```
In [1]: import numpy as np

In [2]: # Now we will look at some commonly used functions in numpy

In [3]: # Arrays
        # asarray() assigns data with same structure
        a = np.asarray([1,2,3], dtype=np.float32)
        print(a[1],a[2])

2.0 3.0


In [4]: # linspace() creates sequences
        a=np.linspace(0, 10, num=5)
        a

Out[4]: array([  0. ,   2.5,   5. ,   7.5,  10. ])

In [5]: # Some basic numerical operations/functions
        np.mean(a)

Out[5]: 5.0

In [6]: np.var(a)

Out[6]: 12.5

In [7]: np.exp(1)

Out[7]: 2.718281828459045

In [8]: np.sin(np.pi/2)

Out[8]: 1.0

In [9]: np.cumsum(a)

Out[9]: array([  0. ,   2.5,   7.5,  15. ,  25. ])

In [10]: # Linear algebra
         # To construct a matrix
         m=np.matrix([[1,2],[3,4]])
         m

Out[10]: matrix([[1, 2],
                 [3, 4]])

In [11]: # Linear algebra is inside package linalg
         from numpy import linalg

In [12]: # Determinant of matrix
         linalg.det(m)
```

```
Out[12]: -2.0000000000000004

In [13]: # Invert of matrix
         linalg.inv(m)

Out[13]: matrix([[-2. ,  1. ],
                 [ 1.5, -0.5]])

In [14]: # Matrix production
         linalg.inv(m)*m
         # notice that different from R, * here is not element-by-element production

Out[14]: matrix([[  1.00000000e+00,   4.44089210e-16],
                 [  0.00000000e+00,   1.00000000e+00]])

In [15]: # Element-by-element multiplication
         np.multiply(linalg.inv(m),m)

Out[15]: matrix([[-2. ,  2. ],
                 [ 4.5, -2. ]])

In [16]: # eigenvalues and eigenvectors
         linalg.eig(m)

Out[16]: (array([-0.37228132,  5.37228132]), matrix([[-0.82456484, -0.41597356],
                  [ 0.56576746, -0.90937671]]))
```