## 1. Flowchart

本题使用 if 语句实现流程框图，当 $a$=10，$b$=5，$c$=1 时，答案为 5。脚本中还给出了其他 $a$，$b$ 和 $c$ 取值下，Print_values 函数的取值。

（注：题目中未指向的箭头，采取了指向 $a>c$ 绿色菱形处的方案计算。）

## 2. Continuous celing function

本题采用了迭代函数的方式描述目标函数 $F$，其中 $F(x)=F(\text{ceil}(x/3)) + 2*x$，并且 $x$=1 时，$F$ 取 1。在脚本中，有 $N$ 个整数的列表为 0 到 29 的整数列表，可以修改 $N$ 得到其他整数列表。

## 3. Dice rolling

### 3.1

本题运用了转换的思想，对于一个整数 $x$ 和 10 个骰子，10 个骰子点数总数为 x 等价于求下列方程的正整数解组的个数，且 $1 \le x_i \le 6 (i=1, 2, \ldots, 10)$。

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} = x$$

而该问题可以使用隔板法解决，想象有 $x$ 个物体，将 9 个隔板插入 $x$ 物体的间隙中，为了保证 $1 \le x_i \le 6 (i=1, 2, \ldots, 10)$，要求两个隔板之间的物体不能超过 6 个，两个隔板不能插入同一个空隙，隔板不能放在两端。

因此脚本中使用了 9 个循环来依次排布隔板的位置，最终解组个数为所求骰子点数和为 $x$ 的组合个数。

### 3.2

定义了一个名为 Number_of_ways 的列表存储 $x$ 从 10 取到 60 时，所求骰子点数和为 $x$ 的组合个数，并用 max()函数求出组合个数取最大时 $x$ 的取值。

得出当组合个数最大时，$x$ 取 35。

## 4. Dynamic programming

### 4.1

导入 random 库，使用 random.randint()函数实现目标函数 Random_integer。

### 4.2

对于一个有 $N$ 个元素的数组，记数组为 A，记 A 的所有元素的和为 S，考虑对于 A 所有的 $k$ 元子集（有 $k$ 个元素的子集），显然对于 A 中的某个元素 $x$，它在所有的 $k$ 元子集总共出现过 $C_{N-1}^{k-1}$ 次（证明：相当于 A 中去除掉 $x$ 元素，记为 A’，然后在 A’取出 $k$-1 元子集，再将 $x$ 放入该 $k$-1 元子集中，A 中含有 $x$ 的 $k$ 元子集与 A’中 $k$-1 元子集一一对应，因此 $x$ 在 $k$ 元子集出现的次数等于 A’中

$k$-1 元子集的个数$C_{N-1}^{k-1}$）。

所以所有的 $k$ 元子集元素和再求和的值为$S \cdot C_{N-1}^{k-1}$，则所有 $k$ 元子集平均值的求和为$S \cdot C_{N-1}^{k-1}/k$。

根据组合数恒等式：

$$C_N^k = \frac{N}{k} C_{N-1}^{k-1}$$

得到所有 $k$ 元子集平均值的求和为$S \cdot C_N^k/N$，因此 Sum_averages 的函数值为（空集的平均值为 0）：

$$\frac{S}{N}(C_N^1 + \boldsymbol{C}_N^2 + \cdots + C_N^N) = \frac{(2^N - 1)S}{N}$$

即为脚本中所给出的公式。

**4.3**

将 N 从 1 取到 100 时 Sum_averages 值存储到 Total_sum_averages 列表中，再导入 matplotlib 库进行绘图，横坐标为 N，纵坐标为对应 Total_sum_averages 函数值，结果如脚本中的图所示。

由图可知，随着 N 增大，Sum_averages 整体呈增加趋势，但 Sum_averages 在增大的过程中出现了波动的现象，不是严格递增的，这可能是初始的随机数大小波动所导致的。

## 5. Path counting

**5.1**

使用 numpy 中的 numpy.random.randint 函数生成行数为 N，列数为 M 的随机数矩阵（矩阵中元素取值是 0 和 1 的二元随机值），再使得该矩阵左上角和右下角的元素取值为 1。

**5.2**

由于只能往下或者往右走，对于一个 N*M 矩阵，从左上角到右下角需要 N+M-2 步，其中 N-1 步向下走，M-1 步向右走。因此我们可以定义一个从 0 到 N+M-3 的列表，然后取出该列表的一个 M-1 元子集，M-1 中包含的元素即向右走的步数时刻。例如，对于一个 5*3 的矩阵，我们定义一个 0 到 5 列表，取出一个 2 元子集，假设是[1, 4]，代表第 1 步和第 4 步为向右走，而第 0 步、第 2 步、第 3 步和第 5 步为向下走。

我们使用 itertools.combinations 函数取出 0 到 N+M-3 的列表所有子集，代表所有可以从左上角到右下角的路径，并构造对应的矩阵（trace_Matrix），再

将 5.1 中随机生成的矩阵（Matrix）减去 trace_Matrix，通过检查相减后得到的矩阵（sub）元素中是否含有-1，判断随机生成的矩阵（Matrix）是否含有 trace_Matrix 对应的路径，遍历所有可能的路径，如果含有一条路径，则路径数（count_path）加 1，这样就可以得到 Count_path(N, M)的函数值。

**5.3**

将上述过程重复 1000 次，并求出路径数的期望，得到值为 0.231。