Assignment 1 Report

Question 1

```
a=float(input("a : "))
b=float(input("b : "))
c=float(input("c : "))
```

将输入的 a、b、c 转化为浮点数

```
if a>b:
    if b>c:
        print("a,b,c")
    elif b<c:
        if a>c:
            print("a,c,b")
        elif a<c:
            print("c,a,b")
elif a<b:
    if b<c:
        print("c,b,a")
    elif b>c:
        if a>c:
            print("b,a,c")
        elif a<c:
            print("b,c,a")
```

建立程序判断值的大小

输出结果

输入　a=1533　b=53531.12　c=0.1215

```
   ...:         elif a<c:
   ...:             print("b,c,a")
a : 1533
b : 53531.12
c : 0.1215
b,a,c
```

Question 2.1

```
import numpy as np
```

导入 numpy 的库

```
M1 = np.random.randint(0, 51, size=(5, 10))
```

选取 0-50 的随机数，按照 5 行 10 列排布

```
M2 = np.random.randint(0, 51, size=(10, 5))
```

选取 0-50 的随机数，按照 5 行 10 列排布

```
print("M1:")
print(M1)
print("M2:")
print(M2)
```

输出 M1、M2

```
...: print(M2)
M1:
[[ 4 28 17 37 20 11 39 16  2 12]
 [30 11  1 27 22 12 12 29  0 41]
 [10 47 32 25 30 42 21 22 19 37]
 [34 34 42  2 37 23 17 29 43 20]
 [37 24 48 15 44  4 13  0 37  8]]
M2:
[[50 33 48 33 12]
 [23 28 14 33 31]
 [ 5  2 35  2 47]
 [45 13  4 14 13]
 [11 18  9 18 12]
 [ 0 32 13 19 11]
 [18 39 15  4 49]
 [41 16 32 18 38]
 [ 2 27 20 13 10]
 [21 44 30 29 32]]
```

Question 2.2

```
#I got inspired by reading
#https://wenku.csdn.net/answer/de84e63cb56d41d08ac01b3c460dc782
import numpy as np
M1 = np.random.randint(0, 51, size=(5, 10))
M2 = np.random.randint(0, 51, size=(10, 5))
def matrix_multiplication(M1, M2):                          定义一个函数
    row_M1, col_M1 = len(M1), len(M1[0])                    选出 M1,M2 的行列长度
    row_M2, col_M2 = len(M2), len(M2[0])
    if col_M1 != row_M2:                                    判断是否可以进行矩阵乘法
        return None
    result = [[0 for _ in range(col_M2)] for _ in range(row_M1)]    建立结果
    for i in range(row_M1):
        for j in range(col_M2):
            for k in range(col_M1):
                result[i][j] += M1[i][k] * M2[k][j]             相乘
    return result

result = matrix_multiplication(M1, M2)
if result is not None:
    for row in result:
        print(row)                                          输出结果
else:
    print("矩阵无法相乘")
```

```
...:     print("矩阵无法相乘")
[2655, 6388, 5732, 6526, 5124]
[3121, 6914, 7181, 6668, 6100]
[6018, 9760, 8588, 9985, 7329]
[3324, 5789, 6907, 5384, 6204]
[5479, 9047, 8499, 7831, 7519]
```

Question 3

```
#I got inspired by reading
#ways:https://www.mathsisfun.com/pascals-triangle.html
#https://blog.csdn.net/m0_62338174/article/details/129746943

def Pascal_triangle(k):                                  定义一个函数
    row = np.zeros(k+1, dtype=float)                     建立一行为 k+1 个 0 的浮点数集
                                                         （ 如果是整数 int 数值太大 超出进制 不好表示）
        row[0] = 1                                       赋值第一个为 1
        for i in range(1, k + 1):                        循环
            row[i] = row[i - 1] * (k - i + 1) /i         利用二项式系数得出公式
        return row                                       调用 row
print("Pascal triangle （k = 100) :")
print(Pascal_triangle(99))
print("Pascal triangle （k = 200) :")
print(Pascal_triangle(199))                              输出
```

```
...: print(Pascal_triangle(199))
Pascal triangle (k = 100) :
[1.00000000e+00 9.90000000e+01 4.85100000e+03 1.56849000e+05
 3.76437600e+06 7.15231440e+07 1.12052926e+09 1.48870315e+10
 1.71200863e+11 1.73103095e+12 1.55792785e+13 1.26050526e+14
 9.24370525e+14 6.18617197e+15 3.80007707e+16 2.15337701e+17
 1.13052293e+18 5.51961194e+18 2.51448989e+19 1.07196674e+20
 4.28786696e+20 1.61305471e+21 5.71901217e+21 1.91462581e+22
 6.06298174e+22 1.81889452e+23 5.17685364e+23 1.39966784e+24
 3.59914587e+24 8.81170195e+24 2.05606379e+25 4.57640004e+25
 9.72485009e+25 1.97443926e+26 3.83273504e+26 7.11793650e+26
 1.26541093e+27 2.15461861e+27 3.51543037e+27 5.49849366e+27
 8.24774049e+27 1.18686997e+28 1.63901091e+28 2.17264238e+28
 2.76518120e+28 3.37966592e+28 3.96743390e+28 4.47391483e+28
 4.84674106e+28 5.04456723e+28 5.04456723e+28 4.84674106e+28
 4.47391483e+28 3.96743390e+28 3.37966592e+28 2.76518120e+28
 2.17264238e+28 1.63901091e+28 1.18686997e+28 8.24774049e+27
 5.49849366e+27 3.51543037e+27 2.15461861e+27 1.26541093e+27
 7.11793650e+26 3.83273504e+26 1.97443926e+26 9.72485009e+25
 4.57640004e+25 2.05606379e+25 8.81170195e+24 3.59914587e+24
 1.39966784e+24 5.17685364e+23 1.81889452e+23 6.06298174e+22
 1.91462581e+22 5.71901217e+21 1.61305471e+21 4.28786696e+20
 1.07196674e+20 2.51448989e+19 5.51961194e+18 1.13052293e+18
 2.15337701e+17 3.80007707e+16 6.18617197e+15 9.24370525e+14
 1.26050526e+14 1.55792785e+13 1.73103095e+12 1.71200863e+11
 1.48870315e+10 1.12052926e+09 7.15231440e+07 3.76437600e+06
 1.56849000e+05 4.85100000e+03 9.90000000e+01 1.00000000e+00]
```

```
Pascal triangle (k = 200) :
[1.00000000e+00 1.99000000e+02 1.97010000e+04 1.29369900e+06
 6.33912510e+07 2.47225879e+09 7.99363675e+10 2.20395985e+12
 5.28950363e+13 1.12255022e+15 2.13284541e+16 3.66461620e+17
 5.74123205e+18 8.25854149e+19 1.09720623e+21 1.35322101e+22
 1.55620416e+23 1.67520801e+24 1.69382143e+25 1.61358779e+26
 1.45222901e+27 1.23785235e+28 1.00153508e+29 7.70746561e+29
 5.65214145e+30 3.95649902e+31 2.64781088e+32 1.69656030e+33
 1.04217276e+34 6.14522558e+34 3.48229449e+35 1.89841216e+36
 9.96666383e+36 5.04373594e+37 2.46252990e+38 1.16090695e+39
 5.28857612e+39 2.32983218e+40 9.93244246e+40 4.10031599e+41
 1.64012640e+42 6.36049017e+42 2.39275583e+43 8.73634104e+43
 3.09743000e+44 1.06689256e+45 3.57177074e+45 1.16272537e+46
 3.68196366e+46 1.13464594e+47 3.40393783e+47 9.94483799e+47
 2.83045389e+48 7.85050418e+48 2.12254372e+49 5.59579709e+49
 1.43891925e+50 3.60992023e+50 8.83808056e+50 2.11215145e+51
 4.92835339e+51 1.12301823e+52 2.49962123e+52 5.43568426e+52
 1.15508290e+53 2.39901834e+53 4.87073421e+53 9.66877089e+53
 1.87687905e+54 3.56335009e+54 6.61765016e+54 1.20236179e+55
 2.13753207e+55 3.71872018e+55 6.33187490e+55 1.05531248e+56
 1.72182563e+56 2.75044873e+56 4.30198392e+56 6.58911461e+56
 9.88367191e+56 1.45204563e+57 2.08952907e+57 2.94548074e+57
 4.06756864e+57 5.50318111e+57 7.29491449e+57 9.47500388e+57
 1.20590958e+58 1.50399959e+58 1.83822173e+58 2.20182602e+58
 2.58475229e+58 2.97385478e+58 3.35349582e+58 3.70649538e+58
 4.01536999e+58 4.26374340e+58 4.43777374e+58 4.52742573e+58
 4.52742573e+58 4.43777374e+58 4.26374340e+58 4.01536999e+58
 3.70649538e+58 3.35349582e+58 2.97385478e+58 2.58475229e+58
 2.20182602e+58 1.83822173e+58 1.50399959e+58 1.20590958e+58
 9.47500388e+57 7.29491449e+57 5.50318111e+57 4.06756864e+57
 2.94548074e+57 2.08952907e+57 1.45204563e+57 9.88367191e+56
 6.58911461e+56 4.30198392e+56 2.75044873e+56 1.72182563e+56
 1.05531248e+56 6.33187490e+55 3.71872018e+55 2.13753207e+55
 1.20236179e+55 6.61765016e+54 3.56335009e+54 1.87687905e+54
 9.66877089e+53 4.87073421e+53 2.39901834e+53 1.15508290e+53
 5.43568426e+52 2.49962123e+52 1.12301823e+52 4.92835339e+51
 2.11215145e+51 8.83808056e+50 3.60992023e+50 1.43891925e+50
 5.59579709e+49 2.12254372e+49 7.85050418e+48 2.83045389e+48
 9.94483799e+47 3.40393783e+47 1.13464594e+47 3.68196366e+46
 1.16272537e+46 3.57177074e+45 1.06689256e+45 3.09743000e+44
 8.73634104e+43 2.39275583e+43 6.36049017e+42 1.64012640e+42
 4.10031599e+41 9.93244246e+40 2.32983218e+40 5.28857612e+39
 1.16090695e+39 2.46252990e+38 5.04373594e+37 9.96666383e+36
 1.89841216e+36 3.48229449e+35 6.14522558e+34 1.04217276e+34
 1.69656030e+33 2.64781088e+32 3.95649902e+31 5.65214145e+30
 7.70746561e+29 1.00153508e+29 1.23785235e+28 1.45222901e+27
 1.61358779e+26 1.69382143e+25 1.67520801e+24 1.55620416e+23
 1.35322101e+22 1.09720623e+21 8.25854149e+19 5.74123205e+18
 3.66461620e+17 2.13284541e+16 1.12255022e+15 5.28950363e+13
 2.20395985e+12 7.99363675e+10 2.47225879e+09 6.33912510e+07
 1.29369900e+06 1.97010000e+04 1.99000000e+02 1.00000000e+00]
```

Question 3

```
def Least_moves(x):                                  定义一个函数
    steps = np.full(x + 1, 0)                        填入 x+1 个 0
    for i in range(2, x + 1):                        开始循环
        if i % 2 == 0:                               判断 x 除 2 的余数
            steps[i] = steps[i // 2] + 1
        else:
            steps[i] = steps[i - 1] + 1
    print(int(steps[x]))                             将 steps 转化成整数
Least_moves(2)
Least_moves(5)                                       输出
```

```
...:     print(int(steps[x]))
...: Least_moves(2)
...: Least_moves(5)
1
3
```

Question 5.1

```
#consult with my roommate he teach how to set up the function and caculate

def Find_expression(target, current_expr="", current_num=1):
#target：目标整数，即要得到的结果。current_expr：当前生成的表达式
#current_num：当前正在处理的数字。
    if current_num == 10:
        # 如果当前数字已经是 9,检查当前表达式是否等于目标整数
      if eval(current_expr) == target:          使用 eval 函数计算当前表达式的结果，
            print(current_expr + " = " + str(target))
        return

    # 选择 1：将当前数字添加到当前表达式并继续递归
    Find_expression(target, current_expr + str(current_num), current_num + 1)

    # 选择 2：在当前表达式中添加加法运算符并继续递归
    Find_expression(target, current_expr + "+" + str(current_num), current_num + 1)

    # 选择 3：在当前表达式中添加减法运算符并继续递归
    Find_expression(target, current_expr + "-" + str(current_num), current_num + 1)

# 测试函数，找到所有满足条件的表达式
Find_expression(50)
```

```
    ...: 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
+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
-12+3+45+6+7-8+9 = 50
-12+3+4+5+67-8-9 = 50
-12+3-4-5+67-8+9 = 50
-12-3+4-5+67+8-9 = 50
-1+23-4+56-7-8-9 = 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
-1-2+34+5+6+7-8+9 = 50
-1-2+3+4+56+7-8-9 = 50
```