PS1_1

第一题就是将三个随机产生的数进行大小比较。

```
import random
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 a < b:
        if b < c:
            print(c,b,a)
        elif b > c:
            if a > c:
                print(a,c,b)
            else: print (c,a,b)
        elif b < c:
            print(c,b,a)

a = random.randint(0,50)
b = random.randint(0,50)
c = random.randint(0,50)
print("a:",a,"b:",b,"c:",c)
Print_values(a, b, c)
```

```
In [1]: runfile('C:/Users/my/.spyder-py3/temp.py', wdir='C:/Users/my/.spyder-py3')
a: 2 b: 7 c: 27
27 7 2

In [2]: runfile('C:/Users/my/.spyder-py3/temp.py', wdir='C:/Users/my/.spyder-py3')
a: 4 b: 42 c: 29
29 4 42

In [3]: runfile('C:/Users/my/.spyder-py3/temp.py', wdir='C:/Users/my/.spyder-py3')
a: 39 b: 5 c: 29
39 29 5

In [4]: runfile('C:/Users/my/.spyder-py3/temp.py', wdir='C:/Users/my/.spyder-py3')
a: 29 b: 8 c: 32
32 29 8

In [5]: runfile('C:/Users/my/.spyder-py3/temp.py', wdir='C:/Users/my/.spyder-py3')
a: 18 b: 44 c: 45
45 44 18

In [6]: runfile('C:/Users/my/.spyder-py3/temp.py', wdir='C:/Users/my/.spyder-py3')
a: 30 b: 38 c: 39
39 38 30
```

PS1_2

第二题的思路是将 M1 中的每一行的元素对应乘以 M2 矩阵中每一列的元素，然后将结果相加作为新矩阵的结果，所以采用了如下的方法。

```
import numpy as np
D = []
def Matrix_multip(M1,M2):
    for i in range(len(M1)):      #矩阵乘法是对应行乘以对应列然后相加
        temp = []
        for k in range(len(M1)):
            result = 0
            for j in range(len(M2)):
                result += ((M1[i][j]))*((M2[j][k]))    #相加的结果为产生新矩阵的每一个元素
            temp.append(result)
        D.append(temp)
    print(D)
M1 = np.random.randint(1,50,[5,10])
M2 = np.random.randint(1,50,[10,5])
print(M1,'\n',M2)
Matrix_multip(M1, M2)
```

PS1_3

第三题是杨辉三角，杨辉三角每一行的第一个和最后一个都是 1，中间的数由上一行相邻两个数相加得到，所以需要先将前面的特殊的 1 构建出来，然后开始向下累加，并将得到的结果带入到新的行中。

```
n=int(input('num:'))
def Pascal_triangle(n):
    triangle = [[1]]    #列出最开始的两特殊项
    print(1,end='\t')
    print()
    for i in range(1,n):    #计数，产生大列表的新元素
        newline = [1]    #新行的首项
        a = triangle[i-1]    #将杨辉三角列表中的当前最后一个元素抽出，为计算下一元素做准备
        for j in range(i-1):    #从 0 开始计数
            b = a[j]+a[j+1]    #将抽出的元素从当前项开始累加
            newline.append(b)    #累加的值循环堆入下一行
        newline.append(1)    #最后堆入最后一项 1
        triangle.append(newline)    #将新生成的元素再堆入杨辉三角列表
```

```
            for item in newline:
                print(item,end='\t')
            print()
    # print(triangle)

Pascal_triangle(n)   #  作业中的 test n=50,100 的情况只需要在 input 的时候输入就可以
```

这个是 number50 时候的 test

```
num:50
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
1    10   45   120  210  252  210  120  45   10   1
1    11   55   165  330  462  462  330  165  55   11   1
1    12   66   220  495  792  924  792  495  220  66   12   1
1    13   78   286  715  1287      1716      1716      1287      715  286  78   13   1
1    14   91   364  1001      2002      3003      3432      3003      2002      1001      364  91
     14   1
1    15   105  455  1365      3003      5005      6435      6435      5005      3003      1365
     455  105  15   1
1    16   120  560  1820      4368      8008      11440     12870     11440     8008      4368
     1820      560  120  16   1
1    17   136  680  2380      6188      12376     19448     24310     24310     19448     12376
     6188      2380      680  136  17   1
1    18   153  816  3060      8568      18564     31824     43758     48620     43758     31824
     18564     8568      3060      816  153  18   1
1    19   171  969  3876      11628     27132     50388     75582     92378     92378     75582
     50388     27132     11628     3876      969  171  19   1
1    20   190  1140      4845      15504     38760     77520     125970    167960    184756
     167960    125970    77520     38760     15504     4845      1140      190  20   1
1    21   210  1330      5985      20349     54264     116280    203490    293930    352716
     352716    293930    203490    116280    54264     20349     5985      1330      210  21   1

1    22   231  1540      7315      26334     74613     170544    319770    497420    646646
     705432    646646    497420    319770    170544    74613     26334     7315      1540
     231  22   1
1    23   253  1771      8855      33649     100947    245157    490314    817190    1144066
```

1352078 1352078 1144066 817190 490314 245157 100947 33649 8855
1771 253 23 1
1 24 276 2024 10626 42504 134596 346104 735471 1307504 1961256
2496144 2704156 2496144 1961256 1307504 735471 346104 134596 42504
10626 2024 276 24 1
1 25 300 2300 12650 53130 177100 480700 1081575 2042975 3268760
4457400 5200300 5200300 4457400 3268760 2042975 1081575 480700 177100
53130 12650 2300 300 25 1
1 26 325 2600 14950 65780 230230 657800 1562275 3124550 5311735
7726160 9657700 10400600 9657700 7726160 5311735 3124550 1562275 657800
230230 65780 14950 2600 325 26 1
1 27 351 2925 17550 80730 296010 888030 2220075 4686825 8436285
13037895 17383860 20058300 20058300 17383860 13037895
8436285 4686825 2220075 888030 296010 80730 17550 2925 351 27 1

1 28 378 3276 20475 98280 376740 1184040 3108105 6906900 13123110
21474180 30421755 37442160 40116600 37442160 30421755
21474180 13123110 6906900 3108105 1184040 376740 98280 20475
3276 378 28 1
1 29 406 3654 23751 118755 475020 1560780 4292145 10015005
20030010 34597290 51895935 67863915 77558760 77558760
67863915 51895935 34597290 20030010 10015005 4292145
1560780 475020 118755 23751 3654 406 29 1
1 30 435 4060 27405 142506 593775 2035800 5852925 14307150
30045015 54627300 86493225 119759850 145422675 155117520
145422675 119759850 86493225 54627300 30045015 14307150
5852925 2035800 593775 142506 27405 4060 435 30 1
1 31 465 4495 31465 169911 736281 2629575 7888725 20160075
44352165 84672315 141120525 206253075 265182525 300540195
300540195 265182525 206253075 141120525 84672315 44352165
20160075 7888725 2629575 736281 169911 31465 4495 465 31 1
1 32 496 4960 35960 201376 906192 3365856 10518300 28048800
64512240 129024480 225792840 347373600 471435600 565722720
601080390 565722720 471435600 347373600 225792840 129024480
64512240 28048800 10518300 3365856 906192 201376 35960 4960
496 32 1
1 33 528 5456 40920 237336 1107568 4272048 13884156 38567100
92561040 193536720 354817320 573166440 818809200 1037158320
1166803110 1166803110 1037158320 818809200 573166440 354817320
193536720 92561040 38567100 13884156 4272048 1107568 237336
40920 5456 528 33 1
1 34 561 5984 46376 278256 1344904 5379616 18156204 52451256
131128140 286097760 548354040 927983760 1391975640 1855967520
2203961430 2333606220 2203961430 1855967520 1391975640 927983760

548354040 286097760 131128140 52451256 18156204 5379616
1344904 278256 46376 5984 561 34 1
1 35 595 6545 52360 324632 1623160 6724520 23535820 70607460
183579396 417225900 834451800 1476337800 2319959400 3247943160
4059928950 4537567650 4537567650 4059928950 3247943160 2319959400
1476337800 834451800 417225900 183579396 70607460 23535820
6724520 1623160 324632 52360 6545 595 35 1
1 36 630 7140 58905 376992 1947792 8347680 30260340 94143280
254186856 600805296 1251677700 2310789600 3796297200 5567902560
7307872110 8597496600 9075135300 8597496600 7307872110 5567902560
3796297200 2310789600 1251677700 600805296 254186856 94143280
30260340 8347680 1947792 376992 58905 7140 630 36 1
1 37 666 7770 66045 435897 2324784 10295472 38608020 124403620
348330136 854992152 1852482996 3562467300 6107086800 9364199760
12875774670 15905368710 17672631900 17672631900 15905368710 12875774670
9364199760 6107086800 3562467300 1852482996 854992152 348330136
124403620 38608020 10295472 2324784 435897 66045 7770 666 37
1
1 38 703 8436 73815 501942 2760681 12620256 48903492 163011640
472733756 1203322288 2707475148 5414950296 9669554100 15471286560
22239974430 28781143380 33578000610 35345263800 33578000610 28781143380
22239974430 15471286560 9669554100 5414950296 2707475148 1203322288
472733756 163011640 48903492 12620256 2760681 501942 73815
8436 703 38 1
1 39 741 9139 82251 575757 3262623 15380937 61523748 211915132
635745396 1676056044 3910797436 8122425444 15084504396 25140840660
37711260990 51021117810 62359143990 68923264410 68923264410 62359143990
51021117810 37711260990 25140840660 15084504396 8122425444 3910797436
1676056044 635745396 211915132 61523748 15380937 3262623 575757
82251 9139 741 39 1
1 40 780 9880 91390 658008 3838380 18643560 76904685 273438880
847660528 2311801440 5586853480 12033222880 23206929840 40225345056
62852101650 88732378800 113380261800 131282408400 137846528820
131282408400 113380261800 88732378800 62852101650 40225345056
23206929840 12033222880 5586853480 2311801440 847660528 273438880
76904685 18643560 3838380 658008 91390 9880 780 40 1
1 41 820 10660 101270 749398 4496388 22481940 95548245 350343565
1121099408 3159461968 7898654920 17620076360 35240152720 63432274896
103077446706 151584480450 202112640600 244662670200
269128937220 269128937220 244662670200 202112640600
151584480450 103077446706 63432274896 35240152720 17620076360
7898654920 3159461968 1121099408 350343565 95548245 22481940
4496388 749398 101270 10660 820 41 1
1 42 861 11480 111930 850668 5245786 26978328 118030185 445891810

1471442973 4280561376 11058116888 25518731280 52860229080 98672427616
166509721602 254661927156 353697121050 446775310800
513791607420 538257874440 513791607420 446775310800
353697121050 254661927156 166509721602 98672427616 52860229080
25518731280 11058116888 4280561376 1471442973 445891810 118030185
26978328 5245786 850668 111930 11480 861 42 1
1 43 903 12341 123410 962598 6096454 32224114 145008513 563921995
1917334783 5752004349 15338678264 36576848168 78378960360 151532656696
265182149218 421171648758 608359048206 800472431850
960566918220 1052049481860 1052049481860 960566918220
800472431850 608359048206 421171648758 265182149218
151532656696 78378960360 36576848168 15338678264 5752004349 1917334783
563921995 145008513 32224114 6096454 962598 123410 12341 903 43
1
1 44 946 13244 135751 1086008 7059052 38320568 177232627 708930508
2481256778 7669339132 21090682613 51915526432 114955808528
229911617056 416714805914 686353797976 1029530696964
1408831480056 1761039350070 2012616400080 2104098963720
2012616400080 1761039350070 1408831480056 1029530696964
686353797976 416714805914 229911617056 114955808528
51915526432 21090682613 7669339132 2481256778 708930508 177232627
38320568 7059052 1086008 135751 13244 946 44 1
1 45 990 14190 148995 1221759 8145060 45379620 215553195 886163135
3190187286 10150595910 28760021745 73006209045 166871334960
344867425584 646626422970 1103068603890 1715884494940
2438362177020 3169870830126 3773655750150 4116715363800
4116715363800 3773655750150 3169870830126 2438362177020
1715884494940 1103068603890 646626422970 344867425584
166871334960 73006209045 28760021745 10150595910 3190187286 886163135
215553195 45379620 8145060 1221759 148995 14190 990 45 1
1 46 1035 15180 163185 1370754 9366819 53524680 260932815
1101716330 4076350421 13340783196 38910617655 101766230790
239877544005 511738760544 991493848554 1749695026860
2818953098830 4154246671960 5608233007146 6943526580276
7890371113950 8233430727600 7890371113950 6943526580276
5608233007146 4154246671960 2818953098830 1749695026860
991493848554 511738760544 239877544005 101766230790
38910617655 13340783196 4076350421 1101716330 260932815 53524680
9366819 1370754 163185 15180 1035 46 1
1 47 1081 16215 178365 1533939 10737573 62891499 314457495
1362649145 5178066751 17417133617 52251400851 140676848445
341643774795 751616304549 1503232609098 2741188875414
4568648125690 6973199770790 9762479679106 12551759587422
14833897694226 16123801841550 16123801841550 14833897694226

12551759587422 9762479679106 6973199770790 4568648125690
2741188875414 1503232609098 751616304549 341643774795
140676848445 52251400851 17417133617 5178066751 1362649145 314457495
62891499 10737573 1533939 178365 16215 1081 47 1
1 48 1128 17296 194580 1712304 12271512 73629072 377348994
1677106640 6540715896 22595200368 69668534468 192928249296
482320623240 1093260079344 2254848913647 4244421484512
7309837001104 11541847896480 16735679449896 22314239266528
27385657281648 30957699535776 32247603683100 30957699535776
27385657281648 22314239266528 16735679449896 11541847896480
7309837001104 4244421484512 2254848913647 1093260079344
482320623240 192928249296 69668534468 22595200368 6540715896
1677106640 377348994 73629072 12271512 1712304 194580 17296
1128 48 1
1 49 1176 18424 211876 1906884 13983816 85900584 450978066
2054455634 8217822536 29135916264 92263734836 262596783764
675248872536 1575580702584 3348108992991 6499270398159
11554258485616 18851684897584 28277527346376 39049918716424
49699896548176 58343356817424 63205303218876 63205303218876
58343356817424 49699896548176 39049918716424 28277527346376
18851684897584 11554258485616 6499270398159 3348108992991
1575580702584 675248872536 262596783764 92263734836 29135916264
8217822536 2054455634 450978066 85900584 13983816 1906884 211876
18424 1176 49 1

PS1_4

这个题的思路是判断随机数的奇偶性质，如果是偶数，就可以除以 2，如果是奇数，就减去 1 当成新的偶数除以 2，知道最后等于 1，并把每一步操作累加，得到最后结果。

```
import random
def Least_moves(n = 0,x = random.randint(1,100)):
    while x!=1:
        if x%2 ==0:
            x = x/2
            n += 1
        else:
            x = x-1
            n +=1
    print("least moves",n)

n = 0
x = random.randint(1,100)
print("initial number",x)
Least_moves(n,x)
```

PS1_5

这个题的思路是将 123456789 中加入 8 个空格，每一种空格可以有三种操作，分别是+，-，“ ”，这三种操作一共有 3^8 种可能性，但是其中大部分都是不合适的，需要将操作结果等于随机产生的数，所以需要八个 for 循环，但是在做题的时候，我在网上查询到一种递归的思路，就是说在我们的函数内部调用本函数进行操作，没进行一次就调用本函数，相当于一直在重复，这样就可以省去八个 for 循环。

```
import random
import matplotlib.pyplot as plt
def Find_expression(res, number, aa: str):

    if len(number) == 1:
        for i in range(len(number)):
            a = number[i]
            b = number[:]
            b.pop(i)                # 借鉴了 python 的 pop 函数，
https://www.runoob.com/python/att-list-pop.html
            operate = str(a)
```

```
                aa = aa + operate
                if eval(aa) == res:             # 字符串的计算函数 eval,https://www.runoob.com/python/python-func-eval.html
                    print(aa + "=" + str(res))
                    Total_solutions.append(aa)
                    y = len(Total_solutions)
                    z.append(y)
                    print(y)
                    plt.scatter(x, y)      #计算出每个数对应的结果，并画出每个的数据个数
                else:
                    return False

    for i in range(len(number)):
        a = number[i]
        b = number[:]
        b.pop(i)
        operate = str(a)
        aa = aa + operate
        Find_expression(res, b, aa + '+')     #python 递归的思想，重新调用 find_expression，https://blog.csdn.net/ruanxingzi123/article/details/82658669
        Find_expression(res, b, aa + '-')
        Find_expression(res, b, aa + '')
        return True

Total_solutions = []
z = []
number = [1, 2, 3, 4, 5, 6, 7, 8, 9]
x = random.randint(1,100)
result = Find_expression(x, number, '')
```