一、 程序说明（也可见 Readme.txt）

功能：1.求方阵 A 的特征值

2.对方阵 A 进行 LU 分解，对矩阵 A 进行 QR（Gram-Schmidt）、Householder 和 Givens 分解，并在此基础上求 Ax=b 的解

3.对矩阵 A 进行 URV 分解

语言：python

依赖库：numpy

1.factorization.py

pp_lu 函数：采用的方法为部分主元 LU 分解，其中输入为 A 矩阵（要求 A 为方阵），xrow 为某一列绝对值最大元素所在行，输出为 P 为行变换矩阵，L 为对角线为 1 的下三角矩阵，U 为上三角矩阵。

qr_gs 函数：gram-schmidt 法将 A 矩阵变为单位正交矩阵 Q 左乘上三角阵 R，其中 r 为 A 矩阵某一列向量在之前已经确定好的方向向量上的投影，然后该列向量减去在已经确定好的方向上的投影向量，anorm 表示新的正交方向的向量的模长，q 为单位正交向量，输出 Q 为单位正交矩阵，R 为上三角矩阵。

householder 函数：输入 A 为一个矩阵，verbose 控制是否打印 P 矩阵或 Q 矩阵，a 为 A 矩阵位于对角线下方的列向量，e 为和 a 长度相同，首个元素为 a 的模长，其余元素为 0 的向量，输出 P 矩阵和 T 矩阵使得 PA=T，当 A 为方阵时，T 为上三角矩阵，P 为正交矩阵。

givens 函数：输入 A 为一个矩阵，r 和 c 为矩阵主对角线下部分的元素位置，s 为主对角线上的元素与其下行中非零元素的平方和，主对角线上的元素除 s 为旋转角度的 cos 值，其下非零元素除 s 为旋转角度的 sin 值，输出 P 为正交矩阵，使得 PA=T，当 A 为方阵时，T 为上三角矩阵。

urv 函数：输入 A 为一个矩阵，P 为 A 做 householder reduction 的一个正交矩阵，B 为使得 PA=B，取 B 矩阵的前非零行，Q 为做 householder reduction 的一个正交矩阵，使得 QB.T=T，Q 为一正交矩阵，取 T 的前非零行作为 T，最后输出 U 为 P.T，V 为 Q.T，R 为和 A 矩阵相同大小，左上角为 T.T，其余元素为 0 的矩阵。

factorization 函数：输入 A 矩阵，输入矩阵分解方法，输出为对应的分解矩阵。

2.solution.py

sub_matrix 函数：输入为 A 为一个方阵，r 和 c 分别为行和列的位置，verbose 控制是否打印子矩阵，输出 SM 矩阵为去掉 r 行和 c 列的一个子矩阵。

det 函数：输入 A 为一个方阵，verbose 控制是否打印行列式值，对于第一行每

个元素求代数余子式，并对乘积求和，输出 det_value 为行列式值

ly_pb 函数：输入 L 为一个下三角阵，b 为一个长度等于 L 行数的列向量，P 默认值为单位阵，当作 PA=LU 分解时，可以输入 P 矩阵，按行从上到下依次消去，最后输出为列向量 y。

ux_y 函数：输入 U 为一个下三角阵，y 为一个长度等于 U 行数的列向量，按行从下到上依次消去，最后输出为列向量 x。

qr_resolution 函数：输入 Q 和 R 为经过 gram-Schmidt 分解得到的矩阵，b 为等于 Q 的行数的列向量，把 R 的前非零行赋值给 R，c 为 Q.T 左乘 b，令 c 的元素个数等于 R 的行数，R 为一个上三角阵，c 为等于 R 行数的列向量，把 R 和 c 输入 ux_y 函数，得到 x 为 Ax=b 的解。

householder_givens_solution 函数：输入为经过 householder reduction 或 givens reduction 后的 P 和 T 矩阵，b 为一列向量，令 R 为 T 前非零行矩阵，c 为 P 左乘 b，令 c 元素个数等于 R 的行数，R 为一上三角阵，把 R，c 输入 ux_y 函数，得到 x 为 Ax=b 的解。

solution 函数：输入 A 为一个矩阵，method 为矩阵分解方法，b 为一个列向量，如果 A 为方阵，打印特征值，输出相应的分解矩阵和 Ax=b 的解向量 x。

二、 例子

1.对方阵 A 进行 LU 分解并求解 Ax=b

```
#partial pivot LU reduction
A = np.array([[1,2,-3,4],
              [4,8,12,-8],
              [2,3,2,1],
              [-3,-1,1,-4]])
b = np.array([[3],
              [60],
              [1],
              [5]])
slt.solution(A,'LU',b)
```

```
det(A):
 120.0
P:
 [[0. 1. 0. 0.]
 [0. 0. 0. 1.]
 [1. 0. 0. 0.]
 [0. 0. 1. 0.]]
L:
 [[ 1.     0.     0.     0.   ]
 [-0.75   1.     0.     0.   ]
 [ 0.25   0.     1.     0.   ]
 [ 0.5   -0.2    0.333  1.   ]]
U:
 [[  4.   8.  12.  -8.]
 [  0.   5.  10. -10.]
 [  0.   0.  -6.   6.]
 [  0.   0.   0.   1.]]
x:
 [[ 12.]
 [  6.]
 [-13.]
 [-15.]]
```

2.对矩阵 A 进行 QR 分解（Gram-Schmidt）并求 Ax=b 的解

```
#Gram-Schmidt solution
A = np.array([[2,4],
              [3,-5],
              [1,2],
              [2,1]])
b = np.array([[11],
              [3],
              [6],
              [7]])
slt.solution(A,'GS',b)
```

```
A不是方阵，没有特征值
Q:
 [[ 0.471  0.642]
 [ 0.707 -0.667]
 [ 0.236  0.321]
 [ 0.471  0.198]]
R:
 [[ 4.243 -0.707]
 [ 0.     6.745]]
x:
 [[3.04 ]
 [1.242]]
```

3.对矩阵 A 进行 Householder reduction 并求 Ax=b 的解

```
#householder reduction
A = np.array([[4,-3,4],
              [2,-14,-3],
              [-2,14,0],
              [1,-7,15]])
b = np.array([[5],
              [-15],
              [0],
              [30]])
slt.solution(A,'Householder',b)
```

```
A不是方阵，没有特征值
P:
 [[ 0.8    0.4   -0.4    0.2  ]
 [ 0.6   -0.533  0.533 -0.267]
 [ 0.    -0.333  0.133  0.933]
 [ 0.    -0.667 -0.733 -0.133]]
T:
 [[  5. -15.   5.]
 [ -0.  15.  -0.]
 [  0.   0.  15.]
 [ -0.   0.   0.]]
x:
 [[-0.8]
 [ 0.2]
 [ 2.2]]
```

4.对矩阵 A 进行 Givens reduction 并求 Ax=b 的解

```
#Givens reduction solution
A = np.array([[4,-3,4],
              [2,-14,-3],
              [-2,14,0],
              [1,-7,15]])
b = np.array([[5],
              [-15],
              [0],
              [30]])
slt.solution(A,'Givens',b)
```

```
A不是方阵，没有特征值
P:
 [[ 0.8    0.4   -0.4    0.2  ]
 [ 0.6   -0.533  0.533 -0.267]
 [ 0.    -0.333  0.133  0.933]
 [-0.     0.667  0.733  0.133]]
T:
 [[  5. -15.   5.]
 [  0.  15.  -0.]
 [ -0.   0.  15.]
 [ -0.  -0.   0.]]
x:
 [[-0.8]
 [ 0.2]
 [ 2.2]]
```

5.对矩阵 A 进行 URV reduction

```
#URV reduction
A = np.array([[-4,-2,-4,-2],
              [2,-2,2,1],
              [-4,1,-4,-2]])
fc.factorization(A,'URV')
```

```
U:
 [[-0.667 -0.667 -0.333]
 [ 0.333 -0.667  0.667]
 [-0.667  0.333  0.667]]
R:
 [[ 9. -0.  0.  0.]
 [-0.  3.  0.  0.]
 [ 0.  0.  0.  0.]]
V:
 [[ 0.667  0.    -0.667 -0.333]
 [ 0.     1.     0.     0.   ]
 [ 0.667 -0.     0.733 -0.133]
 [ 0.333 -0.    -0.133  0.933]]
```