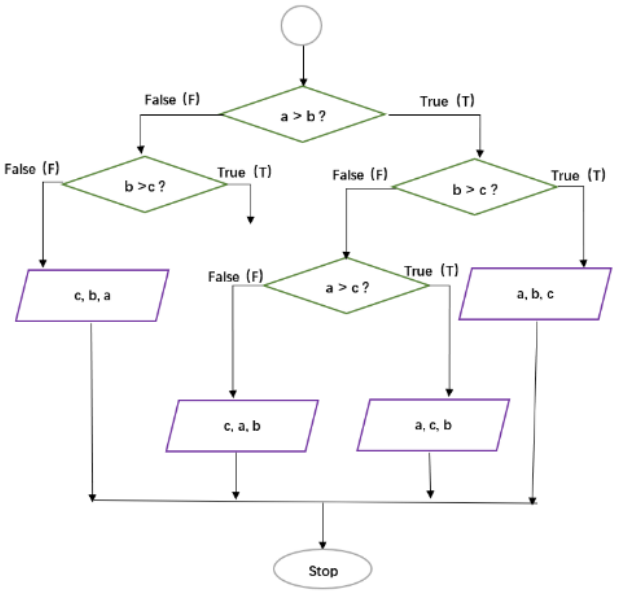
李普双 12332294

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.



Description

When a>b>c, print (a, b, c);

When a>c>b, print (a, c, b);

When c>b>a, print (c, b, a);

When c>a>b, print (c, a, b);

When b>c>a, print (c, a, b);

When b>a>c, print (a, c, b).

Result of Example

When a=6, b=4, c=3, immediately a>b>c, the result is 6, 4, 3, immediately (a, b, c);

When a=6, b=3, c=4, immediately a>c>b, the result is 6, 4, 3, immediately (a, c, b);

When a=3, b=4, c=6, immediately c>b>a, the result is 6, 4, 3, immediately (c, b, a);

When a=4, b=3, c=6, immediately c>a>b, the result is 6, 4, 3, immediately (c, a, b);

When a=3, b=6, c=4, immediately c>a>b, the result is 4, 3, 6, immediately (c, a, b);

When a=4, b=6, c=3, immediately c>a>b, the result is 4, 3, 6, immediately (a, c, b).

Console 1/A

```
    ...:         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)
    ...:
    ...: # Example
    ...: # a=6, b=4, c=3
    ...: Print_values(6,4,3)
    ...: # a=6, b=3, c=4
    ...: Print_values(6,3,4)
    ...: # a=3, b=4, c=6
    ...: Print_values(3,4,6)
    ...: # a=4, b=3, c=6
    ...: Print_values(4,3,6)
    ...: # a=3, b=6, c=4
    ...: Print_values(3,6,4)
    ...: # a=4, b=6, c=3
    ...: Print_values(4,6,3)
6 4 3
6 4 3
6 4 3
6 4 3
4 3 6
4 3 6

In [82]:
```

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.

Result of Example

M1=

[9, 47, 10, 23, 27, 29, 46, 19, 46, 12]

[19, 34, 45, 11, 46, 33, 39, 22, 0, 42]

[19, 31, 21, 31, 22, 23, 49, 45, 32, 11]

[50, 9, 49, 38, 33, 35, 6, 41, 47, 38]

[47, 16, 15, 28, 18, 44, 30, 50, 4, 46]

M2=

[24, 19, 41, 22, 19]

[20, 28, 8, 2, 12]

[2, 36, 34, 50, 4]

[20, 9, 41, 37, 27]

[46, 43, 9, 29, 11]

[31, 6, 45, 25, 10]

[18, 3, 19, 4, 0]

[42, 32, 21, 4, 37]

[24, 1, 28, 35, 30]

[28, 15, 22, 33, 24]

M1*M2=

[6843, 4361, 6401, 5417, 4354]

[7387, 6659, 7058, 6932, 3904]

[7311, 5132, 7259, 5749, 5015]

[8863, 6884, 10345, 9893, 6832]

[8254, 5555, 8599, 6452, 5613]

Console 1/A

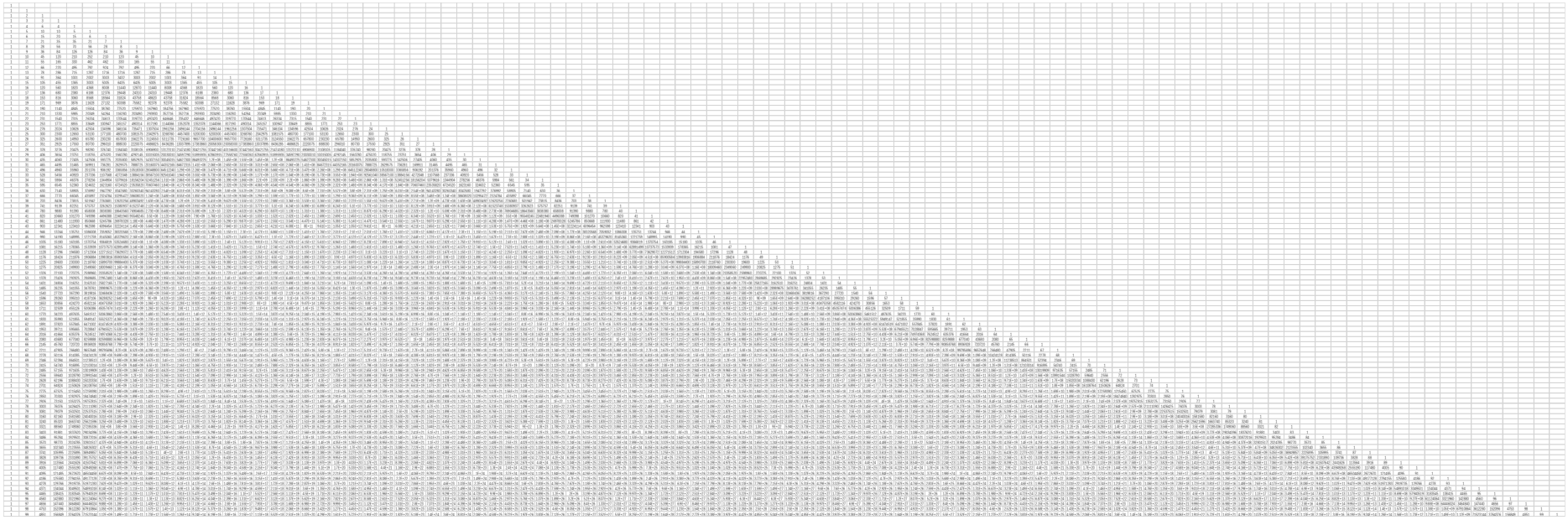
```
In [82]: import random
    ...:
    ...: #2.1
    ...: #定义一个可以生成指定行数和列数的随机矩阵的函数
    ...: def generate_random_matrix(rows,columns):
    ...:     matrix=[]
    ...:     for i in range(rows):
    ...:         row=[]
    ...:         for j in range(columns):
    ...:             random_numbers = random.randint(0,50) #生产一个0~50之间随机数
    ...:             row.append(random_numbers)
    ...:         matrix.append(row)
    ...:     return matrix
    ...:
    ...:
    ...: #生成一个5行10列的矩阵M1
    ...: M1=generate_random_matrix(5,10)
    ...: #生成一个10行5列的矩阵M2
    ...: M2=generate_random_matrix(10,5)
    ...:
    ...: #打印矩阵M1和M2
    ...: print("M1=")
    ...: for row in M1:
    ...:     print(row)
    ...: print("M2=")
    ...: for row in M2:
    ...:     print(row)
    ...:
    ...: #2.2
    ...: #定义矩阵乘法函数
    ...: def Matrix_multip(M1,M2):
    ...: #先判断矩阵是否可以相乘
    ...:     if len(M1) != len(M2[0]):
    ...:         print ("矩阵维度不匹配，无法相乘！")
    ...:     else:
    ...:         multip_matrix = [[0 for _ in range(len(M1))] for _ in range(len(M1))]
    ...:         for i in range(len(M1)):
    ...:             for j in range(len(M1)):
    ...:                 for k in range(len(M2)):
    ...:                     multip_matrix[i][j] = multip_matrix[i][j]+M1[i][k] * M2[k][j]
    ...:         return multip_matrix
    ...:
    ...: #计算并打印M1*M2的结果
    ...: result=Matrix_multip(M1,M2)
    ...: print("M1*M2=")
    ...: for row in result:
    ...:     print(row)
M1=
[11, 45, 11, 29, 43, 18, 11, 28, 13, 28]
[15, 44, 50, 33, 17, 4, 8, 32, 4, 1]
[3, 45, 40, 31, 38, 26, 41, 4, 9, 41]
[38, 2, 43, 23, 25, 9, 2, 10, 50, 34]
[18, 5, 14, 11, 4, 39, 4, 4, 46, 37]
M2=
[36, 31, 50, 40, 27]
[17, 47, 41, 13, 4]
[3, 5, 14, 39, 8]
[37, 43, 20, 23, 34]
[34, 6, 43, 14, 0]
[47, 11, 12, 3, 3]
[2, 10, 45, 4, 39]
[47, 1, 34, 28, 18]
[5, 42, 43, 1, 40]
[0, 46, 15, 33, 2]
M1*M2=
[5978, 6186, 7620, 4542, 3114]
[4965, 4674, 6328, 5096, 3165]
[4969, 6933, 8104, 5226, 3826]
[4379, 6419, 7317, 5589, 4513]
[3577, 5467, 5010, 3152, 3251]

In [83]:
```

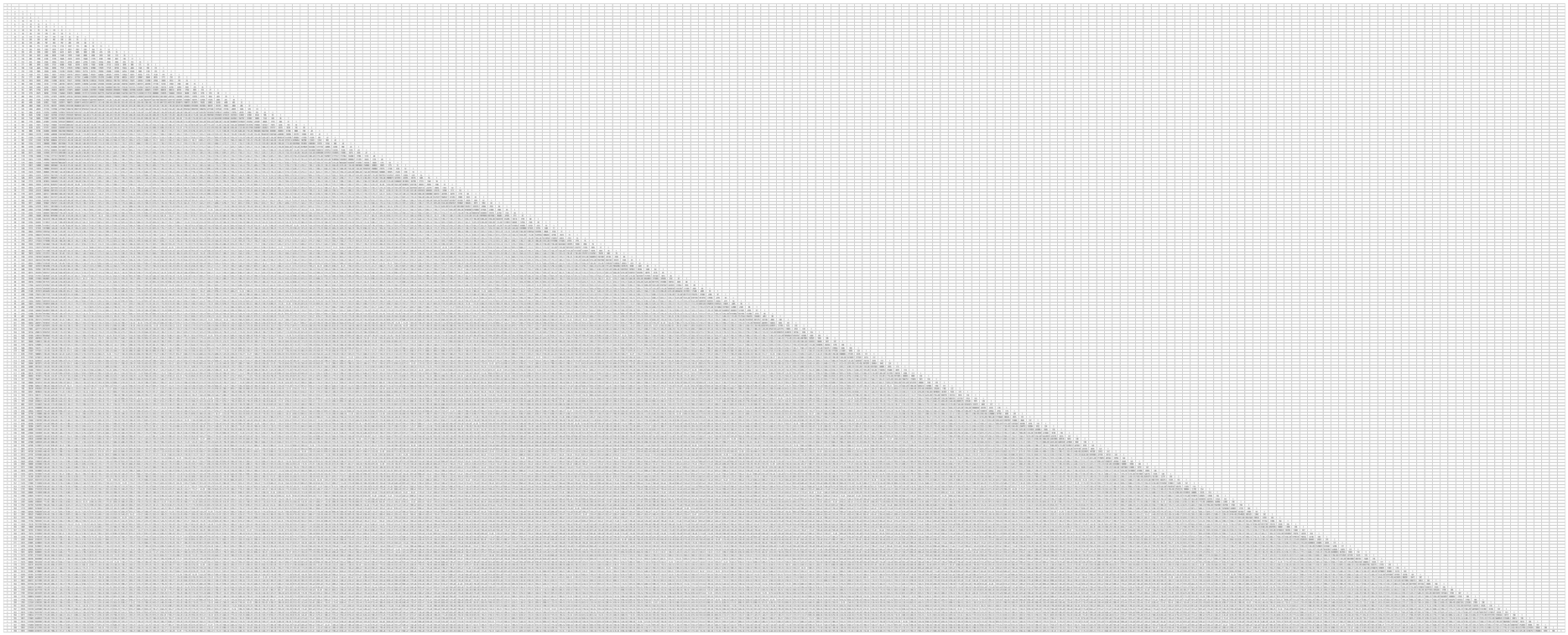
3. Pascal triangle

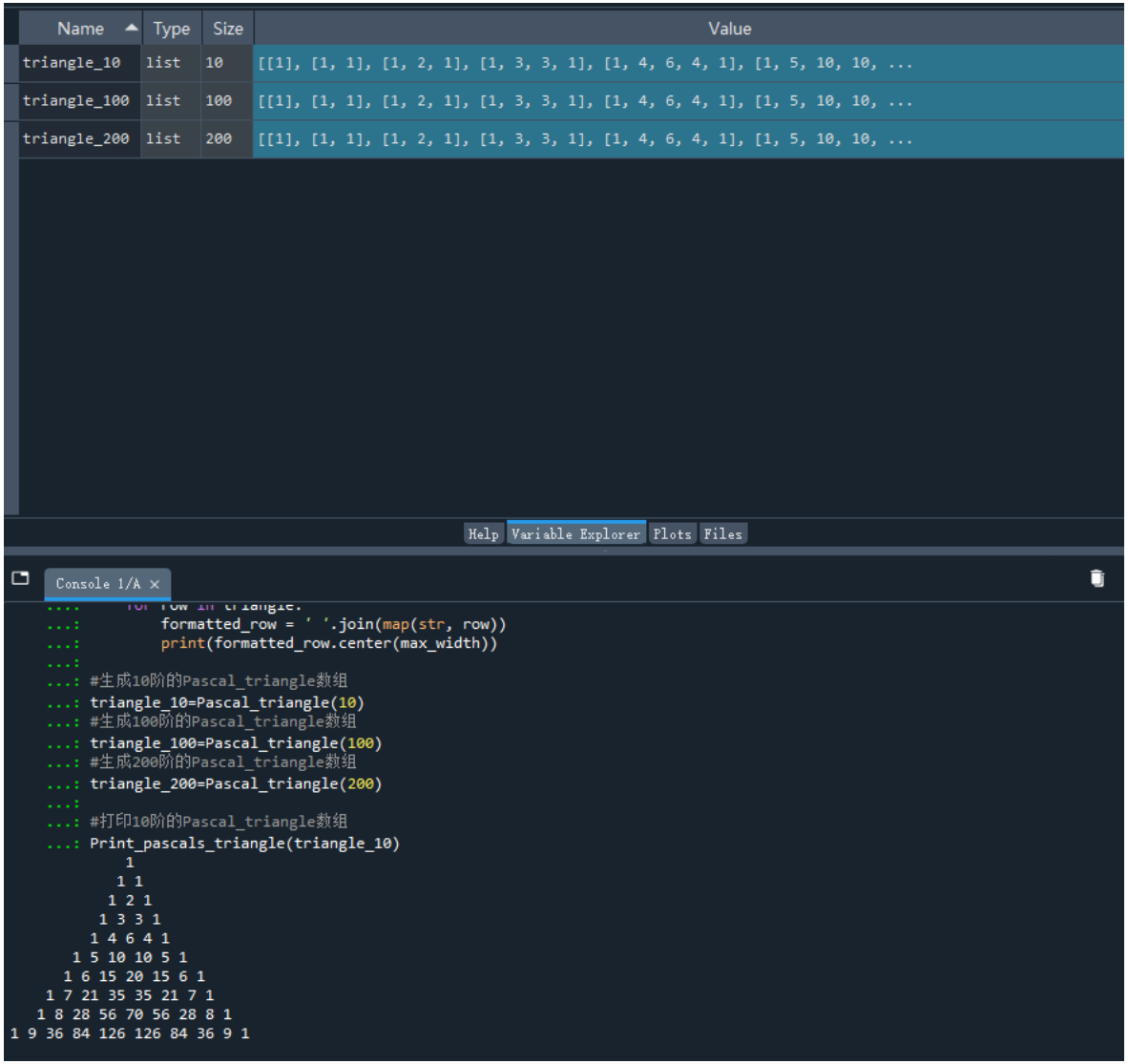
[20 points] One of the most interesting number patterns is Pascal's triangle (named after Blaise Pascal). Write a function `Pascal_triangle` with an argument `k` to print the k^{th} line of the Pascal triangle. Report `Pascal_triangle(100)` and `Pascal_triangle(200)`.

Pascal_triangle(100)



Pascal_triangle(200)




Name
Type
Size
Value
triangle_10
list
10
[[1], [1, 1], [1, 2, 1], [1, 3, 3, 1], [1, 4, 6, 4, 1], [1, 5, 10, 10, ...
triangle_100
list
100
[[1], [1, 1], [1, 2, 1], [1, 3, 3, 1], [1, 4, 6, 4, 1], [1, 5, 10, 10, ...
triangle_200
list
200
[[1], [1, 1], [1, 2, 1], [1, 3, 3, 1], [1, 4, 6, 4, 1], [1, 5, 10, 10, ...
Help
Variable Explorer
Plots
Files
Console 1/A
...: for row in triangle:
...: formatted_row = ' '.join(map(str, row))
...: print(formatted_row.center(max_width))
...:
...: #生成10阶的Pascal_triangle数组
...: triangle_10=Pascal_triangle(10)
...: #生成100阶的Pascal_triangle数组
...: triangle_100=Pascal_triangle(100)
...: #生成200阶的Pascal_triangle数组
...: triangle_200=Pascal_triangle(200)
...:
...: #打印10阶的Pascal_triangle数组
...: Print_pascals_triangle(triangle_10)
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
1 8 28 56 70 56 28 8 1
1 9 36 84 126 126 84 36 9 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`.

Description

To calculate the shortest number of steps from 1 to x, that is, calculate the shortest number of steps from x to 1. Therefore, define a function. When x is an odd number, x=x-1. When x is an even number, x=x/2. The two run alternately until x=1. The shortest number of steps is the number of steps for x to be calculated.

Result of Example

Least_moves(3)=1

Least_moves(5)=3

Least_moves(33)=6

Least_moves(56)=7

Least_moves(84)=8

```
In [87]:
    ...: def Least_moves(x):
    ...:     moves = 0
    ...:     if x == 1:                    #如果x=1,moves=0
    ...:         return moves
    ...:     else:
    ...:         while x > 1:
    ...:             if x % 2 == 0:         #如果x为偶数
    ...:                 x = x // 2         # 除以2
    ...:             else:
    ...:                 x = x - 1          #如果x为奇数，减1
    ...:             moves += 1             #每进行1次运算，步长加1
    ...:         return moves
    ...:
    ...: import random
    ...: x=random.randint(1,100)    #x为1~100的随机数
    ...: moves=Least_moves(x)       #计算到x值的最短步数
    ...: print(f"获得 {x} RMB的最少步数：{moves}")  #输出获得x元的最少步数为moves
获得 56 RMB的最少步数：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:

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

and

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

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 Result of Example (Find_expression(50))

打印出所有 target=50 的等式

1+2+3+4-56+7+89=50

1+2+3-4+56-7+8-9=50

1+2-3+4+56+7-8-9=50

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

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

1+2-34+5-6-7+89=50

1-2+3-45+6+78+9=50

1-2-3+4+56-7-8+9=50

1-2-3-4-5-6+78-9=50

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

1-2+34-5-67+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

12+3+4-56+78+9=50

12-3-4-5+67-8-9=50

12-3+45+6+7-8-9=50

```
打印出所有target=50的等式
1+2+3+4-56+7+89=50
1+2+3-4+56-7+8-9=50
1+2-3+4+56+7-8-9=50
1+2+34-5-6+7+8+9=50
1+2+34-56+78-9=50
1+2-34+5-6-7+89=50
1-2+3-45+6+78+9=50
1-2-3+4+56-7-8+9=50
1-2-3-4-5-6+78-9=50
1-2+34+5+6+7+8-9=50
1-2+34-5-67+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
12+3+4-56+78+9=50
12-3-4-5+67-8-9=50
12-3+45+6+7-8-9=50
Max Solutions: 26 for Target: 1
Min Solutions: 6 for Target: 88
```

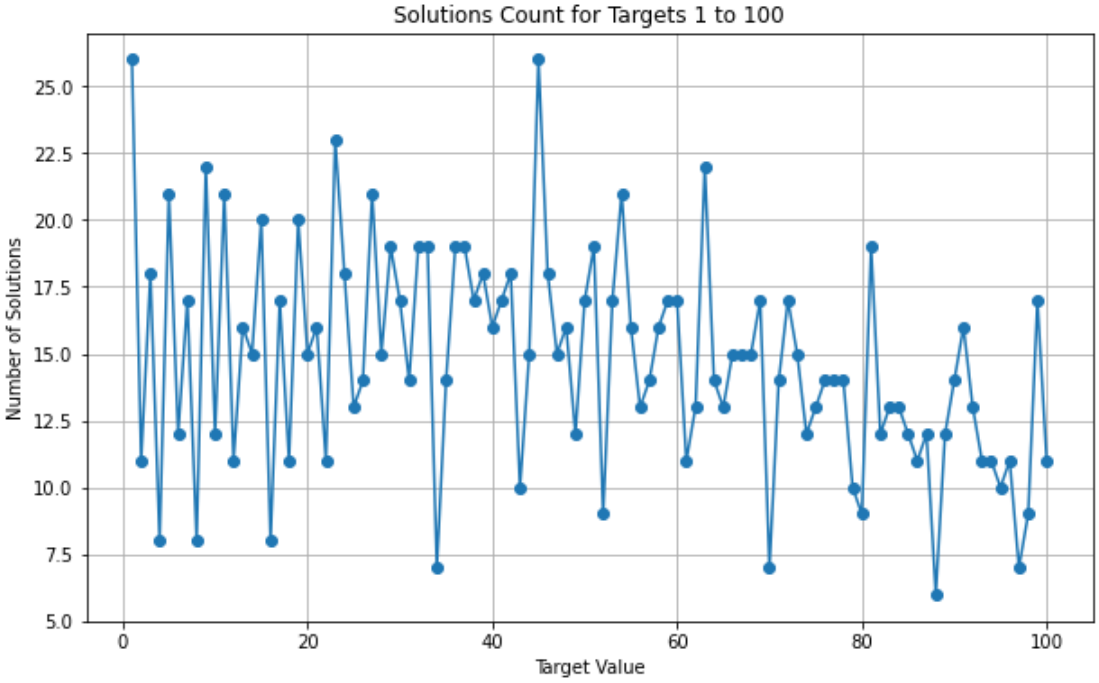
5.2

The maximum of Total_solutions is 26 for target=1.

The minimum of Total_solutions is 6 for target=88.

target	number of solutions	target	number of solutions	target	number of solutions	target	number of solutions	target	number of solutions
1	26	21	16	41	17	61	11	81	19
2	11	22	11	42	18	62	13	82	12
3	18	23	23	43	10	63	22	83	13
4	8	24	18	44	15	64	14	84	13
5	21	25	13	45	26	65	13	85	12
6	12	26	14	46	18	66	15	86	11
7	17	27	21	47	15	67	15	87	12
8	8	28	15	48	16	68	15	88	6
9	22	29	19	49	12	69	17	89	12
10	12	30	17	50	17	70	7	90	14
11	21	31	14	51	19	71	14	91	16
12	11	32	19	52	9	72	17	92	13
13	16	33	19	53	17	73	15	93	11
14	15	34	7	54	21	74	12	94	11
15	20	35	14	55	16	75	13	95	10
16	8	36	19	56	13	76	14	96	11

17	17	37	19	57	14	77	14	97	7
18	11	38	17	58	16	78	14	98	9
19	20	39	18	59	17	79	10	99	17
20	15	40	16	60	17	80	9	100	11



Name	Type	Size	Value
expressions	list	17	[('1+2+3+4-56+7+89', 1), ('1+2+3-4+56-7+8-9', 1), ('1+2-3+4+56+7-8-9', ...
max_solutions	int	1	26
max_target	int	1	1
min_solutions	int	1	6
min_target	int	1	88
target	int	1	100
Total_solutions	list	100	[26, 11, 18, 8, 21, 12, 17, 8, 22, 12, ...]

Help | Variable Explorer | Plots | Files

Console 1/A

```
...:
...: plt.show()
打印出所有target=50的等式
1+2+3+4-56+7+89=50
1+2+3-4+56-7+8-9=50
1+2-3+4+56+7-8-9=50
1+2+34-5-6+7+8+9=50
1+2+34-56+78-9=50
1+2-34+5-6-7+89=50
1-2+3-45+6+78+9=50
1-2-3+4+56-7-8+9=50
1-2-3-4-5-6+78-9=50
1-2+34+5+6+7+8-9=50
1-2+34-5-67+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
12+3+4-56+78+9=50
12-3-4-5+67-8-9=50
12-3+45+6+7-8-9=50
Max Solutions: 26 for Target: 1
Min Solutions: 6 for Target: 88
```