Numpy Basics

What is NumPy?

NumPy is a Python library used for working with arrays.

It also has functions for working in domain of linear algebra, fourier transform, and matrices. NumPy stands for Numerical Python.

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

Creating Arrays

```
In [2]: a = np.array([1,2,3])
        print(a)

[1 2 3]

In [3]: b = np.array([[2,3,4],[5,6,7]])
        print(b)

[[2 3 4]
 [5 6 7]]
```

Inspecting your Array

```
In [4]: b.ndim

Out[4]: 2

In [5]: b.shape

Out[5]: (2, 3)

In [6]: b.dtype

Out[6]: dtype('int32')

In [7]: b.size

Out[7]: 6

In [8]: b.size * b.itemsize

Out[8]: 24

In [9]: b.nbytes

Out[9]: 24

In [10]: a = np.array([[1,2,3,4,5],[6,7,8,9,0]])
         print(a)

[[1 2 3 4 5]
 [6 7 8 9 0]]

In [11]: # get a specific element(r,c)
         a[1,2]

Out[11]: 8

In [12]: #get specific columun
         a[:,0]

Out[12]: array([1, 6])

In [13]: # get specific row
         a[0,:]

Out[13]: array([1, 2, 3, 4, 5])

In [14]: a[0,1:5:2]  # [start:end:step]

Out[14]: array([2, 4])

In [15]: a[1,1] = 10
         print(a)
         a[:,2] = 5
         print(a)

[[ 1  2  3  4  5]
 [ 6 10  8  9  0]]
[[ 1  2  5  4  5]
 [ 6 10  5  9  0]]

In [16]: #3D example
         b= np.array([[[1,2],[3,4]],[[5,6],[7,8]]])
         print(b)

[[[1 2]
  [3 4]]

 [[5 6]
  [7 8]]]

In [17]: b[1,1,1]
         b[0,1,0]

Out[17]: 3

In [18]: #replace
         b[:,1,:] = [[9,9],[8,3]]

In [19]: b

Out[19]: array([[[1, 2],
                 [9, 9]],

                [[5, 6],
                 [8, 3]]])
```

Initial Placeholers

```
In [20]: # all 0s matrix
         np.zeros((2,3))

The history saving thread hit an unexpected error (OperationalError('disk I/O error')).History will not be written to the database.

Out[20]: array([[0., 0., 0.],
                [0., 0., 0.]])

In [21]: np.ones((4,2,3))

Out[21]: 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 [22]: # any other number
         np.full((3,3),99)

Out[22]: array([[99, 99, 99],
                [99, 99, 99],
                [99, 99, 99]])

In [23]: #any other number(full_like)
         np.full_like(a.shape,4)
         np.full_like(a,5)

Out[23]: array([[5, 5, 5, 5, 5],
                [5, 5, 5, 5, 5]])

In [24]: # random decimnal number
         np.random.rand(4,3)

Out[24]: array([[9.50646819e-01, 9.28580512e-01, 4.78222218e-01],
                [6.23965663e-01, 4.88947058e-01, 2.93244930e-01],
                [6.00228932e-04, 3.27949246e-01, 9.33847169e-01],
                [4.58131644e-01, 4.94824843e-01, 8.00703169e-01]])

In [25]: # random integer number
         np.random.randint(3,size=(2,3))  # 3 is exculsive

Out[25]: array([[2, 2, 1],
                [2, 2, 0]])

In [26]: # the identity matrix
         np.identity(4)

Out[26]: array([[1., 0., 0., 0.],
                [0., 1., 0., 0.],
                [0., 0., 1., 0.],
                [0., 0., 0., 1.]])

In [27]: # repeat array
         arr = np.array([[1,2,3]])
         r = np.repeat(arr,3,  axis=0)
         print(r)

[[1 2 3]
 [1 2 3]
 [1 2 3]]

In [28]: # problem
         output = np.ones((5,5))
         print(output)
         z = np.zeros((3,3))
         print(z)
         z[1,1] = 9
         print(z)
         output[1:-1,1:-1] = z
         print(output)

[[1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]]
[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
[[0. 0. 0.]
 [0. 9. 0.]
 [0. 0. 0.]]
[[1. 1. 1. 1. 1.]
 [1. 0. 0. 0. 1.]
 [1. 0. 9. 0. 1.]
 [1. 0. 0. 0. 1.]
 [1. 1. 1. 1. 1.]]
```

Arithmetic Operations

```
In [29]: c=np.array([2,4,6])
         d=np.array([3,7,9])

In [30]: g = c - d       #subtraction
         print(g)

[-1 -3 -3]

In [31]: np.subtract(c,d)  #subtraction

Out[31]: array([-1, -3, -3])

In [32]: c + d            # addition

Out[32]: array([ 5, 11, 15])

In [33]: np.add(c,d)    #addition

Out[33]: array([ 5, 11, 15])

In [34]: c / d            #division

Out[34]: array([0.66666667, 0.57142857, 0.66666667])

In [35]: np.divide(c,d)     # division

Out[35]: array([0.66666667, 0.57142857, 0.66666667])

In [36]: c * d               #multiplication

Out[36]: array([ 6, 28, 54])

In [37]: np.multiply(c,d)       #multiplication

Out[37]: array([ 6, 28, 54])

In [38]: np.exp(c)            #exponentiation

Out[38]: array([  7.3890561 ,  54.59815003, 403.42879349])

In [39]: np.sqrt(c)          #square root

Out[39]: array([1.41421356, 2.        , 2.44948974])

In [40]: np.sin(d)            # print sines of an array

Out[40]: array([0.14112001, 0.6569866 , 0.41211849])

In [41]: np.cos(d)            # element-wise cosine

Out[41]: array([-0.9899925 ,  0.75390225, -0.91113026])

In [42]: np.log(c)            # element-wise natural logarithm

Out[42]: array([0.69314718, 1.38629436, 1.79175947])

In [43]: c.dot(d)              # dot product

Out[43]: 88
```

Aggregate Functions

```
In [44]: a.sum()    # array-wise sum

Out[44]: 47

In [45]: a.min()     # array-wise minimum value

Out[45]: 0

In [46]: b.max(axis=0)   #maximum value of an array row

Out[46]: array([[5, 6],
                [9, 9]])

In [47]: b.cumsum(axis=1)  #cumlative sum of the elemnts

Out[47]: array([[[ 1,  2],
                 [10, 11]],

                [[ 5,  6],
                 [13,  9]]], dtype=int32)

In [48]: a.mean()  #mean

Out[48]: 4.7

In [49]: np.median(b)  #median

Out[49]: 5.5

In [50]: np.corrcoef(a)      #corelation coefficient

Out[50]: array([[ 1.        , -0.52433452],
                [-0.52433452,  1.        ]])

In [51]: np.std(b)            # standard deviation

Out[51]: 2.9553976043842223
```

Sorting Arrays

```
In [52]: a.sort()      # sort an array

In [53]: c.sort(axis=0)   # sort the element of an array's axis
```

Transposing Array

```
In [54]: i = np.transpose(b)

In [55]: i.T

Out[55]: array([[[1, 2],
                 [9, 9]],

                [[5, 6],
                 [8, 3]]])
```

Changing Array Shape

```
In [56]: b.ravel()     #permute array dimensions

Out[56]: array([1, 2, 9, 9, 5, 6, 8, 3])

In [57]: d.reshape(3,-2)

Out[57]: array([[3],
                [7],
                [9]])
```

Thank You!