# # 1.Flowchart

In [33]:

```
#定义函数
def Print_values():
    a = float(input('Enter number a:'))
    b = float(input('Enter number b:'))
    c = float(input('Enter number c:'))
    if a>b:
        if b>c:
            print(a,b,c)
        else:
            if a>c:
                print(a,c,b)
            else:
                print(c,a,b)
    else:
        if b>c:
            if a>c:
                print(a,c,b)
            else:
                print(c,a,b)
        else:
            print(c,b,a)
Print_values()
```

```
Enter number a:9
Enter number b:45
Enter number c:11
11.0 9.0 45.0
```

# # 2.Matrix multiplication

In [247]:

```
# 2.1 创建矩阵M1、M2
import random
r1 = 5      #定义矩阵M1、M2行列数
c1 = 10
r2 = 10
c2 = 5
M1 = [[0 for j in range(c1)] for i in range(r1)]     #创建空的二维list作为矩阵M1、M2
M2 = [[0 for j in range(c2)] for i in range(r2)]
for i in range(r1):
    for j in range(c1):
        M1[i][j] = random.randint(0,50)
for j in range(r2):
    for k in range(c2):
        M2[j][k] = random.randint(0,50)
print(M1)
print(M2)
# 2.2 矩阵乘法运算
def Matrix_multip(M1, M2):                           #定义矩阵乘法函数
    result = [[0] * len(M2[0]) for _ in range(r1)]  #矩阵result行数为M1的行数，列数为M2的列数
    for i in range(len(M1)):                         #按行遍历M1矩阵
        for j in range(len(M2[0])):                  #按列遍历M2矩阵
            for k in range(len(M2)):                 #按行遍历M2矩阵
                result[i][j] += M1[i][k] * M2[k][j]
    return result

print(result)                                       #打印矩阵乘法结果
```

```
[[42, 42, 50, 20, 43, 26, 9, 12, 22, 42], [34, 50, 50, 29, 2, 40, 17, 48, 33, 41], [14, 4, 16, 19, 4, 34, 39, 19, 29, 24], [30,
27, 30, 22, 28, 32, 0, 29, 14, 6], [23, 33, 23, 14, 0, 42, 46, 35, 39, 12]]
[[42, 22, 24, 2, 4], [21, 0, 50, 40, 2], [36, 41, 16, 9, 18], [30, 42, 12, 11, 13], [23, 34, 3, 46, 11], [5, 38, 33, 28, 24], [4
4, 17, 36, 1, 39], [8, 36, 0, 10, 23], [2, 2, 41, 37, 35], [47, 40, 25, 33, 46]]
[[4044, 5612, 4729, 2577, 4679], [6640, 5375, 6093, 5052, 7017], [5558, 5091, 4351, 4439, 6158], [4796, 6323, 5761, 3241, 5222],
[4584, 3394, 5596, 3304, 4966]]
```

# # 3 Pascal triangle

```
In  [333]: def Pascal_triangle(n):                   #定义帕斯卡函数，参数n为行数
               result = []
               for i in range(n):
                   k = [1]
                   if i > 0:
                       for j in range(1, i):
                           new_k = k[j - 1] + result[i - 1][j]   # 当前行各元素等于上一行对应位置元素之和
                           k.append(new_k)
                       k.append(1)
                   result.append(k)
               return result

           n1 = 100
           n2 = 200
           for r in Pascal_triangle(n1):  #打印前100行、前200行帕斯卡三角
               print(r)
           print("\n—————————————————————————————————————\n")
           for r in Pascal_triangle(n2):
               print(r)
```

```
[1]
[1, 1]
[1, 2, 1]
[1, 3, 4, 1]
[1, 4, 8, 9, 1]
[1, 5, 13, 22, 23, 1]
[1, 6, 19, 41, 64, 65, 1]
[1, 7, 26, 67, 131, 196, 197, 1]
[1, 8, 34, 101, 232, 428, 625, 626, 1]
[1, 9, 43, 144, 376, 804, 1429, 2055, 2056, 1]
[1, 10, 53, 197, 573, 1377, 2806, 4861, 6917, 6918, 1]
[1, 11, 64, 261, 834, 2211, 5017, 9878, 16795, 23713, 23714, 1]
[1, 12, 76, 337, 1171, 3382, 8399, 18277, 35072, 58785, 82499, 82500, 1]
[1, 13, 89, 426, 1597, 4979, 13378, 31655, 66727, 125512, 208011, 290511, 290512, 1]
[1, 14, 103, 529, 2126, 7105, 20483, 52138, 118865, 244377, 452388, 742899, 1033411, 1033412, 1]
[1, 15, 118, 647, 2773, 9878, 30361, 82499, 201364, 445741, 898129, 1641028, 2674439, 3707851, 3707852, 1]
[1, 16, 134, 781, 3554, 13432, 43793, 126292, 327656, 773397, 1671526, 3312554, 5986993, 9694844, 13402696, 13402697, 1]
[1, 17, 151, 932, 4486, 17918, 61711, 188003, 515659, 1289056, 2960582, 6273136, 12260129, 21954973, 35357669, 48760366, 4876
0367, 1]
```

# # 4 Add or double

In [362]:

```python
import random
x = random.randint(1,100)
def Least_moves(x):                #定义函数
    if x == 1:
        return 0                    #x==1时，返回0，表示0次即获得1元
    else:
        tt = 0                     #初始化次数tt
        while x > 1:
            if x % 2 == 0:
                x = x // 2         #若x为偶数，则将x除以2（即钱数double）
            else:
                x -= 1             #若x为奇数，则将x减去1（即钱数增加1元）
            tt += 1
        return tt

#Least_moves(x)
print(x)                          #打印随机整数x
print(Least_moves(x))             #打印计算所需最小次数
```

```
15
6
```

# # 5 Dynamic programming

In [690]:

```
#5.1
import random
#定义函数，参数为数字串、目标为1到100随机整数、当前表达式和当前结果
def Find_expression(str_nums, target, exp="", result=0):
    if not str_nums:           #若数字串为空，说明已经遍历完所有数字
        if result == target:   #若当前结果等于目标整数，就打印出表达式
            print(str(exp) + "=" + str(target))
            return 1           #找到1个对应结果的表达式
        else:
            return 0           #未找到对应结果的表达式
            pass               #do nothing
    else:
        count = 0
        for i in range(1, len(str_nums) + 1):  #遍历数字串中每个数字
            num = int(str_nums[:i])            #提取当前已遍历的数字串
            if not exp:              #若为第一个数，则直接加入表达式和结果中，再递归剩余数字
                count += Find_expression(str_nums[i:], target, str(num), num)
            else:   # 否则进行加法或减法
                count += Find_expression(str_nums[i:], target, exp + "+" + str(num), result + num)   #插入加号，更新表达式和结果，
                count += Find_expression(str_nums[i:], target, exp + "-" + str(num), result - num)   #插入减号，更新表达式和结果，
        return count

rr = random.randint(1,101)
Find_expression("123456789", rr)
print("结果等于"+ str(rr)+"的表达式有"+ str(count)+"个")
```

```
1+2+34-5+67-89=10
1-2+34-5+6-7-8-9=10
1-2-34-5+67-8-9=10
12+3+4+5-6-7+8-9=10
12+3+4-5+6+7-8-9=10
12+3-4-5-6-7+8+9=10
12+3+45-67+8+9=10
12-3+4-5-6+7-8+9=10
12-3-4+5+6-7-8+9=10
12-3-4+5-6+7+8-9=10
12-34+56-7-8-9=10
123-45-67+8-9=10
结果等于10的表达式有26个
```

In [691]:

```
#5.2
#5.1
import random
import matplotlib.pyplot as plt
#定义函数，参数为数字串、目标为1到100随机整数、当前表达式和当前结果
def Find_expression(str_nums, target, exp="", result=0):
    if not str_nums:           #若数字串为空，说明已经遍历完所有数字
        if result == target:   #若当前结果等于目标整数，就打印出表达式
            print(str(exp) + "=" + str(target))
            return 1           #找到1个对应结果的表达式
        else:
            return 0           #未找到对应结果的表达式
            pass               #do nothing
    else:
        count = 0
        for i in range(1, len(str_nums) + 1):  #遍历数字串中每个数字
            num = int(str_nums[ :i])           #提取当前已遍历的数字串
            if not exp:               #若为第一个数，则直接加入表达式和结果中，再递归剩余数字
                count += Find_expression(str_nums[i: ], target, str(num), num)
            else:   # 否则进行加法或减法
                count += Find_expression(str_nums[i: ], target, exp + "+" + str(num), result + num)   #插入加号，更新表达式和结果，
                count += Find_expression(str_nums[i: ], target, exp + "-" + str(num), result - num)   #插入减号，更新表达式和结果，
        return count

rr = random.randint(1,101)
Find_expression("123456789", rr)
print("结果为"+ str(rr)+"的表达式有"+ str(count)+"个")

#5.2

counts = np.zeros((100))
for i in range(100):
    counts[i] = Find_expression("123456789", i)
    print("结果为"+ str(i)+"的表达式有"+ str(count)+"个")
    # Append the number of solutions to the list
#     counts.append(str(count))
print(counts)

import numpy as np
print(np.where(counts==max(counts)))  #打印最大个数表达式
print(np.where(counts==min(counts)))  #打印最小个数表达式
```

```
plt.plot(range(1, 101), counts)  # 用matplotlib库来绘制列表，横轴是整数，纵轴是解的数量
plt.title("counts of the expression")   # 添加标题和坐标轴标签
plt.xlabel("Integer")
plt.ylabel("counts of solutions")
plt.show()
```

```
1-2+34-56+78-9=46
1-2-34+5-6-7+89=46
1+23-4-56-7+89=46
1-23+4+56+7-8+9=46
12+3+4+5-67+89=46
12+3-4+5+6+7+8+9=46
12+3+45-6-7+8-9=46
12+3-45-6-7+89=46
12-34+5-6+78-9=46
123+4-5+6+7-89=46
123-4-5-67+8-9=46
结果为46的表达式有26个
1+2+3-4-5+67-8-9=47
1+2+3+45+6+7-8-9=47
1+2-3+45-6+7-8+9=47
1-2+3+45+6-7-8+9=47
1-2+3+45-6+7+8-9=47
1-2+3-45-6+7+89=47
1-2-3-4+5+67-8-9=47
```

In [ ]: