Assaiment1

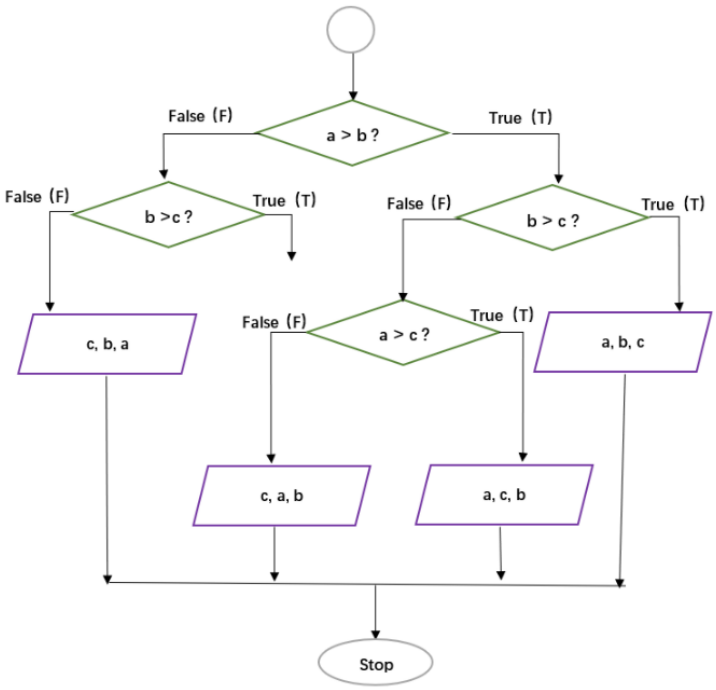
12132834 曹喆

第一题

题目：

1. Flowchart

[10 points] Write a function `Print_values` with arguments `a`, `b`, and `c` to reflect the following flowchart. Here the purple parallelogram operator is to print values in the given order. Report your output with some random `a`, `b`, and `c` values.



代码：

```python
#1
def Print_values(a, b, c):
    if(a>b):
        if(b>c):
            print(str(a)+' , '+str(b)+' , '+str(c))
        else:
            if(a>c):
                print(str(a)+' , '+str(c)+' , '+str(b))
            else:
                print(str(c)+' , '+str(a)+' , '+str(b))
    else:
        if(b>c):
            if(a>c):
                print(str(a)+' , '+str(c)+' , '+str(b))
            else:
                print(str(c)+' , '+str(a)+' , '+str(b))
        else:
            print(str(c)+' , '+str(b)+' , '+str(a))
result_1 = Print_values(1,2,3)
result_2 = Print_values(3,2,1)
result_3 = Print_values(2,1,3)
print(result_1)
print(result_2)
print(result_3)
```

代码运行结果：

举例：a = 1 b = 2 c = 3

a = 3 b = 2 c = 1

a = 2 b = 1 c = 3

运行结果为：

```
PS C:\Users\Cao Zhe\Desktop\南科大课后作业\环境编程\PS1> & python "c:/Users/Cao Zhe/Desktop/南科大课后作业/环境编程/PS1/PS1_1.py"
3 , 2 , 1
3 , 2 , 1
3 , 2 , 1
```

运行完的结果恰巧都输出为 3，2，1

第二题

题目：

2. Matrix multiplication

2.1 [5 points] Make two matrices `M1` (`5` rows and `10` columns) and `M2` (`10` rows and `5` columns); both are filled with random integers from `0` and `50`.

2.2 [10 points] Write a function `Matrix_multip` to do matrix multiplication, *i.e.*, `M1 * M2`. Here you are **ONLY** allowed to use `for` loop, `*` operator, and `+` operator.

代码：

```python
#2.1
import numpy as np
M1 = np.random.randint(0,51,size=(5,10))
M2 = np.random.randint(0,51,size=(10,5))#low,high,size
print(M1)
print(M2)
#2.2
def Matrix_multip(a,b):
    A = np.shape(a)
    B = np.shape(b)
    a_row = A[0]
    a_col = A[1]
    b_row = B[0]
    b_col = B[1]
    if(a_col!=b_row):
        print("can not mulitp!")
        exit
    else:
        pass
    result = np.zeros((a_row,b_col))
    for i in range(a_row):
        for j in range(b_col):
            mul_sum=0
            for t in range(a_col):
                mul_sum+=a[i][t]*b[t][j]
            result[i][j]=mul_sum
    print(result)

Matrix_multip(M1,M2)
```

运行结果：
随机输出的 5 行 10 列矩阵为 M1 为:

```
[[22 42  6  0  6 25 14 39 36  3]
 [36 48 47 13  7  0 27 13 19 34]
 [31 24 31  6 13 30 24 36 16 48]
 [27 17 40 40 47  8 22  0 20  1]
 [ 8 16  1 17 16 43 36 43  8 25]]
```

随机输出的 10 行 5 列矩阵 M2 为：

```
[[ 8 30 37 39 16]
 [34 18 28 31 39]
 [39  2 17 46  2]
 [ 9 46 18  9  8]
 [40 30 49 21 16]
 [ 2 44 49 41 11]
 [50 44 19 45 23]
 [40 26  3 30 10]
 [17  3 42 44 50]
 [44 13 28 29 40]]
```

M1 和 M2 的乘积为：

```
[[5132. 4485. 5590. 7058. 5005.]
 [7839. 4871. 6354. 8745. 5819.]
 [7931. 6074. 7141. 9192. 5712.]
 [6094. 5839. 6856. 6994. 3881.]
 [6282. 6735. 5807. 7093. 4277.]]
```

引用：在网上查找了得到矩阵的行列的值的方法：np.shape()
链接为：(38 条消息) Numpy shape 的用法_杨鑫 newlife 的专栏-CSDN 博客

第三题：
题目：

3. Pascal triangle

[20 points] One of the most interesting number patterns is Pascal's triangle (named after Blaise Pascal). Write a function `Pascal_triangle` with an argument `k` to print the k^{th} line of the Pascal triangle. Report `Pascal_triangle(100)` and `Pascal_triangle(200)`.

代码：

```python
#3
def Pascal_triangle(row_num):
    if row_num == 0:
        result = [[1]]
        exit
    if row_num == 1:
        result = [[1],[1,1]]
        exit
    result = [[1],[1,1]]
    row_num =  row_num -2
    row = 1
    while(row <= row_num):
        new_list = [1]
        ex_line = len(result[row])
        for i in range(ex_line-1):
            num = result[row][i]+result[row][i+1]
            new_list.append(num)
        new_list.append(1)
        result.append(new_list)
        row = row + 1
    print(result[row_num+1])

Pascal_triangle(100)
Pascal_triangle(200)
```

运行结果：

输出 Pascal_triangle(100)为：

```
[1, 99, 4851, 156849, 3764376, 71523144, 1120529256, 14887031544, 171200862756, 1731030945644, 15579278510796, 126050526132804, 924370524973896, 618617197482530
4, 38000770702498296, 215337700647490344, 1130522928399324306, 5519611944537877494, 25144898858450330806, 107196674080761936594, 428786696323047746376, 16130547
14739084379224, 5719012170438571889976, 19146258135816088501224, 60629817430084280253876, 181889452290252840761628, 517685364210719623706172, 139966783656972342
7057428, 3599145865465003098147672, 8811701946483283447189128, 20560637875127661376774632, 45764000431735762419272568, 97248500917438495140954207, 1974439261051
02399225573693, 383273503615787010261407757, 711793649572175876199757263, 1265410932572757113244012912, 2154618614921181030658724688, 3515430371713505892127392 9
12, 5498493658321124600506947888, 8247740487481686900760421832, 11868699725888281149874753368, 16390109145274293016493707032, 21726423750712434928840495368, 276
51812046361280818524266832, 33796659167774898778196326128, 39674339023040098565708730672, 44739148260023940935799206928, 48467410615025936013782474172, 50445672
272782096667406248628, 50445672272782096667406248628, 48467410615025936013782474172, 44739148260023940935799206928, 39674339023040098565708730672, 3379665916777
4898778196326128, 27651812046361280818524266832, 21726423750712434928840495368, 16390109145274293016493707032, 11868699725888281149874753368, 824774048748168690
0760421832, 5498493658321124600506947888, 3515430371713505892127392912, 2154618614921181030658724688, 1265410932572757113244012912, 711793649572175876199757263,
 383273503615787010261407757, 197443926105102399225573693, 97248500917438495140954207, 45764000431735762419272568, 20560637875127661376774632, 88117019464832834
47189128, 3599145865465003098147672, 1399667836569723427057428, 517685364210719623706172, 181889452290252840761628, 60629817430084280253876, 1914625813581608850
1224, 5719012170438571889976, 1613054714739084379224, 428786696323047746376, 107196674080761936594, 25144898858450330806, 5519611944537877494, 1130522928399243
06, 215337700647490344, 38000770702498296, 6186171974825304, 924370524973896, 126050526132804, 15579278510796, 1731030945644, 171200862756, 14887031544, 1120529
256, 71523144, 3764376, 156849, 4851, 99, 1]
```

输出 Pascal_triangle(200)为：

```
[1, 199, 19701, 1293699, 63391251, 2472258789, 79936367511, 2203959847089, 52895036330136, 1122550215450664, 21328454093562616, 366461620334848584, 574123205191
2627816, 82585414900589338584, 1097206226536401212616, 13532210127282281622264, 1556204164637462386560 36, 1675208012521503627885564, 169382143488285366819540 36,
 161358778796735007338614764, 1452229009170615066047532876, 12378523459120956991548018324, 100153507987433197477070330076, 770746561468507650149628192324, 56521
41450769056101097273410376, 39564990155383392707680913872632, 264781087962950397351403038993768, 1696560304355200694140471323923032, 1042172758389623283543432 38
46955768, 61452255753319166029629978545842632, 348229449268808607501236545093108248, 1898412158917053376377708907120493352, 99666638343145302259829717623825900 9
8, 504373594039553499314895843732694711 02, 2462529900310761202537432648812568294 98, 1160906953003644566910503963011639339062, 528857611923882524925896249816413 4
766838, 23298321822592662584573267221641999107962, 99324424612105561544759718155421154091838, 410031599039717830992469605718533482276562, 1640126396158871323969
87842287413392910 6248, 6360490170469769280761235835048470603119352, 23927558260338655865720839569944246554591848, 87363410392399278393445856104215039745835352,
30974300048214289612221712618767150455341 6248, 1066892557216269975532081212424201849017322632, 3571770735028382091998706667681023581492775768, 11627253669347711
91650642808840843846741265 3032, 36819636619601087735603688946626721813473401268, 113464594480811515266860347570217040690499665132, 34039378344243454580058104271
065112207149899 5396, 994483798684759751456599516938961121346144123804, 2830453888564316215684167855903197037677487121596, 78505041814897072397277863173164144252
56426544804, 2122543723143513438889364448755919455717 4782880396, 5595797088287444520708324455811060383255170 0321044, 14389192512739143053249977172085583842656 15
15111256, 3609920226880170976517099536154804367543560817 70344, 88380805554652461838866919678272796584687178 6403256, 21121514547806774778441077414638075116001512
18353544, 4928353394488247448302918063415550860400352842824936, 11230182325145350742854190341225599501568017133650264, 249962122720977161695786817272440763 09941
715555544136, 5435684255995852563860760947035616594384150843 0310264, 11550829043991186698204117012450685263066320541 4409311, 2399018339905861852703931994893 6038
6232915888168388569, 4870734205263416488823134656299135114425868032509 70731, 9668770885075140194230998646086342839084185795877 47869, 187687905416164486123307620
776970184523398900 7435039981, 35633500883358764756743910611279846626906168112172 49819, 66176501640523420262524405420948286592825740779748925 21, 1202361790370073
41040361243652145478457387613529 40297679, 21375320717690193962730887760381418392424464627449418096, 37187201796529515524203051309156714189560369968302412304, 63
318749004901607514183573850726297133575765081163566896, 10553124834150267919030595641787716188929294180193927 8160, 17218256308350437131049919205022063255621479 9
78211145384 0, 275044873497026463262225982106196594862524939911684530160, 4301983918799644681793791002173844176054877265285322 13840, 65891146098070507151525 15332
44348285193215378606992378160, 9883671914710576072728772998665224277898230679104885672 40, 14520456269759982131539802306681008507035672232652024 0760, 2089529072
96546084331914228315653537052464551635035839 5240, 2945480741409143598413730688304995642787753318228818460760, 40675686428983411597141995219449939828973736299350
35017240, 5503181105097755686672152294396168329802329028735635611560, 7294914488152838933495643739083292902296110572975144880440, 947500387541690574120698554616
5656298384603387887257143560, 12059095841439698216081617967847198925216767948220145455440, 15039995937076477550393928027315045850551249912948720736560, 18382217
256426805894925912033385056039562638782492880900240, 22018260230225514753262905622406275915520083816392571627760, 258475228789603868842651500784769325964800983 9
3156497128240, 29738547828481305339960979122548728901326564817932744007760, 33534958189564025170594295606278353867453360326605009200240, 370649537884655015043 41
06356483397006402739825572132595 8160, 401536999375042932963694855285701342360296814436981031213 40, 42637433954257136180681000097347668312485125656710356922660,
44377737380961509086014918468667981304831457316167922511340, 45274257328051640582702088538742081937252294837706668420660, 452742573280516405827020885387420819 37
25229483770666842066 0, 44377737380961509086014918468667981304831457316167922511340, 42637433954257136180681000097347668312485125656710356922660, 401536999375042
93296369485528570134236029681443698103121340, 37064953788465501504341063564833970064027398255721325958160, 33534958189564025170594295606278353867453360326605009
200240, 29738547828481305339960979122548728901326564817932744007760, 25847522878960386884265150078476932596480098393156497128240, 22018260230225514753262905622 4
06275915520083816392571627760, 18382217256426805894925912033385056039562638782492880900240, 15039995937076477550393928027315045850551249912948720736560, 1205909
58414396982160816179678471989252167679482201454554 40, 94750038754169057412069855461656562983846033878872571435 60, 7294914488152838933495643739083292902296110572
975144880440, 5503181105097755686672152294396168329802329028735635611560, 4067568642898341159714199521944993982897373629935035017240, 29454807414091435984137306
8830499564278775331822881846076 0, 2089529072965460843319142283156535370524645516350358395240, 1452045626975998213153980230668100850703567223226520240760, 988367
19147105760727287729986652242778982306791048856 7240, 658911460980705071515251533244348285193215378606992378160, 4301983918799644681793791002173844176054877265 28
532213840, 275044873497026463262225982106196594862524939911684530160, 172182563083504371310499192050220632556214799782111453840, 10553124834150267919030595641 78
77161889292941801939278160, 633187490049016075141835738507262971335757650811635 66896, 37187201796529515524203051309156714189560369968302412304, 2137532071769019
39627308877603814183924244646274494180 96, 12023617903700734104036124365214547845738761352940297679, 66176501640523420262524405420948286592825740779748925 21, 356
3350088335876475674391061127984662690616811217249819, 1876879054161644861233076207769701845233989007435039981, 96687708850751401942309986460863428390841857958 77
47869, 487073420526341648882313465629913511442586803250970731, 239901833990586185270393199489360386232915888168388569, 11550829043991186698204117012450685263066
3205414409311, 5435684255995852563860760947035616594384150843 0310264, 24996212272097716169578681727244076309941715555544136, 112301823251453507428541903412255 99
50156801713 3650264, 4928353394488247448302918063415550860400352842824936, 2112151454780677477844107741463807511600151218353544, 88380805554652461838866919678272
7965846871786403256, 360992022688017097651709953615480436754356081770344, 143891925127391430532499771720855838426561515111256, 559579708828744452070832445581106
0383255170032 1044, 212254372314351343888936444875591945571747828803 96, 785050418148970723972778631731641442525642654480 4, 28304538885643162156841678559031970376
77487121596, 994483798684759751456599516938961121346144123804, 340393783442434545800581042710651122071498995396, 11346459448081151526686034757021704069049966513
2, 36819636619601087735603688946626721813473401268, 11627253669347711916506428088408438467412653032, 3571770735028382091998706667681023581492775768, 10668925572
162699755320812124242018490173226 32, 30974300048214289612221712618767150455341 6248, 87363410392399278393445856104215039745835352, 239275582603386558657208395699
44246554591848, 6360490170469769280761235835048470603119352, 1640126396158871323969878422874133929106248, 410031599039717830992469605718533482276562, 9932442461
2105561544759718155421154091838, 23298321822592662584573267221641999107962, 5288576119238825249258962498164134766838, 1160906953003644566910503963011639339062,
2462529900310761202537432648812568294 98, 504373594039553499314895843732694711 02, 99666638343145302259829717623825900 98, 189841215891705337637770890712049335 2, 3
48229449268808607501236545093108248, 61452255753319166029629978545842632, 10421727583896232835434323846955768, 1696560304355200694140471323923032, 2647810879629
50397351403038993768, 39564990155383392707680913872632, 56521414507690561010972734103 76, 770746561468507650149628192324, 100153507987433197477070330076, 1237852
3459120956991548018324, 1452229009170615066047532876, 161358778796735007338614764, 169382143488285366819540 36, 1675208012521503627885564, 155620416463746238656 0
36, 13532210127282281622264, 1097206226536401212616, 82585414900589338584, 5741232051912627816, 366461620334848584, 21328454093562616, 1122550215450664, 5289503
6330136, 2203959847089, 79936367511, 2472258789, 63391251, 1293699, 19701, 199, 1]
```

引用：从网上搜索矩阵尾部插入值的办法 append：
(38 条消息) append() 方法_成都都成-CSDN 博客_append

第四题：
题目：

4. Add or double

[20 points] If you start with `1` RMB and, with each move, you can either double your money or add another `1` RMB, what is the smallest number of moves you have to make to get to exactly x RMB? Here x is an integer randomly selected from `1` to `100`. Write a function `Least_moves` to print your results. For example, `Least_moves(2)` should print `1`, and `Least_moves(5)` should print `3`.

代码：

```
#4
import numpy as np
def Least_moves(target_money,least_step=0):
        while(target_money%2==0):
            target_money = target_money / 2
            least_step += 1
        if(int(target_money) == 1):
            return least_step
        else:
            target_money = target_money - 1
            least_step += 1
            least_step = Least_moves(target_money, least_step)
            return least_step


num = np.random.randint(1,101,1)
print(str(num[0])+' RMB')
result = Least_moves(num[0])
print('least_moves = '+str(result))
```

运行结果：

```
85 RMB
least_moves = 9
```

得到 1-100 的随机数为 85，从 1 经过加或乘最少要经过 9 步
引用：
网上查找得到范围内的随机整数的方法 np.random.randint()：
(38 条消息) random.randint()用法_还没想好的博客-CSDN 博客

第五题：

题目：

5. Dynamic programming

Insert `+` or `-` operation anywhere between the digits `123456789` in a way that the expression evaluates to an integer number. You may join digits together to form a bigger number. However, the digits must stay in the original order.

5.1 [30 points] Write a function `Find_expression`, which should be able to print every possible solution that makes the expression evaluate to a random integer from `1` to `100`. For example, `Find_expression(50)` should print lines include:

$$1 - 2 + 34 + 5 + 6 + 7 + 8 - 9 = 50$$

and

$$1 + 2 + 34 - 56 + 78 - 9 = 50$$

5.2 [5 points] Count the total number of suitable solutions for any integer *i* from `1` to `100`, assign the count to a list called `Total_solutions`. Plot the list `Total_solutions`, so which number(s) yields the maximum and minimum of `Total_solutions`?

5.1 代码：

```python
#5.1
import numpy as np
def Find_expression(sum_num):
    plist = all_strings(9)
    result = []
    for command in plist:
        if eval(command) == sum_num:
            result.append(command + '=' + str(sum_num))
    return result

def all_strings(n):
    if n == 1:
        return ['1']
    result = []
    for s in all_strings(n - 1):
        result.append(s + str(n))
        result.append(s + "+" + str(n))
        result.append(s + '-' + str(n))
    return result


num = np.random.randint(1,101,1)
res = Find_expression(num[0])
for i in res:
    print(i)
print('Total counts: ', len(res))
```

5.1 运行结果：

```
123+4-56-7-8-9=47
123+4-5-6-78+9=47
12+34+5+6+7-8-9=47
1+23+45+67-89=47
1+23+4+5+6+7-8+9=47
1+23-45+67-8+9=47
1+23-4+5-67+89=47
1+2+3+45+6+7-8-9=47
1+2+3-4-5+67-8-9=47
1+2-3+45-6+7-8+9=47
1-23-4+5+67-8+9=47
1-2+3+45+6-7-8+9=47
1-2+3+45-6+7+8-9=47
1-2+3-45-6+7+89=47
1-2-3-4+5+67-8-9=47
Total counts:  15
```

得到的 1 到 100 的随机数为 47，共得到 15 个结果为 47 的式子

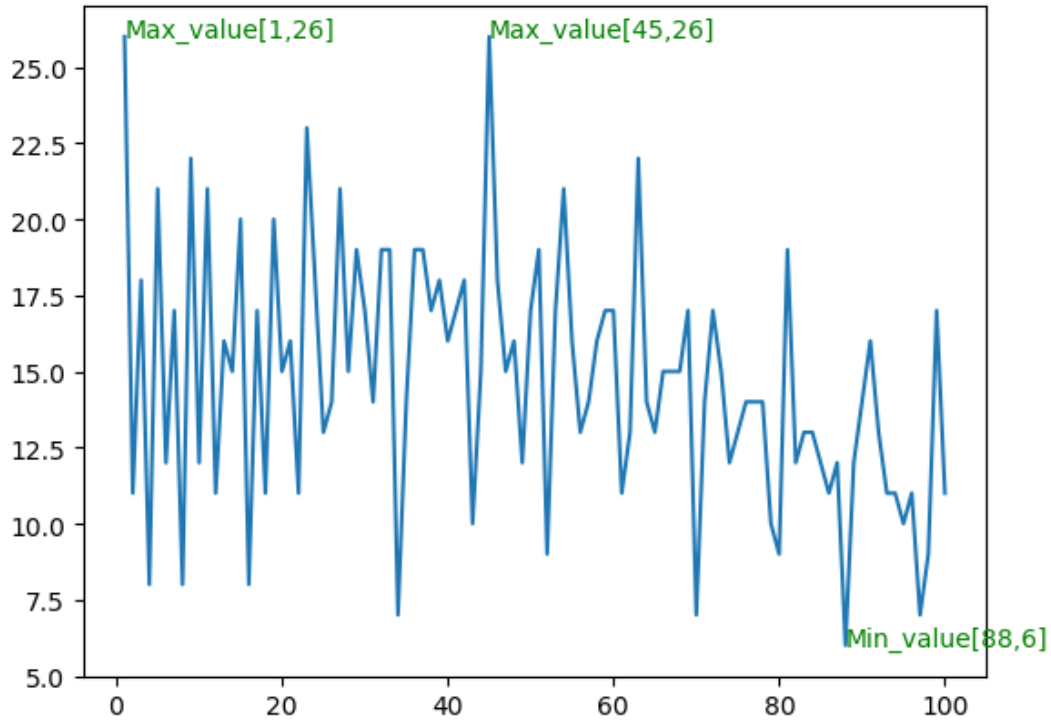
5.2 代码：

```python
#5.2
import numpy as np
import matplotlib.pyplot as plt
def Find_expression(sum_num):
    plist = all_strings(9)
    result = []
    for command in plist:
        if eval(command) == sum_num:
            result.append(command + '=' + str(sum_num))
    return result

def all_strings(n):
    if n == 1:
        return ['1']
    result = []
    for s in all_strings(n - 1):
        result.append(s + str(n))
        result.append(s + "+" + str(n))
        result.append(s + '-' + str(n))
    return result

Total_solutions = []
for j in range(1,101):
    solution = 0
    res = Find_expression(j)
    Total_solutions.append(len(res))
X = []
for i in range(1,101):
    X.append(i)
Y = Total_solutions
plt.plot(X,Y)
max_solution = []
max_solution_ = max(Total_solutions)
for i in range(100):
    if Total_solutions[i] == max_solution_:
        max_solution.append(i)
min_solution = []
min_solution_ = min(Total_solutions)
for i in range(100):
    if Total_solutions[i] == min_solution_:
        min_solution.append(i)
print(max_solution)
print(min_solution)#得到最大值最小值的位置
plt.text(0 + 1, 26, 'Max_value[1,26]', color = 'g')
plt.text(44 + 1, 26, 'Max_value[45,26]', color = 'g' )
plt.text(87 + 1, 6, 'Min_value[88,6]', color = 'g')
plt.show()
```

5.2 运行结果：



得到最大值两个：当 1，45 时，得到的式子最多，为 26 个
得到最小值一个：当 88 时，得到的式子最少，为 6 个
引用：
在网络上查找 plot 的用法：
(38 条消息) plt.plot()函数详解 Fighting Hua-CSDN 博客