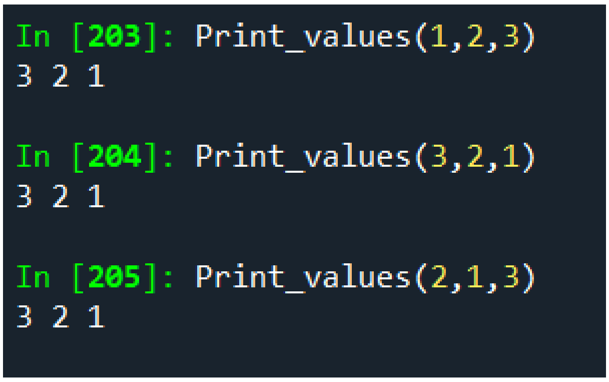
PS1_1:

CODE:

```
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:
            '''
            if a > c:
                print(b,a,c)
            else:
                print(b,c,a)
                '''
        else:
            print(c,b,a)
    return
```

RESULTS:



```
In [203]: Print_values(1,2,3)
3 2 1

In [204]: Print_values(3,2,1)
3 2 1

In [205]: Print_values(2,1,3)
3 2 1
```

PS1_2:

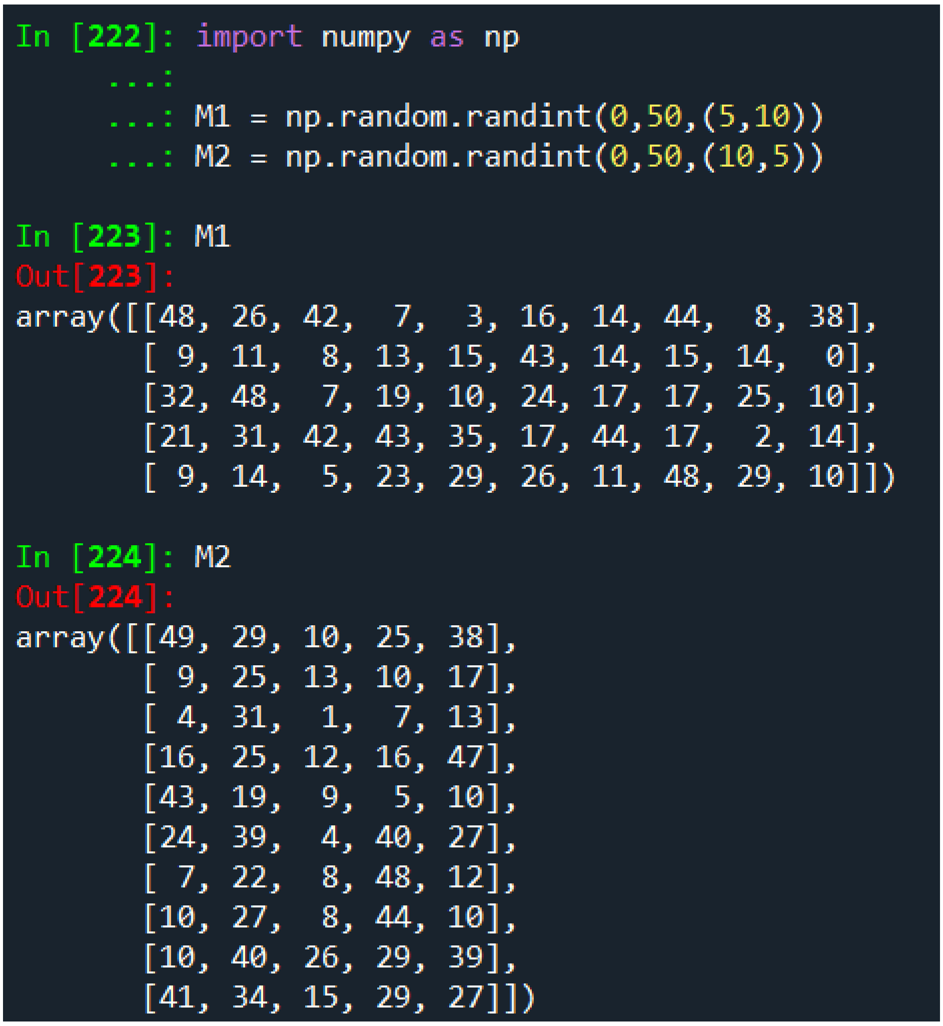
2.1

CODE:

```
import numpy as np

M1 = np.random.randint(0,50,(5,10))
M2 = np.random.randint(0,50,(10,5))
```

RESULT:



```
In [222]: import numpy as np
     ...:
     ...: M1 = np.random.randint(0,50,(5,10))
     ...: M2 = np.random.randint(0,50,(10,5))

In [223]: M1
Out[223]:
array([[48, 26, 42,  7,  3, 16, 14, 44,  8, 38],
       [ 9, 11,  8, 13, 15, 43, 14, 15, 14,  0],
       [32, 48,  7, 19, 10, 24, 17, 17, 25, 10],
       [21, 31, 42, 43, 35, 17, 44, 17,  2, 14],
       [ 9, 14,  5, 23, 29, 26, 11, 48, 29, 10]])

In [224]: M2
Out[224]:
array([[49, 29, 10, 25, 38],
       [ 9, 25, 13, 10, 17],
       [ 4, 31,  1,  7, 13],
       [16, 25, 12, 16, 47],
       [43, 19,  9,  5, 10],
       [24, 39,  4, 40, 27],
       [ 7, 22,  8, 48, 12],
       [10, 27,  8, 44, 10],
       [10, 40, 26, 29, 39],
       [41, 34, 15, 29, 27]])
```

2.2

CODE:

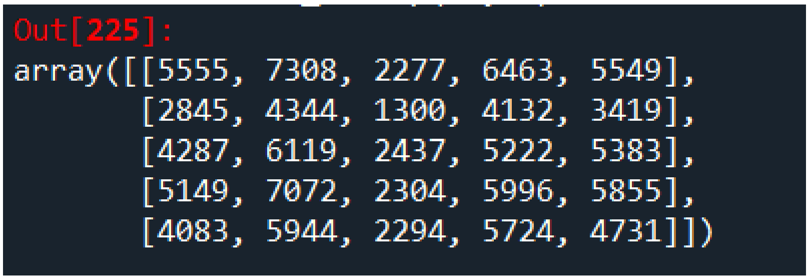
```
import numpy as np

M1 = np.random.randint(0,50,(5,10))
M2 = np.random.randint(0,50,(10,5))
```

```
def Matrix_multip(A,B):
    m=np.zeros([5,5],dtype=int)
    for i in range(5):
        for j in range(5):
            for k in range(10):
                m[i,j]+= M1[i,k] * M2[k,j]
    return m

Matrix_multip(M1,M2)
```

RESULTS:



```
Out[225]:
array([[5555, 7308, 2277, 6463, 5549],
       [2845, 4344, 1300, 4132, 3419],
       [4287, 6119, 2437, 5222, 5383],
       [5149, 7072, 2304, 5996, 5855],
       [4083, 5944, 2294, 5724, 4731]])
```

PS1_3:

CODE:

```
def Pascal_triangle(k):
    x = []
    for i in range(k):
        if i == 0:
            x.append([1])
        elif i == 1:
            x.append([1,1])
        else:
            y = []
            for j in range(i+1):
                if j == 0 or j == i:
                    y.append(1)
                else:
                    y.append(x[i-1][j-1]+x[i-1][j])
            x.append(y)
    return x[k-1]
```

RESULT:

```
In [211]:Pascal_triangle(100)
Out[211]:
[1,
 99,
 4851,
 156849,
 3764376,
 71523144,
 1120529256,
 14887031544,
 171200862756,
 1731030945644,
 15579278510796,
 126050526132804,
 924370524973896,
 6186171974825304,
 38000770702498296,
 215337700647490344,
 1130522928399324306,
 5519611944537877494,
 25144898858450330806,
 107196674080761936594,
 428786696323047746376,
 1613054714739084379224,
 5719012170438571889976,
 19146258135816088501224,
 60629817430084280253876,
 181889452290252840761628,
 517685364210719623706172,
 1399667836569723427057428,
 3599145865465003098147672,
 8811701946483283447189128,
 20560637875127661376774632,
 45764000431735762419272568,
 97248500917438495140954207,
 197443926105102399225573693,
 383273503615787010261407757,
 711793649572175876199757263,
 1265410932572757113244012912,
 2154618614921181030658724688,
 3515430371713505892127392912,
 5498493658321124600506947888,
 8247740487481686900760421832,
 11868699725888281149874753368,
 16390109145274293016493707032,
 21726423750712434928840495368,
 27651812046361280818524266832,
 33796659167774898778196326128,
 39674339023040098565708730672,
 44739148260023940935799206928,
 48467410615025936013782474172,
 50445672272782096667406248628,
 50445672272782096667406248628,
 48467410615025936013782474172,
 44739148260023940935799206928,
 39674339023040098565708730672,
 33796659167774898778196326128,
 27651812046361280818524266832,
 21726423750712434928840495368,
 16390109145274293016493707032,
 11868699725888281149874753368,
 8247740487481686900760421832,
 5498493658321124600506947888,
 3515430371713505892127392912,
 2154618614921181030658724688,
 1265410932572757113244012912,
 711793649572175876199757263,
```

Removing all variables...

```
def Pascal_triangle(k):
    x = []
    for i in range(k):
        if i == 0:
            x.append([1])
        elif i == 1:
            x.append([1,1])
        else:
            y = []
            for j in range(i+1):
                if j == 0 or j == i:
                    y.append(1)
                else:
                    y.append(x[i-1][j-1]+x[i-1][j])
            x.append(y)
    return x[k-1]
```

```
Pascal_triangle(100)
Out[211]:
[1,
 99,
 4851,
 156849,
 3764376,
 71523144,
 1120529256,
 14887031544,
 171200862756,
 1731030945644,
 15579278510796,
 126050526132804,
 924370524973896,
 6186171974825304,
 38000770702498296,
 215337700647490344,
 1130522928399324306,
 5519611944537877494,
 25144898858450330806,
 107196674080761936594,
 428786696323047746376,
 1613054714739084379224,
 5719012170438571889976,
 19146258135816088501224,
 60629817430084280253876,
 181889452290252840761628,
 517685364210719623706172,
```

1399667836569723427057428,
3599145865465003098147672,
8811701946483283447189128,
20560637875127661376774632,
45764000431735762419272568,
97248500917438495140954207,
197443926105102399225573693,
383273503615787010261407757,
711793649572175876199757263,
1265410932572757113244012912,
2154618614921181030658724688,
3515430371713505892127392912,
5498493658321124600506947888,
8247740487481686900760421832,
11868699725888281149874753368,
16390109145274293016493707032,
21726423750712434928840495368,
27651812046361280818524266832,
33796659167774898778196326128,
39674339023040098565708730672,
44739148260023940935799206928,
48467410615025936013782474172,
50445672272782096667406248628,
50445672272782096667406248628,
48467410615025936013782474172,
44739148260023940935799206928,
39674339023040098565708730672,
33796659167774898778196326128,
27651812046361280818524266832,
21726423750712434928840495368,
16390109145274293016493707032,
11868699725888281149874753368,
8247740487481686900760421832,
5498493658321124600506947888,
3515430371713505892127392912,
2154618614921181030658724688,
1265410932572757113244012912,
711793649572175876199757263,
383273503615787010261407757,
197443926105102399225573693,
97248500917438495140954207,
45764000431735762419272568,
20560637875127661376774632,
8811701946483283447189128,

```
 3599145865465003098147672,
 1399667836569723427057428,
 517685364210719623706172,
 181889452290252840761628,
 60629817430084280253876,
 19146258135816088501224,
 5719012170438571889976,
 1613054714739084379224,
 428786696323047746376,
 107196674080761936594,
 25144898858450330806,
 5519611944537877494,
 1130522928399324306,
 215337700647490344,
 38000770702498296,
 6186171974825304,
 924370524973896,
 126050526132804,
 15579278510796,
 1731030945644,
 171200862756,
 14887031544,
 1120529256,
 71523144,
 3764376,
 156849,
 4851,
 99,
 1]
In [211]:Pascal_triangle(200)
Out[213]:
[1,
 199,
 19701,
 1293699,
 63391251,
 2472258789,
 79936367511,
 2203959847089,
 52895036330136,
 1122550215450664,
 21328454093562616,
 366461620334848584,
 5741232051912627816,
```

825854149005893385 84,

```
360992022688017097651709953615480436754356081770344,
883808055546524618388669196782727965846871786403256,
2112151454780677477844107741463807511600151218353544,
4928353394488247448302918063415550860400352842824936,
11230182325145350742854190341225599501568017133650264,
24996212272097716169578681727244076309941715555544136,
54356842559958525638607609470356165943841508430310264,
115508290439911866982041170124506852630663205414409311,
239901833990586185270393199489360386232915888168388569,
487073420526341648882313465629913511442586803250970731,
966877088507514019423099864608634283908418579587747869,
1876879054161644861233076207769701845233989007435039981,
3563350088335876475674391061127984662690616811217249819,
6617650164052342026252440542094828659282574077974892521,
12023617903700734104036124365214547845738761352940297679,
21375320717690193962730887760381418392424464627449418096,
37187201796529515524203051309156714189560369968302412304,
63318749004901607514183573850726297133575765081163566896,
105531248341502679190305956417877161889292941801939278160,
172182563083504371310499192050220632556214799782111453840,
275044873497026463262225982106196594862524939911684530160,
430198391879964468179379100217384417605487726528532213840,
658911460980705071515251533244348285193215378606992378160,
988367191471057607272877299866522427789823067910488567240,
1452045626975998213153980230668100850703567223226520240760,
2089529072965460843319142283156535370524645516350358395240,
2945480741409143598413730688304995642787753318228818460760,
4067568642898341159714199521944993982897373629935035017240,
5503181105097755686672152294396168329802329028735635611560,
7294914488152838933495643739083292902296110572975144880440,
9475003875416905741206985546165656298384603387887257143560,
12059095841439698216081617967847198925216767948220145455440,
15039995937076477550393928027315045850551249912948720736560,
18382217256426805894925912033385056039562638782492880900240,
22018260230225514753262905622406275915520083816392571627760,
25847522878960386884265150078476932596480098393156497128240,
29738547828481305339960979122548728901326564817932744007760,
33534958189564025170594295606278353867453360326605009200240,
37064953788465501504341063564833970064027398255721325958160,
40153699937504293296369485528570134236029681443698103121340,
42637433954257136180681000097347668312485125656710356922660,
44377737380961509086014918468667981304831457316167922511340,
45274257328051640582702088538742081937252294837706668420660,
45274257328051640582702088538742081937252294837706668420660,
44377737380961509086014918468667981304831457316167922511340,
42637433954257136180681000097347668312485125656710356922660,
40153699937504293296369485528570134236029681443698103121340,
37064953788465501504341063564833970064027398255721325958160,
33534958189564025170594295606278353867453360326605009200240,
29738547828481305339960979122548728901326564817932744007760,
25847522878960386884265150078476932596480098393156497128240,
22018260230225514753262905622406275915520083816392571627760,
18382217256426805894925912033385056039562638782492880900240,
15039995937076477550393928027315045850551249912948720736560,
12059095841439698216081617967847198925216767948220145455440,
9475003875416905741206985546165656298384603387887257143560,
7294914488152838933495643739083292902296110572975144880440,
5503181105097755686672152294396168329802329028735635611560,
4067568642898341159714199521944993982897373629935035017240,
2945480741409143598413730688304995642787753318228818460760,
2089529072965460843319142283156535370524645516350358395240,
1452045626975998213153980230668100850703567223226520240760,
988367191471057607272877299866522427789823067910488567240,
658911460980705071515251533244348285193215378606992378160,
430198391879964468179379100217384417605487726528532213840,
275044873497026463262225982106196594862524939911684530160,
172182563083504371310499192050220632556214799782111453840,
105531248341502679190305956417877161889292941801939278160,
63318749004901607514183573850726297133575765081163566896,
37187201796529515524203051309156714189560369968302412304,
21375320717690193962730887760381418392424464627449418096,
12023617903700734104036124365214547845738761352940297679,
6617650164052342026252440542094828659282574077974892521,
3563350088335876475674391061127984662690616811217249819,
1876879054161644861233076207769701845233989007435039981,
966877088507514019423099864608634283908418579587747869,
487073420526341648882313465629913511442586803250970731,
239901833990586185270393199489360386232915888168388569,
115508290439911866982041170124506852630663205414409311,
54356842559958525638607609470356165943841508430310264,
24996212272097716169578681727244076309941715555544136,
11230182325145350742854190341225599501568017133650264,
4928353394488247448302918063415550860400352842824936,
2112151454780677477844107741463807511600151218353544,
883808055546524618388669196782727965846871786403256,
360992022688017097651709953615480436754356081770344,
143891925127391430532499771720855838426561515111256,
55957970882874445207083244558110603832551700321044,
21225437231435134388893644487559194557174782880396,
7850504181489707239727786317316414425256426544804,
21328454093562616,
1122550215450664,
52895036330136,
2203959847089,
79936367511,
2472258789,
63391251,
1293699,
19701,
199,
1]
```

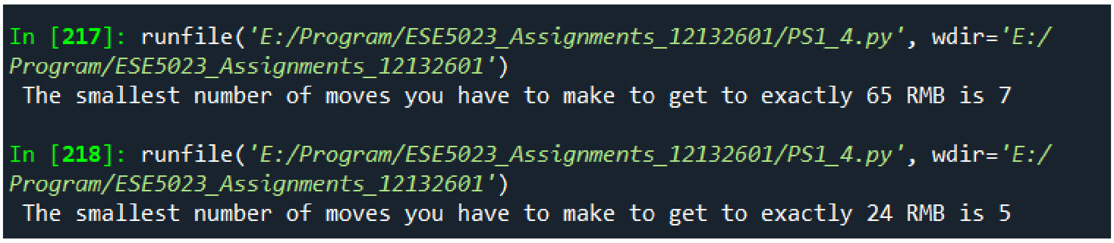
PS1_4:

CODE:

```
import numpy as np

y = np.random.randint(1,101)
def Least_moves(x):
    m = x
    n = 0
    while x > 1:
        if x % 2:
            x = (x-1) / 2
            n += 2
        else:
            x /= 2
            n += 1
    print(" The smallest number of moves you have to make to get to exactly",m, "RMB is", n)
    return
Least_moves(y)
```

RESULTS:



```
In [217]: runfile('E:/Program/ESE5023_Assignments_12132601/PS1_4.py', wdir='E:/Program/ESE5023_Assignments_12132601')
 The smallest number of moves you have to make to get to exactly 65 RMB is 7

In [218]: runfile('E:/Program/ESE5023_Assignments_12132601/PS1_4.py', wdir='E:/Program/ESE5023_Assignments_12132601')
 The smallest number of moves you have to make to get to exactly 24 RMB is 5
```

PS1_5:

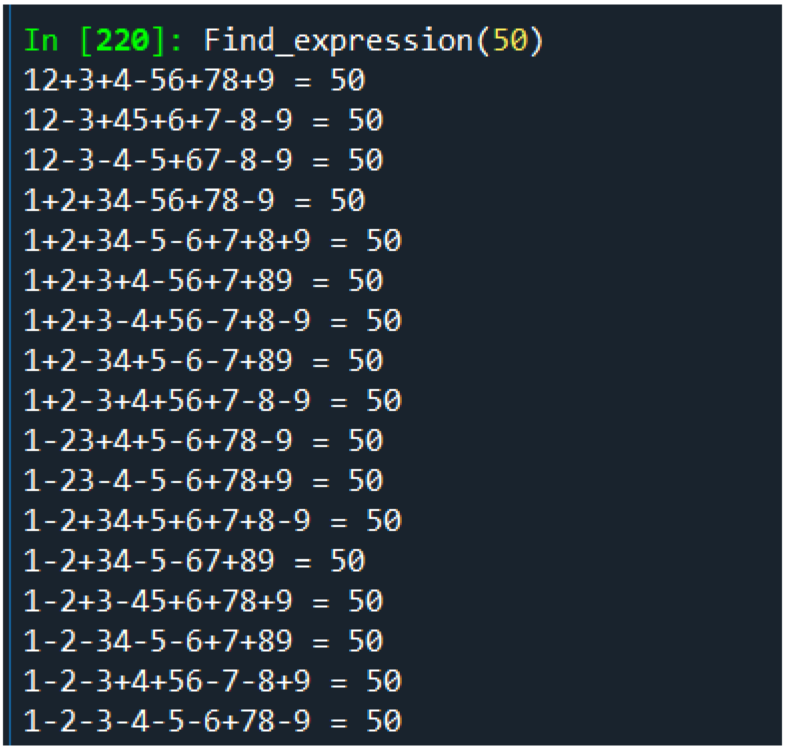
5.1

COODE:

```
import matplotlib.pyplot as plt

def Find_expression(x):
    a = ['','+','-']
    m = 0
    for i0 in range(3):
        for i1 in range(3):
            for i2 in range(3):
                for i3 in range(3):
                    for i4 in range(3):
                        for i5 in range(3):
                            for i6 in range(3):
                                for i7 in range(3):
                                    n                                       =
'1'+a[i0]+'2'+a[i1]+'3'+a[i2]+'4'+a[i3]+'5'+a[i4]+'6'+a[i5]+'7'+a[i6]+'8'+a[i7]+'9'
                                    #m.append(eval(n))
                                    if eval(n) == x:
                                        print(n, '=', x)
                                        m+=1
    return m
```

RESULT:



```
In [220]: 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
```

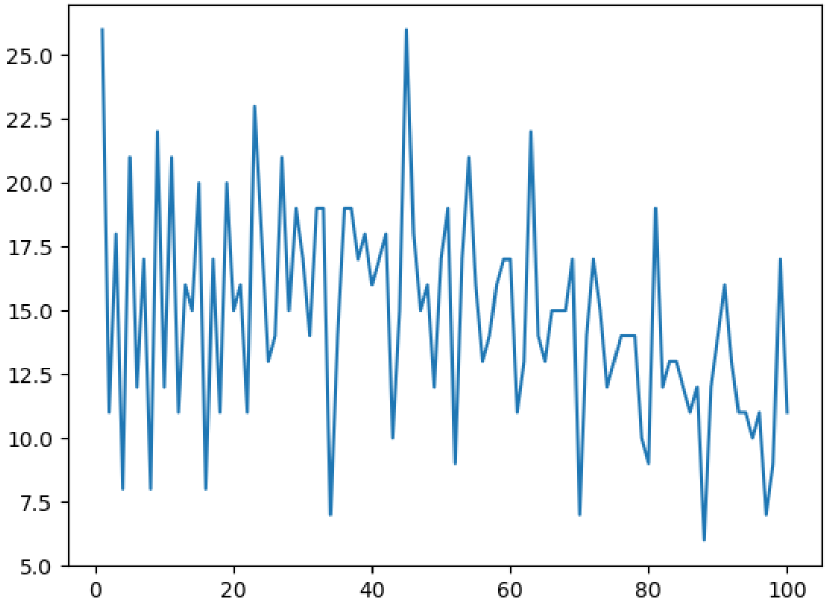
5.2

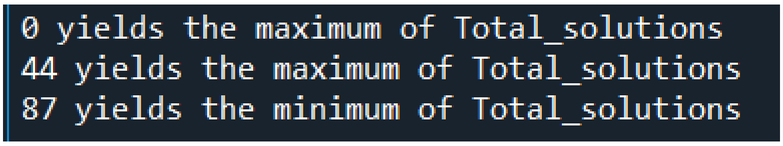
CODE:

```
Total_solutions = [0]*100
for i in range(1,101):
        Total_solutions[i-1]=Find_expression(i)
num = range(1,101)
plt.plot(num, Total_solutions)
plt.show()

Total_solutions_max = max(Total_solutions)
Total_solutions_min = min(Total_solutions)
for i in range(100):
        if Total_solutions[i] == Total_solutions_max:
                print(i,'yields the maximum of Total_solutions')
        elif Total_solutions[i] == Total_solutions_min:
                print(i,'yields the minimum of Total_solutions')
```

RESULT:





```
0 yields the maximum of Total_solutions
44 yields the maximum of Total_solutions
87 yields the minimum of Total_solutions
```