1.
首先随机生成三个随机整数 a、b 和 c，然后定义了一个名为 Print_values(a, b, c)的函数。在函数中，使用 if…else 结构来执行流程图中的内容。最后，调用函数 Print_values(a, b, c)输出这些值的不同排列顺序。

2.（2.2 询问过龙诗倩如何使用 for 循环准确表示出矩阵中的其中一行或一列）
首先生成了两个随机的5×10和10×5的矩阵M1和M2，然后定义了一个函数Matrix_multip()来执行矩阵乘法运算。在函数中创建一个 5×5 的全零矩阵并储存在 result 中，并使用三个嵌套的循环来计算矩阵乘积并将结果存储在 result 中。最后，它将结果转换为整数类型并返回。最后，调用 Matrix_multip()函数来执行矩阵乘法计算，并将结果输出。

3.（第二个循环中，triangle[i-1][j-1] + triangle[i-1][j]最开始的行列没弄明白，运行不出结果，咨询了计算机系的同学了解了错误）
首先定义了一个名为 Pascal_triangle(k)的函数，用于输出帕斯卡三角形的第 k 行。在函数中，创建了一个空列表 triangle 来储存帕斯卡三角形的行。然后通过外层循环 for i in range(k) 遍历从 0 到 k-1 的范围，表示要生成的行数。在每一行的生成过程中，内层的循环 for j in range(i+1)遍历从 0 到 i 的范围，表示当前行的元素个数。在内层循环中，使用条件语句判断当前元素的位置。如果 j 等于 0 或者 j 等于 i，即在当前行的开头或结尾位置，将值设为 1。对于其他位置的元素，使用递推关系 triangle[i-1][j-1] + triangle[i-1][j]计算它的值。这个递推关系表示当前位置的值等于上一行相邻两个元素的和。生成完一行的元素后，将该行添加到 triangle 列表中。重复这个过程，直到生成了 k 行。接下来，通过条件语句 `if k == 100 or k == 200` 判断当前行的行号是否为 100 或 200。如果是，则执行 `print(triangle[k-1])` 语句，输出该行的元素列表。最后，调用函数 Pascal_triangle(100)和 Pascal_triangle(200)分别输出帕斯卡三角形的第 100 行和第 200 行。

4.
初始化步数计数器 count 为 0，初始化 RMB 的值为 1。生成一个介于 1 和 100（包含 100）之间的随机整数 x。定义一个名为 Least_moves(x)的函数。在函数中进行 whlie 循环，当 RMB 的值小于 x 时，执行循环体内的代码。使用一个 if...else 结构判断 RMB 是翻倍还是增加 1，然后每执行一次循环，步数计数器 count 增加 1，之后输出步数 count。最后，调用函数 Least_moves(x)函数，输出步数计数器 count。

5.（chaptgpt 帮助我理清这题的思路和改正自己写的代码中的错误，并且学习了如何使用 matplotlib 库绘图）
首先定义一个函数 Find_expression(target, current='', total=0, idx=0)，其中目标值 target、当前表达式 current、当前总和 total 和当前数字索引 idx。在函数中，定义一个字符串 nums，首先判断当前数字索引 idx 是否等于字符串长度 len(nums)，并且当前总和 total 是否等于目标值 target。使用 for 循环遍历从 idx+1 到 len(nums)+1 的范围，将字符串切片转换为整数 num。然后，如果 idx 等于 0，表示当前数字是表达式的第一个数字，则调用 Find_expression 函数，并更新 current 为当前数字的字符串形式，total 为当前数字的值，idx 为下一个数字的索引。如果 idx 不等于 0，表示当前数字不是第一个数字，那么调用 Find_expression 函数两次。一次是将当前数字添加到 current 后面，并将 total 增加当前数字的值，另一次是将当前数字添加到 current 后面，并将 total 减去当前数字的值。同时，更新 idx 为下一个数字的索引。最后，调用 Find_expression(i)来演示函数的使用，其中 i 是从 1 到 101 之间的随机整数。这将

生成并输出所有等于随机整数的数学表达式。
首先定义一个函数 Find_expression_count(target, current='', total=0, idx=0)，初始化步数计数器 count 为 0，首先判断当前数字索引是否等于字符串长度，并且当前总和是否等于目标值，若是，则将 count 加 1。使用 for 循环遍历从 idx+1 到 len(nums)+1 的范围，将字符串切片转换为整数 num。如果 idx 等于 0，表示当前数字是表达式的第一个数字，调用 Find_expression_count 函数，并将解决方案数量累加到 count 上；如果 idx 不等于 0，表示当前数字不是第一个数字，两次调用 Find_expression_count，第一次调用是将当前数字添加到当前表达式 current 后面，并将当前总和 total 增加当前数字的值 num。第二次调用是将当前数字添加到当前表达式 current 后面，并将当前总和 total 减去当前数字的值 num。然后，将这两次递归调用的解决方案数量累加到 count 上。第一次调用是将当前数字添加到当前表达式 current 后面，并将当前总和 total 增加当前数字的值 num。第二次调用是将当前数字添加到当前表达式 current 后面，并将当前总和 total 减去当前数字的值 num。然后，将这两次递归调用的解决方案数量累加到 count 上。然后，函数返回累计的解决方案数量 count。最后，调用 Find_expression_count(n)来计算从 1 到 100 之间每个数字的解决方案数量，并将结果存储在 Total_solutions 列表中。使用 matplotlib 库绘制了一个折线图，横轴表示目标数值，纵轴表示解决方案的数量。通过 plt.plot()函数将目标数值和对应的解决方案数量传递给 plot 函数，并使用 plt.xlabel()和 plt.ylabel()函数设置横轴和纵轴的标签。最后，使用 plt.show()函数显示绘制的折线图。通过使用 max()和 min()函数找到解决方案数量的最大值和最小值，以及它们在 Total_solutions 列表中的索引。通过列表推导式，找到所有等于最大值和最小值的目标数值的索引，并将结果存储在 max_numbers_indices 和 min_numbers_indices 列表中。最后，输出最大解决方案数量和对应的目标数值索引，以及最小解决方案数量和对应的目标数值索引。