##1.Flowchart 定义一个函数判断参数 abc 的大小
首先对 a>b 进行判断，分两步走:
1,若满足，则，对 b>c；a>c 分别进行判断
2,若不满足，认定 a<b;
下一步则对 b>c；a>c 分别进行判断。

###2.Matrix multiplication
#2.1 生成矩阵
#参考 https://www.jianshu.com/p/85ec63259c4e-np.random 用法
#numpy 是 python 进行高效数据处理的模块
import numpy as np#导入 numpy 模块并将其简写成 np
#np.random.randint(low,high=(),size=,dtype=)
#其中，low 和 high 是指随机生产整数的范围
#size 是指 array 的维度，dtype 是指数据类型,默认为 int 型
M1=np.random.randint(low=0,high=50,size=(5,10))
M2=np.random.randint(low=0,high=50,size=(10,5))

#2.2 定义一个矩阵相乘的函数
矩阵的算法是由 M1 的行与 M2 的列对应的每个数字相乘相加得到。
```
def Matrix_multip():
    M4=np.zeros((5,5))#首先生成一个 5*5 的 0 矩阵，用来装 M1*M2 的结果
    for i in range(5):
        n=0
        while n < 5:
            for j in range(10):
                M4[i,n]=M4[i,n]+M1[i,j]*M2[j,n] # M4=0 矩阵+M1*M2 相乘得到的各行各列元素
            n=n+1
    print(M4)
Matrix_multip()
```

###3 定义 Pascal_triangle
```
def Pascal_triangle(K):
    k=int(K)#将 K 调整为整数形式
    a=[]#生成一个空矩阵
    if k==0:
        a.append([1])#k 为 0 时，杨辉三角为 1*1 的单元素列表[1]
     elif k==1: #k 为 1 时，杨辉三角为 2 维列表[1],[1,1]
        a.append([1],[1,1])
    k-=2#k=k-2
    a=[[1],[1,1]]#设计一个基础列表
```

```
    while k>0:
        b=[1]
        for i in range(len(a)-1):#当 a 的长度大于 1 时,b b 的第 i 行数组如下
            b.append(a[-1][i]+a[-1][i+1])#即倒数第一行的第 i 个数字与第 i+1 个数字相
加
        b.append(1)
        k-=1#k=k-1
        a.append(b)
    return a

m=Pascal_triangle(100)
n=m=Pascal_triangle(200)
```

###4 Add or double

首先判断 x 在大于 1 的基础上的奇偶性，若为偶数则除以 2 即可得到最少步数，若为奇数，则在则是发生了+1 的情况，同时被认定为一步，即(x-1)/2+1=a 简化为(x+1)/2=a

```
import random
def   Least_moves():
    x=random.randint(0,100)#x 为 0，100 的任意数字
        if x>1:
判断最少需要走多少步，首先判断 x 的奇偶性
        if x%2==0:
            a=x/2
        else:
            a=(x+1)/2
    else:
        a=0
    return(a)
Least_moves()
```