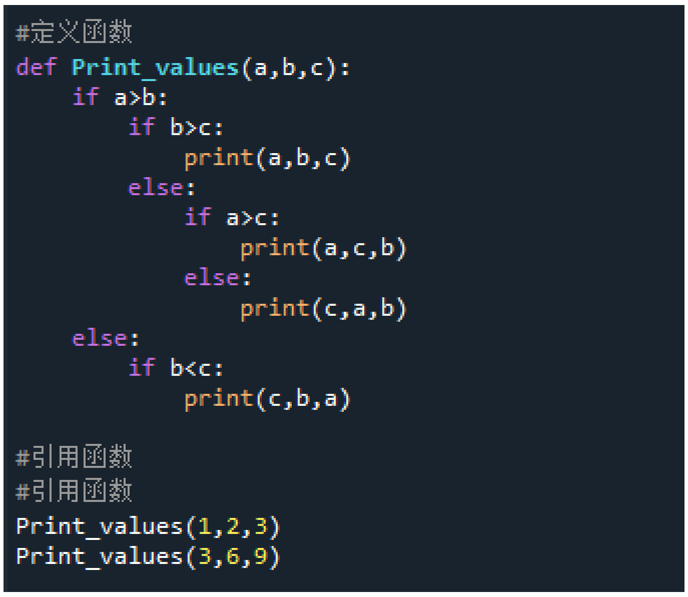
蒙芳秀，12331305，Assignment 01

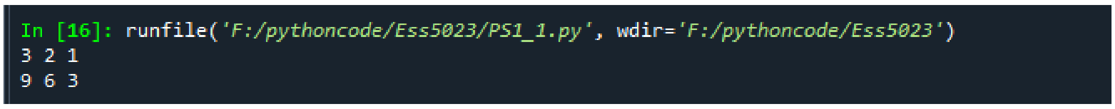
1. Flowchart

[10 points] Write a function Print_values with arguments a, b, and c to reflect the following flowchart. Here the purple parallelogram operator is to print values in the given order. Report your output with some random a, b, and c values.

思路：利用if语句，根据流程，编写条件语句解决



```python
#定义函数
def Print_values(a,b,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:
            print(c,b,a)

#引用函数
#引用函数
Print_values(1,2,3)
Print_values(3,6,9)
```



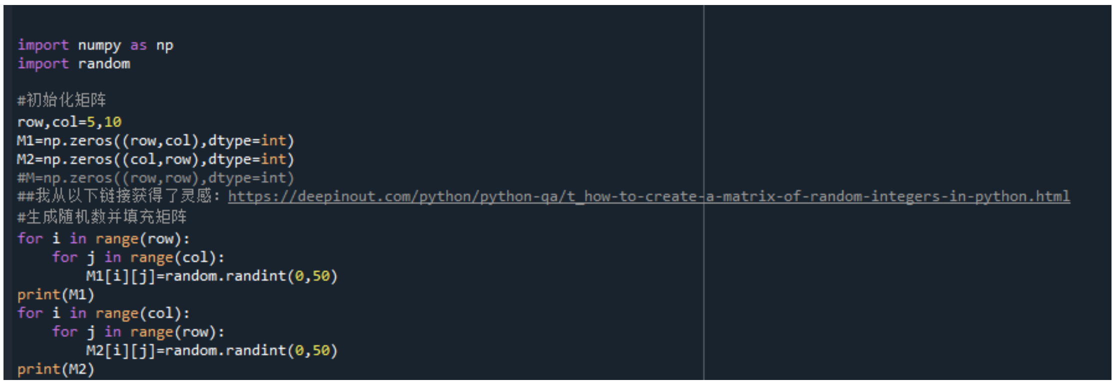
```
In [16]: runfile('F:/pythoncode/Ess5023/PS1_1.py', wdir='F:/pythoncode/Ess5023')
3 2 1
9 6 3
```

2. Matrix multiplication

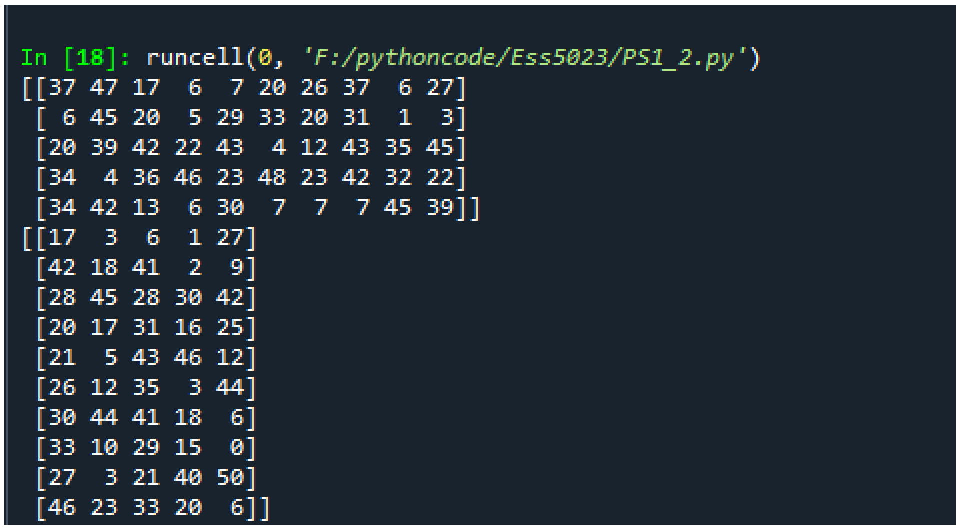
2.1 [5 points] Make two matrices M1 (5 rows and 10 columns) and M2 (10 rows and 5 columns); both are filled with random integers from 0 and 50.

2.2 [10 points] Write a function Matrix_multip to do matrix multiplication, *i.e.*, M1 * M2. Here you are **ONLY** allowed to use for loop, * operator, and + operator.

2.1思路：我从以下链接获得了灵感：https://deepinout.com/python/python-qa/t_how-to-create-a-matrix-of-random-integers-in-python.html。首先初始化矩阵，生成随机数并填充矩阵。



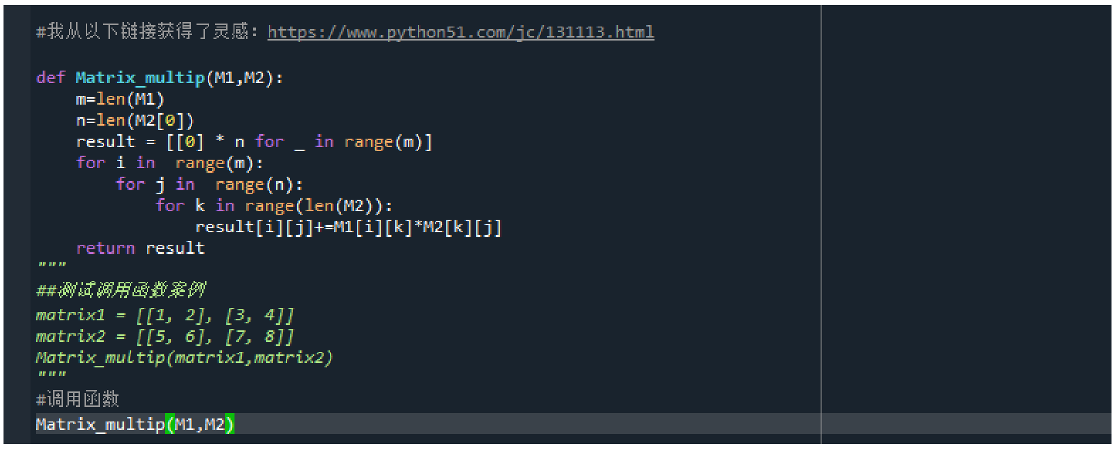
```python
import numpy as np
import random

#初始化矩阵
row,col=5,10
M1=np.zeros((row,col),dtype=int)
M2=np.zeros((col,row),dtype=int)
#M=np.zeros((row,row),dtype=int)
##我从以下链接获得了灵感：https://deepinout.com/python/python-qa/t_how-to-create-a-matrix-of-random-integers-in-python.html
#生成随机数并填充矩阵
for i in range(row):
    for j in range(col):
        M1[i][j]=random.randint(0,50)
print(M1)
for i in range(col):
    for j in range(row):
        M2[i][j]=random.randint(0,50)
print(M2)
```



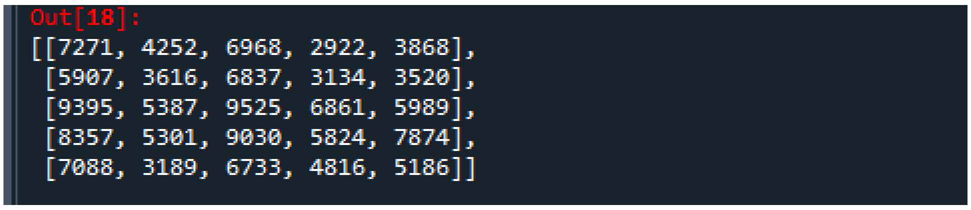
```
In [18]: runcell(0, 'F:/pythoncode/Ess5023/PS1_2.py')
[[37 47 17  6  7 20 26 37  6 27]
 [ 6 45 20  5 29 33 20 31  1  3]
 [20 39 42 22 43  4 12 43 35 45]
 [34  4 36 46 23 48 23 42 32 22]
 [34 42 13  6 30  7  7  7 45 39]]
[[17  3  6  1 27]
 [42 18 41  2  9]
 [28 45 28 30 42]
 [20 17 31 16 25]
 [21  5 43 46 12]
 [26 12 35  3 44]
 [30 44 41 18  6]
 [33 10 29 15  0]
 [27  3 21 40 50]
 [46 23 33 20  6]]
```

2.2

思路：我从以下链接获得了灵感：https://www.python51.com/jc/131113.html



```python
#我从以下链接获得了灵感：https://www.python51.com/jc/131113.html

def Matrix_multip(M1,M2):
    m=len(M1)
    n=len(M2[0])
    result = [[0] * n for _ in range(m)]
    for i in  range(m):
        for j in  range(n):
            for k in range(len(M2)):
                result[i][j]+=M1[i][k]*M2[k][j]
    return result
"""
##测试调用函数案例
matrix1 = [[1, 2], [3, 4]]
matrix2 = [[5, 6], [7, 8]]
Matrix_multip(matrix1,matrix2)
"""
#调用函数
Matrix_multip(M1,M2)
```



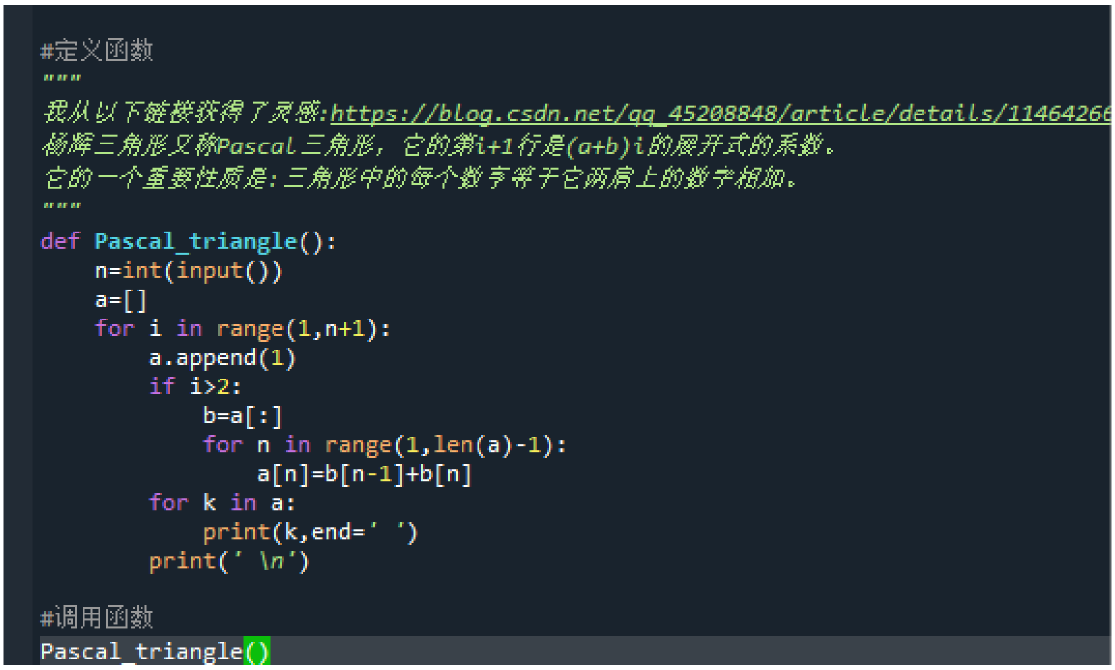
```
Out[18]:
[[7271, 4252, 6968, 2922, 3868],
 [5907, 3616, 6837, 3134, 3520],
 [9395, 5387, 9525, 6861, 5989],
 [8357, 5301, 9030, 5824, 7874],
 [7088, 3189, 6733, 4816, 5186]]
```

3. Pascal triangle

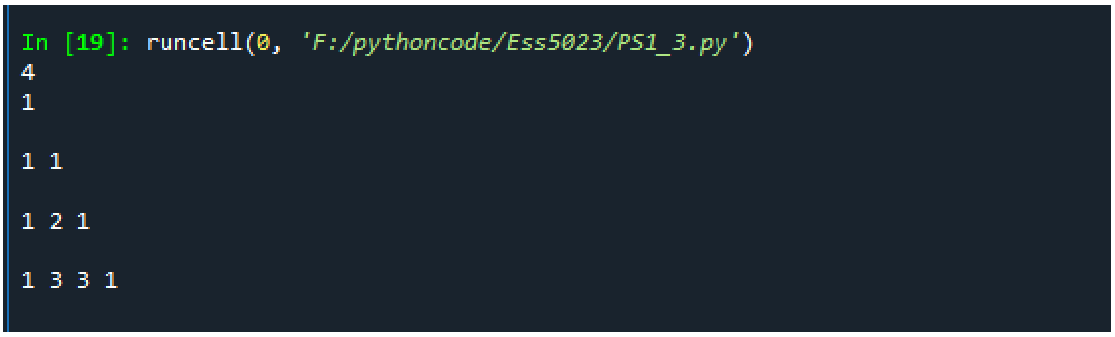
[20 points] One of the most interesting number patterns is Pascal's triangle (named after Blaise Pascal). Write a function Pascal_triangle with an argument k to print the *k*th line of the Pascal triangle.
Report Pascal_triangle(100) and Pascal_triangle(200).

思路：首先了解了 Pascal_triangle 的性质，我从以下链接获得了灵感：

https://blog.csdn.net/qq_45208848/article/details/114642662?utm_medium=distribute.pc_relevant.none-task-blog-2~default~baidujs_baidulandingword~default-0-114642662-blog-113983632.235^v38^pc_relevant_anti_vip&spm=1001.2101.3001.4242.1&utm_relevant_index=3



```
#定义函数
"""
我从以下链接获得了灵感:https://blog.csdn.net/qq_45208848/article/details/1146426
杨辉三角形又称Pascal三角形，它的第i+1行是(a+b)i的展开式的系数。
它的一个重要性质是:三角形中的每个数字等于它两肩上的数字相加。
"""
def Pascal_triangle():
    n=int(input())
    a=[]
    for i in range(1,n+1):
        a.append(1)
        if i>2:
            b=a[:]
            for n in range(1,len(a)-1):
                a[n]=b[n-1]+b[n]
        for k in a:
            print(k,end=' ')
        print(' \n')

#调用函数
Pascal_triangle()
```



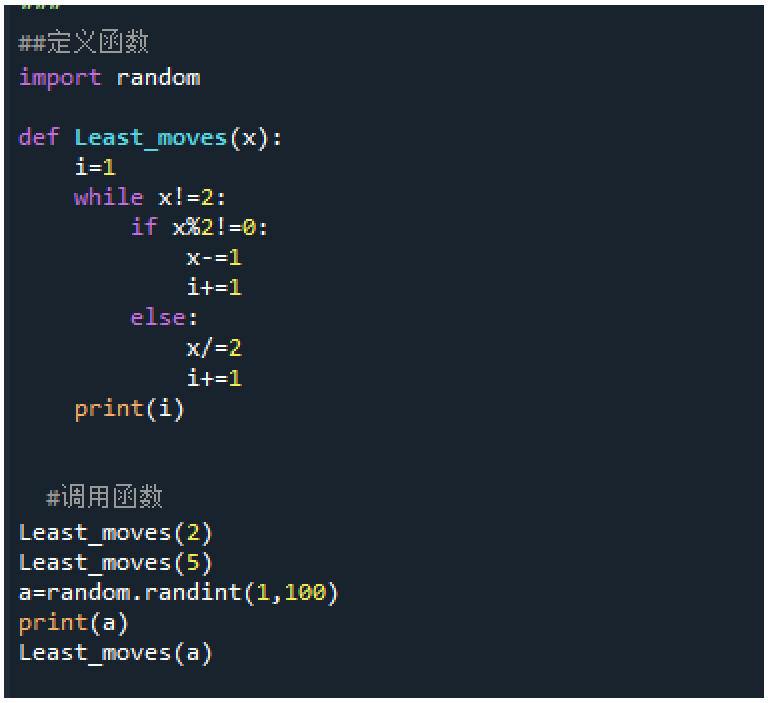
```
In [19]: runcell(0, 'F:/pythoncode/Ess5023/PS1_3.py')
4
1

1 1

1 2 1

1 3 3 1
```

4. Add or double

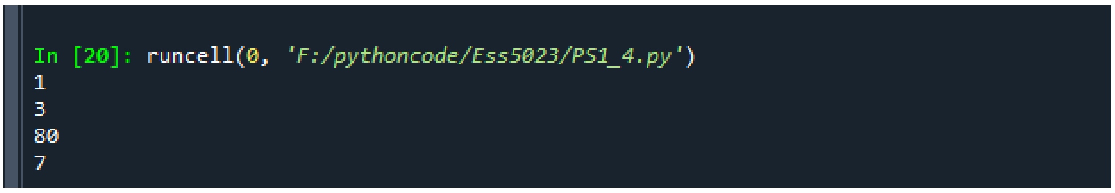
[20 points] If you start with 1 RMB and, with each move, you can either double your money or add another 1 RMB, what is the smallest number of moves you have to make to get to exactly *x* RMB? Here *x* is an integer randomly selected from 1 to 100. Write a function Least_moves to print your results. For example, Least_moves(2) should print 1, and Least_moves(5) should print 3

思路：通过与冯汇然同学和助教张鹏的交流，主要思路为做一个逆运算，从 x 块

钱变成 1 块钱，需要的次数是多少？



```
##定义函数
import random

def Least_moves(x):
    i=1
    while x!=2:
        if x%2!=0:
            x-=1
            i+=1
        else:
            x/=2
            i+=1
    print(i)


  #调用函数
Least_moves(2)
Least_moves(5)
a=random.randint(1,100)
print(a)
Least_moves(a)
```



```
In [20]: runcell(0, 'F:/pythoncode/Ess5023/PS1_4.py')
1
3
80
7
```

5. Dynamic programming

Insert + or - operation anywhere between the digits 123456789 in a way that the expression evaluates to an integer number. You may join digits together to form a bigger number. However, the digits must stay in the original order.

5.1 [30 points] Write a function Find_expression, which should be able to print every possible solution that makes the expression evaluate to a random integer from 1 to 100. For example, Find_expression(50) should print lines include:

$$1-2+34+5+6+7+8-9=501-2+34+5+6+7+8-9=50$$

and

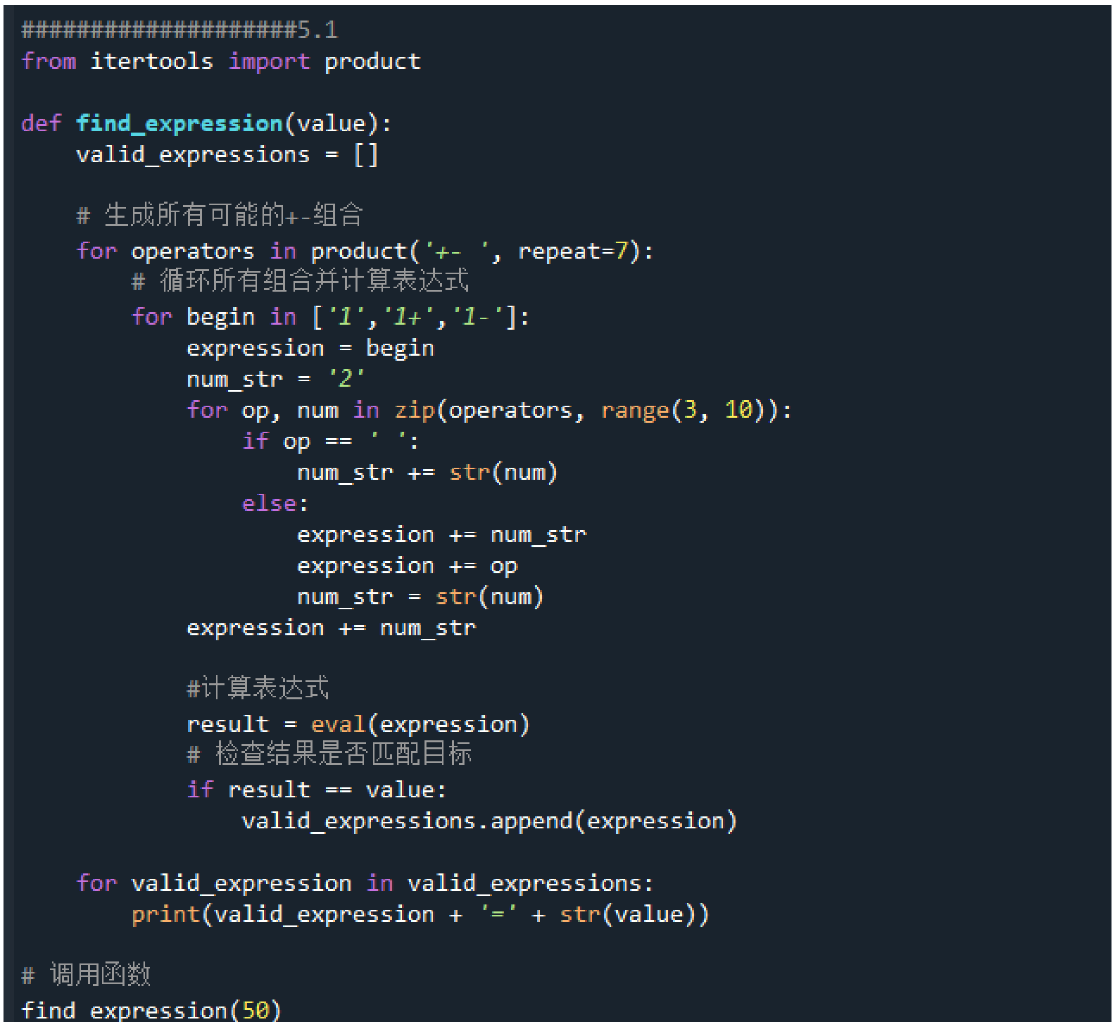
$$1+2+34-56+78-9=501+2+34-56+78-9=50$$

5.2 [5 points] Count the total number of suitable solutions for any integer *i* from 1 to 100, assign the count to a list called Total_solutions. Plot the list Total_solutions, so which number(s) yields the maximum and minimum of Total_solutions?

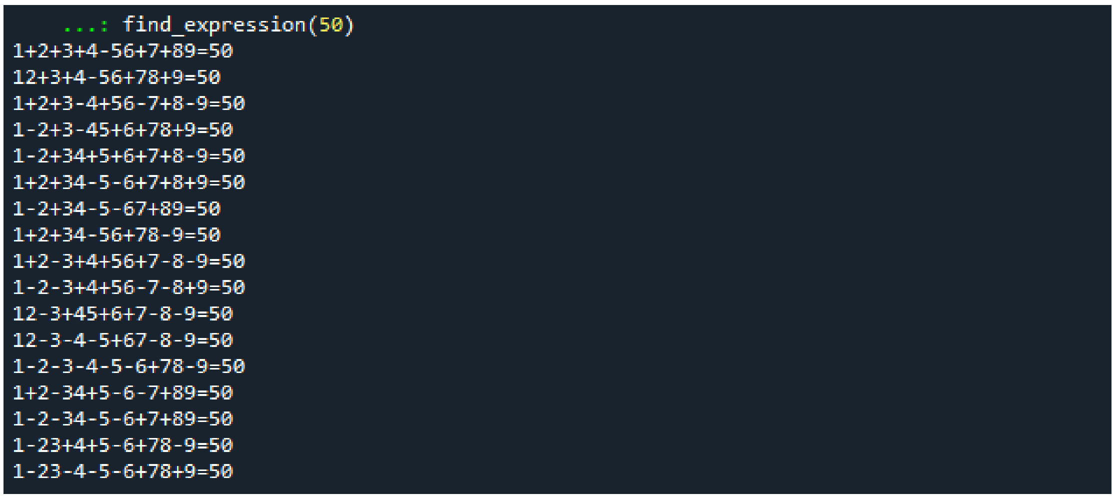
5.1 通过与冯汇然同学的交流，主要思路为生成 8 个符号（遍历加、减、空白），

放在 1 和 2、2 和 3……8 和 9 之间，得到所有可能的算式，再用 eval( )给算式，

最后匹配目标，打印结果。



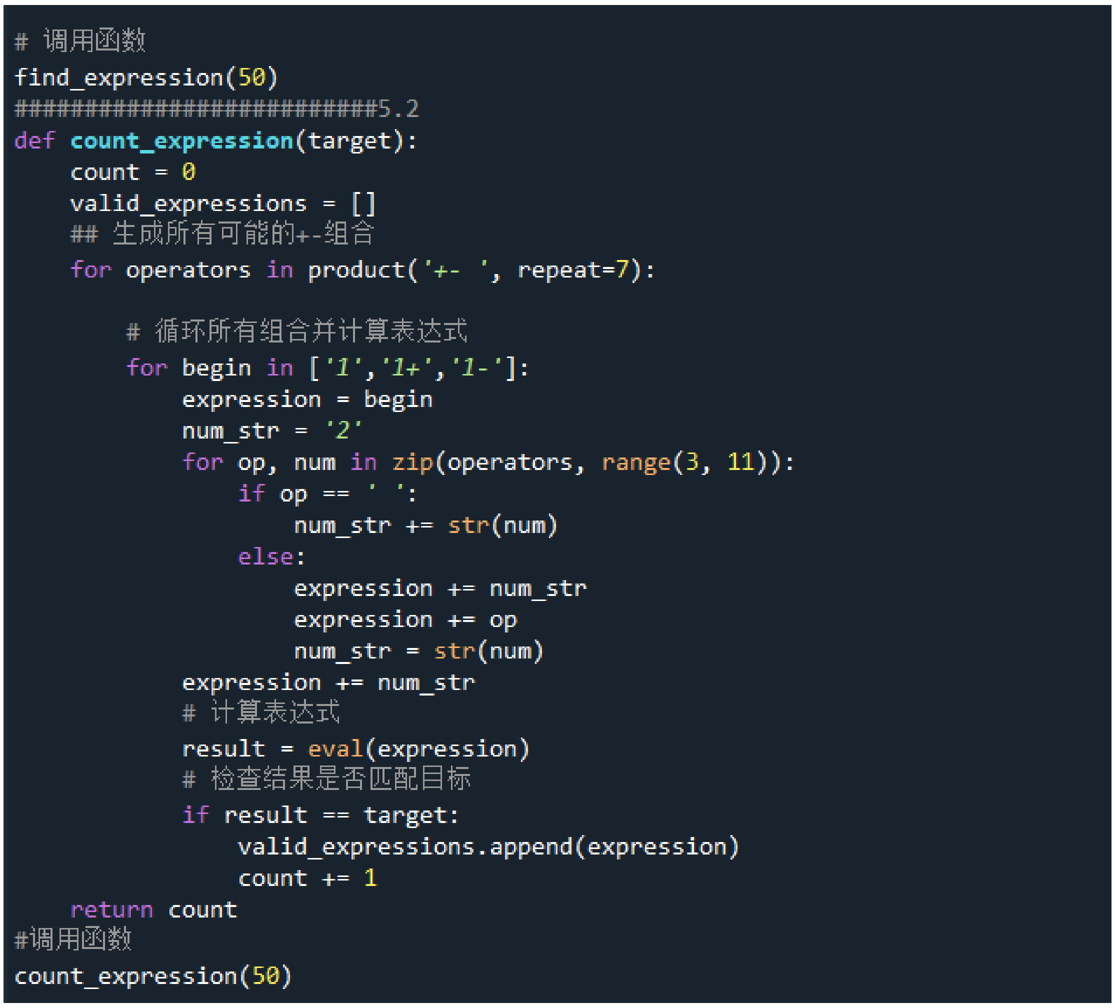
```
####################5.1
from itertools import product

def find_expression(value):
    valid_expressions = []

    # 生成所有可能的+-组合
    for operators in product('+- ', repeat=7):
        # 循环所有组合并计算表达式
        for begin in ['1','1+','1-']:
            expression = begin
            num_str = '2'
            for op, num in zip(operators, range(3, 10)):
                if op == ' ':
                    num_str += str(num)
                else:
                    expression += num_str
                    expression += op
                    num_str = str(num)
            expression += num_str

            #计算表达式
            result = eval(expression)
            # 检查结果是否匹配目标
            if result == value:
                valid_expressions.append(expression)

    for valid_expression in valid_expressions:
        print(valid_expression + '=' + str(value))

# 调用函数
find_expression(50)
```



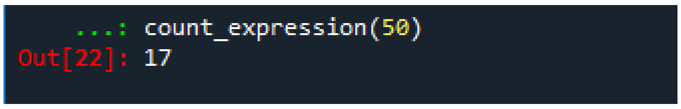
```
    ...: find_expression(50)
1+2+3+4-56+7+89=50
12+3+4-56+78+9=50
1+2+3-4+56-7+8-9=50
1-2+3-45+6+78+9=50
1-2+34+5+6+7+8-9=50
1+2+34-5-6+7+8+9=50
1-2+34-5-67+89=50
1+2+34-56+78-9=50
1+2-3+4+56+7-8-9=50
1-2-3+4+56-7-8+9=50
12-3+45+6+7-8-9=50
12-3-4-5+67-8-9=50
1-2-3-4-5-6+78-9=50
1+2-34+5-6-7+89=50
1-2-34-5-6+7+89=50
1-23+4+5-6+78-9=50
1-23-4-5-6+78+9=50
```

5.2 思路：通过与冯汇然同学的交流，主要思路为生成 8 个符号（遍历加、减、空白），放在 1 和 2、2 和 3……8 和 9 之间，得到所有可能的算式，再用 eval( ) 给算式，最后匹配目标，打印计数。通过计数程序，计算累计计数，并画图，找

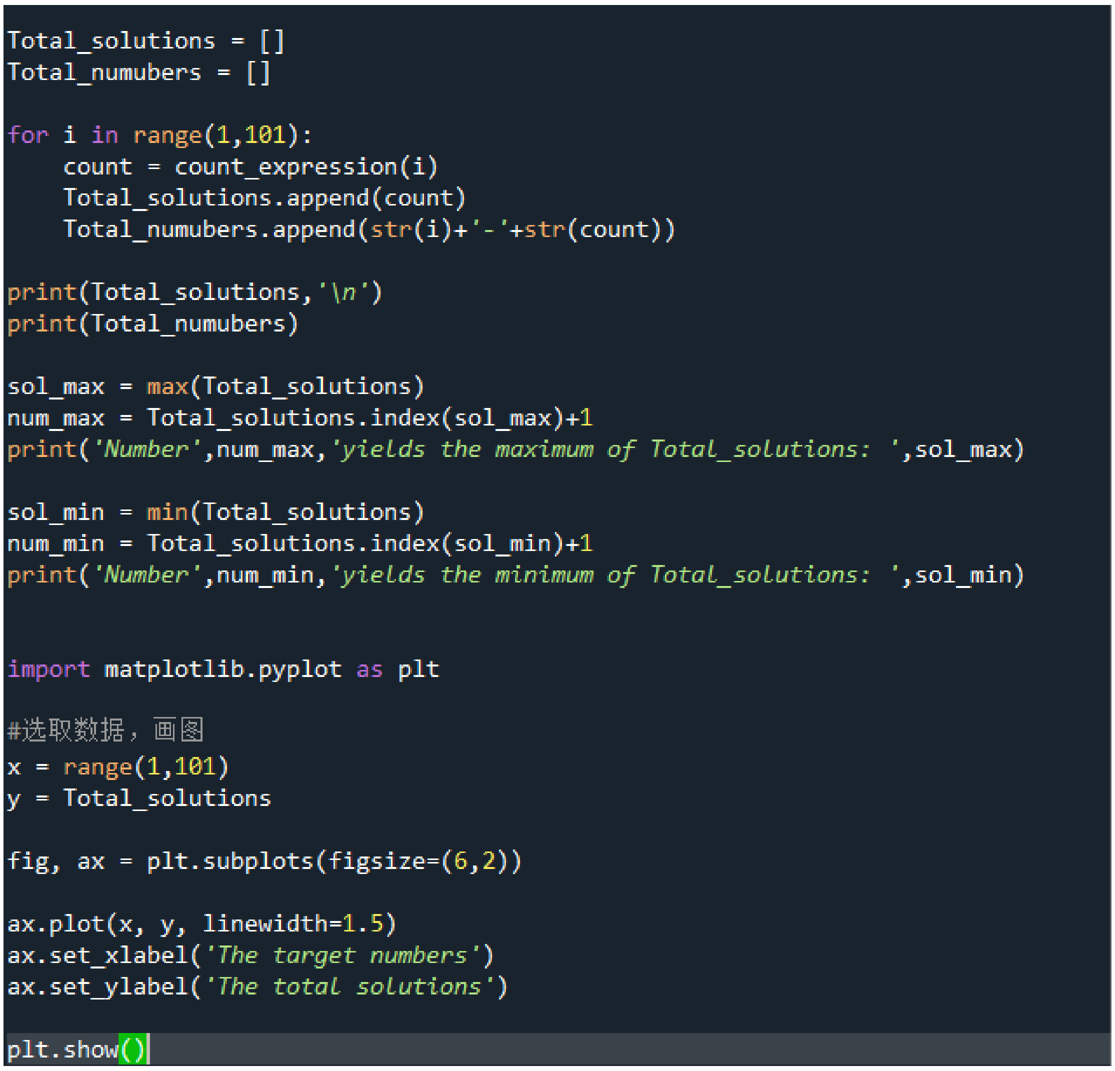
最大、最小情况所对应的值。



```python
# 调用函数
find_expression(50)
##########################5.2
def count_expression(target):
    count = 0
    valid_expressions = []
    ## 生成所有可能的+-组合
    for operators in product('+- ', repeat=7):

        # 循环所有组合并计算表达式
        for begin in ['1','1+','1-']:
            expression = begin
            num_str = '2'
            for op, num in zip(operators, range(3, 11)):
                if op == ' ':
                    num_str += str(num)
                else:
                    expression += num_str
                    expression += op
                    num_str = str(num)
            expression += num_str
            # 计算表达式
            result = eval(expression)
            # 检查结果是否匹配目标
            if result == target:
                valid_expressions.append(expression)
                count += 1
    return count
#调用函数
count_expression(50)
```

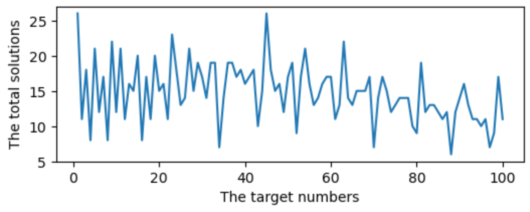


```
    ...: count_expression(50)
Out[22]: 17
```



```python
Total_solutions = []
Total_numubers = []

for i in range(1,101):
    count = count_expression(i)
    Total_solutions.append(count)
    Total_numubers.append(str(i)+'-'+str(count))

print(Total_solutions,'\n')
print(Total_numubers)

sol_max = max(Total_solutions)
num_max = Total_solutions.index(sol_max)+1
print('Number',num_max,'yields the maximum of Total_solutions: ',sol_max)

sol_min = min(Total_solutions)
num_min = Total_solutions.index(sol_min)+1
print('Number',num_min,'yields the minimum of Total_solutions: ',sol_min)


import matplotlib.pyplot as plt

#选取数据，画图
x = range(1,101)
y = Total_solutions

fig, ax = plt.subplots(figsize=(6,2))

ax.plot(x, y, linewidth=1.5)
ax.set_xlabel('The target numbers')
ax.set_ylabel('The total solutions')

plt.show()
```

```
[26, 11, 18, 8, 21, 12, 17, 8, 22, 12, 21, 11, 16, 15, 20, 8, 17, 11, 20, 15, 16, 11, 23, 18, 13, 14, 21, 15, 19, 17, 14, 19, 19, 7, 14, 19,
19, 17, 18, 16, 17, 18, 10, 15, 26, 18, 15, 16, 12, 17, 19, 9, 17, 21, 16, 13, 14, 16, 17, 17, 11, 13, 22, 14, 13, 15, 15, 15, 17, 7, 14, 1
7, 15, 12, 13, 14, 14, 14, 10, 9, 19, 12, 13, 13, 12, 11, 12, 6, 12, 14, 16, 13, 11, 11, 10, 11, 7, 9, 17, 11]

['1-26', '2-11', '3-18', '4-8', '5-21', '6-12', '7-17', '8-8', '9-22', '10-12', '11-21', '12-11', '13-16', '14-15', '15-20', '16-8', '17-1
7', '18-11', '19-20', '20-15', '21-16', '22-11', '23-23', '24-18', '25-13', '26-14', '27-21', '28-15', '29-19', '30-17', '31-14', '32-19',
'33-19', '34-7', '35-14', '36-19', '37-19', '38-17', '39-18', '40-16', '41-17', '42-18', '43-10', '44-15', '45-26', '46-18', '47-15', '48-1
6', '49-12', '50-17', '51-19', '52-9', '53-17', '54-21', '55-16', '56-13', '57-14', '58-16', '59-17', '60-17', '61-11', '62-13', '63-22', '6
4-14', '65-13', '66-15', '67-15', '68-15', '69-17', '70-7', '71-14', '72-17', '73-15', '74-12', '75-13', '76-14', '77-14', '78-14', '79-10',
'80-9', '81-19', '82-12', '83-13', '84-13', '85-12', '86-11', '87-12', '88-6', '89-12', '90-14', '91-16', '92-13', '93-11', '94-11', '95-1
0', '96-11', '97-7', '98-9', '99-17', '100-11']
Number 1 yields the maximum of Total_solutions:  26
Number 88 yields the minimum of Total_solutions:  6
```


The total solutions
The target numbers