PS1 Report

Name:左小幸　SID:12132243

PS1_1: Flowchart

In this case, I created a Print_values function with parameters a, b, and c. In the function, I use nesting of **if** statements to implement the judgment shown in the flowchart.

Here is the code and some results:

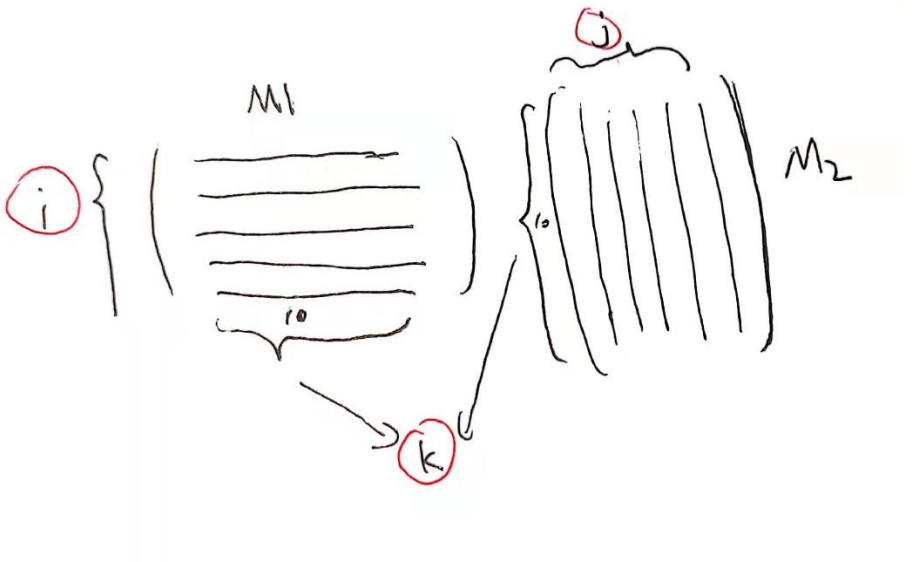
```python
# Flowchart
import random


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 b > c:
        pass
    else:
        print(c, b, a)


a = random.randint(0, 100)
b = random.randint(0, 100)
c = random.randint(0, 100)
print("a: ", a)
print("b: ", b)
print("c: ", c)
Print_values(a, b, c)
```

```
In [3]: runfile('D:/github-repo/ESE5023_Assignments_12132243/PS1_1.py', wdir='D:/github-repo/
ESE5023_Assignments_12132243')
a:  91
b:  30
c:  64
91 64 30

In [4]: runfile('D:/github-repo/ESE5023_Assignments_12132243/PS1_1.py', wdir='D:/github-repo/
ESE5023_Assignments_12132243')
a:  67
b:  18
c:  20
67 20 18

In [5]: runfile('D:/github-repo/ESE5023_Assignments_12132243/PS1_1.py', wdir='D:/github-repo/
ESE5023_Assignments_12132243')
a:  50
b:  62
c:  44

In [6]: runfile('D:/github-repo/ESE5023_Assignments_12132243/PS1_1.py', wdir='D:/github-repo/
ESE5023_Assignments_12132243')
a:  52
b:  42
c:  37
52 42 37
```

PS2: Matrix multiplication

2.1: The M1 and M2 matrices are created using **np.random.randint()** function, both are filled with random integers from 0 and 50. The first parameter in the function specifies the range of values in the matrix, and the second parameter specifies the size of the matrix.

2.2: I've created a Matrix_multip() function that uses a for loop to multiply matrices. Where **i** represents the 5 rows of matrix M1, **j** represents the 5 columns of matrix M2, and **k** represents the 10 elements in each row of M1 multiplied by the 10 corresponding elements in each column of M2.



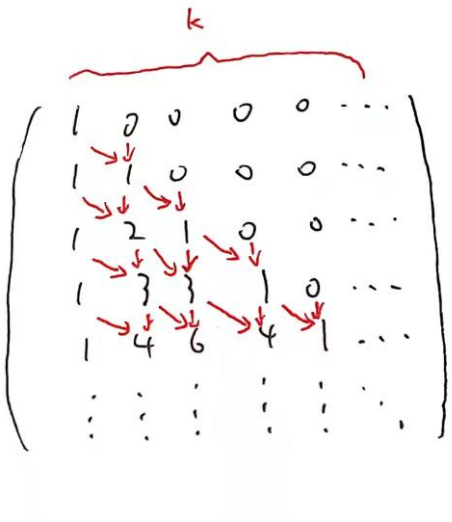
Here is the code and a result:

```python
import numpy as np

M1 = np.random.randint(51, size=(5, 10))
M2 = np.random.randint(51, size=(10, 5))
result = np.empty((5, 5), dtype = int)

print(M1)
print(M2)


def Matrix_multip(m1, m2):
    for i in range(5):
        for j in range(5):
            for k in range(10):
                result[i, j] += m1[i, k] * m2[k, j]
    print(result)


Matrix_multip(M1, M2)
```

```
In [11]: runfile('D:/github-repo/ESE5023_Assignments_12132243/PS1_2.py', wdir='D:/github-repo/
ESE5023_Assignments_12132243')
[[25  8 25 49 25 17 50 11 38 45]
 [22 16 45 28  4 43 18 15 42 30]
 [22 11 36 25 14 45 47 21 36 17]
 [23  4 39 32 24 24 42 23 45 18]
 [ 6 49 32 29 13 20 42  5 36 11]]
[[50 14 33 18 13]
 [ 8 45  4 28 23]
 [ 0 12 42 25  9]
 [47 26 37 19 23]
 [16 28  5  6 24]
 [14 50 50 18 20]
 [33 30 27  5  4]
 [20  4 16 45  6]
 [26 17 44 24  0]
 [45 14 39 12 42]]
[[4137998 6036029 3810804 5575535 6624208]
 [7543300 3741936 3352016 3347816 6361165]
 [6363991 7281466 6563089 3347654 7081682]
 [6429865 6363127 7939072 6493400 7278197]
 [7607728 7543591 6495278 3019463 7933500]]
```

PS3: Pascal triangle

In this case, I've created a function Pascal_triangle(k) that takes an argument k. In the function, a square matrix of size k will be created first, and then the values of PASCAL triangles will be generated line by line and stored in the square matrix according to the rule, as shown in the following figure.



Here is the code and a result:

```python
import numpy as np


def Pascal_triangle(k):
    m1 = np.zeros((k, k))
    m1[0, 0] = 1
    m1[1, 0] = 1
    m1[1, 1] = 1

    if k == 1:                                          #第1行
        print(int(m1[0, 0]))
    elif k == 2:                                        #第2行
        print(int(m1[1, 0]), int(m1[1, 1]))
    else:
        for i in range(2, k):                           #第3行及以后,i表示行号
            m1[i, 0] = 1                                #每一行第一个值均为1
            for j in range(1, i + 1):
                m1[i, j] = m1[i - 1][j - 1] + m1[i - 1][j]
                #自第二个值开始，元素的值等于上方元素与左上方元素相加

    for j in range(k):
        print(int(m1[k-1,j]),end =' ')
    print('\n_____________________________________________________________')


Pascal_triangle(100)
Pascal_triangle(200)
```

Pascal_triangle(100)

```
In [28]: runfile('D:/github-repo/ESE5023_Assignments_12132243/PS1_3.py', wdir='D:/github-repo/ESE5023_Assignments_12132243')
1 99 4851 156849 3764376 71523144 1120529256 14887031544 171200862756 1731030945644 15579278510796 126050526132804
924370524973896 6186171974825304 38000770702498304 215337700647490368 1130522928399324288 5519611944537877504
25144898858450329600 107196674080761921536 428786696323047686144 1613054714739083968512 5719012170438570672128
19146258135816085700608 60629817430084281171968 181889452290252818350080 517685364210719585730560 139966783656972328212889 6
3599145865465002725474304 881170194648328342395840 205606378751276623949987 84 4576400043173575748118118 4
97248500917438495384928256 19744392610510239972091 4944 383273503615787035476885504 711793649572175834674888704
1265410932572756856170086400 215461861492118075637917 2864 351543037171350556035671 6544 549849365832112416616139980 8
824774048748168624924209971 2 118686997258882796223421 48096 163901091452742898598899875 84 217264237507124360630732062 72
2765181204636127906351389081 6 337966591677749024972456591 36 39674339023040099107160981504 447391482600239415463730216 96
484674106150259469373459660 80 504456722727821186280435220 48 504456722727821186280435220 48 484674106150259469373459660 80
447391482600239415463730216 96 39674339023040099107160981504 337966591677749024972456591 36 27651812046361279063513890816
217264237507124360630732062 72 163901091452742898598899875 84 118686997258882796223421 48096 824774048748168624924209971 2
549849365832112416616139980 8 351543037171350556035671 6544 215461861492118075637917 2864 126541093257275685617008640 0
711793649572175834674888704 38327350361578703547688 5504 19744392610510239972091 4944 97248500917438495384928256
4576400043173575748118118 4 205606378751276623949987 84 881170194648328342395840 359914586546500272547430 4
139966783656972328212889 6 517685364210719585730560 181889452290252818350080 606298174300842811719 68 191462581358160857006 8
571901217043857067212 8 161305471473908396851 2 428786696323047686144 107196674080761921536 251448988584503296 0
5519611944537877504 1130522928399324288 215337700647490368 38000770702498304 6186171974825304 924370524973896 126050526132804
15579278510796 1731030945644 171200862756 14887031544 1120529256 71523144 3764376 156849 4851 99 1
```

Pascal_triangle(200)

```
1 199 19701 1293699 63391251 2472258789 79936367511 2203959847089 52895036330136
1122550215450664 21328454093562616 366461620334848640 5741232051912627200 8258541490058933043
109720622653640068300 8 135322101272822761390 08 155620416463746214395904
167520801252150359569 2032 169382143488285370126172 16 161358778796735012183998464
145222900917061493785729 4336 123785234591209582811540357 12 100153507987433208310977789952
770746561468507584372469137408 56521414507690567446313471836 16
395649901553833904570199892295 68 264781087962950344181906157862912
169656030435520068038981610949 8368 104217275838962325450871731356958 72
614522557533191672595206188658 85184 348229449268808675959763752605188096
189841215891705348776234457284 1099264 996666383431453110590021418676859 6992
504373594039553533469145081328 91754496 246252990031076074704930118366440456192
116090695300364456805999548919 7000491008 528857611923882537202998969109750533 3248
232983218225926640239553610554377 20911872 993244246121055608092839856373949750 31296
410031599039717872593535946991964168126464 164012639615887118088913396662278794772 4800
636049017046976803898137577928346413983334 4 239275582603386552028310170461548742 94632448
873634103923992570873442768613222076844605 44 309743000482142843990723178995514843 519254528
106689255721626983118056317843536783591918 7968 357177073502838120914941361238522445 7159966720
116272536693477105124854892581583345428893 53216
368196366196010811767863822704523902246442 76224
113464594480811484913789968028007868106400 595968
340393783442434454741369904084023604319201 787904
994483798684759347023246541916435480586619 977728
283045388856431479001200244678570139746067 0218240
785050418148970366513046578301802199966776 9769984
212254372314351246761648069454299518090045 59663104
559579708828744226689735438085316534463053 06378240
143891925127391408181681856889327990213902 759624704
360992022688017060161562538180328751242269 224337408
883808055546524569573178836522826627086904 618647552
211215145478067737526416463246255889803615 1930585088
492835339448824632279772028580205549287964 7651135488
112301823251453477205753312826762685660708 64879026176
249962122720977099142324455528624881613150 75475374080
543568425599585165086695261451480306937660 65389830144
115508290439911855556290717767934802647095 250635456512
239901833990586189351528017940873623177468 627954696192
487073420526341628811389944529022976624552 352594001920
966877088507514007514445980410770222979489 608503394304
187687905416164509833220513680914366954076 7864329338880
356335008833587684691305800270988993182386 7700273217536
661765016405234320181334617780188624963662 2060176277504
120236179037007356072767496249657204089813 53024920223744
213753207176901966351586659999390585048557 38710969286656
371872017965295133119110475941211261921815 98311373864960
633187490049016137534489515786670053583517 18361631555584
105531248341502682329724424984424455045261 238724997414912
172182563083504381017565426646179197527736 731737241157632
275044873497026456990247315468770043485734 44490899599523 84
430198391879964440500174327936230755888169 368569904103424
658911460980705107411308795383056118642599 471857941872640
988367191471057661116963193074584177963899 207786912808960
```

(The complete results can be viewed by running the code.)

PS4: Add or double

In this case, I did the reverse. That is, if I have an integer between 1 and 100, I can divide by 2(remainder must be 0) or subtract by 1 each time to find the smallest number of operations so that the final result is 1.

Here is the code and some results:

```python
import random


def Least_moves(x):
    i = 1              #i表示操作的次数
    while x != 2:
        if x % 2 != 0:
            x -= 1
            i += 1
        else:
            x /= 2
            i += 1
    print(i)


b = random.randint(2, 100)
print(b)
Least_moves(b)

```

```
In [1]: runfile('D:/github-repo/ESE5023_Assignments_12132243/PS1_4.py', wdir='D:/github-repo/ES
16
4

In [2]: runfile('D:/github-repo/ESE5023_Assignments_12132243/PS1_4.py', wdir='D:/github-repo/ES
75
9

In [3]: runfile('D:/github-repo/ESE5023_Assignments_12132243/PS1_4.py', wdir='D:/github-repo/ES
68
7

In [4]: runfile('D:/github-repo/ESE5023_Assignments_12132243/PS1_4.py', wdir='D:/github-repo/ES
41
7

In [5]: runfile('D:/github-repo/ESE5023_Assignments_12132243/PS1_4.py', wdir='D:/github-repo/ES
98
8
```

PS5: Dynamic programming

5.1: In this case, I've created a function Find_expression(a) that takes argument “a”. The nine integers from 1 to 9 are sequentially stored in a string "num", and the list "sym" is used to store three possible symbols between each two numbers. A for loop is then used to list all possible cases, and an eval function is used to evaluate the result of the string operator re, and finally to print the result equal to a.

Here is the code and some results:

```
import random

def Find_expression(a):
    num = '123456789'
    sym = ['+', '-', '']
    for s1 in sym:
        for s2 in sym:
            for s3 in sym:
                for s4 in sym:
                    for s5 in sym:
                        for s6 in sym:
                            for s7 in sym:
                                for s8 in sym:
                                    re = num[-9] + s1 + num[-8] + s2 + num[-7] + s3 + num[-6] + s4 + num[
                                        -5] + s5 + num[-4] + s6 + num[-3] + s7 + num[-2] + s8 + num[-1]
                                    if eval(re) == a:
                                        print(re, ' = ', eval(re))


Find_expression(random.randint(1,100))
```

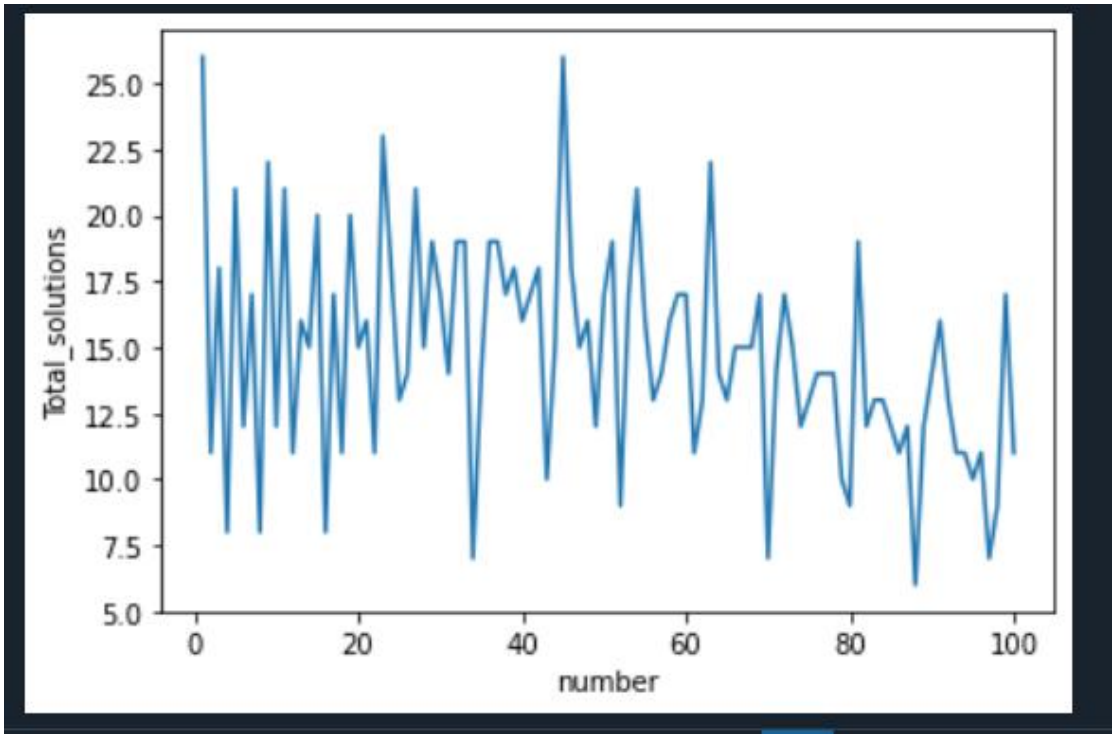
```
In [16]: runfile('D:/github-repo/ESE5023_Assignments_12132243')
1+2+3-4-5+6-7+89  =  85
1+2+34+56-7+8-9  =  85
1+2-3+4+5-6-7+89  =  85
1+2-3-4+5+67+8+9  =  85
1+23+45+6-7+8+9  =  85
1-2+3+4-5+67+8+9  =  85
1-2-3+4-5-6+7+89  =  85
1-2-3-4+5+6-7+89  =  85
12+3+4+56-7+8+9  =  85
12-3-4+5+6+78-9  =  85
12-3-4+56+7+8+9  =  85
12-34+5+6+7+89  =  85
```

```
In [19]: runfile('D:/github-repo/ESE5023_Assignments_12132243')
1+2+3-4+56+7+8-9  =  64
1+2+34+5-67+89  =  64
1+2-3-4+5-6+78-9  =  64
1+2-34+5-6+7+89  =  64
1-2+3+4-5-6+78-9  =  64
1-2+3-4+56-7+8+9  =  64
1-2+34-56+78+9  =  64
1-2-3+4+56+7-8+9  =  64
12+3+4-5+67-8-9  =  64
12+3+45-6-7+8+9  =  64
12-34+5-6+78+9  =  64
123+4+5-67+8-9  =  64
123-4-5-67+8+9  =  64
123-45-6-7+8-9  =  64
```

5.2: In this case, we only need to change the 5.1 code slightly. Use a list to store solutions with integers 1-100, chart the results, and find the best value.

Here is the code and some results:

```python
import matplotlib.pyplot as plt

Total_solutions = []
num = range(1,101)


def Find_expression(a):
    num = '123456789'
    sym = {'+', '-', ''}
    i = 0

    for s1 in sym:
        for s2 in sym:
            for s3 in sym:
                for s4 in sym:
                    for s5 in sym:
                        for s6 in sym:
                            for s7 in sym:
                                for s8 in sym:
                                    re = num[-9] + s1 + num[-8] + s2 + num[-7] + s3 + num[-6] + s4 + num[-5] + s5 + num[
                                        -4] + s6 + num[-3] + s7 + num[-2] + s8 + num[-1]
                                    if eval(re) == a:
                                        i += 1
    Total_solutions.append(i)


for j in range(1, 101):
    Find_expression(j)

# plot--------------------------------------------------------------------
# Plot a line
plt.plot(num, Total_solutions)

# Add x and y labels
plt.xlabel("number")
plt.ylabel("Total_solutions")

# find the max and min----------------------------------------------------
for i in range(100):
    if Total_solutions[i] == max(Total_solutions):
        print('max:',i+1,' Total_solutions:',max(Total_solutions))
    elif Total_solutions[i] == min(Total_solutions):
        print('min:',i+1,' Total_solutions:',min(Total_solutions))
```




```
In [26]: runfile('D:/github-repo/ESE5023_Assignments_12132243/PS1_5.2.py', wdir='D:/github-
repo/ESE5023_Assignments_12132243')
max: 1  Total_solutions: 26
max: 45  Total_solutions: 26
min: 88  Total_solutions: 6
```

Reference:

1.https://www.cnblogs.com/oNull/p/13472480.html#random_1 help me know how to create random numbers in python. **In in problem set 1.**

2.https://www.runoob.com/numpy/numpy-array-creation.html and https://www.cnblogs.com/pipiyan/p/10445948.html helped me understand how to create matrices using the Numpy library, **in problem set2**.

3.https://blog.csdn.net/weixin_34273481/article/details/92068609 tells me how do I create a 0 matrix, **in problem set3.**

4. This is where I found the use of the eval function(https://www.runoob.com/python/python-func-eval.html) , which helps me evaluate string expressions, **in problem set5**.

5. What have I learned from this website(https://zhu-group.github.io/ese5023/Lab_02.html) about how to chart data, **in problem set5**.