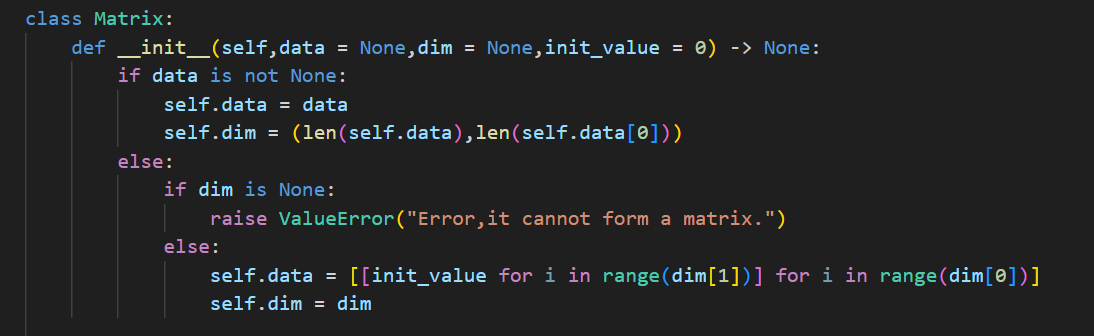
**技术分析报告**

**项目名称：构造一个仿照numpy库的mini矩阵运算库**

**项目参与成员：李青雅，陈卓妍，侯湘贻**

**PART ONE: 代码设计说明**

****

对class中的变量进行初始化：

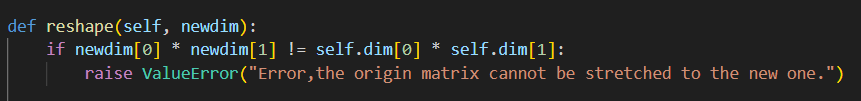
如果data存在，则矩阵类型为data，dim为data维数。

如果data不存在，检查dim是否存在。若dim存在，则以dim为维数，init\_value为值，填充一个矩阵；若dim不存在，则抛出ValueError

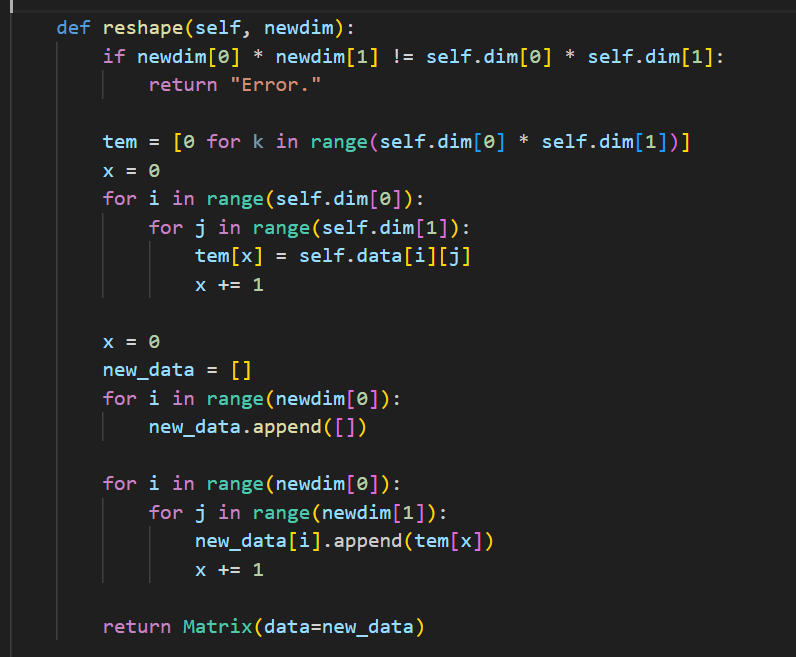
**（1）函数1**：shape:返回矩阵形状



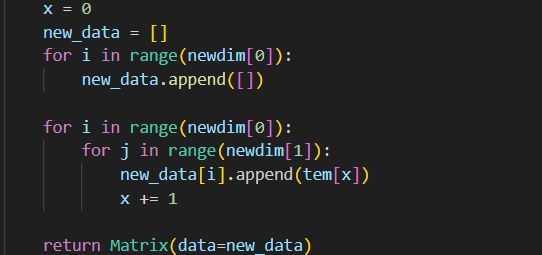
**（2）函数2**：reshape：将矩阵从（m，n）维拉伸到（m1，n1）维



首先判断拉伸操作是否合法

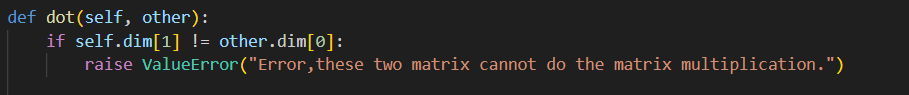


其次将原二维数组存储到一个一维列表中去

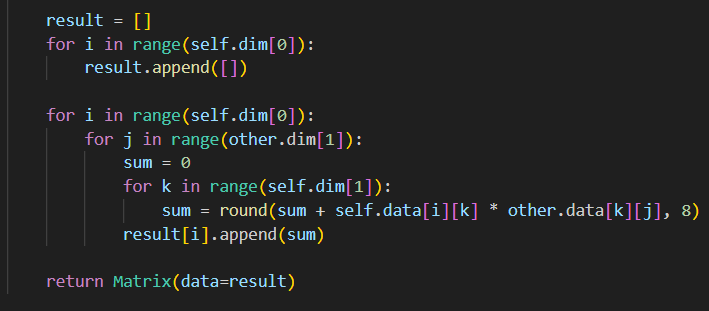


最后将列表中的数据输入到新的二维数组中

**（3）函数3**：dot：矩阵乘法

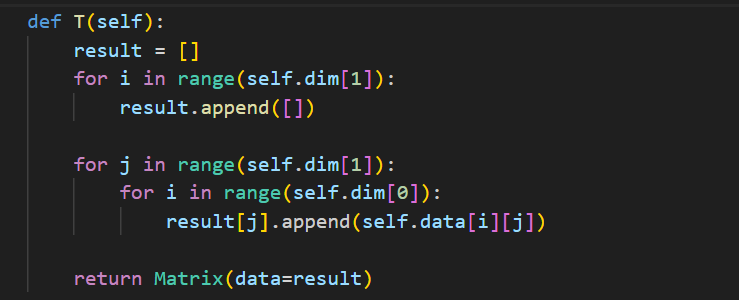


首先判断是否符合乘法的条件

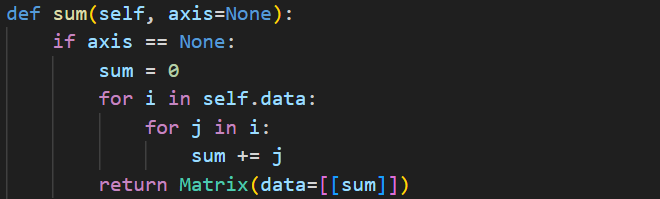


其次遍历二位列表进行矩阵乘法

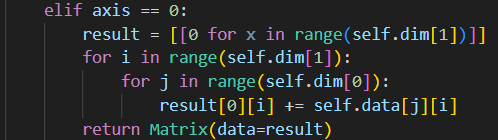
**（4）函数4**：T：转置



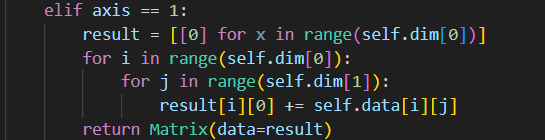
**（5）函数5**：sum：根据制定坐标轴对矩阵进行求和



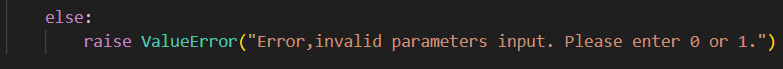
axis = None 表示对矩阵全部元素进行求和，得到形状为 (1, 1) 的矩阵



axis = 0 表示对矩阵进行按列求和，得到形状为 (1, self.dim[1]) 的矩阵

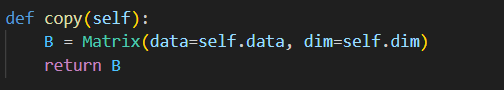


axis = 1 表示对矩阵进行按行求和，得到形状为 (self.dim[0], 1) 的矩阵

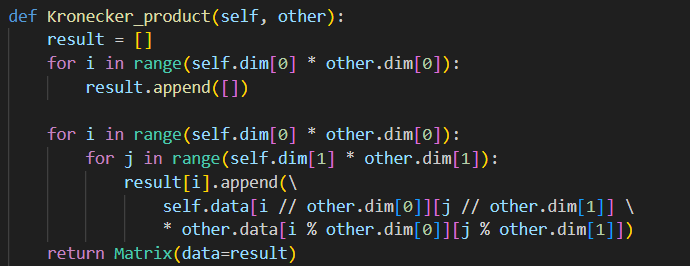


若输入的axis不是0、1、none，则抛出ValueError

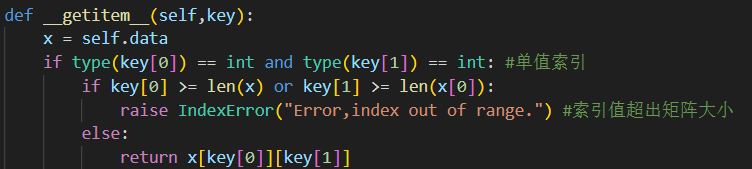
**（6）函数6**：copy



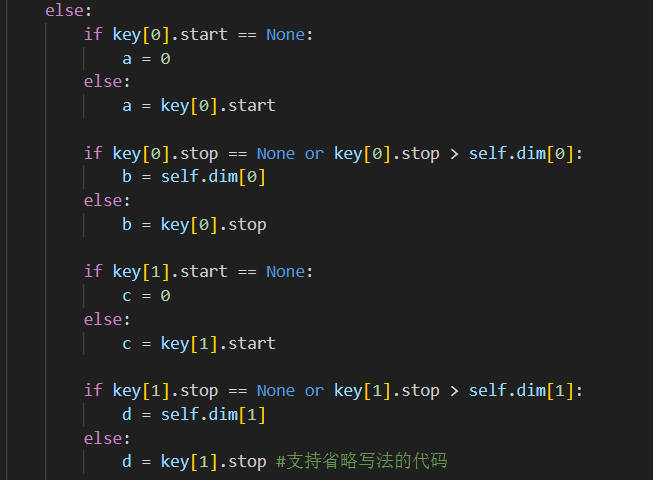
**（7）函数7**：Kronecker\_product



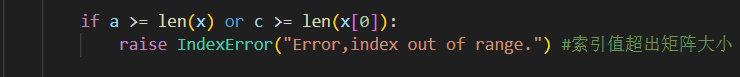
**(8) 函数8**：\_\_getitem\_\_



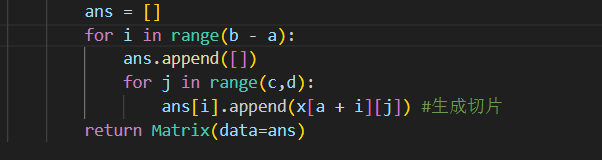
第一种情况：单值索引



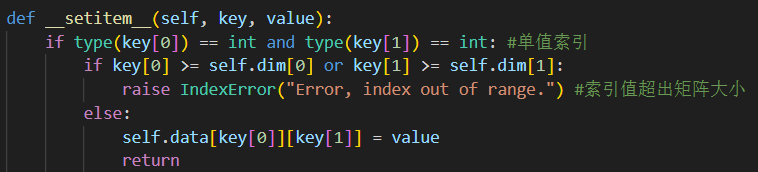
第二种情况：矩阵切片



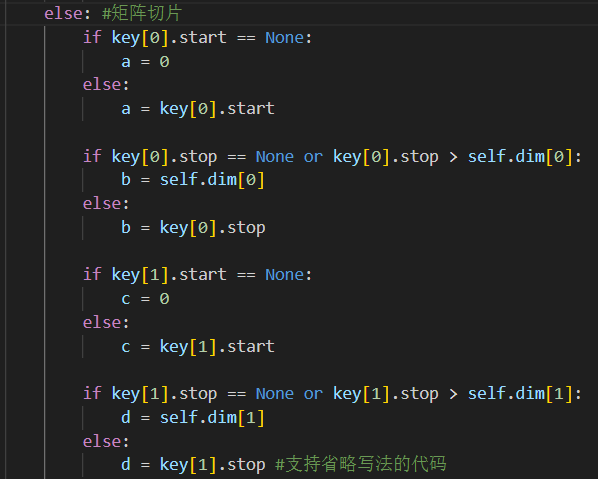
当索引值超出矩阵大小，抛出ValueError



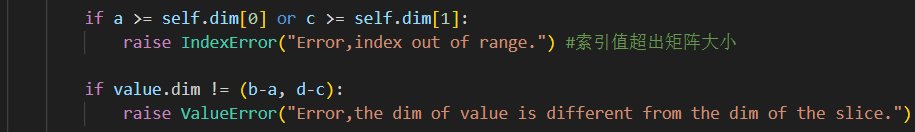
**（9）函数9**：\_\_setitem\_\_



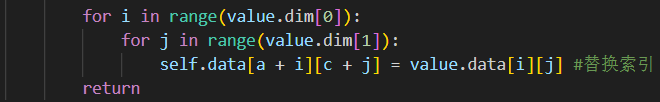
第一种情况：单个元素赋值



第二种情况：矩阵赋值：

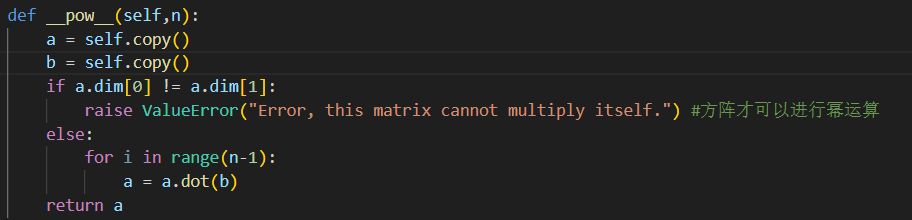


当索引值超出矩阵大小或者索引值与切片大小不匹配时，抛出ValueError

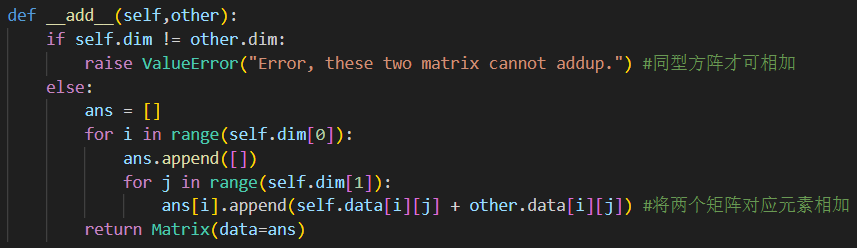


第三步操作将矩阵赋值

**(10) 函数10**：\_\_pow\_\_

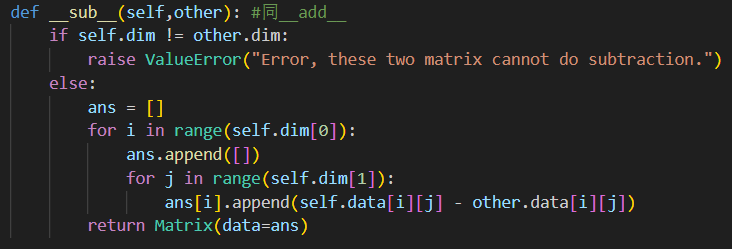


**（11）函数11：\_\_add\_\_**



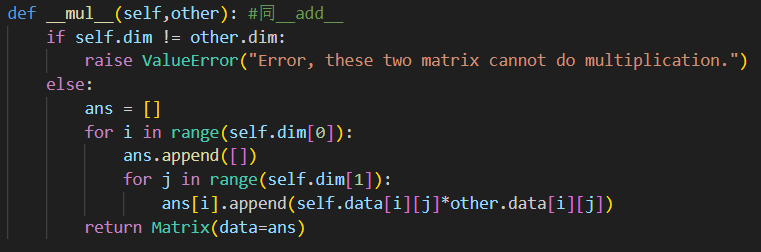
首先判断矩阵是否符合各个元素相加的条件，若不符合，则抛出ValueError；若符合，则对应元素相加。

**（12）函数12：\_\_sub\_\_**



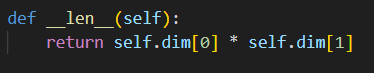
首先判断矩阵是否符合各个元素相减的条件，若不符合，则抛出ValueError；若符合，则对应元素相减。

**（13）函数13：**\_\_mul\_\_

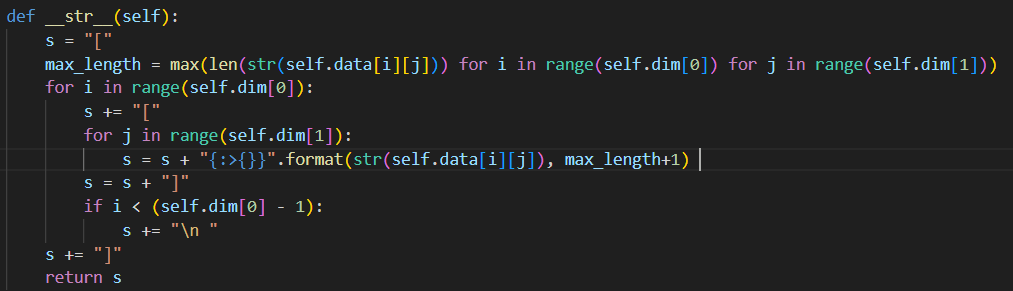


首先判断矩阵是否符合各个元素相乘的条件，若不符合，则抛出ValueError；若符合，则对应元素相乘。

**（14）函数14**：\_\_len\_\_ 返回矩阵元素的数目

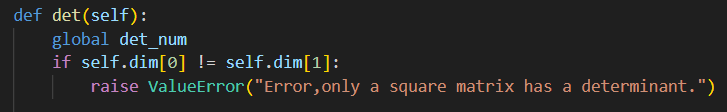


**（15）函数15：\_\_str\_\_** 将矩阵以字符串的形式返回输出

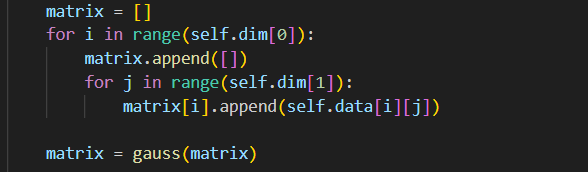


重点：去除多余的“，”并将元素右对齐

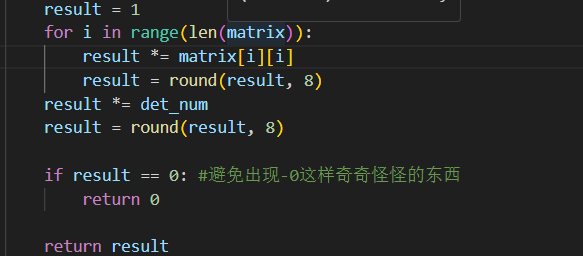
**(16) 函数16：**det 计算矩阵行列式



首先判断矩阵有没有行列式（即是否是方阵），并引入全局变量det\_num记录行变换中行列式的变化

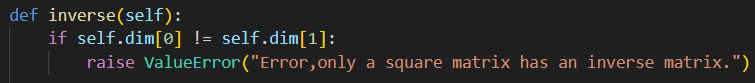


接着创建一个和self.data相同的二维数组

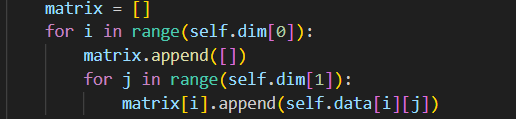


然后对matrix进行高斯消元[1]，并计算简化阶梯性的行列式（即对角线元素乘积），最后乘以det\_num计算原矩阵行列式，输出结果。

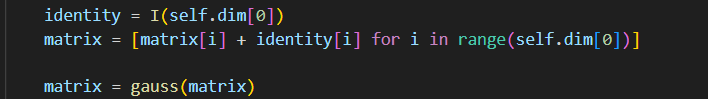
1. **函数17：**inverse 计算逆矩阵



首先判断可逆条件1：矩阵是否是方阵



接着创建一个和self.data相同的二维数组



然后再矩阵右边添加一个单位矩阵，并通过初等行变换把左边矩阵变为单位矩阵

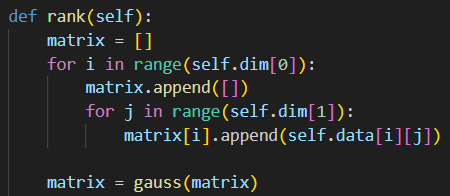


判断矩阵可逆条件2：行列式不为零。当左边无法化到单位矩阵时，说明原矩阵行列式为0，原矩阵不可逆。

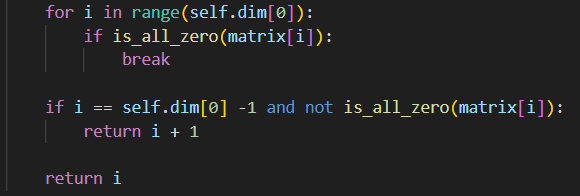


输出行变化后右侧矩阵，即为逆矩阵

1. **函数18：**rank 计算矩阵的秩

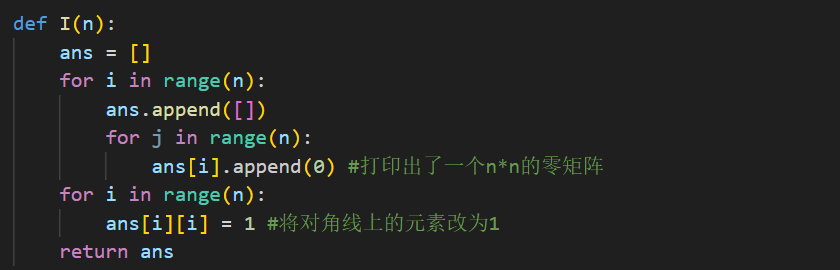


首先创建一个等于self.data的矩阵matrix, 并进行高斯消元

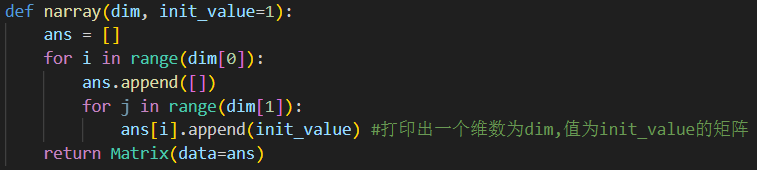


清点非零行数，输出结果

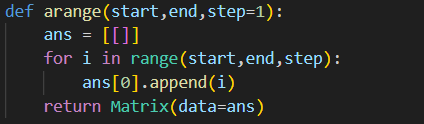
**（19）函数19**：I：返回一个单位矩阵



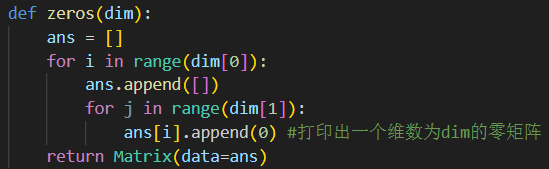
**（20）函数20：**narray 返回一个matrix，维数为dim，初始值为init\_value



**（21）函数21**：arrange：返回一个1\*n 的narray，其中的元素类同 range(start, end, step)



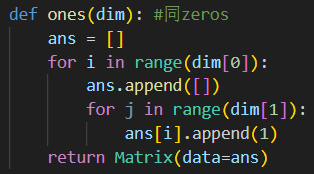
**（22）函数22**：zeros返回一个维数为dim 的全0 narray



**（23）函数23**：zeros\_like：返回一个形状和matrix一样的全0 narray



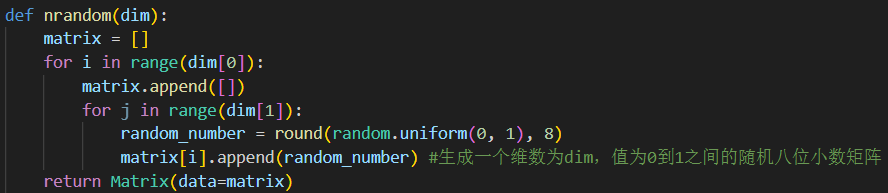
**（24）函数24 ：**ones



**（25）函数25：**ones\_like



**（26）函数26**：nrandom

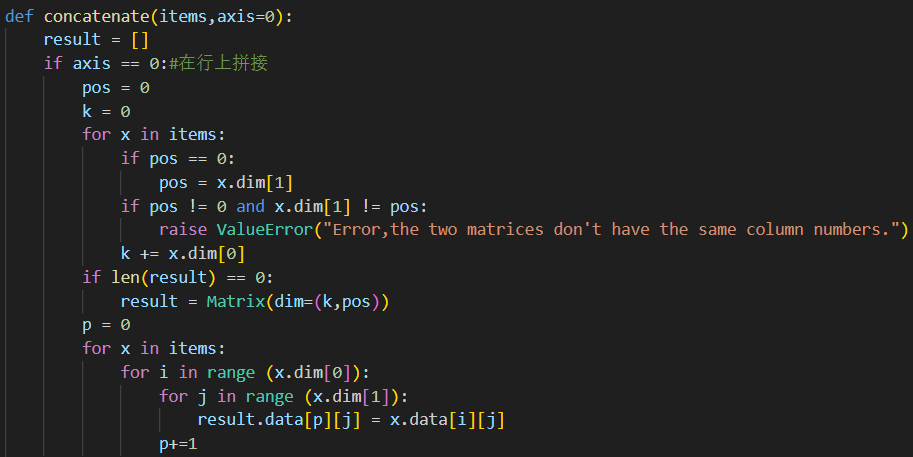


主要操作同zero和one，引入random，输入（0,1）内的任意值

**（27）函数27**：nrandomlike



**（28）函数28**：concatenate 将若干矩阵按照指定的方向拼接起来，若给定的输入在形状上不对应，应抛出异常,且该函数应当不改变 items 中的元素。

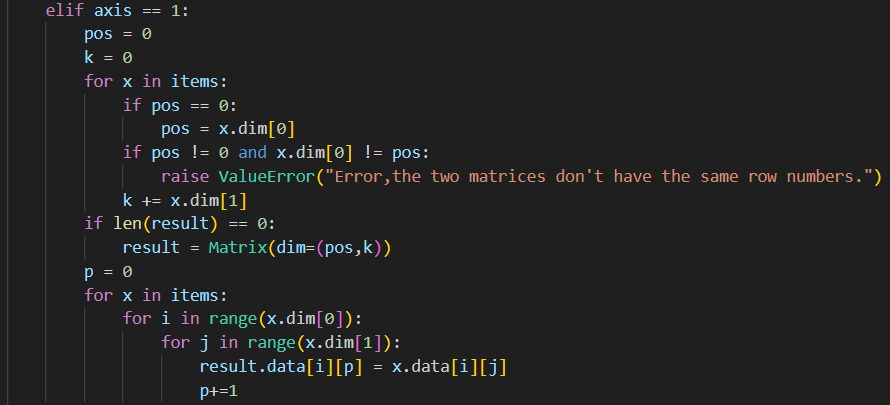


第一种情况：axis=0 在行上拼接

首先：判断矩阵是否匹配并计算出匹配后矩阵的长宽：k，pos

其次：将结果矩阵初始化

最后：执行操作拼接

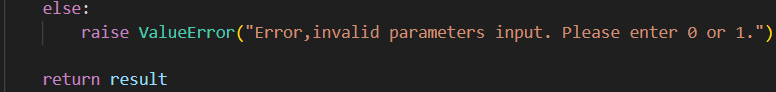


第一种情况：axis=1 在列上拼接

首先：判断矩阵是否匹配并计算出匹配后矩阵的长宽：pos，k

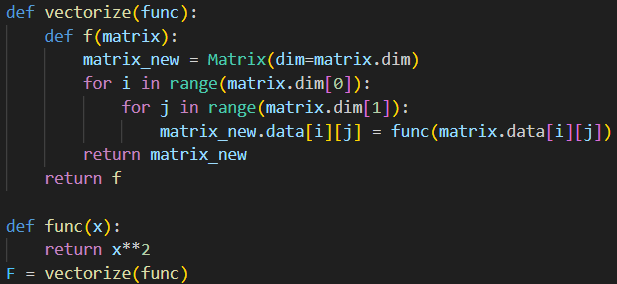
其次：将结果矩阵初始化

最后：执行操作拼接



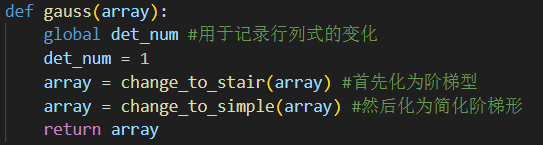
axis参数输入错误，抛出ValueError

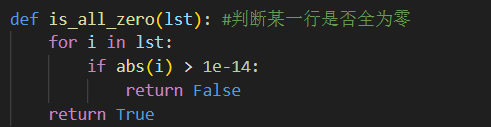
**（29）函数29：**vectorize 将给定函数进行向量化

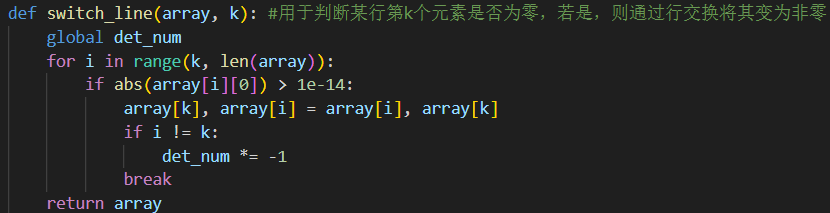


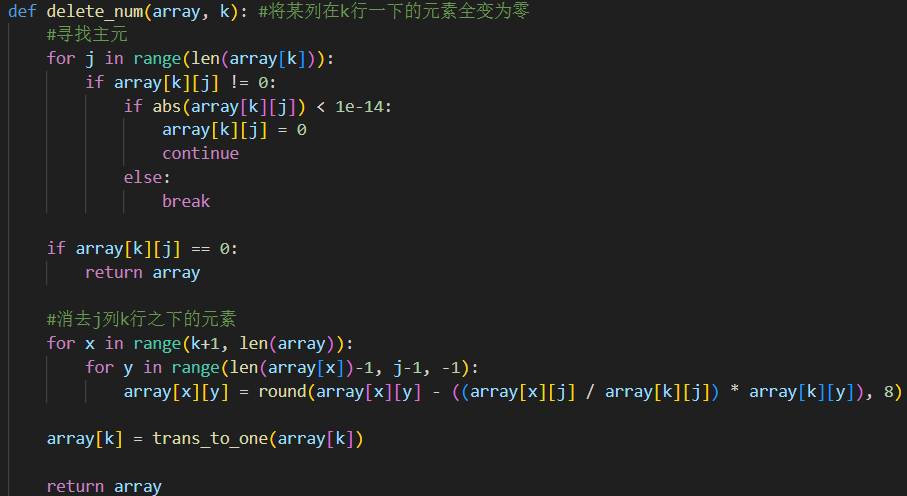
根据输入的func函数对矩阵执行操作并将操作记录为函数输出，所得结果为一个对矩阵进行操作的函数

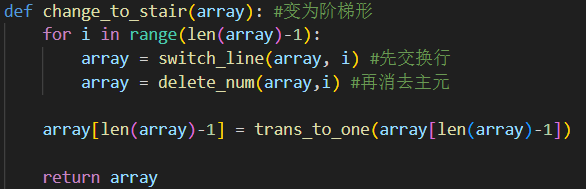
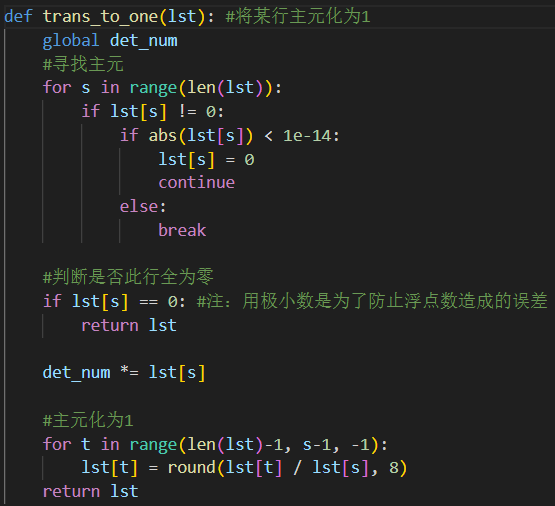
注【1】：高斯消元系列函数

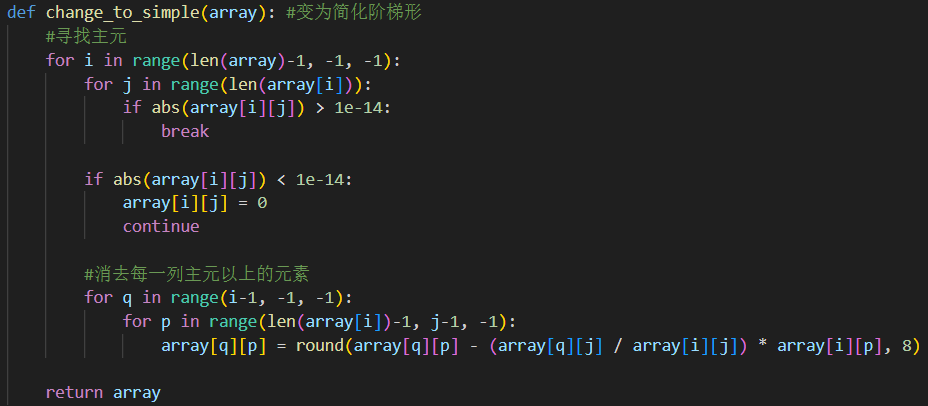






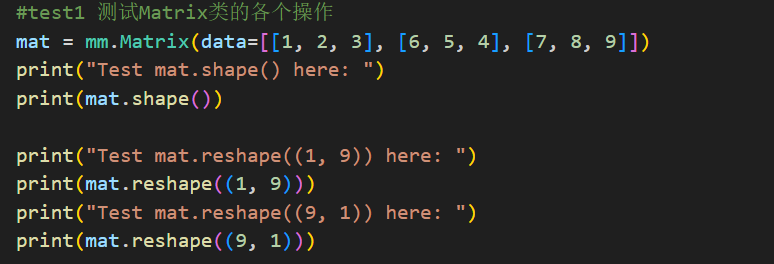


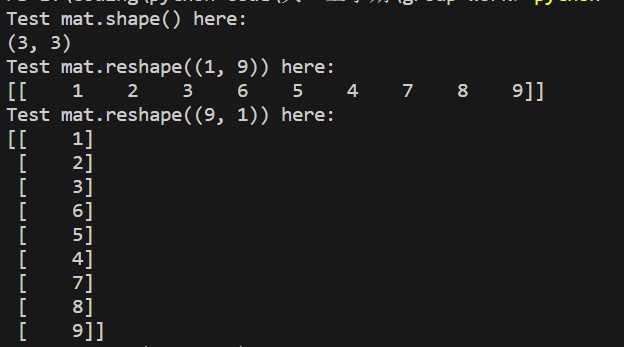


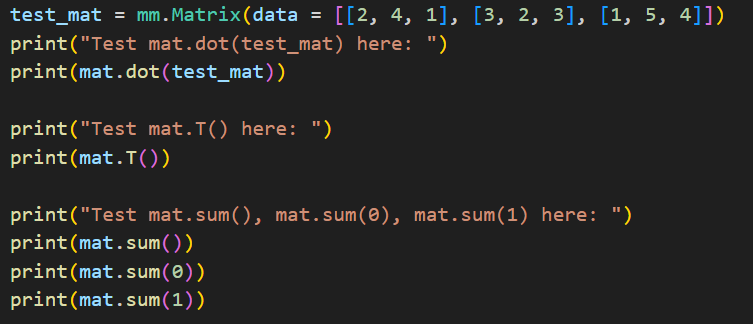


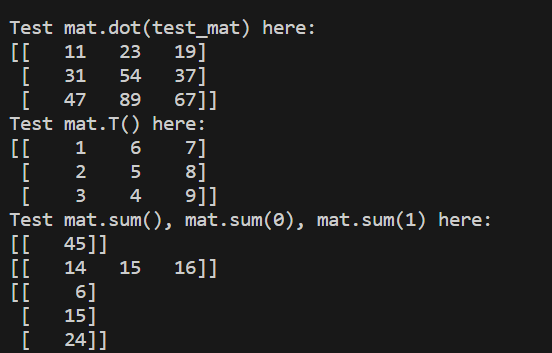
**PART TWO 执行结果测试**

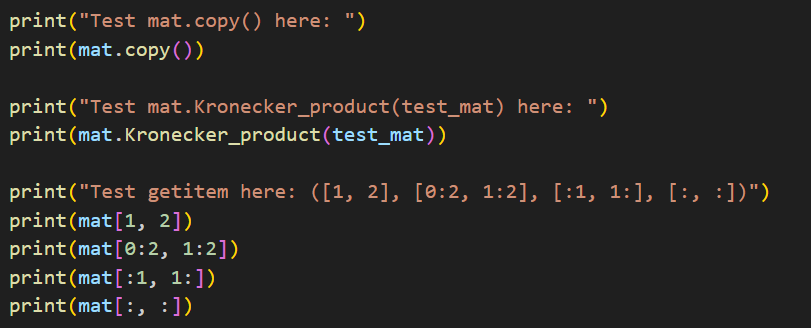
**Test1:测试Matrix 类的各个操作**

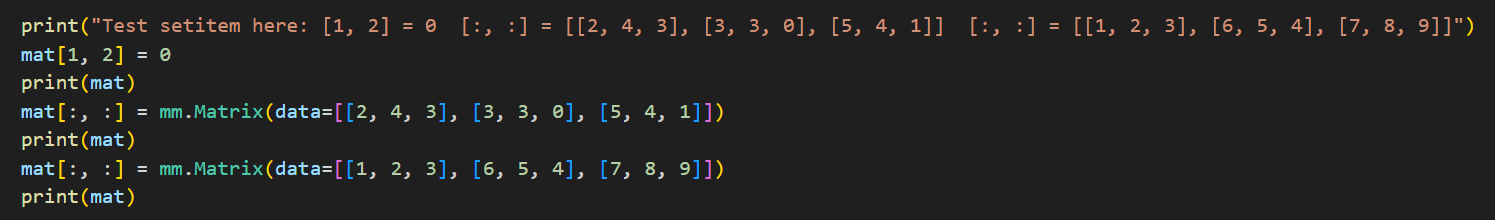
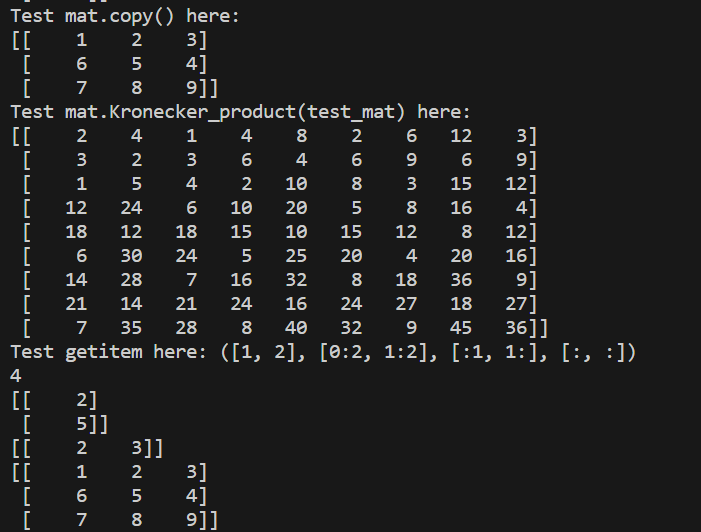
****

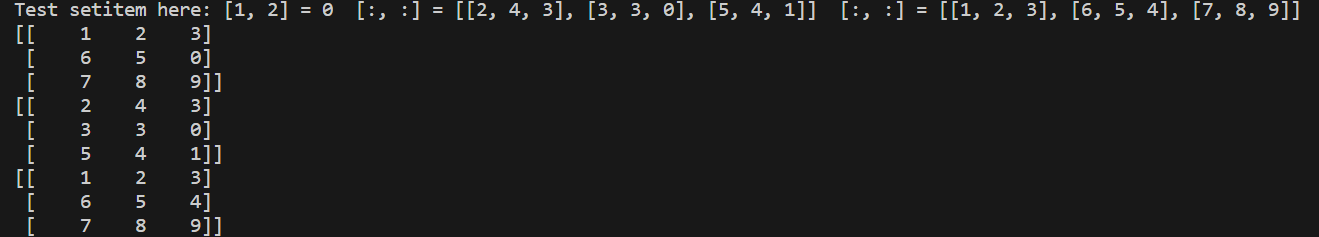
****

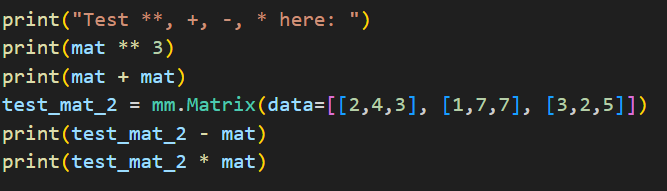
****

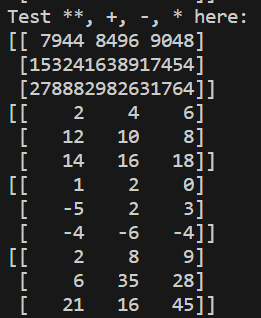
****

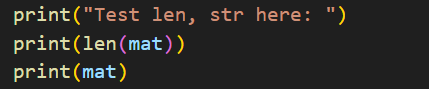
****

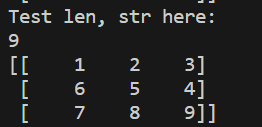
****

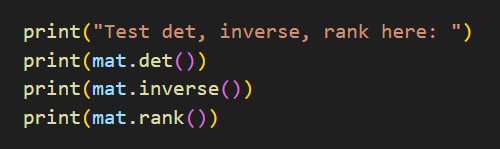
****

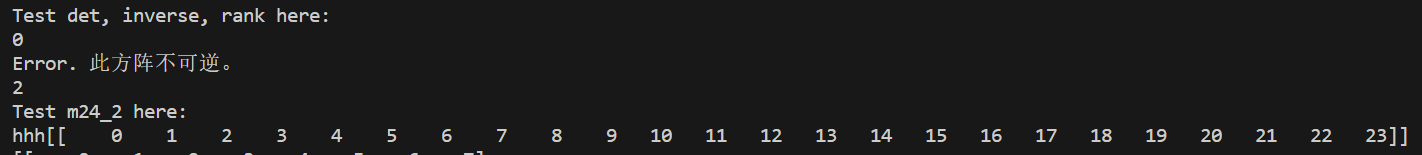
****

****

****

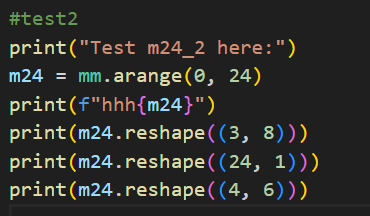
****

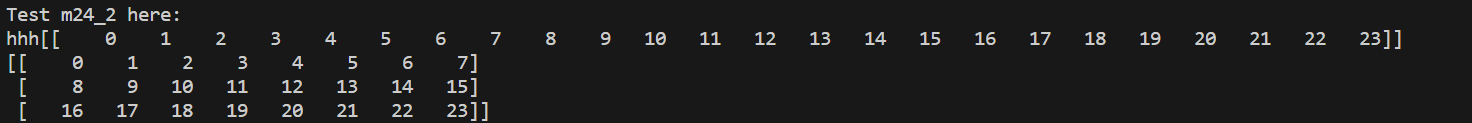
****

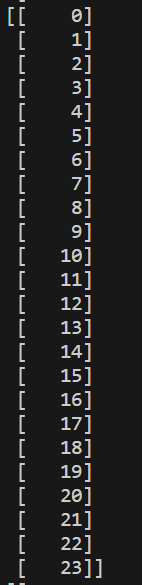
****

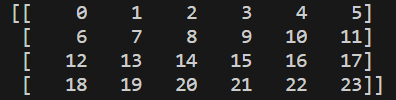
**经过测验test1结果均正确**

**Test2：通过arange()函数生成0到24（不包括24）的维矩阵m24，测试reshape([3,8]), reshape([24,1]), reshape([4,6])，并输出。**

****

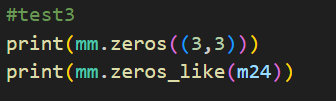
****

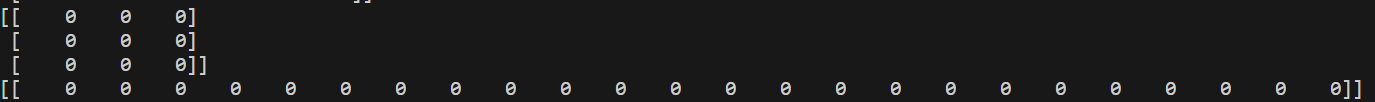
****

****

**经过测验test2结果均正确**

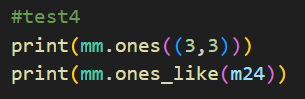
**Test3：测试zeros(), 生成一个维全0矩阵的并输出。测试zeros\_like(m24)**

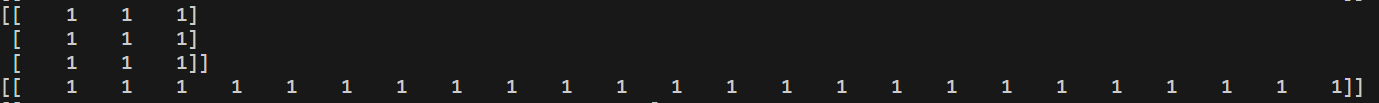
****

****

**经过测验test3结果均正确**

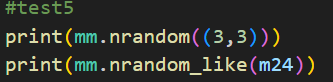
**Test 4：测试ones(), 生成一个维全1矩阵的并输出。测试ones\_like(m24)**

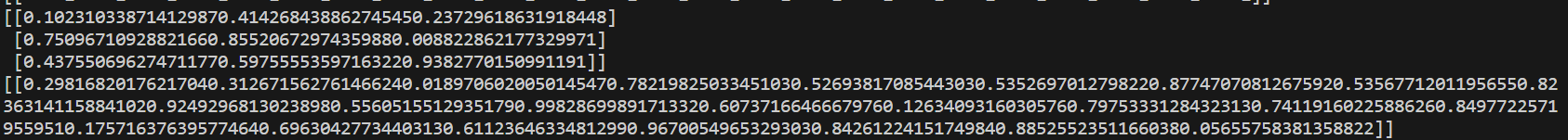
****

****

**经过测验test4结果均正确**

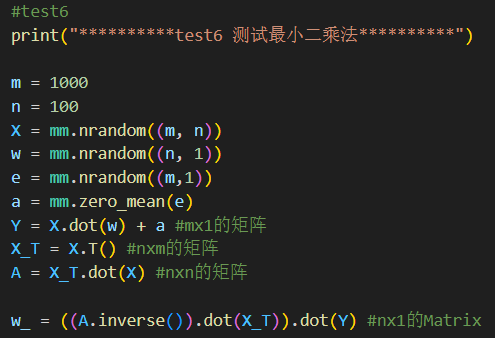
**Test5：测试nrandom(), 生成一个维的随机矩阵并输出。测试nrandom\_like(m24)**

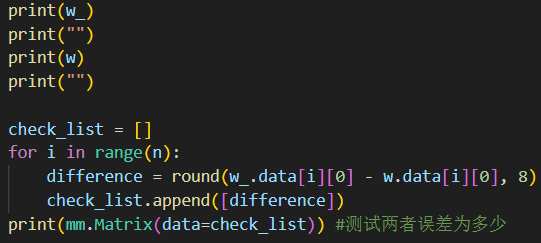
****

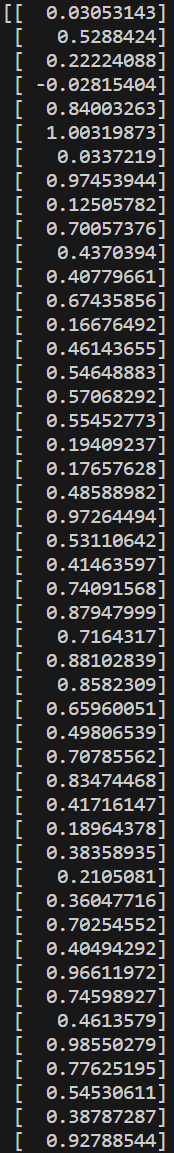
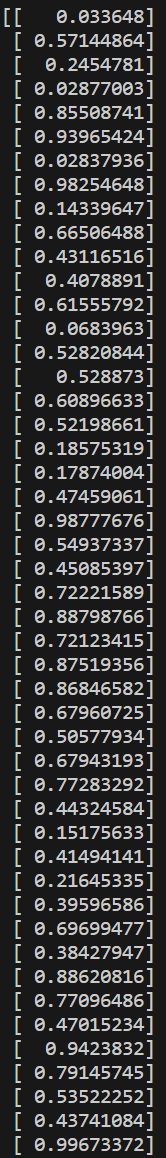
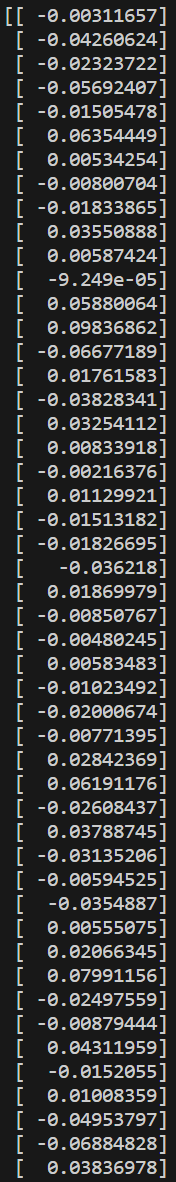
****

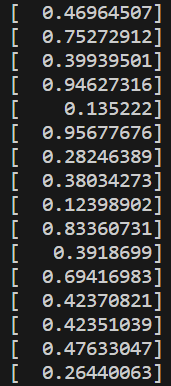
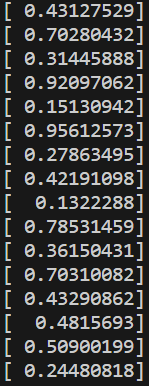
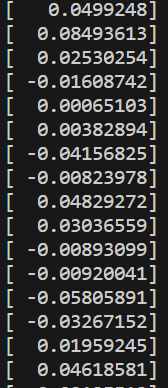
**经过测验test5结果均正确**

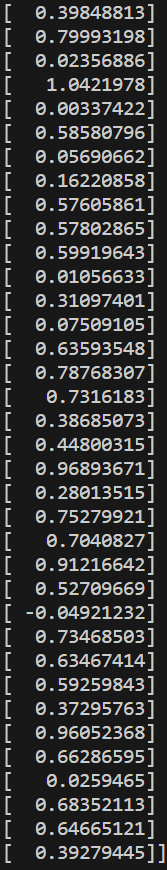
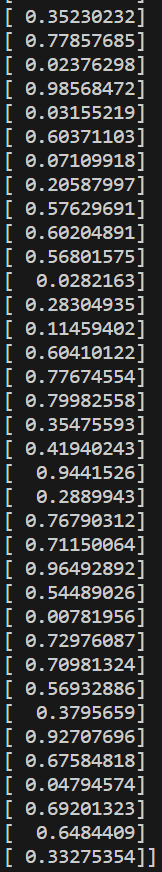
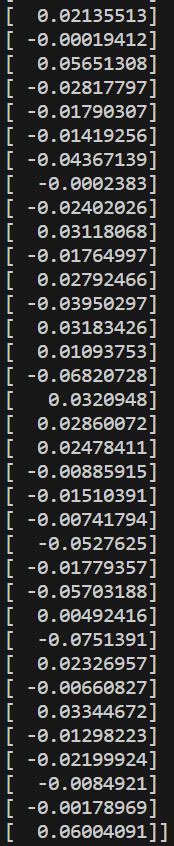
**Test6：测试使用自定义的Matrix类解决最小二乘法问题：利用nrandom() 生成 的随机矩阵 以及 的随机向量 ，并生成的零均值的随机向量。计算得到 。利用自己实现的矩阵乘法、矩阵求逆等功能计算最小二乘法的估计  , 并与 比较。测试中取 m=1000, n=100.**





****

****

****