```
------------------------------#1 solve_quadratic_equation-----------------------------
------------------1-------------------
a=1, b=0, c=-1
(1.0, -1.0)
------------------2-------------------
a=1, b=0, c=1
n real No
0
------------------3-------------------
a=0, b=1, c=-1          -5
float division by zero
------------------4-------------------
a=1, b=-2, c=1          -5
n real No
0
------------------5-------------------
a=1.2, b=-2.5, c=0
(2.0833333333333335, 0.0)
--------------------------------#2 array35--------------------------------
(array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
       18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
       35]), array([[ 1,  2,  3,  4,  5,  6,  7],
       [ 8,  9, 10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19, 20, 21],
       [22, 23, 24, 25, 26, 27, 28],
       [29, 30, 31, 32, 33, 34, 35]]), array([[ 1,  6, 11, 16, 21, 26, 31],
       [ 2,  7, 12, 17, 22, 27, 32],
       [ 3,  8, 13, 18, 23, 28, 33],
       [ 4,  9, 14, 19, 24, 29, 34],
       [ 5, 10, 15, 20, 25, 30, 35]]))
------------------------------#3 solve_linear_system------------------------------
------------------6-------------------
A
[[1 2]
 [3 4]]
b
[[1]
 [1]]
solution
[[-1.]
 [ 1.]]
------------------7-------------------
A
[[1 2 0]
 [4 5 6]
 [3 7 8]]
b
[[1]
 [1]
 [1]]
solution
[[ 0.2]
 [ 0.4]
 [-0.3]]
```

```
import numpy as np
def solve_quadratic_equation(a,b,c):
    sqrt=b**2-4*a*c
    if sqrt>0:
        x1=(-b+sqrt**(1/2))/(2*a)
        x2=(-b-sqrt**(1/2))/(2*a)
        return x1,x2
    else:
        print('n real No ')
        return 0
def array35():
    v = np.arange(1,36)
    rows = v.reshape((5, 7))
    b = v.reshape((35,1))
    columns = b.reshape(7,5).swapaxes(0,1)
    return v, rows, columns
def solve_linear_system(matrix1, matrix2):
    return np.linalg.solve(matrix1,matrix2)
```