NumPy

Numpy is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely. NumPy stands for Numerical Python.

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

Creating Arrays

```
In [2]: a=np.array([10,20,30])
        print(a)

[10 20 30]

In [3]: b=np.array([(1.1,6,7.7),(7,9,2)],dtype=float)
        print(b)

[[1.1 6.  7.7]
 [7.  9.  2. ]]

In [4]: c=np.array([illegible],dtype=float)
        print(c)

[illegible]

In [5]: #Create an array of zeros
        np.zeros((3,4))

Out[5]: array([[0., 0., 0., 0.],
               [0., 0., 0., 0.],
               [0., 0., 0., 0.]])

In [6]: #Create an array of ones
        np.ones((2,3,4))

Out[6]: array([[[1., 1., 1., 1.],
                [1., 1., 1., 1.],
                [1., 1., 1., 1.]],

               [[1., 1., 1., 1.],
                [1., 1., 1., 1.],
                [1., 1., 1., 1.]]])

In [7]: #Evenly spaced Values(step value)
        d=np.arange(10,25,5)
        print(d)

[10 15 20]

In [8]: #Evenly spaced Values(number of samples)
        np.linspace(0,2,9)

Out[8]: array([0.  , 0.25, 0.5 , 0.75, 1.  , 1.25, 1.5 , 1.75, 2.  ])

In [9]: #Constant array
        e=np.full((3,3),9)
        print(e)

[[9 9 9]
 [9 9 9]
 [9 9 9]]

In [10]: #Identity matrix
         f=np.eye(3)
         print(f)

[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]

In [11]: #Random values
         np.random.random((3,2))

Out[11]: [illegible]
```

Data Types

np.int64 np.[illegible] np.complex np.bool np.object np.string_ np.unicode

```
In [12]: #Array dimensions
         a.shape

Out[12]: (3,)

In [13]: #Array length
         len(a)

Out[13]: 3

In [14]: #Number of Array Dimensions
         b.ndim

Out[14]: 2

In [15]: #Number of Array Elements
         e.size

Out[15]: 9

In [16]: #DataType Name
         b.dtype

Out[16]: dtype('float64')

In [17]: b.dtype.name

Out[17]: 'float64'

In [18]: #Array Conversion
         b.astype(int)

Out[18]: array([[1, 6, 7],
                [7, 9, 2]])
```

Airthmetic Operations

```
In [19]: #Addition
         g=a+b
         print(g)

[[11.1 26.  37.7]
 [17.  29.  32. ]]

In [20]: #Subtraction
         g=a-b
         print(g)

[[ 8.9 14.  22.3]
 [ 3.  11.  28. ]]

In [21]: #Multiplication
         g=a*b
         print(g)

[[ 11. 120. 231.]
 [ 70. 180.  60.]]

In [22]: #Division
         g=a/b
         print(g)

[[ 9.09090909  3.33333333  3.8961039 ]
 [ 1.42857143  2.22222222 15.        ]]

In [23]: np.subtract(a,b)

Out[23]: array([[ 8.9, 14. , 22.3],
                [ 3. , 11. , 28. ]])

In [24]: np.add(a,b)

Out[24]: array([[11.1, 26. , 37.7],
                [17. , 29. , 32. ]])

In [25]: np.divide(a,b)

Out[25]: array([[ 9.09090909,  3.33333333,  3.8961039 ],
                [ 1.42857143,  2.22222222, 15.        ]])

In [26]: np.multiply(a,b)

Out[26]: array([[ 11., 120., 231.],
                [ 70., 180.,  60.]])

In [27]: #Exponentiation
         np.exp(b)

Out[27]: array([[3.00416602e+00, 4.03428793e+02, 2.20834799e+03],
                [1.09663316e+03, 8.10308393e+03, 7.38905610e+00]])

In [28]: #Square Root
         np.sqrt(b)

Out[28]: array([[1.04880885, 2.44948974, 2.77488739],
                [2.64575131, 3.        , 1.41421356]])

In [29]: #Array Sines
         np.sin(a)

Out[29]: array([-0.54402111,  0.91294525, -0.98803162])

In [30]: #Cosines
         np.cos(b)

Out[30]: array([[ 0.45359612,  0.96017029,  0.15337386],
                [ 0.75390225, -0.91113026, -0.41614684]])

In [31]: #Logrithmic
         np.log(a)

Out[31]: array([2.30258509, 2.99573227, 3.40119738])

In [32]: #Dot Product
         e.dot(f)

Out[32]: array([[9., 9., 9.],
                [9., 9., 9.],
                [9., 9., 9.]])
```

Comparison

```
In [33]: a==b

Out[33]: array([[False, False, False],
                [False, False, False]])

In [34]: b==c

Out[34]: array([[[False,  True, False],
                 [False, False, False]],

                [[False, False, False],
                 [False, False, False]]])

In [35]: #Element wise comparison
         a<2

Out[35]: array([False, False, False])

In [36]: #Array wise comparison
         np.array_equal(e,f)

Out[36]: False
```

Aggregate Functions

```
In [37]: #Array Wise Sum
         a.sum()

Out[37]: 60

In [38]: #Array Wise Minimun Value
         a.min()

Out[38]: 10

In [39]: #Array Wise Maximum Value
         a.max()

Out[39]: 30

In [40]: c.sum()

Out[40]: 53.7

In [41]: b.max(axis=0)

Out[41]: array([7. , 9. , 7.7])

In [42]: #Cumulative Sum of the elements
         b.cumsum(axis=1)

Out[42]: array([[ 1.1,  7.1, 14.8],
                [ 7. , 16. , 18. ]])

In [43]: #Mean
         d.mean()

Out[43]: 15.0

In [44]: #Median
         np.median(b)

Out[44]: 6.5

In [45]: #Correlation Coefficient
         np.corrcoef(a)

Out[45]: 1.0

In [46]: #Standard Deviation
         np.std(d)

Out[46]: 4.08248290463863
```

Copy Array

```
In [47]: #View of Array with Same data
         h=a.view()

In [48]: #Copy of Array
         np.copy(a)

Out[48]: array([10, 20, 30])

In [49]: #Deep Copy of Array
         h=a.copy()

In [50]: print(h)

[10 20 30]
```

Sorting

```
In [51]: #Sort an Array
         b.sort()

In [52]: print(b)

[[1.1 6.  7.7]
 [2.  7.  9. ]]

In [53]: c.sort(axis=0)

In [54]: print(c)

[[[1.7 2.  6. ]
  [1.  2.  3. ]]

 [[6.  6.  9. ]
  [6.  8.  3. ]]]
```

Subsetting

```
In [55]: #Select the element at the 2nd index
         a[2]

Out[55]: 30

In [56]: #Select the element at row 1 column 2
         b[1,2]

Out[56]: 9.0
```

Slicing

```
In [57]: #Select items at rows 0 and 1
         a[0:2]

Out[57]: array([10, 20])

In [58]: #Select items at rows 0 and 1 in column 1
         b[0:2,1]

Out[58]: array([6., 7.])

In [59]: #Select all items at row 0
         d[:1]

Out[59]: array([10])

In [60]: c[1,...]

Out[60]: array([[6., 6., 9.],
                [6., 8., 3.]])

In [61]: #Reversed Array
         a[: :-1]

Out[61]: array([30, 20, 10])
```

Boolean Indexing

```
In [62]: #Select elements from a less than 2
         a[a<20]

Out[62]: array([10])

In [63]: b[b<2]

Out[63]: array([1.1])

In [64]: #Select elements (1,0),(0,1),(1,2) and (0,0)
         b[[1,0,1,0],[0,1,2,0]]

Out[64]: array([2. , 6. , 9. , 1.1])

In [65]: #Select Subset of Matrix
         b[[1,0,1,0]][:,[0,1,2,0]]

Out[65]: array([[2. , 7. , 9. , 2. ],
                [1.1, 6. , 7.7, 1.1],
                [2. , 7. , 9. , 2. ],
                [1.1, 6. , 7.7, 1.1]])
```

Changing Array Shape

```
In [66]: #Flatten the Array
         b.ravel()

Out[66]: array([1.1, 6. , 7.7, 2. , 7. , 9. ])

In [67]: #Reshape
         g.reshape(3,-2)

Out[67]: array([[ 9.09090909,  3.33333333],
                [ 3.8961039 ,  1.42857143],
                [ 2.22222222, 15.        ]])
```

Adding Removing Elements

```
In [68]: #Returned New Array
         h.resize((2,6))

In [69]: #Append Items
         np.append(h,g)

Out[69]: array([10.        , 20.        , 30.        ,  0.        ,  0.        ,
                 0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
                 0.        ,  0.        ,  9.09090909,  3.33333333,  3.8961039 ,
                 1.42857143,  2.22222222, 15.        ])

In [70]: #Insert Items
         np.insert(a,1,5)

Out[70]: array([10,  5, 20, 30])

In [71]: #Delete Items
         np.delete(a,[1])

Out[71]: array([10, 30])
```

Combining Array

```
In [72]: #Concatenation
         np.concatenate((a,d),axis=0)

Out[72]: array([10, 20, 30, 10, 15, 20])

In [73]: #Stack Array Vertically
         np.vstack((a,b))

Out[73]: array([[10. , 20. , 30. ],
                [ 1.1,  6. ,  7.7],
                [ 2. ,  7. ,  9. ]])

In [74]: #Stack Array Vertically Raw Wise
         np.r_[e,f]

Out[74]: array([[9., 9., 9.],
                [9., 9., 9.],
                [9., 9., 9.],
                [1., 0., 0.],
                [0., 1., 0.],
                [0., 0., 1.]])

In [75]: #Stack Array Horizontally
         np.hstack((e,f))

Out[75]: array([[9., 9., 9., 1., 0., 0.],
                [9., 9., 9., 0., 1., 0.],
                [9., 9., 9., 0., 0., 1.]])

In [76]: #Column Wise
         np.column_stack((a,d))

Out[76]: array([[10, 10],
                [20, 15],
                [30, 20]])

In [77]: #Column Wise
         np.c_[a,d]

Out[77]: array([[10, 10],
                [20, 15],
                [30, 20]])
```

Splitting Arrays

```
In [78]: #Horizontally Split the array at 3rd Index
         np.hsplit(a,3)

Out[78]: [array([10]), array([20]), array([30])]

In [79]: #Vertically Split the array at 2nd index
         np.vsplit(c,2)

Out[79]: [array([[[1.7, 2. , 6. ],
                  [1. , 2. , 3. ]]]),
          array([[[6., 6., 9.],
                  [6., 8., 3.]]])]

In [ ]:
```