Getting started with NumPy

1. Creating Arrays

In [32]:

```
import numpy as np  #import numpy to begin with
```

In [33]:

```
# Ex:1 create an 3x3 array

a = np.arange(9)
print(a)
b = a.reshape(3,3)
print(b)
```

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

In [34]:

```
# Ex:2 create an 3x3 array by explicitly listing elements

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

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

In [35]:

```
# Ex: 3 create an array from data in a CSV file.

x = np.genfromtxt('numpy_ex.csv',delimiter=',',dtype='int8')
print (x)
```

```
[[-1  2  3]
 [ 4  5  6]
 [ 7  8  9]]
```

In [37]:

```
# Ex:4 create an 3x3 array by explicitly listing elements

a = np.zeros((3,3))
print (a)
```

```
[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
```

In [44]:

```
print(x.shape)  # Array Dimensions
print(x.ndim)   # Number of dimensions
print(x.size)  # number of elements in the array
print(x.dtype)  # type of all elements in the array [for https://docs.scipy.org/doc/numpy/u
print(x.itemsize) # size of items in bytes
```

```
(5, 5)
2
25
int32
4
```

In [45]:

```
# changing the type of elements
print (x.dtype)
x = np.array(x, dtype='int16') # note that we cannot change the type of array inplace.
print (x.dtype)
```

```
int32
int16
```

In [40]:

```
# Creating copy of an array

xCpy = x  # No new xCpy is created. xCpy and x are two names for the same object.

xCpy [0,0] = 999
print (x)
```

```
[[999   2   3]
 [  4   5   6]
 [  7   8   9]]
```

In [41]:

```
# Creating copy of an array

print (x)       #
xCpy = x.copy() # xCpy is a different object  - caled as deep copy

x [0,0] = 11111 # changing an element of x

print (x)
print (xCpy)
```

```
[[999   2   3]
 [  4   5   6]
 [  7   8   9]]
[[11111     2     3]
 [    4     5     6]
 [    7     8     9]]
[[999   2   3]
 [  4   5   6]
 [  7   8   9]]
```

2. Accessing elements, subarrays

In [46]:

```
# assume an array
x = np.arange(25).reshape(5,5)
print (x)
```

```
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]
 [15 16 17 18 19]
 [20 21 22 23 24]]
```

In [47]:

```
# access elements as x[i,j]
x[1,2]
```

Out[47]:

```
7
```

In [48]:

```
# accessing a row as x[i,...]
x[1,...]
```

Out[48]:

```
array([5, 6, 7, 8, 9])
```

In [49]:

```
# accessing a column as x[...,j]
x[...,1]
```

Out[49]:

```
array([ 1,  6, 11, 16, 21])
```

In [50]:

```
# accessing a part of a row as x [i, low:high]
x[1, 2:4]
```

Out[50]:

```
array([7, 8])
```

In [51]:

```
# accessing a part of a column as x [ low:high, j]
x[2:4,1]
```

Out[51]:

```
array([11, 16])
```

In [52]:

```
# extracting a subarray
y = x[2:4, 3:5]
print (y)
```

```
[[13 14]
 [18 19]]
```

In [54]:

```
# Accessing with lesser indices
x[1] # the last index is assumed to be full here.
```

Out[54]:

```
array([5, 6, 7, 8, 9])
```

In [55]:

```
# reshaping array shape

a = np.ones((2,3))
print(a)
a=a.reshape (3,2)
print(a)
```

```
[[1. 1. 1.]
 [1. 1. 1.]]
[[1. 1.]
 [1. 1.]
 [1. 1.]]
```

3. Some sample operations

In [56]:

```
# Assume a smaller array
x = np.array ([[1,2], [2,1]])
y = np.array([[1,1],[1,1]])
print(' Matrix x is ')
print (x)
print(' Matrix y is ')
print (y)
```

```
 Matrix x is
[[1 2]
 [2 1]]
 Matrix y is
[[1 1]
 [1 1]]
```

In [57]:

```
# addition
z = x + y
print (z)
```

```
[[2 3]
 [3 2]]
```

In [58]:

```
# subtraction
z = x-y
print (z)
```

```
[[0 1]
 [1 0]]
```

In [60]:

```
# transpose of a matrix

x = np.array([[1,2],[3,4]]) #sample matrix
print(x)
x.transpose()
```

```
[[1 2]
 [3 4]]
```

Out[60]:

```
array([[1, 3],
       [2, 4]])
```

In [61]:

```
# (Pseudo) Inverse of a matrix

x = np.array([[1,2,3],[3,4,2]]) #sample matrix
print(x)
np.linalg.pinv(a)
```

```
[[1 2 3]
 [3 4 2]]
```

Out[61]:

```
array([[0.16666667, 0.16666667, 0.16666667],
       [0.16666667, 0.16666667, 0.16666667]])
```

In [62]:

```
# identity matrix

x=np.eye(3)
print(x)
```

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

In [64]:

```
# Compute eigen values of a matrix

x=np.arange(9).reshape(3,3)
print(x)
np.linalg.eigvals(x)
```

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

Out[64]:

```
array([ 1.33484692e+01, -1.34846923e+00, -1.15433316e-15])
```

In [65]:

```
# compute dot product
x = np.arange(9).reshape (3,3)
y = np.eye (3)
z=np.dot(x,y)
print(z)
```

```
[[0. 1. 2.]
 [3. 4. 5.]
 [6. 7. 8.]]
```

In [66]:

```
# Some basic Stat
x = np.array ([1,2,3,4,5,6,7,8,9,10]) #taking a simple matrix
print (x)
print (np.median(x))
print (np.mean(x))
print (np.std (x))
print (np.var(x))
```

```
[ 1  2  3  4  5  6  7  8  9 10]
5.5
5.5
2.8722813232690143
8.25
```