numpy library

In [2]:

```
# importing the numpy module
import numpy as np
```

Creating arrays

In [17]:

```
# 1 dimensional
a = np.array([1,2,3,4,5])
print("The array: ", a)
print("Dimension of the array: ", a.ndim)
print("Shape of the array: ", a.shape)
print("Legth of the array: ", len(a))
print("\n")

# 2 dimensional
b = np.array(([1,2,3],[4,5,6]))
print("The array: \n", b)
print("Dimension of the array: ", b.ndim)
print("Shape of the array: ", b.shape)
print("Legth of the array: ", len(b))
```

```
The array:  [1 2 3 4 5]
Dimension of the array:  1
Shape of the array:  (5,)
Legth of the array:  5


The array: 
 [[1 2 3]
 [4 5 6]]
Dimension of the array:  2
Shape of the array:  (2, 3)
Legth of the array:  2
```

Generating Sequence of array

In [21]:

```
# Evenly spaced
a = np.arange(10)
print(a)
print(np.arange(10,50,10))
print("\n")

# By number of points
print(np.linspace(0,1,6))       # dividing 6 partitions from 0 to 1
```

```
[0 1 2 3 4 5 6 7 8 9]
[10 20 30 40]


[0.  0.2 0.4 0.6 0.8 1. ]
```

Common arrays

In [30]:

```
print(np.ones((3,3)))
print("\n")

print(np.zeros((3,3)))
print("\n")

print(np.eye(3))
print("\n")

print(np.diag(np.array([1,2,3])))
```

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


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


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


[[1 0 0]
 [0 2 0]
 [0 0 3]]
```

Generating random numbers in numpy

In [37]:

```
r = np.random.rand(4)       # generates random floating point numbers between 0 to 1
print("Random numbers generated are: ", r)
r.dtype
```

```
Random numbers generated are:  [0.32782768 0.70724533 0.53552084 0.1775902 ]
```

Out[37]:

```
dtype('float64')
```

Datatypes in numpy

In [50]:

```
a = np.array([1,2,3,4,5], dtype='float64')
print(a)
print(a.dtype)

b = np.array([1,2,3,4,5], dtype='int64')
print(b)
print(b.dtype)

com = np.array([1+ 1j, 2+5j])
print(com)
print(com.dtype)
```

```
[1. 2. 3. 4. 5.]
float64
[1 2 3 4 5]
int64
[1.+1.j 2.+5.j]
complex128
```

Indexing and Slicing

In [66]:

```
num = np.arange(10)
print(num)
print(num[5])
print(num[:5])
print(num[-1])
print(num[: : -1])  # for reversing the sequence
print(num[2:9:3])
```

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

In [64]:

```
# for mutli dimensional ndarrays
mat = np.diag(np.array([1,2,3,4]))
print(mat)

print(mat[2,2])

mat[2,1] = 99   # updating the matrices
mat
```

```
[[1 0 0 0]
 [0 2 0 0]
 [0 0 3 0]
 [0 0 0 4]]
3
```

Out[64]:

```
array([[ 1,  0,  0,  0],
       [ 0,  2,  0,  0],
       [ 0, 99,  3,  0],
       [ 0,  0,  0,  4]])
```

- A slicing operation craetes a view on the original array, which is just a way of accessing array data
- Thus the original array is not copied into the memory it's just a view
- we can use np.share_memory() to check whether if two arrays share the same memory or not

In [8]:

```
a = np.arange(10)
print("a: ", a)

b = a[::2]          # view
print("b: ", b)

np.may_share_memory(a,b)
b[0] = 12

print("a: ", a)
print("b: ", b)     # when modifying the view(b) even the original array(a) also has been m
```

```
a:  [0 1 2 3 4 5 6 7 8 9]
b:  [0 2 4 6 8]
a:  [12  1  2  3  4  5  6  7  8  9]
b:  [12  2  4  6  8]
```

In [12]:

```
a = np.arange(10)
print("a: ", a)

b = a[::2].copy()   # forcing a copy to avoid data redundancy while doing array manipulatio
print("b: ", b)

b[0] = 12
print("a: ", a)
print("b: ", b)
```

```
a:  [0 1 2 3 4 5 6 7 8 9]
b:  [0 2 4 6 8]
a:  [0 1 2 3 4 5 6 7 8 9]
b:  [12  2  4  6  8]
```

Numerical operations on ndarray

In [22]:

```
# Element-wise operations
a = np.array([0,1,2,3,4,5])
print("a:   ", a)
print("a+1: ", a+1)
print("2**a:", a**2)

# array multiplication
c = np.ones((3,3))
print("Array multiplicaiton: \n", c*c)

# matrix muliplication
print("Matrix multiplication: \n", c.dot(c))
```

```
a:    [0 1 2 3 4 5]
a+1:  [1 2 3 4 5 6]
2**a: [ 0  1  4  9 16 25]
Array multiplicaiton: 
 [[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]
Matrix multiplication: 
 [[3. 3. 3.]
 [3. 3. 3.]
 [3. 3. 3.]]
```

Array shape manipulations

In [42]:

```
# Flattening
a = np.array([[1,2,3],[4,5,6]])
print("a:\n", a)

print("Flattened a: ", a.ravel())   # flattening 2d to 1d
```

```
a:
 [[1 2 3]
 [4 5 6]]
Flattened a:  [1 2 3 4 5 6]
```

In [43]:

```
# Reshaping
print("Reshaped a:\n", a.reshape(3,2))  # reverse of flattening
```

```
Reshaped a:
 [[1 2]
 [3 4]
 [5 6]]
```

In [47]:

```
# Resizing
a = np.arange(4)
print("a: ", a)
a.resize((8,))
print("resized a: \n", a)
```

```
a:  [0 1 2 3]
resized a: 
 [0 1 2 3 0 0 0 0]
```

In [53]:

```
# Sorting data
a = np.array([[3,2,1],[10,5,6]])
print("a:\n", a)

a.sort()
print("Sorted a: \n",a)
```

```
a:
 [[ 3  2  1]
 [10  5  6]]
Sorted a: 
 [[ 1  2  3]
 [ 5  6 10]]
```