In [67]:

```
#  1. Flowchart
def Print_values(a,b,c):
    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(c,b,a)
import random
a = random.randint(0,50)
b = random.randint(0,50)
c = random.randint(0,50)
print("a:",a,"\nb:",b," \nc:",c)
Print_values(a,b,c)
```

```
a: 36
b: 15
c: 19
36 19 15
```

In [71]:

```
#  2. Matrix multiplication
#  2.1
import numpy as np
M1 = np.random.randint(0,50,(5,10))
M2 = np.random.randint(0,50,(10,5))
print("M1: \n",M1)
print("M2: \n",M2)
```

```
M1:
 [[ 9 44 11 17 41  7 42 21 24 41]
 [24  1 45 36 26 48 19 25 41 44]
 [46  2 22  1 22 25 22 45 41 26]
 [31 20 15  2 37 10 33  2  7 22]
 [19  7  3  2 19 46 17  1 32 19]]
M2:
 [[38 28  7  5 24]
 [44 23  4 26 18]
 [44 28 15 12 26]
 [28 17 34 16  9]
 [21  7 40  3 29]
 [46 12 23 35 31]
 [42 14 17 27 34]
 [26  6  9 47 15]
 [47  2 20 13 27]
 [34  7 49 10 36]]
```

In [72]:

```
#  2.2
def Matrix_multip(M1,M2):
    import numpy as np
    row1 = 5
    column1 = 5
    #  K为M1的行数及M2的列数
    K = 10
    M3 = np.zeros((row1,column1))
#  使用三重for循环时，我参考了网址：https://blog.csdn.net/m0_52025744/article/details/121947127
    for i in range(row1):
        for j in range(column1):
            for k in range(K):
                M3[i,j] = M3[i,j] + M1[i,k] * M2[k,j]
    print (M3)
Matrix_multip(M1,M2)
```

```
[[ 9253.   3281.   6175.   4804.   6720.]
 [11569.   4131.   7739.   5681.   8042.]
 [ 9349.   3263.   5022.   5005.   6600.]
 [ 6526.   2803.   4097.   2644.   5028.]
 [ 6623.   1937.   3961.   3124.   4796.]]
```

In [73]:

```
#  3. Pascal triangle
N = [1]
Rows = 100
#  当行数为200时，将上行Rows的值改为200

for i in range(Rows):
    N.append(0)
    N = [N[K]+N[K-1]for K in range(i+1)]
print(N)
#  以下结果目前显示的是第100行的数字集
```

```
[1, 99, 4851, 156849, 3764376, 71523144, 1120529256, 14887031544, 171200862756, 1731030945644, 15
579278510796, 126050526132804, 924370524973896, 6186171974825304, 38000770702498296, 215337700647
490344, 1130522928399324306, 5519611944537877494, 25144898858450330806, 107196674080761936594, 42
8786696323047746376, 1613054714739084379224, 5719012170438571889976, 19146258135816088501224, 606
29817430084280253876, 181889452290252840761628, 517685364210719623706172, 139966783656972342705742
8, 3599145865465003098147672, 8811701946483283447189128, 20560637875127661376774632, 45764000431
735762419272568, 97248500917438495140954207, 197443926105102399225573693, 38327350361578701026140
7757, 711793649572175876199757263, 1265410932572757113244012912, 2154618614921181030658724688, 35
15430371713505892127392912, 5498493658321124600506947888, 8247740487481686900760421832, 118686997
25888281149874753368, 16390109145274293016493707032, 21726423750712434928840495368, 2765181204636
1280818524266832, 33796659167774898778196326128, 39674339023040098565708730672, 44739148260023940
935799206928, 48467410615025936013782474172, 50445672272782096667406248628, 504456722727820966674
06248628, 48467410615025936013782474172, 44739148260023940935799206928, 396743390230400985657087
30672, 33796659167774898778196326128, 27651812046361280818524266832, 217264237507124349288404953
68, 16390109145274293016493707032, 11868699725888281149874753368, 8247740487481686900760421832, 54
98493658321124600506947888, 3515430371713505892127392912, 2154618614921181030658724688, 126541093
2572757113244012912, 711793649572175876199757263, 383273503615787010261407757, 197443926105102399
225573693, 97248500917438495140954207, 45764000431735762419272568, 20560637875127661376774632, 88
11701946483283447189128, 3599145865465003098147672, 1399667836569723427057428, 517685364210719623
706172, 181889452290252840761628, 60629817430084280253876, 19146258135816088501224, 5719012170438
571889976, 1613054714739084379224, 428786696323047746376, 107196674080761936594, 2514489885845033
0806, 5519611944537877494, 1130522928399324306, 215337700647490344, 38000770702498296, 6186171974
825304, 924370524973896, 126050526132804, 15579278510796, 1731030945644, 171200862756, 1488703154
4, 1120529256, 71523144, 3764376, 156849, 4851, 99, 1]
```

In [74]:

```
#  4. Add or double
def Least_moves(a):
    if a%2 ==0:
        return Least_moves(a/2)+1
    elif a==1:
        return 0
    else:
        return Least_moves(a-1)+1
Least_moves(5)
print(Least_moves(5))
```

```
3
```

In [75]:

```
#  5. Dynamic programming
# 5.1
import numpy as np
N1 = np.random.randint(0,100)
print(N1)
def Find_expression(num):
    dig = "123456789"
    opration = ['+', '-', '']
    def All_expression(dig):
        if len(dig) == 1:
            return [dig]
        else:
            return [dig[0] + j + i for i in All_expression(dig[1:]) for j in opration]
    return [i for i in All_expression(dig) if eval(i) == N1]
print(Find_expression(N1))

#  以下结果说明当输入值等于67时，有以下15个式子可以通过加减运算得到67
```

```
67
['12+34+5+6-7+8+9', '1+2+3+45+6-7+8+9', '12-34+5+67+8+9', '1+2-3+4-5+67-8+9', '1+2-34+5+6+78+9',
'12-3-4+56+7+8-9', '1+23+45+6-7+8-9', '12+3+4+56-7+8-9', '1+2-3-4+5+67+8-9', '1-2+3+4-5+67+8-9',
'1-23+4-5-6+7+89', '1-23-4+5+6-7+89', '1+23-45+6-7+89', '1+2-3-4-5-6-7+89', '1+2-3+45-67+89']
```

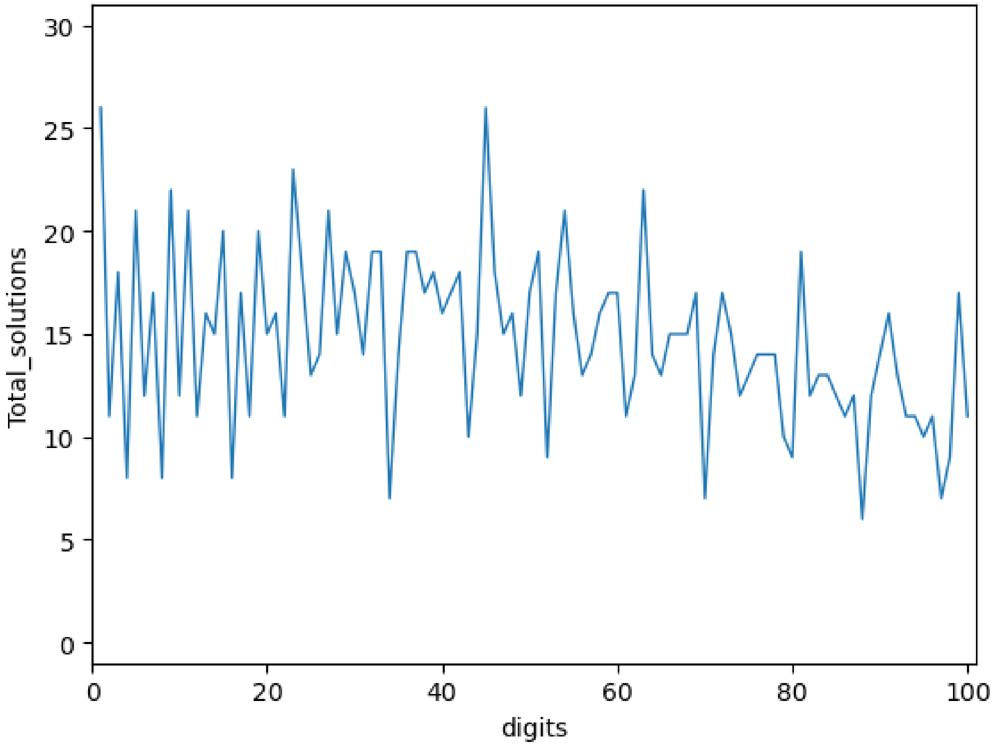
```
In  [76]: #  5.2
Total_solutions=[]
for N2 in range(1,101):
    def Find_expression(num):
        dig = "123456789"
        opration = ['+', '-', '']
        def All_expression(dig):
            if len(dig) == 1:
                return [dig]
            else:
                return [dig[0] + j + i for i in All_expression(dig[1:]) for j in opration]
        return [i for i in All_expression(dig) if eval(i) == N2]
    Total_solutions.append(len(Find_expression(N2)))
print(Total_solutions)
#  以上代码基本与5.1相同，仅将输入随机数改为：依次输入1到100。并添加了一步：计算list中的元素个数，即

# 使用matplotlib模块画坐标图时，我参考了网址：https://blog.csdn.net/HHG20171226/article/details/1012
# 横坐标为输入的数字，纵坐标为数字对应的解决方式的总数
import matplotlib.pyplot as plt
import numpy as np

x_axis_data = list(range(1,101))
y_axis_data = Total_solutions

plt.plot(x_axis_data, y_axis_data, '-', alpha=1, linewidth=1)
plt.xlabel('digits')
plt.ylabel('Total_solutions')
plt.ylim(-1,31)
plt.xlim(0,101)
plt.show()
#  以下集合是1-100按顺序所对应的Total_solutions，为了更好看清图片，横纵坐标的范围，左右均扩大了1个单
#  可以看出Total_solutions（max）=26 对应的数字是1和45；  Total_solutions（min）=6 对应的数字是88
```

```
[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, 1
7, 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, 1
1]
```


Total_solutions
digits