NumPy array and perform operation on it

In [4]:

```
#! pip install numpy
```

```
Requirement already satisfied: numpy in c:\users\student\anaconda3\lib\site-
packages (1.16.2)
```

import numpy

In [6]:

```
import numpy as np
```

Numpy array creation1

In [19]:

```
a=np.arange(15)
print(a)
```

```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14]
```

In [16]:

```
s=np.arange(20).reshape(4,5)
print(s)
```

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

In [22]:

```
j=np.arange(27).reshape(3,3,3)
print(j)
```

```
[[[ 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 25 26]]]
```

attributes of numpy array

In [43]:

```
#to check dimensions of array
#syntax: ndarray.ndim
print("Dimension of array j: ",j.ndim)

#shape of array
print("nriows and ncols in aaray 1: ",j.shape)

#total no. of elements
print("Size of array j: ",j.size)

#datatype of array elements
print("Datatype of array element j:  ",j.dtype)

#itemsize(no. of bytes data each element takes)
print("Itemsize of array j:", j.itemsize)

#memory location  for array
print("Memory of array j: ", j.data)
```

```
Dimension of array j:  3
nriows and ncols in aaray 1:  (3, 3, 3)
Size of array j:  27
Datatype of array element j:   int32
Itemsize of array j: 4
Memory of array j:  <memory at 0x0000022177437048>
```

array creation using array()

In [42]:

```
#1-d array
k=np.array([1,2,3])
print("k =",k)
print("dimension =",k.ndim)

#2-d array(2x3)
u=np.array([(3,5,6),(2,6,4)])
print("u=",u)
print("dimension =",u.ndim)
print("datatype =",u.dtype)

#3-d array
o=np.array([[(2,3),(4,8),(7,6)]])
print("o=",o)
print("dimension =",o.ndim)
print("datatype =",o.dtype)
```

```
k = [1 2 3]
dimension = 1
u= [[3 5 6]
 [2 6 4]]
dimension = 2
datatype = int32
o= [[[2 3]
  [4 8]
  [7 6]]]
dimension = 3
datatype = int32
```

Type conversion in numpy array

In [45]:

```
#implicit conversion(automatic)
f=np.array([(1,4,2),(8,5,1.4)])
print("f=",f)
print("dimension =",f.ndim)
print("datatype =",f.dtype)

#explicit conversion(user defined)
f=np.array([(1,4,2),(8,5,1.4)],dtype='int')
print("f=",f)
print("dimension =",f.ndim)
print("datatype =",f.dtype)

```

```
f= [[1.  4.  2. ]
 [8.  5.  1.4]]
dimension = 2
datatype = float64
f= [[1 4 2]
 [8 5 1]]
dimension = 2
datatype = int32
```

Unique or standard array creation in numpy

In [54]:

```
#zeros array
z=np.zeros((2,3))
print("Zeros array z= ",z)
print("Memory location of array z =",z.data)

#ones array
y=np.ones((2,3))
print("Ones array y= ",y)
print("Memory location of array y =",y.data)

#empty array(when size is changes it gives diff memory location with garbage values)
e=np.empty((3,3))
print("Empty array e= ",e)
print("Memory location of array e =",e.data)
```

```
Zeros array z=  [[0. 0. 0.]
 [0. 0. 0.]]
Memory location of array z = <memory at 0x00000221762B2EA0>
Ones array y=  [[1. 1. 1.]
 [1. 1. 1.]]
Memory location of array y = <memory at 0x00000221762B2EA0>
Empty array e=  [[0.00000000e+000 0.00000000e+000 9.76118064e-313]
 [1.95820216e-306 6.23054972e-307 8.90106955e-307]
 [1.20161797e-306 1.86920872e-306 1.08225056e-312]]
Memory location of array e = <memory at 0x00000221762B2EA0>
```

Numpy arange()

In [57]:

```
#arange() creates range from lower to upper-1step and it will increment as per the step
#by default step =1
#synatx
#np.arange(lower,upper,step)

x=np.arange(10,50,2)
print("x=",x)
```

```
x= [10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48]
```

In [60]:

```
#problem in arange()-you cannot predict the no. of elements created due to step size in
m=np.arange(0,3,0.12)
print(m)
print("m=",m.size)

#solution-linspace() in numpy
```

```
[0.   0.12 0.24 0.36 0.48 0.6  0.72 0.84 0.96 1.08 1.2  1.32 1.44 1.56
 1.68 1.8  1.92 2.04 2.16 2.28 2.4  2.52 2.64 2.76 2.88]
m= 25
```

linspace() in numpy

In [65]:

```
#syntax-linspace(lower,upper,no. of elements)
from math import pi
t=np.linspace(0,3*pi,10)
print(t)

#in linspace upper bound is included but in arange() upper bound is neglected
```

```
[0.         1.04719755 2.0943951  3.14159265 4.1887902  5.23598776
 6.28318531 7.33038286 8.37758041 9.42477796]
```

Basic operations on numpy array

In [76]:

```
c=np.array([5,3,7,1])
d=np.arange(4)
print(c)
print(d)

print(c-d)

print("Square of elements of c: ",c**2)
#d**2

print("Sin() of c: ",np.sin(c))

#Loical operations of each elemen of array to check array c element if greater than 5
print(c>5)
```

```
[5 3 7 1]
[0 1 2 3]
[ 5  2  5 -2]
Square of elements of c:  [25  9 49  1]
Sin() of c:  [-0.95892427  0.14112001  0.6569866   0.84147098]
[False False  True False]
```

Matrix multiplication of ndarray

In [78]:

```
M1=np.array([[2,4],[4,3]])
M2=np.array([[5,3],[1,6]])

#element wise product
print("element wise productof M1 AND M2: ",M1*M2)

#matrix multiplication
print ("matrix product :",M1@M2)

#another matrix product array1.dot(array2)
print("another matrix product: ",M1.dot(M2))
```

```
element wise productof M1 AND M2:  [[10 12]
 [ 4 18]]
matrix product : [[14 30]
 [23 30]]
another matrix product:  [[14 30]
 [23 30]]
```