In [48]:

```
print("Summation of 1st row: ")
print(np.sum(X[0,0:]))
print("Summation of 2nd row: ")
print("Summation of 3rd row: ")
print("Summation of 4th row: ")
print("Summation of 5th row: ")
print("Summation of 6th row: ")
```

```
Summation of 1st row:
21
Summation of 2nd row:
Summation of 3rd row:
Summation of 4th row:
Summation of 5th row:
Summation of 6th row:
```

K.

In [44]:

```
print("The sum of all the elements in the matrix is: ")
np.sum(X[:,:])
```

```
The sum of all the elements in the matrix is:
```

Out[44]:

666

24/06/19

In [4]:

```
import sympy as sp
```

In [16]:

```
x = np.array([[1,2,-1],[2,1,4],[3,3,4]]) - 1
y = np.array([[1,2,-1],[2,1,5],[3,3,4]]) - 2
z = np.array([[1,2],[2,4]]) -3
```

In [18]:

```
print(np.linalg.solve(x,y))
```

```
[[ -9. -10.  -3.]
 [  5.   6.   1.]
 [  3.   3.   2.]]
```

01/07/2019 (LAB 2)

1.

In [15]:

```
import numpy as np
a = np.array([[1,1,-2],[2,-3,1],[3,1,4]])
b = np.array([1,-8,7])
np.linalg.solve(a,b)
```

Out[15]:

```
array([ -2.96059473e-16,   3.00000000e+00,   1.00000000e+00])
```

2.

In [3]:

```
a = np.array([[1,2,-1],[2,1,4],[3,3,4]])
b = np.array([1,2,1])
np.linalg.solve(a,b)
```

Out[3]:

```
array([ 7., -4., -2.])
```

3.

In [16]:

```
a = np.array([[1,-1,1,-1],[1,-1,1,1],[4,-4,4],[-2,2,-2,1]])
b = np.array([2,0,4,-3])
np.linalg.solve(a,b)
# The matrix is singular and the determinant is 0.
```

```
---------------------------------------------------------------------------
LinAlgError                               Traceback (most recent call last)
<ipython-input-16-dda4fb9f66d2> in <module>()
      1 a = np.array([[1,-1,1,-1],[1,-1,1,1],[4,-4,4],[-2,2,-2,1]])
      2 b = np.array([2,0,4,-3])
----> 3 np.linalg.solve(a,b)
      4 # The matrix is singular and the determinant is 0.

C:\Users\Jeevan\Anaconda3\lib\site-packages\numpy\linalg\linalg.py in solve
(a, b)
    355     """
    356     a, _ = _makearray(a)
--> 357     _assertRankAtLeast2(a)
    358     _assertNdSquareness(a)
    359     b, wrap = _makearray(b)

C:\Users\Jeevan\Anaconda3\lib\site-packages\numpy\linalg\linalg.py in _asser
tRankAtLeast2(*arrays)
    200         if len(a.shape) < 2:
    201             raise LinAlgError('%d-dimensional array given. Array mus
t be '
--> 202                     'at least two-dimensional' % len(a.shape))
    203
    204 def _assertSquareness(*arrays):

LinAlgError: 1-dimensional array given. Array must be at least two-dimension
al
```

4.

In [13]:

```
a = np.array([[2,1],[4,2]])
b = np.array([3,6])
np.linalg.solve(a,b)
# The matrix form is singular and hence cannot be solved
```

```
---------------------------------------------------------------------------
LinAlgError                               Traceback (most recent call last)
<ipython-input-13-5eb615042530> in <module>()
      1 a = np.array([[2,1],[4,2]])
      2 b = np.array([3,6])
----> 3 np.linalg.solve(a,b)
      4 # The matrix form is singular and hence cannot be solved

C:\Users\Jeevan\Anaconda3\lib\site-packages\numpy\linalg\linalg.py in solve
(a, b)
    382     signature = 'DD->D' if isComplexType(t) else 'dd->d'
    383     extobj = get_linalg_error_extobj(_raise_linalgerror_singular)
--> 384     r = gufunc(a, b, signature=signature, extobj=extobj)
    385
    386     return wrap(r.astype(result_t, copy=False))

C:\Users\Jeevan\Anaconda3\lib\site-packages\numpy\linalg\linalg.py in _raise
_linalgerror_singular(err, flag)
     88
     89 def _raise_linalgerror_singular(err, flag):
---> 90     raise LinAlgError("Singular matrix")
     91
     92 def _raise_linalgerror_nonposdef(err, flag):

LinAlgError: Singular matrix
```

5.

In [6]:

```
a = np.array([[1,2,-1],[2,1,5],[3,3,4]])
b = np.array([1,2,1])
np.linalg.solve(a,b)
```

Out[6]:

```
array([ -1.04293886e+16,   6.63688366e+15,   2.84437871e+15])
```

6.

In [7]:

```
a = np.array([[2,1,-2],[1,-3,1],[4,-1,-2]])
b = np.array([-3,8,3])
np.linalg.solve(a,b)
```

Out[7]:

```
array([ 2., -1.,  3.])
```

7.

In [14]:

```
a = np.array([[2,1],[2,-1],[1,-2]])
b = np.array([[3,0,4]])
np.linalg.solve(a,b)
# Since the matrix is square matrix it cannot be solved
```

```
---------------------------------------------------------------------------
LinAlgError                               Traceback (most recent call last)
<ipython-input-14-7460f17ef1cf> in <module>()
      1 a = np.array([[2,1],[2,-1],[1,-2]])
      2 b = np.array([[3,0,4]])
----> 3 np.linalg.solve(a,b)
      4 # Since the matrix is square matrix it cannot be solved

C:\Users\Jeevan\Anaconda3\lib\site-packages\numpy\linalg\linalg.py in solve
(a, b)
    356     a, _ = _makearray(a)
    357     _assertRankAtLeast2(a)
--> 358     _assertNdSquareness(a)
    359     b, wrap = _makearray(b)
    360     t, result_t = _commonType(a, b)

C:\Users\Jeevan\Anaconda3\lib\site-packages\numpy\linalg\linalg.py in _asser
tNdSquareness(*arrays)
    210     for a in arrays:
    211         if max(a.shape[-2:]) != min(a.shape[-2:]):
--> 212             raise LinAlgError('Last 2 dimensions of the array must b
e square')
    213
    214 def _assertFinite(*arrays):

LinAlgError: Last 2 dimensions of the array must be square
```

8.

In [11]:

```
a = np.array([[2,1,-2],[3,2,2],[5,4,3]])
b = np.array([10,1,4])
np.linalg.solve(a,b)
```

Out[11]:

```
array([ 1.,  2., -3.])
```

9.

In [12]:

```
a = np.array([[1,2,-3],[3,-1,2],[5,3,-4]])
b = np.array([-1,7,2])
np.linalg.solve(a,b)
```

Out[12]:

```
array([  8.77324603e+14,  -9.65057063e+15,  -6.14127222e+15])
```