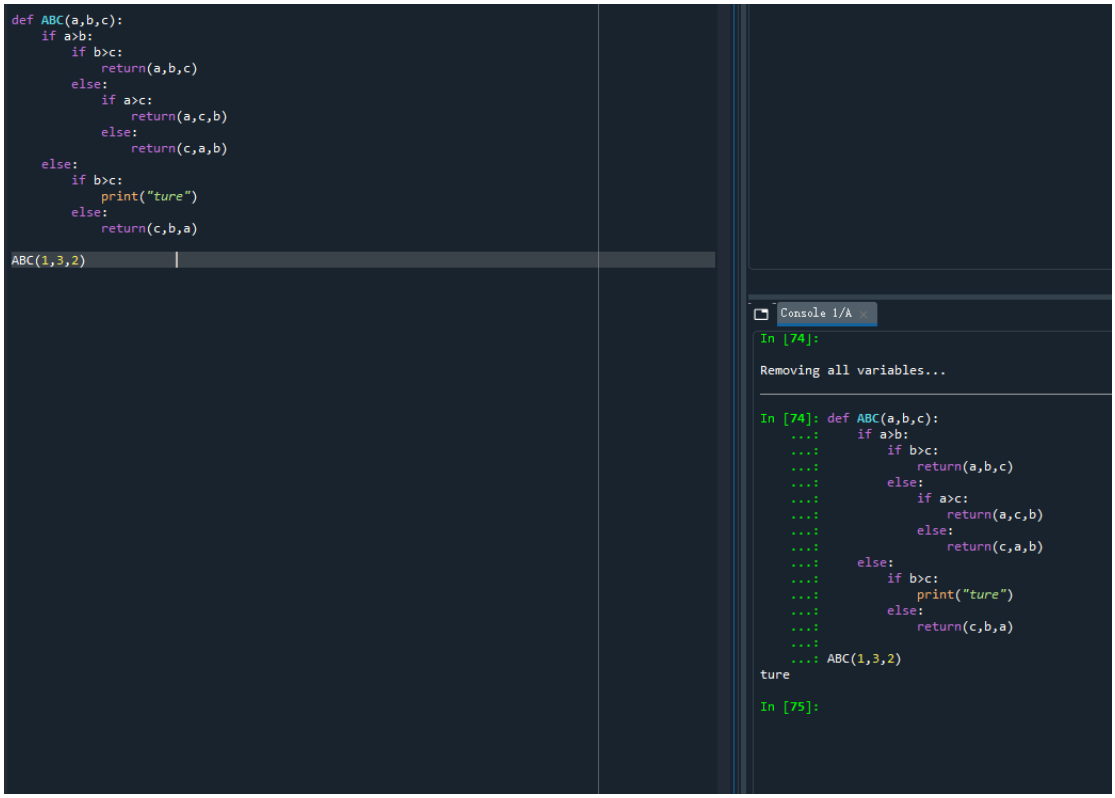
1. Flowchart

根据第一图的流程图，分析可能用上的语法，由菱形框表示判断用 if，平行四边形表输出的结果用 def 函数后 return。依据流程图的层次写出代码。



2. Matrix multiplication

2.1

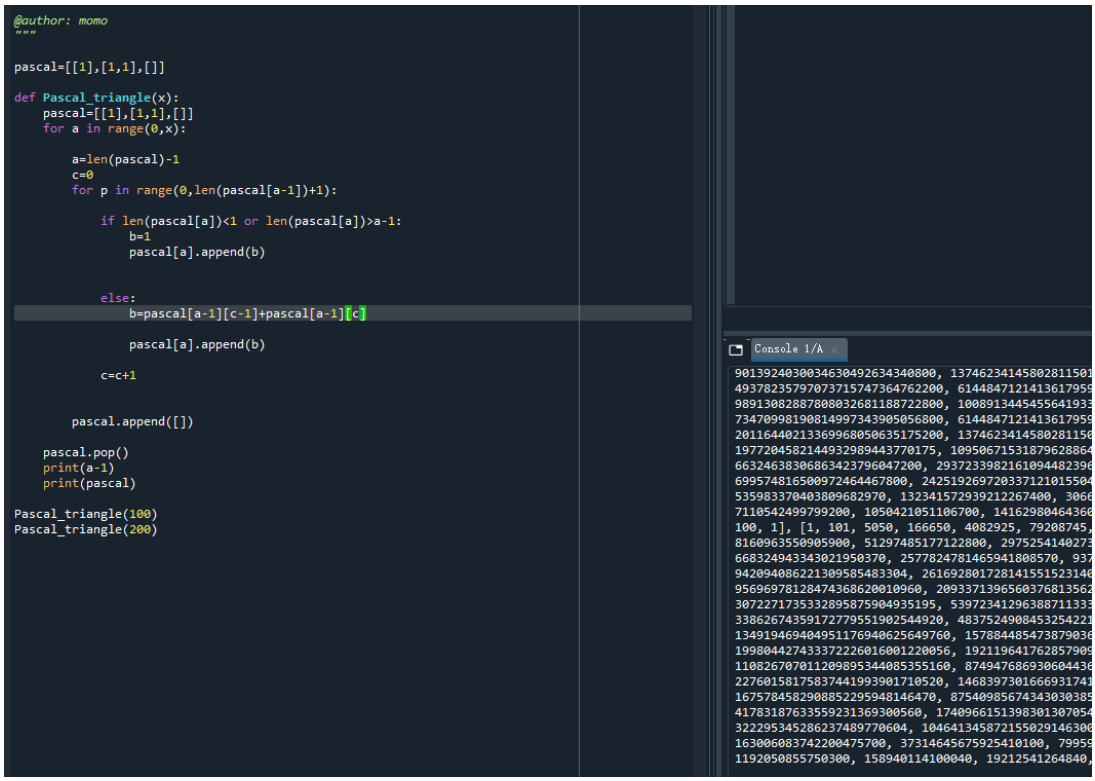
题目提到用到随机数，需要用到 random。题目要求列出矩阵，首先想到的是利用列表（list）来实现，思路是通过创建列表 M1，M1 中的每一个元素都是一个列表[]，每个列表[]代表矩阵的一行。利用 random 得到的随机量依次插入到每个列表[]中，最终建立 M1 和 M2“矩阵“。

2.2

为了实现 M1*M2，思路是将 M1 中每个列表[]中的元素分别提取出来作为 i，与 M2 中每个列表[]中的元素相乘后列入一个新的列表 M3，并且 M3 的结构与 M1 和 M2 类似。

3. Pascal triangle

帕斯卡三角形我理解的也是一种矩阵，还是基于第二题的思路，把这个三角形用列表来进行计算。首先建立一个初始的三角形 pascal=[[1],[1,1],[]]，然后将列表中的元素提取出来计算出的结果再填入到 pascal 列表中相应的位置。

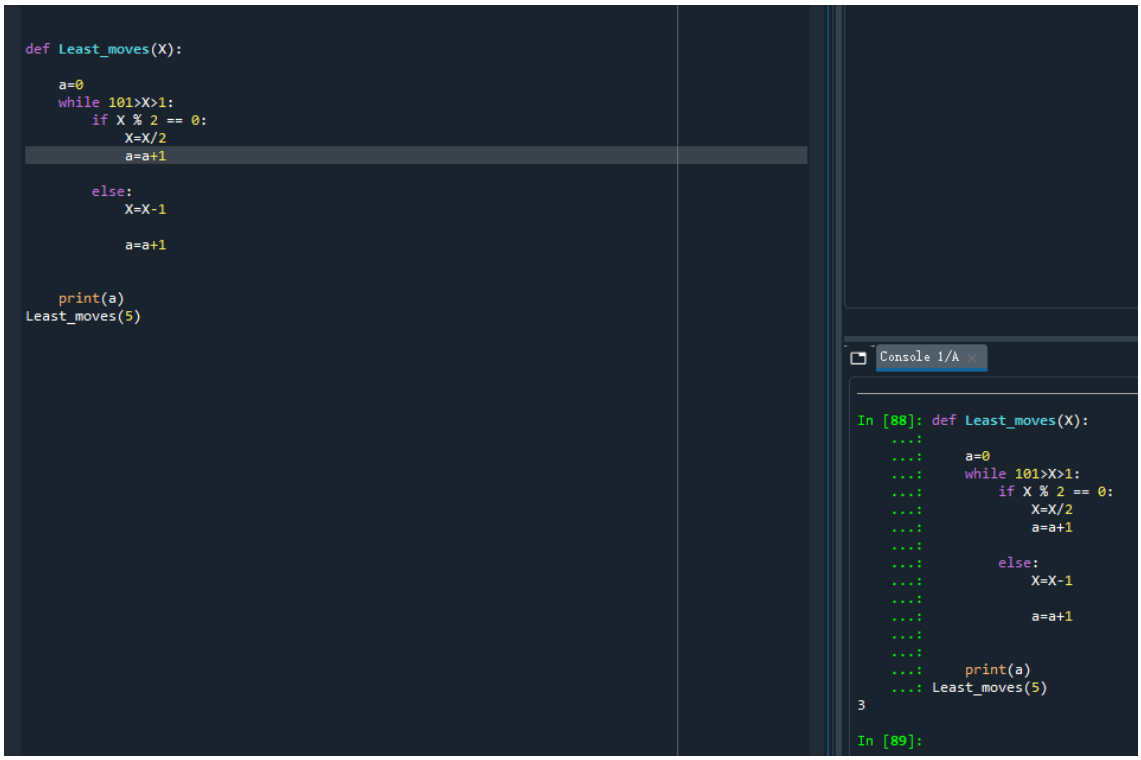


```
@author: momo
"""

pascal=[[1],[1,1],[]]

def Pascal_triangle(x):
    pascal=[[1],[1,1],[]]
    for a in range(0,x):

        a=len(pascal)-1
        c=0
        for p in range(0,len(pascal[a-1])+1):

            if len(pascal[a])<1 or len(pascal[a])>a-1:
                b=1
                pascal[a].append(b)


            else:
                b=pascal[a-1][c-1]+pascal[a-1][c]

                pascal[a].append(b)

            c=c+1


        pascal.append([])

    pascal.pop()
    print(a-1)
    print(pascal)

Pascal_triangle(100)
Pascal_triangle(200)
```

```
Console 1/A
901392403003463049263434080O, 13746234145802811501
493782357970737157473647622OO, 61448471214136179591
989130828878080326811887228OO, 10089134454556419331
734709981908149973439050568OO, 61448471214136179591
201164402133699680506351752OO, 13746234145802811501
197720458214493298944377017S, 10950671531879628864
663246383068634237960472OO, 29372339821610944823961
699574816500972464467800, 24251926972033712101550401
535983370403809682970, 13234157293921226740O, 306601
7110542499799200, 1050421051106700, 14162980464360
100, 1], [1, 101, 5050, 166650, 4082925, 79208745,
8160963550905900, 5129748517712280O, 29752541402731
668324943343021950370, 25778247814659418085TO, 937
942094086221309585483304, 26169280172814155152314O
956969781284743686200109GO, 20933713965603768135621
307227173533289587590493519S, 53972341296388711331
338626743591727795519025449ZO, 48375249084532542211
134919469404951176940625649760, 15788448547387903G
199804427433372226016001200056, 19211964176285790S
110826707011209895344085355160, 87494768693060443G
227601581758374419939017105ZO, 14683973016669317411
167578458290885229594846470, 87540985674343030385
417831876335592313693005GO, 17409661513983013070541
322295345286237489770604, 10464134587215502914630O
163006083742200475700, 37314645675925410100, 799591
1192050855750300, 158940114100040, 19212541264840,
```

4. Add or double

题目要求加 1 元人民币或者翻倍，并用最少的步数。翻倍既是乘以 2，反之便是想要的结果能被 2 整除，并且一个大于 2 的数以最少的步数到 1 肯定是被除以 2 的。所以 X 块钱首先被判断能否被 2 整除，可以就被除以 2，如果不可以则被减去 1，并将以上步骤建立 whlie 循环，直到 X=1 时。步数引入 a 来计数。当 X 大于 100 时输出 0。



5. Dynamic programming

我的思路是通过 A、B 两个函数实现对 1~9 的数字进行随机分组，分组的后的数字添加进入到 m 列表，并复制到 f 列表中。C、D 函数提取 m 列表中的数字出来进行随机计算，结果为 x，计算的过程用 v 列表记录。只有当计算结果 x 与需要的 y 值相等时，将 f 和 v 列表的数字和符号组合到 find 列表，形成算式。

这是一个随机的计算，所以把以上随机计算设为 Find_expression(y)函数，重复运行 n 次此函数并排除重复项，就得到了最终的所以符合结果的算式，算式的形式是表格。