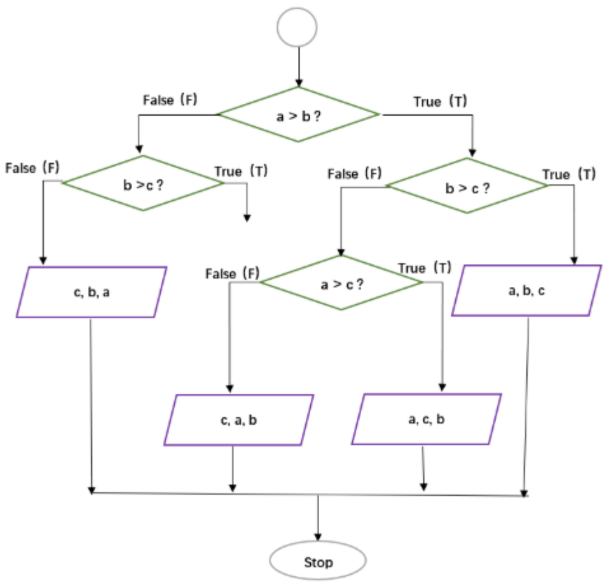
Homework#1

1. Flowchart

[10 points] Write a function `Print_values` with arguments `a`, `b`, and `c` to reflect the following flowchart. Here the purple parallelogram operator on a list `[x, y, z]` is to compute and print `x+y-10z`. Try your output with some random `a`, `b`, and `c` values. Report your output when `a = 5, b = 15, c = 10`.



思路：该题主要使用 if　else 语句进行判断，然后根据分支判断的结果赋值给相应的 x,yz,最后进行计算。具体而言，根据“a 是否大于 b”“b 是否大于 c”“a 是否大于 c”三个条件，将 a、b、c 排序为特定列表，再计算列表中“第一个元素+第二个元素-10×第三个元素”的结果并输出；若“a≤b 且 b>c”，则无输出。

2. Continuous celing function

[10 points] Given a list with `N` positive integers. For every element `x` of the list, find the value of continuous ceiling function defined as `F(x) = F(ceil(x/3)) + 2x`, where `F(1) = 1`.

思路: 该题主要运用递归思想，定义递归的终止条件为 F(1)=1；当 x>1 时，先计算 x 除以 3 的上取整结果，再递归调用 F 函数，最终结果为“递归返回值+2×x”。当用户输入一个正整数列表时，即可批量计算每个数的 F(x)结果。

3. Dice rolling

3.1 [15 points] Given `10` dice each with `6` faces, numbered from `1` to `6`. Write a function `Find_number_of_ways` to find the number of ways to get sum `x`, defined as the sum of values on each face when all the dice are thrown.

3.2 [5 points] Count the number of ways for any `x` from `10` to `60`, assign the number of ways to a list called `Number_of_ways`, so which `x` yields the maximum of `Number_of_ways`?

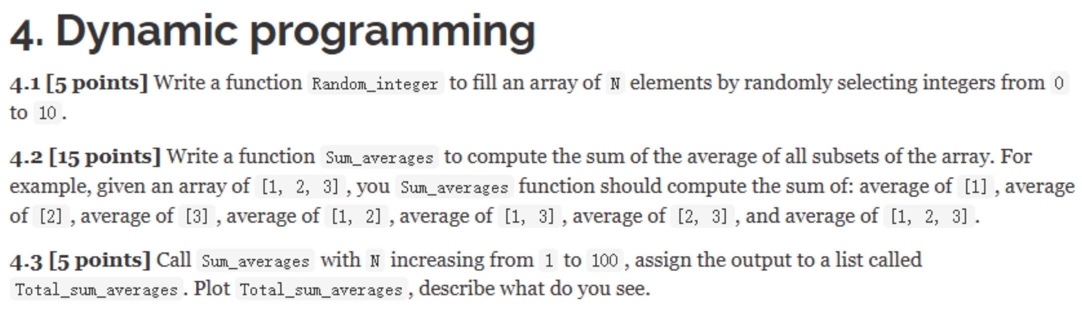
3.1

　　思路：此题也是通过递归遍历的方式，统计 10 个 6 面骰子掷出和为 x 的方式数。递归的终止条件就是当 10 个骰子全部掷完，且与所求的和相等则 return。遍历每个骰子的 1-6 点，累计 10 个骰子点数和等于 x 的次数；若 x 不在 10-60 范围内，直接返回 0。

3.2

思路：遍历 x 从 10 到 60 的所有可能，调用 3.1 的函数 count_ways 统计每个 x 的方式数，最终找到方式数最多的 x 并打印输出结果。

代码来源：学计算机的朋友的指导。



4. Dynamic programming

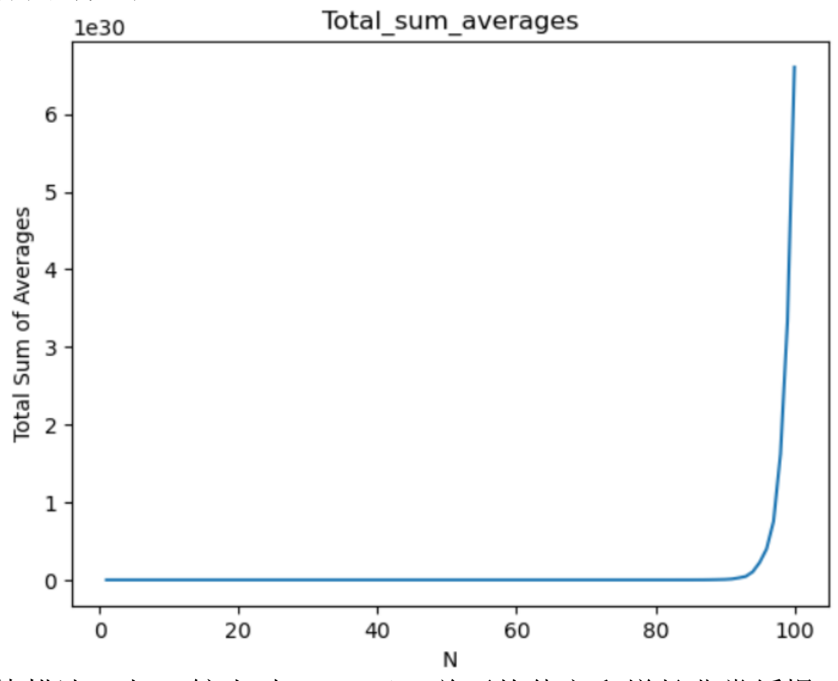
4.1 [5 points] Write a function `Random_integer` to fill an array of `N` elements by randomly selecting integers from `0` to `10`.

4.2 [15 points] Write a function `Sum_averages` to compute the sum of the average of all subsets of the array. For example, given an array of `[1, 2, 3]`, you `Sum_averages` function should compute the sum of: average of `[1]`, average of `[2]`, average of `[3]`, average of `[1, 2]`, average of `[1, 3]`, average of `[2, 3]`, and average of `[1, 2, 3]`.

4.3 [5 points] Call `Sum_averages` with `N` increasing from `1` to `100`, assign the output to a list called `Total_sum_averages`. Plot `Total_sum_averages`, describe what do you see.

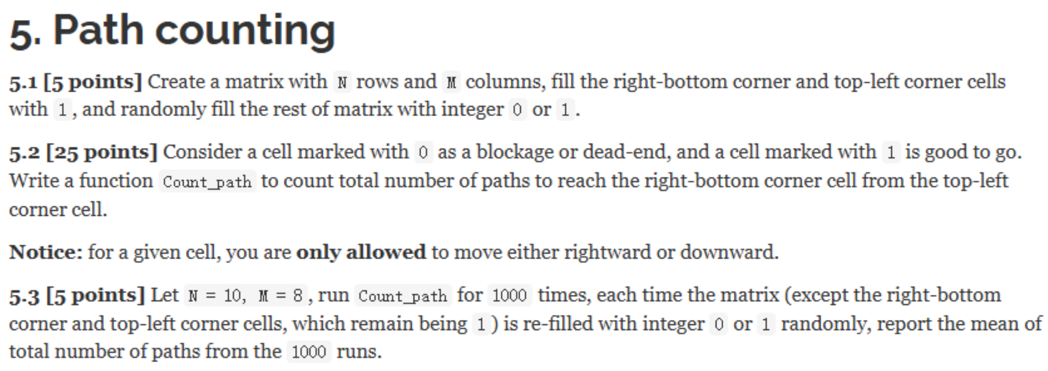
思路：此题先通过 Random_integer 生成含 n 个 0-10 随机整数的数组；再利用数学优化思路（避免枚举所有子集（使用枚举导致第三问当 N 到 21 时，计算数量太大，运行时间过长）），考虑时间复杂度，所以通过优化成数学公式：总和×(非空子集数)/元素个数，即“数组总和×(2^n-1)/n”计算所有非空子集的平均值之和。第三问即遍历 n 从 1 到 100，生成数组并计算子集平均和，将结果存入列表后绘图。

运行图片如下：



规律描述：当 N 较小时（N<90），总平均值之和增长非常缓慢，几乎趋近于 0；而当 N 接近 100 时，由于公式中 2^N 的指数增长特性，总平均值之和出现了爆发式上升。

思路及代码来源：学计算机的朋友的指导。



5. Path counting

5.1 [5 points] Create a matrix with `N` rows and `M` columns, fill the right-bottom corner and top-left corner cells with `1`, and randomly fill the rest of matrix with integer `0` or `1`.

5.2 [25 points] Consider a cell marked with `0` as a blockage or dead-end, and a cell marked with `1` is good to go. Write a function `Count_path` to count total number of paths to reach the right-bottom corner cell from the top-left corner cell.

Notice: for a given cell, you are **only allowed** to move either rightward or downward.

5.3 [5 points] Let `N = 10, M = 8`, run `Count_path` for `1000` times, each time the matrix (except the right-bottom corner and top-left corner cells, which remain being `1`) is re-filled with integer `0` or `1` randomly, report the mean of total number of paths from the `1000` runs.

思路：首先使用 numpy 的 np.zeros()函数创建初始矩阵全为 0 的矩阵，再设置左上角和右下角为 1，其余位置随机填充 0 或 1。其次通过递归的思想定义函数来计算并统计从(0,0)到(N-1,M-1)的路径数返回(仅允许向右或向下走),最后重复生成矩阵并统计路径 1000 次，计算路径数的平均值并输出结果。
思路及代码来源：学计算机的朋友的指导。