In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

1

In [2]:
```
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)
```

对代码进行测试

In [3]:
```
Print_values(5,8,9)
Print_values(51,82,19)
Print_values(25,48,39)
Print_values(65,48,29)
Print_values(55,68,99)
```

```
9 8 5
65 48 29
99 68 55
```

未输出情况为流程图未包含情况

2

生了两个随机整数矩阵，范围为[0,50],矩阵大小分别为5，10和10，5

In [4]:
```
M1=np.random.randint(0,51,(5,10))
print(M1)
M2=np.random.randint(0,51,(10,5))
print(M2)
```

```
[[32 27 35  6 11 22  6 14  3  0]
 [45  3 43 42 45  9  2  8 11 21]
 [34 34  8  0  3 13 23 35 26 17]
 [ 8 28 34 15  3 27 11  8 22 18]
 [42 45 31 46  3 11 23 26 16 38]]
[[ 8 25 45 50 50]
 [10 36 48 43  5]
 [21 30 18 41 10]
 [43 26 26 31 25]
 [ 2 40 24  3  9]
 [22 18 44 27 14]
 [33  4 31 45  6]
 [43 25 27 34 29]
 [41 24 18 49 41]
 [16 48 12  4 37]]
```

进行矩阵乘法运算，当输入矩阵不满足乘法规则时，输出"the two matrix can not be multipled following array multiple law"

In [5]:
```
def Matrix_multip(M1,M2):
    if M1.shape[0]!=M2.shape[1] or M1.shape[1]!=M2.shape[0]:
        print("the two matrix can not be multipled following array multiple law")
    else:
        M3=np.zeros((M1.shape[0],M1.shape[0]))
        for i in range(0,M1.shape[0],1):
            for j in range(0,M1.shape[0],1):
                for k in range(0,M1.shape[1],1):
                    M3[i,j]=M3[i,j]+M1[i,k]*M2[k,j]
        print(M3)
```

进行测试

In [6]:
```
Matrix_multip(M1,M2)
```

```
[[2948. 4260. 5372. 5902. 3207.]
 [4584. 7057. 6239. 6807. 5748.]
 [4674. 5075. 6280. 7417. 5007.]
 [4200. 4860. 5135. 6118. 3526.]
 [6804. 8064. 8519. 9893. 6920.]]
```

In [7]:
```
Matrix_multip(M2,M1)
```

```
[[5411. 5471. 4965. 4148. 1648. 2886. 2833. 3587. 3369. 4090.]
 [4126. 3439. 3899. 2447. 2018. 2384. 1824. 2582. 2700. 2536.]
 [3382. 2867. 3873. 2461. 1788. 2183. 1281. 1752. 1923. 2054.]
 [4728. 4116. 4660. 2965. 1889. 2630. 1824. 2618. 2173. 2496.]
 [3082. 1479. 2363. 2151. 1930.  896.  884. 1446. 1280. 1644.]
 [3814. 3530. 3248. 1937. 1307. 2101. 1799. 2572. 2226. 2144.]
 [2902. 3487. 3291. 1317.  789. 2446. 1552. 2095. 2035. 1649.]
 [4909. 4411. 4851. 3152. 1868. 2759. 1970. 2773. 2318. 2698.]
 [5118. 5008. 5548. 3875. 1855. 3126. 2190. 2854. 2589. 3250.]
 [4666. 2761. 4003. 3874. 2495. 1455. 1363. 2022. 1568. 2690.]]
```

In [8]:
```
M1=np.random.randint(0,51,(4,10))
print(M1)
M2=np.random.randint(0,51,(10,5))
print(M2)
```

```
[[ 8 32 48 23 41 33 34  2 14  5]
 [27  9  0  8 18 18 23 32 46 22]
 [37 48  3 16 44  5 25 26 22 16]
 [ 8 24 20 28 28 25  0 37 21 26]]
[[26 36 40 47 22]
 [41 46 11 18  1]
 [31 16  3  1 16]
 [12 40 44 39 31]
 [28 47 14 13 35]
 [17 18 30 27 15]
 [19 34  2 50  3]
 [12  6 24  2 25]
 [23 45 19 19  2]
 [27 38 18  9 33]]
```

In [9]:
```
Matrix_multip(M1,M2)
```

```
the two matrix can not be multipled following array multiple law
```

3

In [10]:
```
def Pascal_triangle(k):
    x=np.ones((k,k))
    for i in range(2,k,1):
        for j in range(1,i,1):
            x[i,j]=x[i-1,j-1]+x[i-1,j]
    print(x[k-1,:])
```

对特殊行第一行和第二行进行测试

In [11]:
```
Pascal_triangle(1)
```

```
[1.]
```

In [12]:
```
Pascal_triangle(2)
```

```
[1. 1.]
```

对一般行进行测试

In [13]:
```
Pascal_triangle(5)
```

```
[1. 4. 6. 4. 1.]
```

In [14]:
```
Pascal_triangle(100)
```

```
[1.00000000e+00 9.90000000e+01 4.85100000e+03 1.56849000e+05
 3.76437600e+06 7.15231440e+07 1.12052926e+09 1.48870315e+10
 1.71200863e+11 1.73103095e+12 1.55792785e+13 1.26050526e+14
 9.24370525e+14 6.18617197e+15 3.80007707e+16 2.15337701e+17
 1.13052293e+18 5.51961194e+18 2.51448989e+19 1.07196674e+20
 4.28786696e+20 1.61305471e+21 5.71901217e+21 1.91462581e+22
 6.06298174e+22 1.81889452e+23 5.17685364e+23 1.39966784e+24
 3.59914587e+24 8.81170195e+24 2.05606379e+25 4.57640004e+25
 9.72485009e+25 1.97443926e+26 3.83273504e+26 7.11793650e+26
 1.26541093e+27 2.15461861e+27 3.51543037e+27 5.49849366e+27
 8.24774049e+27 1.18686997e+28 1.63901091e+28 2.17264238e+28
 2.76518120e+28 3.37966592e+28 3.96743390e+28 4.47391483e+28
 4.84674106e+28 5.04456723e+28 5.04456723e+28 4.84674106e+28
 4.47391483e+28 3.96743390e+28 3.37966592e+28 2.76518120e+28
 2.17264238e+28 1.63901091e+28 1.18686997e+28 8.24774049e+27
 5.49849366e+27 3.51543037e+27 2.15461861e+27 1.26541093e+27
 7.11793650e+26 3.83273504e+26 1.97443926e+26 9.72485009e+25
 4.57640004e+25 2.05606379e+25 8.81170195e+24 3.59914587e+24
 1.39966784e+24 5.17685364e+23 1.81889452e+23 6.06298174e+22
 1.91462581e+22 5.71901217e+21 1.61305471e+21 4.28786696e+20
 1.07196674e+20 2.51448989e+19 5.51961194e+18 1.13052293e+18
 2.15337701e+17 3.80007707e+16 6.18617197e+15 9.24370525e+14
 1.26050526e+14 1.55792785e+13 1.73103095e+12 1.71200863e+11
 1.48870315e+10 1.12052926e+09 7.15231440e+07 3.76437600e+06
 1.56849000e+05 4.85100000e+03 9.90000000e+01 1.00000000e+00]
```

In [15]:
```
Pascal_triangle(200)
```

```
[1.00000000e+00 1.99000000e+02 1.97010000e+04 1.29369900e+06
 6.33912510e+07 2.47225879e+09 7.99363675e+10 2.20395985e+12
 5.28950363e+13 1.12255022e+15 2.13284541e+16 3.66461620e+17
 5.74123205e+18 8.25854149e+19 1.09720623e+21 1.35322101e+22
 1.55620416e+23 1.67520801e+24 1.69382143e+25 1.61358779e+26
 1.45222901e+27 1.23785235e+28 1.00153508e+29 7.70746561e+29
 5.65214145e+30 3.95649902e+31 2.64781088e+32 1.69656030e+33
 1.04217276e+34 6.14522558e+34 3.48229449e+35 1.89841216e+36
 9.96666383e+36 5.04373594e+37 2.46252990e+38 1.16090695e+39
 5.28857612e+39 2.32983218e+40 9.93244246e+40 4.10031599e+41
 1.64012640e+42 6.36049017e+42 2.39275583e+43 8.73634104e+43
 3.09743000e+44 1.06689256e+45 3.57177074e+45 1.16272537e+46
 3.68196366e+46 1.13464594e+47 3.40393783e+47 9.94483799e+47
 2.83045389e+48 7.85050418e+48 2.12254372e+49 5.59579709e+49
 1.43891925e+50 3.60992023e+50 8.83808056e+50 2.11215145e+51
 4.92835339e+51 1.12301823e+52 2.49962123e+52 5.43568426e+52
 1.15508290e+53 2.39901834e+53 4.87073421e+53 9.66877089e+53
 1.87687905e+54 3.56335009e+54 6.61765016e+54 1.20236179e+55
 2.13753207e+55 3.71872018e+55 6.33187490e+55 1.05531248e+56
 1.72182563e+56 2.75044873e+56 4.30198392e+56 6.58911461e+56
 9.88367191e+56 1.45204563e+57 2.08952907e+57 2.94548074e+57
 4.06756864e+57 5.50318111e+57 7.29491449e+57 9.47500388e+57
 1.20590958e+58 1.50399959e+58 1.83822173e+58 2.20182602e+58
 2.58475229e+58 2.97385478e+58 3.35349582e+58 3.70649538e+58
 4.01536999e+58 4.26374340e+58 4.43777374e+58 4.52742573e+58
 4.52742573e+58 4.43777374e+58 4.26374340e+58 4.01536999e+58
 3.70649538e+58 3.35349582e+58 2.97385478e+58 2.58475229e+58
 2.20182602e+58 1.83822173e+58 1.50399959e+58 1.20590958e+58
 9.47500388e+57 7.29491449e+57 5.50318111e+57 4.06756864e+57
 2.94548074e+57 2.08952907e+57 1.45204563e+57 9.88367191e+56
 6.58911461e+56 4.30198392e+56 2.75044873e+56 1.72182563e+56
 1.05531248e+56 6.33187490e+55 3.71872018e+55 2.13753207e+55
 1.20236179e+55 6.61765016e+54 3.56335009e+54 1.87687905e+54
 9.66877089e+53 4.87073421e+53 2.39901834e+53 1.15508290e+53
 5.43568426e+52 2.49962123e+52 1.12301823e+52 4.92835339e+51
 2.11215145e+51 8.83808056e+50 3.60992023e+50 1.43891925e+50
 5.59579709e+49 2.12254372e+49 7.85050418e+48 2.83045389e+48
 9.94483799e+47 3.40393783e+47 1.13464594e+47 3.68196366e+46
 1.16272537e+46 3.57177074e+45 1.06689256e+45 3.09743000e+44
 8.73634104e+43 2.39275583e+43 6.36049017e+42 1.64012640e+42
 4.10031599e+41 9.93244246e+40 2.32983218e+40 5.28857612e+39
 1.16090695e+39 2.46252990e+38 5.04373594e+37 9.96666383e+36
 1.89841216e+36 3.48229449e+35 6.14522558e+34 1.04217276e+34
 1.69656030e+33 2.64781088e+32 3.95649902e+31 5.65214145e+30
 7.70746561e+29 1.00153508e+29 1.23785235e+28 1.45222901e+27
 1.61358779e+26 1.69382143e+25 1.67520801e+24 1.55620416e+23
 1.35322101e+22 1.09720623e+21 8.25854149e+19 5.74123205e+18
 3.66461620e+17 2.13284541e+16 1.12255022e+15 5.28950363e+13
 2.20395985e+12 7.99363675e+10 2.47225879e+09 6.33912510e+07
 1.29369900e+06 1.97010000e+04 1.99000000e+02 1.00000000e+00]
```

4

In [16]:
```
def least_moves(x):
    n=0
    while x>1:
        if x%2==1:
            x=(x-1)/2
            n=n+2
        else:
            x=x/2
            n=n+1
    print(n)
```

进行测试

In [17]:
```
least_moves(2)
least_moves(3)
least_moves(5)
least_moves(245)
least_moves(1543)
```

```
1
2
3
12
14
```

5

In [18]:
```
def Find_expression(a):
    y=np.array(["","+","-"])
    for i in y:
        for j in y:
            for k in y:
                for l in y:
                    for m in y:
                        for n in y:
                            for o in y:
                                for q in y:
                                    z="1"+i+"2"+j+"3"+k+"4"+l+"5"+m+"6"+n+"7"+o+"8"
                                    if eval(z)==a:
                                        print(z+"="+str(a))
```

In [19]:
```
def Find_expression_count(a):
    y=np.array(["","+","-"])
    count=0
    for i in y:
        for j in y:
            for k in y:
                for l in y:
                    for m in y:
                        for n in y:
                            for o in y:
                                for q in y:
                                    z="1"+i+"2"+j+"3"+k+"4"+l+"5"+m+"6"+n+"7"+o+"8"
                                    if eval(z)==a:
                                        count+=1
    return count
```

进行测试

In [20]:
```
Find_expression(50)
```

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

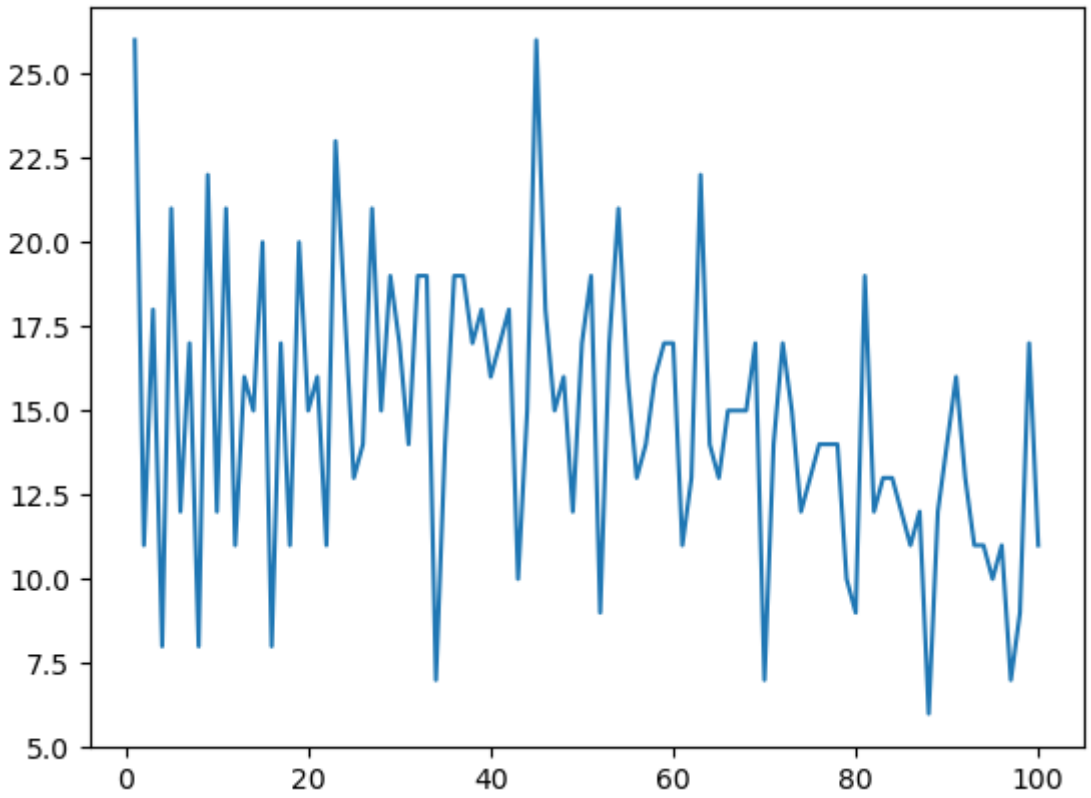
In [21]:
```
Find_expression(75)
```

```
123+45-6-78-9=75
12+3-4+56+7-8+9=75
12-34-5+6+7+89=75
12-3+4+56+7+8-9=75
12-3-4-5+6+78-9=75
1+23-4+5+67-8-9=75
1+2+3-4+5+67-8+9=75
1+2-3+45+6+7+8+9=75
1+2-3+4+5+67+8-9=75
1+2-3+4-5-6-7+89=75
1+2-3-4-5+67+8+9=75
1-23+4+5+6-7+89=75
1-2-3-4-5+6-7+89=75
```

获取值为1-100时每个值可能结果的个数

In [22]:
```
x=np.linspace(1,100,100)
Total_solutions=np.zeros((100))
for i in range(0,100,1):
    Total_solutions[i]=Find_expression_count(i+1)
plt.plot(x,Total_solutions)
```

Out[22]: [<matplotlib.lines.Line2D at 0x24991e40cd0>]



In [23]:
```
print(max(Total_solutions))
print(np.where(Total_solutions==max(Total_solutions)))
```

```
26.0
(array([ 0, 44], dtype=int64),)
```

当值为1和45时，有最多可能的结果，有26种结果

In [24]:
```
print(min(Total_solutions))
print(np.where(Total_solutions==min(Total_solutions)))
```

```
6.0
(array([87], dtype=int64),)
```

当值为88时，有最少可能的结果，有6种结果