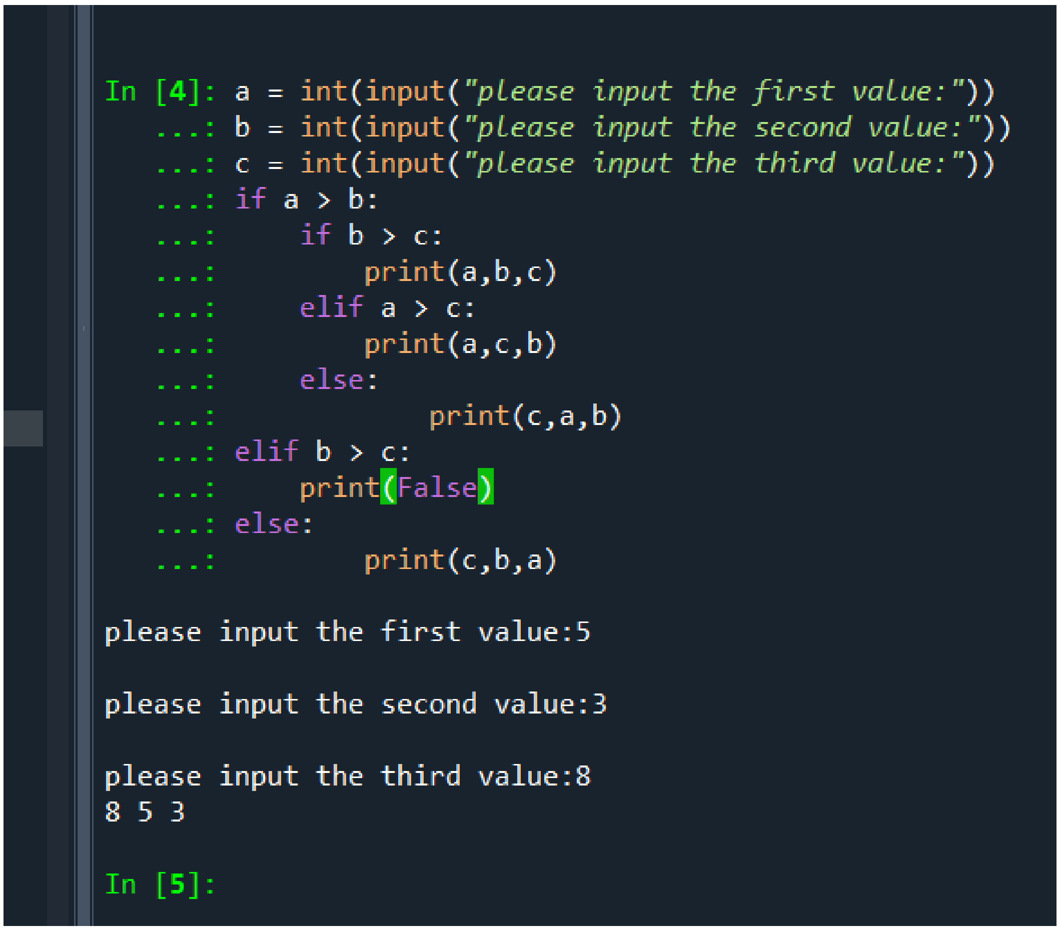
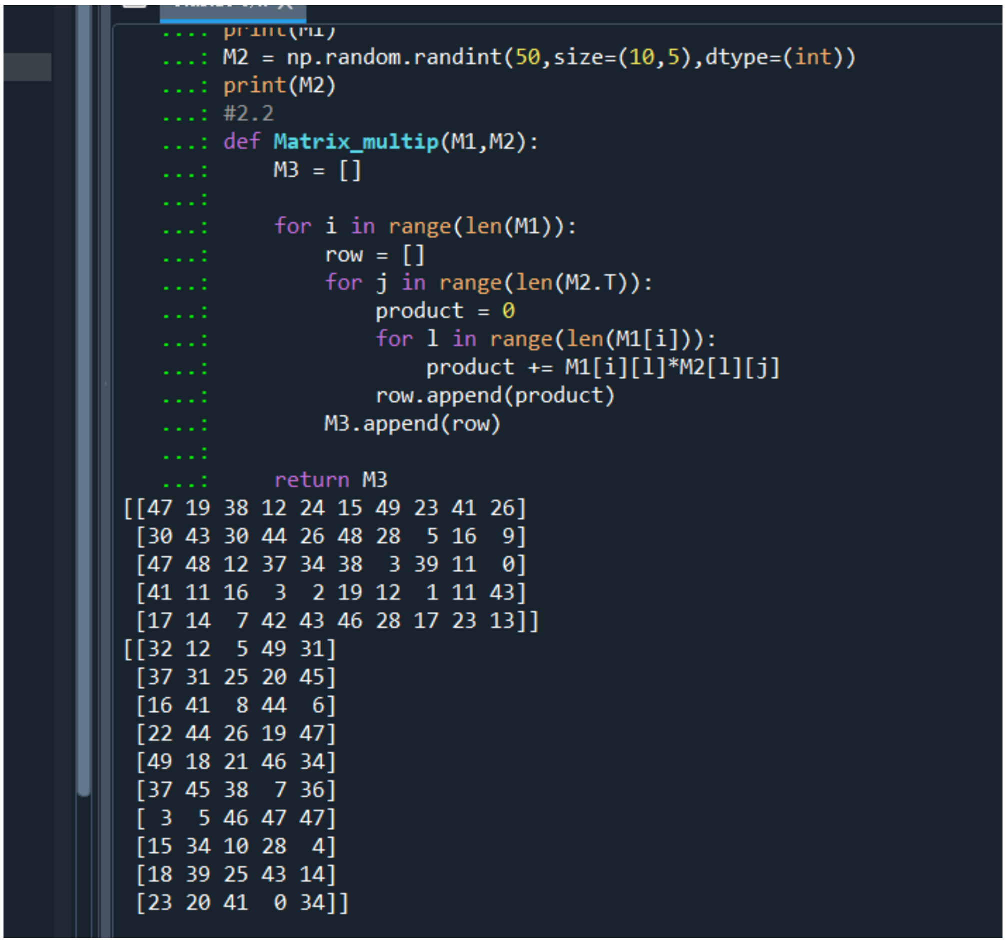
运行结果截图

1.



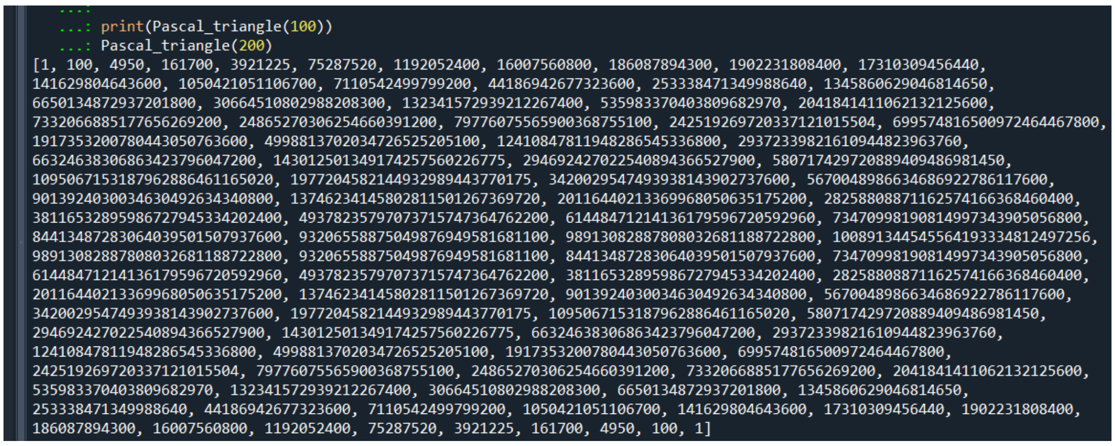
```
In [4]: a = int(input("please input the first value:"))
   ...: b = int(input("please input the second value:"))
   ...: c = int(input("please input the third value:"))
   ...: if a > b:
   ...:     if b > c:
   ...:         print(a,b,c)
   ...:     elif a > c:
   ...:         print(a,c,b)
   ...:     else:
   ...:             print(c,a,b)
   ...: elif b > c:
   ...:     print(False)
   ...: else:
   ...:         print(c,b,a)

please input the first value:5

please input the second value:3

please input the third value:8
8 5 3

In [5]:
```

2.

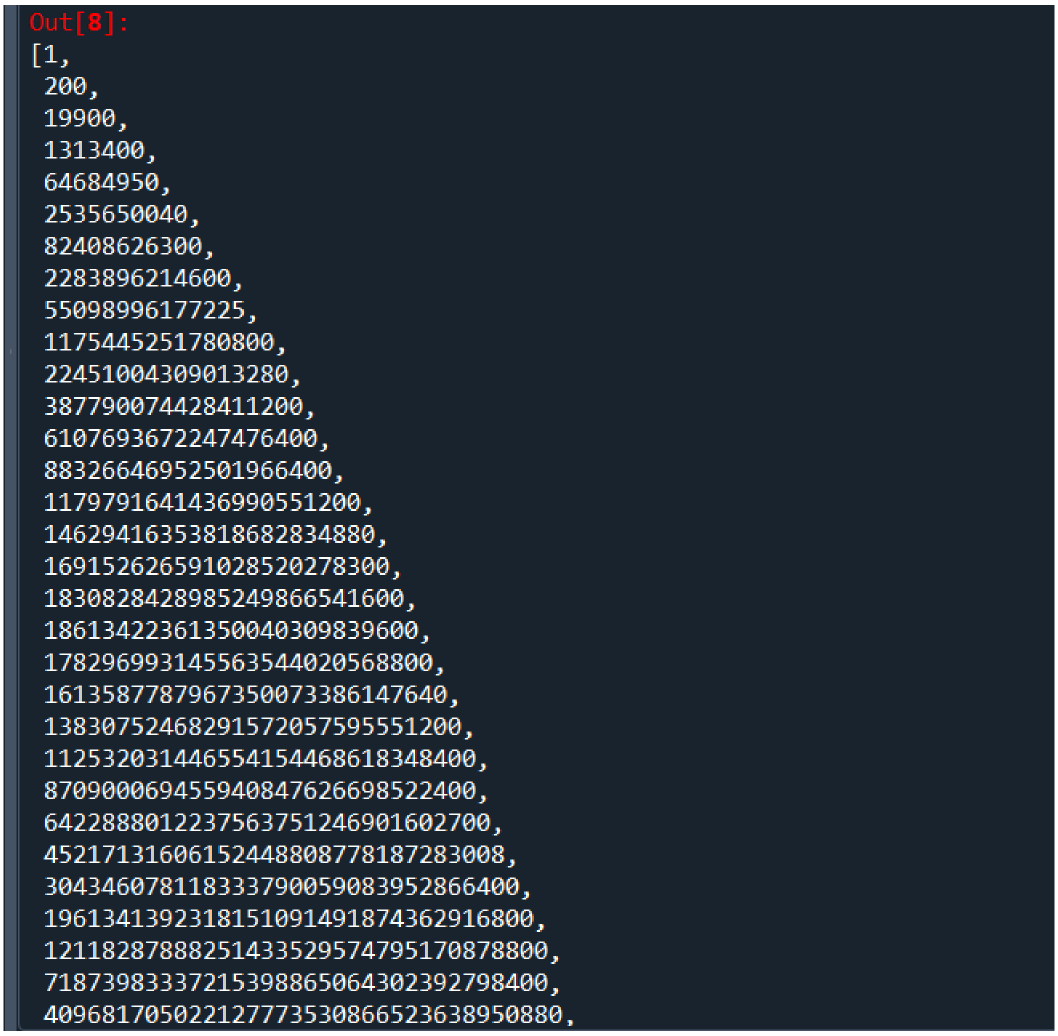


```
   ...: print(M1)
   ...: M2 = np.random.randint(50,size=(10,5),dtype=(int))
   ...: print(M2)
   ...: #2.2
   ...: def Matrix_multip(M1,M2):
   ...:     M3 = []
   ...:
   ...:     for i in range(len(M1)):
   ...:         row = []
   ...:         for j in range(len(M2.T)):
   ...:             product = 0
   ...:             for l in range(len(M1[i])):
   ...:                 product += M1[i][l]*M2[l][j]
   ...:             row.append(product)
   ...:         M3.append(row)
   ...:
   ...:     return M3
[[47 19 38 12 24 15 49 23 41 26]
 [30 43 30 44 26 48 28  5 16  9]
 [47 48 12 37 34 38  3 39 11  0]
 [41 11 16  3  2 19 12  1 11 43]
 [17 14  7 42 43 46 28 17 23 13]]
[[32 12  5 49 31]
 [37 31 25 20 45]
 [16 41  8 44  6]
 [22 44 26 19 47]
 [49 18 21 46 34]
 [37 45 38  7 36]
 [ 3  5 46 47 47]
 [15 34 10 28  4]
 [18 39 25 43 14]
 [23 20 41  0 34]]
```

3.

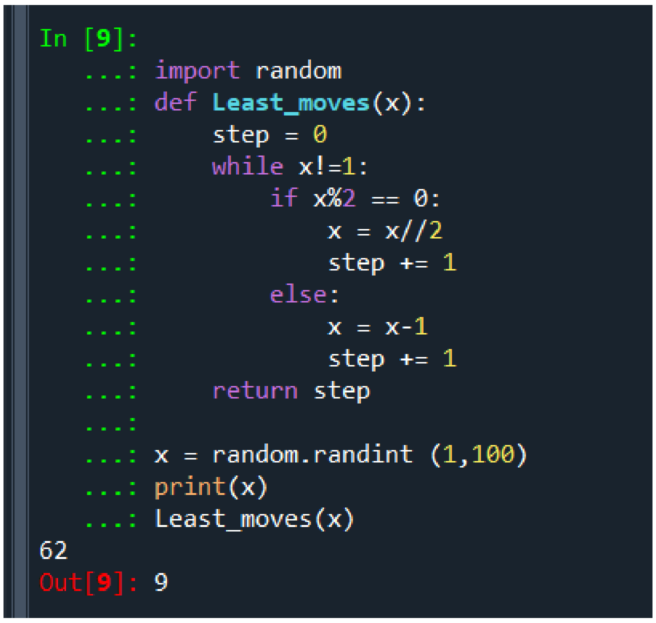


```
   ...:
   ...: print(Pascal_triangle(100))
   ...: Pascal_triangle(200)
[1, 100, 4950, 161700, 3921225, 75287520, 1192052400, 16007560800, 186087894300, 1902231808400, 17310309456440,
141629804643600, 1050421051106700, 7110542499799200, 44186942677323600, 253338471349988640, 1345860629046814650,
6650134872937201800, 30664510802988208300, 132341572939212267400, 535983370403809682970, 2041841411062132125600,
7332066885177656269200, 24865270306254660391200, 79776075565900368755100, 242519269720337121015504, 699574816500972464467800,
1917353200780443050763600, 4998813702034726525205100, 12410847811948286545336800, 29372339821610944823963760,
66324638306863423796047200, 143012501349174257560226775, 294692427022540894366527900, 580717429720889409486981450,
1095067153187962886461165020, 1977204582144932989443770175, 3420029547493938143902737600, 5670048986634686922786117600,
9013924030034630492634340800, 13746234145802811501267369720, 20116440213369968050635175200, 28258808871162574166368460400,
38116532895986727945334202400, 49378235797073715747364762200, 61448471214136179596720592960, 73470998190814997343905056800,
84413487283064039501507937600, 93206558875049876949581681100, 98913082887808032681188722800, 100891344545564193334812497256,
98913082887808032681188722800, 93206558875049876949581681100, 84413487283064039501507937600, 73470998190814997343905056800,
61448471214136179596720592960, 49378235797073715747364762200, 38116532895986727945334202400, 28258808871162574166368460400,
20116440213369968050635175200, 13746234145802811501267369720, 9013924030034630492634340800, 5670048986634686922786117600,
3420029547493938143902737600, 1977204582144932989443770175, 1095067153187962886461165020, 580717429720889409486981450,
294692427022540894366527900, 143012501349174257560226775, 66324638306863423796047200, 29372339821610944823963760,
12410847811948286545336800, 4998813702034726525205100, 1917353200780443050763600, 699574816500972464467800,
242519269720337121015504, 79776075565900368755100, 24865270306254660391200, 7332066885177656269200, 2041841411062132125600,
535983370403809682970, 132341572939212267400, 30664510802988208300, 6650134872937201800, 1345860629046814650,
253338471349988640, 44186942677323600, 7110542499799200, 1050421051106700, 141629804643600, 17310309456440, 1902231808400,
186087894300, 16007560800, 1192052400, 75287520, 3921225, 161700, 4950, 100, 1]
```



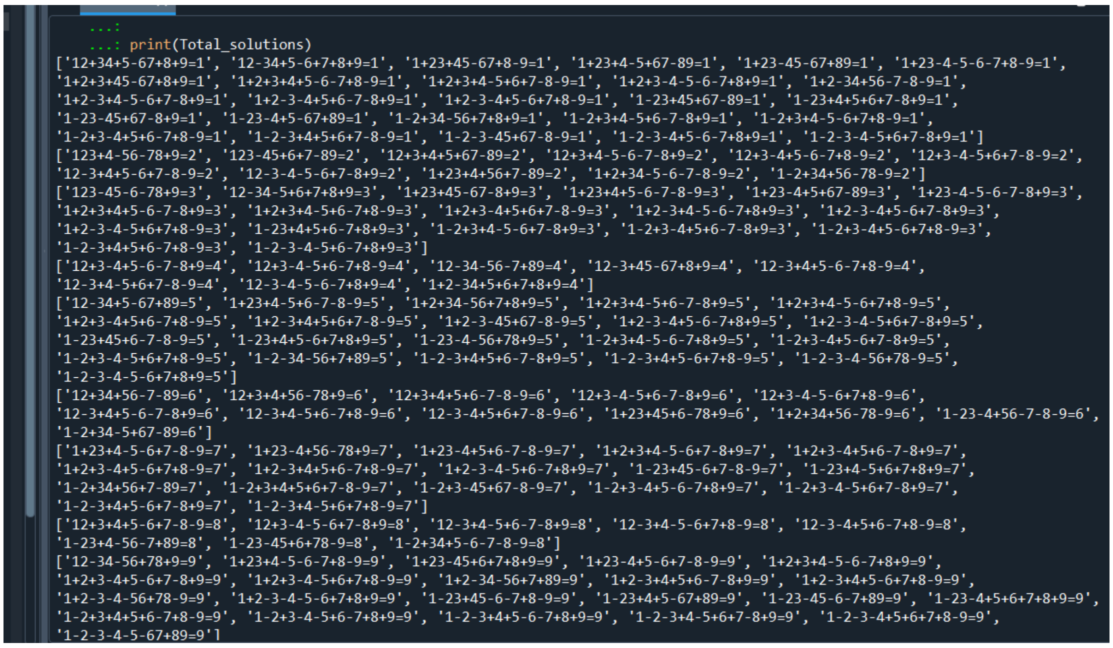
```
Out[8]:
[1,
 200,
 19900,
 1313400,
 64684950,
 2535650040,
 82408626300,
 2283896214600,
 55098996177225,
 1175445251780800,
 22451004309013280,
 387790074428411200,
 6107693672247476400,
 88326646952501966400,
 1179791641436990551200,
 14629416353818682834880,
 169152626591028520278300,
 1830828428985249866541600,
 18613422361350040309839600,
 178296993145563544020568800,
 1613587787967350073386147640,
 13830752468291572057595551200,
 112532031446554154468618348400,
 870900069455940847626698522400,
 6422888012237563751246901602700,
 45217131606152448808778187283008,
 304346078118333790059083952866400,
 1961341392318151091491874362916800,
 12118287888251433529574795170878800,
 71873983337215398865064302392798400,
 409681705022127773530866523638950880,
```

4.

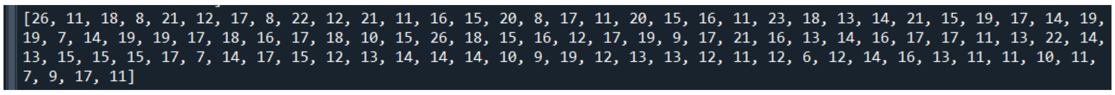


```
In [9]:
   ...: import random
   ...: def Least_moves(x):
   ...:     step = 0
   ...:     while x!=1:
   ...:         if x%2 == 0:
   ...:             x = x//2
   ...:             step += 1
   ...:         else:
   ...:             x = x-1
   ...:             step += 1
   ...:     return step
   ...:
   ...: x = random.randint (1,100)
   ...: print(x)
   ...: Least_moves(x)
62
Out[9]: 9
```

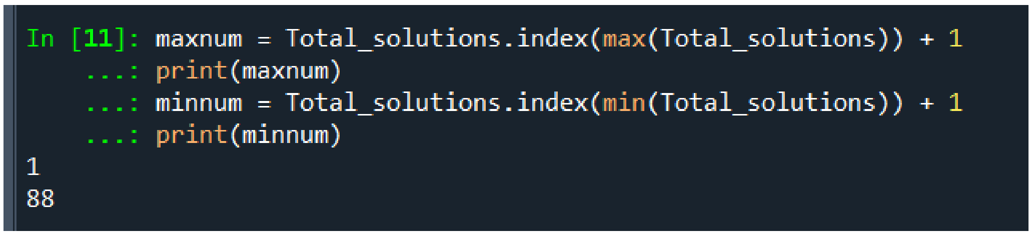
5.



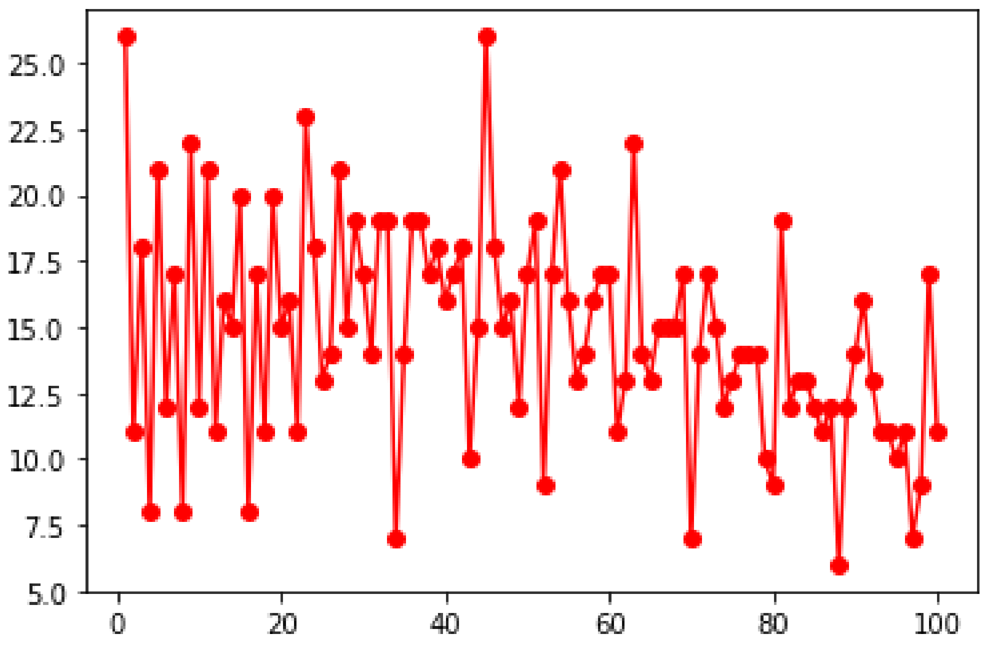
```
   ...:
   ...: print(Total_solutions)
['12+34+5-67+8+9=1', '12-34+5-6+7+8+9=1', '1+23+45-67+8-9=1', '1+23+4-5+67-89=1', '1+23-45-67+89=1', '1+23-4-5-6-7+8-9=1',
'1+2+3+45-67+8+9=1', '1+2+3+4+5-6-7+8-9=1', '1+2+3+4-5+6+7-8-9=1', '1+2+3-4-5-6-7+8+9=1', '1+2-34+56-7-8-9=1',
'1+2-3+4-5-6+7-8+9=1', '1+2-3-4+5+6-7-8+9=1', '1+2-3-4+5-6+7+8-9=1', '1-23+45+67-89=1', '1-23+4+5+6+7-8+9=1',
'1-23-45+67-8+9=1', '1-23-4+5-67+89=1', '1-2+34-56+7+8+9=1', '1-2+3+4-5+6-7-8+9=1', '1-2+3+4-5-6+7+8-9=1',
'1-2+3-4+5+6-7+8-9=1', '1-2-3+4+5+6+7-8-9=1', '1-2-3-45+67-8-9=1', '1-2-3-4+5-6-7+8+9=1', '1-2-3-4-5+6+7-8+9=1']
['123+4-56-78+9=2', '123-45+6+7-89=2', '12+3+4+5+67-89=2', '12+3+4-5-6-7-8+9=2', '12+3-4+5-6-7+8-9=2', '12+3-4-5+6+7-8-9=2',
'12-3+4+5-6+7-8-9=2', '12-3-4-5-6+7-8+9=2', '1+23+4+56+7-89=2', '1+2+34-5-6-7-8-9=2', '1-2+34+56-78-9=2']
['123-45-6-78+9=3', '12-34-5+6+7+8+9=3', '1+23+45-67-8+9=3', '1+23+4+5-6-7-8-9=3', '1+23-4+5+67-89=3', '1+23-4-5-6-7-8+9=3',
'1+2+3+4+5-6-7-8+9=3', '1+2+3+4-5+6-7+8-9=3', '1+2+3-4+5+6+7-8-9=3', '1+2-3+4-5-6-7+8+9=3', '1+2-3-4+5-6+7-8+9=3',
'1+2-3-4-5+6+7+8-9=3', '1-23+4+5+6-7+8+9=3', '1-2+3+4-5-6+7-8+9=3', '1-2+3-4+5+6-7-8+9=3', '1-2+3-4+5-6+7+8-9=3',
'1-2-3+4+5+6-7+8-9=3', '1-2-3-4-5+6-7+8+9=3']
['12+3-4+5-6-7-8+9=4', '12+3-4-5+6-7+8-9=4', '12-34-56-7+89=4', '12-3+45-67+8+9=4', '12-3+4+5-6-7+8-9=4',
'12-3+4-5+6+7-8-9=4', '12-3-4-5-6-7+8+9=4', '1+2-34+5+6+7+8+9=4']
['12-34+5-67+89=5', '1+23+4-5+6-7-8-9=5', '1+2+34-56+7+8+9=5', '1+2+3+4-5+6-7-8+9=5', '1+2+3+4-5-6+7+8-9=5',
'1+2+3-4+5+6-7+8-9=5', '1+2-3+4+5+6+7-8-9=5', '1+2-3-45+67-8-9=5', '1+2-3-4+5-6-7+8+9=5', '1+2-3-4-5+6+7-8+9=5',
'1-23+45+6-7-8-9=5', '1-23+4+5-6+7+8+9=5', '1-23-4-56+78+9=5', '1-2+3+4-5-6-7+8+9=5', '1-2+3-4+5-6+7-8+9=5',
'1-2+3-4-5+6+7+8-9=5', '1-2-34-56+7+89=5', '1-2-3+4+5+6-7-8+9=5', '1-2-3+4+5-6+7+8-9=5', '1-2-3-4-56+78-9=5',
'1-2-3-4-5-6+7+8+9=5']
['12+34+56-7-89=6', '12+3+4+56-78+9=6', '12+3+4+5+6-7-8-9=6', '12+3-4-5+6-7-8+9=6', '12+3-4-5-6+7+8-9=6',
'12-3+4+5-6-7-8+9=6', '12-3+4-5+6-7+8-9=6', '12-3-4+5+6+7-8-9=6', '1+23+45+6-78+9=6', '1+2+34+56-78-9=6', '1-23-4+56-7-8-9=6',
'1-2+34-5+67-89=6']
['1+23+4-5-6+7-8-9=7', '1+23-4+56-78+9=7', '1+23-4+5+6-7-8-9=7', '1+2+3+4-5-6+7-8+9=7', '1+2+3-4+5+6-7-8+9=7',
'1+2+3-4+5-6+7+8-9=7', '1+2-3+4+5+6-7+8-9=7', '1+2-3-4-5+6-7+8+9=7', '1-23+45-6+7-8-9=7', '1-23+4-5+6+7+8+9=7',
'1-2+34+56+7-89=7', '1-2+3+4+5+6+7-8-9=7', '1-2+3-45+67-8-9=7', '1-2+3-4+5-6-7+8+9=7', '1-2+3-4-5+6+7-8+9=7',
'1-2-3+4+5-6+7-8+9=7', '1-2-3+4-5+6+7+8-9=7']
['12+3+4+5-6+7-8-9=8', '12+3-4-5-6+7-8+9=8', '12-3+4-5+6-7-8+9=8', '12-3+4-5-6+7+8-9=8', '12-3-4+5+6-7+8-9=8',
'1-23+4-56-7+89=8', '1-23-45+6+78-9=8', '1-2+34+5-6-7-8-9=8']
['12-34-56+78+9=9', '1+23+4-5-6-7+8-9=9', '1+23-45+6+7+8+9=9', '1+23-4+5-6+7-8-9=9', '1+2+3+4-5-6-7+8+9=9',
'1+2+3-4+5-6+7-8+9=9', '1+2+3-4-5+6+7+8-9=9', '1+2-34-56+7+89=9', '1+2-3+4+5+6-7-8+9=9', '1+2-3+4+5-6+7+8-9=9',
'1+2-3-4-56+78-9=9', '1+2-3-4-5-6+7+8+9=9', '1-23+45-6-7+8-9=9', '1-23+4+5-67+89=9', '1-23-45-6-7+89=9', '1-23-4+5+6+7+8+9=9',
'1-2+3+4+5+6-7+8-9=9', '1-2+3-4-5+6-7+8+9=9', '1-2-3+4+5-6-7+8+9=9', '1-2-3+4-5+6+7-8+9=9', '1-2-3-4+5+6+7+8-9=9',
'1-2-3-4-5-67+89=9']
```



```
[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, 17, 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]
```



```
In [11]: maxnum = Total_solutions.index(max(Total_solutions)) + 1
    ...: print(maxnum)
    ...: minnum = Total_solutions.index(min(Total_solutions)) + 1
    ...: print(minnum)
1
88
```



25.0
22.5
20.0
17.5
15.0
12.5
10.0
7.5
5.0
0
20
40
60
80
100