
print(num)
total_sum=sum(num)
print(total_sum)
     n=len(arr)
      for i in range(1,n+1):
           #利用combinations找出所有大小为i的子集
                                                                                                 (1,)
(2,)
(3,)
(1, 2)
(1, 3)
(2, 3)
(1, 2, 3)
(1, 2, 3)
(1, 2, 3)
(1, 2, 3)
(1, 4, 2, 3)
(1, 2, 3)
(1, 2, 3)
(1, 4, 2, 3)
(1, 2, 3)
           for sub in combinations(arr, i):
               print(sub)
#计算方法的调整来自Ai,计算每个生成的子集的和,然后计算平均值
                sets_sum=sum(sub)
                average=sets_sum/i
subsets.append(average)
     return(subsets)
arr=(1,2,3)
num=Sum_averages(arr)
print(num)
total_sum=sum(num)
print(total_sum)
```

4.3

关于数组来源,个人产生了两个理解所以写了两个版本。但出现相同问题,计算小数目的时候能正确输出,当计算 N 逐渐增加到 100 时出现 MemoryError 的错误。

4.3.1

思路:将 N 是数组,其长度和元素从 1 依次递增到 100。先写出一个函数依次生成数组 N (个数先从 1 到 6 开始),再调用 4.2 的函数进行计算。将结果放在一个列表中,再调用结果进行画图。

通过 Google 搜索如何写一个依次增加的列表,通过案例写成用于得到数组 N 的函数。

调用函数时出现问题。询问 AI 后得出,上一个函数返回的是一个包含多个数组的列表,不是单个的数组,所以直接带入 4.2 函数时会出现错误。要分别带入列表中的每一个函数。

根据课程网站中的画图代码,实现绘图。向吴星沂助教询问如何查看绘图。

```
#4.3.1(最开始理解,N是数组,其长度和数组内的元素都是从1依次递增到100)
#调用函数生成一个数组的每个子集和绘图的模组
from itertools import combinations
import matplotlib.pyplot as plt
 #N的个数变化,由N产生的列表的长度也会随着变化,由此建立函数 def generate_increasing_arrays():
        increasing_arrays=[]
#使N能从1到100. 做实验先从1到20
for N in range(1, 21):
#创建一个随时变化并包含M的数组、并返回到之前的列表
arr =list(range(1, N + 1))
increasing_arrays.append(arr)
 return(increasing_arrays)
#实验这个函数能否生成长度和元素都从1依次递增的数组
#arrays = generate_increasing_arrays()
#print(arrays)
 def Sum_averages(arr):
       新用丁森根数は近へ返

malen(anr)

郵館定毎个可能子集的大小

for in range(1,n+1);

#利用combinations找出所有大小为i的子集

for sub in combinations(arr, i);
                                                                                                                                                                                                  帮助 变量浏览器 绘图 文件
                                                                                                                                                                                   □ 控制台 1/A X
                                                                                                                                                                                                                                         .
                                                                                                                                                                                   12, 13, 14, 15, 16, 17, 18, 19,
20)
                       #计算方法的调整来自Ai. 计算每个生成的子集的和,然后计算平均值,再添加到列表中sets_sum(sub)
average=sets_sum(i
subsets.append(average)
                                                                                                                                                                                 20)
(1, 2, 3, 4, 5, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20)
(1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20)
        return(subsets)
arrays=generate_increasing_arrays()
                                                                                                                                                                                (1, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20)
(1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20)
(1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20)
(2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20)
(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20)
(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20)
 Total_sum_averages = []
 #generate_increasing_arrays函数返回的是一个包含多个数组的列表,不是单个的数组,所以直接带入4
for arr in arrays:
        subset_averages=Sum_averages(arr)
        total_sum_of_averages=sum(subset_averages)
        Total sum averages.append(total sum of averages)
print(total_sum_of_averages)
                                                                                                                                                                                   20)
(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20)
11010037.5
plt.plot(Total_sum_averages)
plt.xlabel('Array Index')
plt.ylabel('Total Sum of Averages')
plt.title('Total Sum of Averages for Arrays from 1 to 5')
plt.show()
```

4. 3. 2

思路: N 是数组的个数,利用第一问,只不过将个数从 1 依次递增到 100。其余同上。

```
from itertools import combinations import matplotlib.pyplot as plt
import random
def Random integer(N):
         random_number = random.randint(0, 10)
random_int.append(random_number)
     return(random int)
random_integers_list=[]
业计列表中的元素个数N从1选增到100(做试验将个数设为20个),元素取值范围仍然在0到10
#让列表中的元素介数N从口
for N in range(1,21):
random_integers_list.append(Random_integer(N))
#print(random integers list)
subsets=[]
#田干莊取数组长度
                                                                                                                          帮助 变量浏览器 绘图 文件
    朝日子教授を知られ

mlen(arp)

地域定義う可能子集的大小

for i in range(1,n+1);

郵利用combinations提出所有大小方i的子集

for sub in combinations(arr, i);
                                                                                                               □ 控制台 1/A X
     return(subsets)
#不能如4.3.1一样直接调用(rrays=Random_integer(N)),该函数生成的整数的长度,应直接让arr在r
for arr in random_integers_list:
     subset_averages=Sum_averages(arr)
Total_sum_averages.append(total_sum_of_averages)
print(Total_sum_averages)
```

5. Path counting

5. 1

思路: 先随机生成一个数组,再利用 reshape 将数组变为矩阵,再为特殊位置赋值。

```
#5.1
import numpy as np

RB 变量浏览器 绘图 文件

N = 3
M = 3

#生成一个一维数组,包含 N*M 个随机整数 (0 或 1)
random_array = np.random.randint(0, 2, (N,M))

#将一维数组转换成 N x M 的二维矩阵
matrix = random_array.reshape(N, M)

#设置左上角和右下角的元素为 1
matrix[0, 0] = 1
matrix[1, -1] = 1

print(matrix)

In [41]:
```

5.2

思路: 从初始位置开始往下探路。

从 Google 上搜索出来的案例都是从每个格子的末位置去找来路,不太好理解。所以向 AI 提问从初始位置向下和右摸索道路,根据 AI 提示完成代码。



5.3

思路: 调用前两题的函数重复 1000 次

