Test1:

对代码的解释:

- 1. if a > b:这是最外层的条件语句, 检查 a 是否大于 b。
- 2. if b > c:这是第二层的条件语句,如果 b 大于 c ,表示 a 、b 、c 已经按升序排列,print(a, b, c),如果条件成立,即 a > b 且 b > c,那么这一行代码会打印 a 、b 、c 按升序排列的结果。
- 3. el if a > c: 这是第二层条件语句的另一个分支,如果 a 大于 c,表示 a、 c、b 已经按升序排列,print(a, c, b): 如果条件成立,即 a > b 且 c > b,那么这一行代码会打印 a、c、b 按升序排列的结果。
- 4. else:如果前两个条件都不成立,那么只有 c 大于 a 和 b,表示 c、a、b 已 经按升序排列, print(c, a, b):这一行代码会打印 c、a、b 按升序排列的结果。
- 5. 如果最外层的条件 if a > b 不成立,表示 a 不大于 b,那么进入下一个条件分支。
- 6. if(b>c):检查 b 是否大于 c, if (a > c):如果 b > c 且 a > c, 表示 b、a、c 已经按升序排列,print(b, a, c): 这一行代码会打印 b、a、c 按升序排列的结果。else:如果 a 不大于 c,表示 b、c、a 已经按升序排列,print(b, c, a),这一行代码会打印 b、c、a 按升序排列的结果。
- 7. 如果条件 if(b>c)不成立,表示 b 不大于 c,那么只有 c 大于 b,表示 c、b、a 已经按升序排列,print(c,b,a),这一行代码会打印 c、b、a 按升序排列的结果。

Test2:

代码包含 test2.1 和 test2.2 两部分,前者生成随机矩阵,后者进行矩阵乘法操作。

Test2.1:

Import random 引入随机数,紧接着定义 M1、M2 两个矩阵,行、列可进行 更改,from_value 到 to_value 表示生成随机数的范围,使用 random.randint 生成 两个随机矩阵,打印即可。

结果如下:

```
Matrix M1:

[48, 7, 49, 36, 50, 42, 42, 45, 1, 42]

[36, 23, 20, 46, 23, 26, 48, 10, 35, 35]

[8, 9, 11, 17, 1, 13, 23, 41, 41, 18]

[39, 1, 44, 5, 39, 39, 27, 1, 7, 41]

[10, 3, 0, 17, 24, 24, 24, 19, 12, 10]

Matrix M2:

[21, 24, 44, 30, 40]

[9, 38, 34, 50, 6]

[122, 7, 27, 37, 37]

[24, 10, 14, 35, 8]

[13, 4, 40, 12, 31]

[41, 7, 50, 8, 28]

[13, 46, 46, 3, 50]

[44, 45, 18, 9, 48]

[49, 4, 2, 22, 24]

[19, 49, 25, 21, 1]
```

打印结果为符合条件的矩阵,分别为 5 行 10 列和 10 行 5 列矩阵,且为 50 随机整数。

Test2.2:

首先定义一个函数,用于两个乘法操作,首先检查 M1、M2 矩阵是否符合 行列规范, 否则 error, 其次创建 result 矩阵, 大小为两矩阵乘积, 接着使用嵌 套循环执行乘法操作, 调用 M1、M2 相乘即可打印。

```
Matrix Multiplication Result:
[8758, 8634, 12071, 7234, 11115]
[7316, 7125, 9103, 6803, 7882]
[5899, 4825, 4211, 3526, 5432]
[5539, 5085, 8817, 4908, 7142]
[3867, 3285, 4660, 2242, 4380]
```

打印结果为上述两矩阵的乘积。

Test3:

代码解释:

定义 generate pascals triangle line(k) 函数:

这个函数接受一个整数 k 作为参数,表示生成帕斯卡三角形的第 k 行,初始化 line,并将其第一个元素设置为 1,使用一个循环来生成行中的每个元素,从 1 到 k。

在循环中,根据帕斯卡三角形的性质,计算下一个元素的值,通过将前一个元素乘以 (k-i+1) 再除以 i 得到,将计算出的元素值添加到 line 中。

Pascal triangle(k) 函数:

这个函数接受一个整数 k 作为参数,表示生成帕斯卡三角形的前 k 行。初始化一个空列表 triangle,用于存储帕斯卡三角形的前 k 行。使用一个循环来生成前 k 行的三角形,通过调用第一个定义函数,将每一行添加到 triangle 列表中。

部分打印结果如下:

```
Pascal's Triangle (Line 100): [1, 99, 4851, 156849, 3764376, 71523144, 1120529256, 14887031544, 171200862756, 1731030435644, 15579278510796, 126050526132804, 924370524973896, 6186171974825304, 380009770702498296, 121337706647499344, 1130522928399924309. 551961194453774949, 25144888884503380806, 1071986740807619394, 428786696523047746376, 1613054714739084379224, 5719612170448571889976, 19186758135816888501224, 6062981743084286233947746376, 181899452296252849761628, 1613054714739084379224, 5719612170448571889976, 19186758135816888501224, 606298174308428623876, 181889452296252849761628, 1613054714739084379224, 5719612170448571889976, 19186758135816888501224, 606298174308428623876, 181889452296252849761628, 1613054714739084379224, 571960405172758, 19186745897713761376747492761851747824, 16130547613761376747824, 1613054761376747489778180461727589, 1918674597477492912, 5498404051378546147787489971713056892173792912, 5498404051386, 284774084784186869007604212, 2154618614907180180585724688, 381543037171350589212739142860023940955992606928, 384747408478418686900760451282, 379665591677748987781805625128, 95074339029409985557897390672, 473794892781805625128, 9507433902940998555789739672, 47379488978196325128, 950743390294098555789739672, 47379488978196325128, 950743390294098555789739672, 47379488778196325128, 950743390239409855578739672, 473794887781895236128, 2757437748987781895236128, 275743497848891784891784891784891784891784918491849785388, 22477404874818686000760421832, 549849365893211246005606947888, 38374339819459499145744993104592795771324404878488, 12545400957599260927, 475764000431735762419272568, 26566637875127678957610497757, 197443926105102399225573693, 972485009174889514095427977785976995757258, 383727536913787010251407757, 197443926105102399225573693, 97248500917488951409546767, 13996697579757482, 15901279448571849785786914778488, 12554406478488, 12554406478548, 12554406478548, 12554406478548, 12554406478548, 12554406478548, 12554406478548, 12554406478548, 12554406478548, 12554406478548, 12554406478548, 125
```

Test4:

代码解释:

这段代码定义了一个函数 least_moves(x),其目的是计算将某个金额 x 的人民币变为 1 所需的最小操作步数。这个函数通过模拟将 x 转换为 1 的过程来实现,每一步可以进行两种操作:如果 x 是偶数,则将其翻倍;如果 x 是奇数,则将其减一。使用 while 循环,只要当前金额小于目标金额,就会继续循环,当当前金额不再小于 x,结束循环。

打印结果如下:

```
...:
...: # 示例用法:
...: x1 = 2
...: x2 = 5
...:
...: print(f"Least moves to reach {x1} RMB: {least_moves(x1)}")
...: print(f"Least moves to reach {x2} RMB: {least_moves(x2)}")
Least moves to reach 2 RMB: 1
Least moves to reach 5 RMB: 3
```

Test5:

这段代码的目标是寻找从数字1到9组成的表达式,通过添加加法和减法运算符,使得表达式的结果等于给定的目标值。它还计算了在目标范围内有多少种不同的解决方案,以及找到解决方案数量的最小和最大值

origin=['1','2','3','4','5','6','7','8','9']: 定义了一个初始数字列表,从1到9。

str2int(list,i,j) 函数用于将从 i 到 j 范围内的字符转换为整数。

ret 初始化为 0, 然后遍历列表中的字符并将它们转换为整数,返回最终整数值。

cal(list) 函数计算给定表达式列表的结果,使用 + 和 - 运算符,遍历表达式列表,执行相应的操作,最后返回结果。

number, data, 和 result 是列表,用于存储数字,表达式列表,和它们的计算结果。

insert(list,idx,type) 函数根据给定的索引位置 idx 和操作类型 type (0表示无操作,1表示加法,2表示减法),将操作插入到表达式列表中。

使用递归方式生成所有可能的表达式。

bti() 函数:初始化 number 列表,将所有可能的数字组合的值存储在其中。

使用 insert 函数生成所有可能的表达式,并将它们存储在 data 和 result 列表中。

Find expression(x) 函数: 寻找结果等于 x 的所有表达式。

调用 bti() 函数,生成所有可能的表达式,并找到结果等于 50 的表达式。

遍历结果列表 Total_solutions,它存储了不同结果值对应的解决方案数量。

计算解决方案数量的最小值 min 和最大值 max,以及具有最小和最大解决方案数量的目标值,打印具有最小、大的解决方案数量的目标值。

打印结果如下:

```
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
88
1 45
```