Single Dimensional Array

A single dimensional array is an array of elements of similar data type starting from index 0 and ending till n+1.

An integer array is created using “int” argument; string array is created using “str” argument; array of float elements is created using “float” argument.

An array can also be created from an existing array using view() and copy() command.

In [3]:
```
#Program to create array from numpy library.
import numpy
#Creating an array of integers.
myarray1=numpy.array([11,22,33,44, 55, 66], int)
print ("The array of integers is: \n",myarray1)
#Creating an array of characters.
myarray2=numpy.array(['a', 'b', 'c', 'd', 'e', 'f'])
print ("The array of characters is: \n",myarray2)
#Creating an array of strings.
myarray3=numpy.array(['JAVA', 'PYTHON', 'C++', 'C'], dtype=str)
print ("The array of strings is: \n", myarray3)
#Creating a float array.
myarray4=numpy.array([6.3,24.5,16.7,28.9,85.8], float)
print ("The array of decimal numbers is: \n", myarray4)
#Using view() function to create a new array from an existing array
myarray5=myarray3.view()
print("Using view function to create a new array:\n",myarray5)
#Using copy() function to create a new array from an existing array
myarray6=myarray4.copy()
print ("Using copy function to create a new array: \n",myarray6)
```

```
The array of integers is:
 [11 22 33 44 55 66]
The array of characters is:
 ['a' 'b' 'c' 'd' 'e' 'f']
The array of strings is:
 ['JAVA' 'PYTHON' 'C++' 'C']
The array of decimal numbers is:
 [ 6.3 24.5 16.7 28.9 85.8]
Using view function to create a new array:
 ['JAVA' 'PYTHON' 'C++' 'C']
Using copy function to create a new array:
 [ 6.3 24.5 16.7 28.9 85.8]
```

numpy array has an access to many other mathematical, trigonometrical, statistical, logarithmic and exponential functions.

In [5]:
```
#Creating an integer array using arange() function.
myarray7=numpy.arange(14)
print ("The array of integers using arange() function is: \n",myarray7)
#Creating a float array using arange() function.
```

```python
myarray8= numpy.arange(14,19, dtype = 'float')
print ("The array of decimal numbers using arange() function is: \n", myarray8)
# Creating an integer array using linspace () for equal spacing.
myarray9=numpy.linspace (12, 24, 5)
print ("The array of integers using linspace () function is: \n",myarray9)
#Creating float array using linspace () for equal spacing.
myarray10=numpy.linspace (1., 4., 5)
print ("The array of decimal numbers using linspace () function is:\n", myarray10)
#Creating array using Zeros () function.
myarray11=numpy.zeros((4))
print ("An array using Zeros: \n", myarray11)
#Creating array using Ones () function.
myarray12=numpy.ones ((5))
print ("An array using Ones: \n",myarray12)
```

```
The array of integers using arange() function is:
 [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13]
The array of decimal numbers using arange() function is:
 [14. 15. 16. 17. 18.]
The array of integers using linspace () function is:
 [12. 15. 18. 21. 24.]
The array of decimal numbers using linspace () function is:
 [1.   1.75 2.5  3.25 4.  ]
An array using Zeros:
 [0. 0. 0. 0.]
An array using Ones:
 [1. 1. 1. 1. 1.]
```

In [7]:

```python
myarray13=numpy.array([13, 24, 22,13,11,28,16,24,18], int)
print ("Original array is: \n",myarray13)
#Modifying an element at a specified index.
myarray13 [3]=45
print ("Modified array is: \n", myarray13)
#Determine characteristics of the array. #The size returns the size of the array.
print ("Size of the array is: ", myarray13.size)
#The dtype returns the datatype of the elements of the array.
print ("Data type of the array is: ",myarray13.dtype)
#Functions returning one single numeric output.
#The mean () function returns the mean of all array elements.
print ("Mean of all array elements is: ", numpy.mean (myarray13))
#The median () function returns the median of all array elements.
print ("Median of all array elements is: ", numpy.median (myarray13))
#The min() function returns the minimum element of the array.
print ("Minimum element of the array is: ", numpy.min (myarray13))
#The max() function returns the maximum element of the array.
print ("Maximum element of the array is: ", numpy.max (myarray13))
#The sum() function returns the sum of all array elements.
print ("Sum of all array elements is: ", numpy.sum (myarray13))
#The prod() function returns the product of all array elements.
print("Product of all array elements is: ", numpy.prod (myarray13))
#The cov() function returns the covariance of all array elements.
print ("Co-variance of all array elements is: ", numpy.cov (myarray13))
#The var() function returns the variance of all array elements.
print ("Variance of all array elements is: ", numpy.var (myarray13))
#The std() function returns standard deviation of all array elements.
print ("standard deviation of array elements is: ", numpy.std (myarray13))
```

```
#Mathematical functions return an array output (all elements).
#The sort() function returns the elements in a sorted order.
print ("The sorted array is: \n", numpy.sort (myarray13))
#The power() function returns all elements raised to power of a number.
print ("Power raised to element (3) of array: \n", numpy.power (myarray13,3))
#The sqrt() function returns the square root of all elements.
print ("Square root of the array elements is: \n", numpy.sqrt (myarray13))
#The abs() function returns the absolute value of all the elements.
print ("Absolute value of the array elements is: \n", numpy.abs (myarray13))
#Trignometric functions return an array output (all elements).
#The sin() function returns the sine value of all the elements.
print ("Sin value of the array elements is: \n", numpy.sin (myarray13))
#The cos() function returns the cosine value of all the elements.
print ("Cosine value of the array elements is: \n", numpy.cos (myarray13))
#The tan() function returns the tangent value of all the elements.
print ("Tangent value of the array elements is: \n", numpy.tan (myarray13))
#Logarithmic/exponential functions return an array output.
#The log() function returns log with exponential base of all elements.
print ("Log of all elements with 'e' base is: \n", numpy.log (myarray13))
#The log10() function returns the log with base 10 of all elements.
print ("Log of all elements with base 10 is: \n", numpy. log10 (myarray13))
#The exp() function returns the exponential value of all the elements.
print ("Exponential of the array elements is: \n", numpy.exp (myarray13))
```

```
Original array is:
 [13 24 22 13 11 28 16 24 18]
Modified array is:
 [13 24 22 45 11 28 16 24 18]
Size of the array is:  9
Data type of the array is:  int64
Mean of all array elements is:  22.333333333333332
Median of all array elements is:  22.0
Minimum element of the array is:  11
Maximum element of the array is:  45
Sum of all array elements is:  201
Product of all array elements is:  657573396480
Co-variance of all array elements is:  103.25
Variance of all array elements is:  91.77777777777777
standard deviation of array elements is:  9.580071908799942
The sorted array is:
 [11 13 16 18 22 24 24 28 45]
Power raised to element (3) of array:
 [ 2197 13824 10648 91125  1331 21952  4096 13824  5832]
Square root of the array elements is:
 [3.60555128 4.89897949 4.69041576 6.70820393 3.31662479 5.29150262
 4.         4.89897949 4.24264069]
Absolute value of the array elements is:
 [13 24 22 45 11 28 16 24 18]
Sin value of the array elements is:
 [ 0.42016704 -0.90557836 -0.00885131  0.85090352 -0.99999021  0.27090579
 -0.28790332 -0.90557836 -0.75098725]
Cosine value of the array elements is:
 [ 0.90744678  0.42417901 -0.99996083  0.52532199  0.0044257  -0.96260587
 -0.95765948  0.42417901  0.66031671]
Tangent value of the array elements is:
 [ 4.63021133e-01 -2.13489670e+00  8.85165604e-03  1.61977519e+00
 -2.25950846e+02 -2.81429605e-01  3.00632242e-01 -2.13489670e+00
 -1.13731371e+00]
Log of all elements with 'e' base is:
 [2.56494936 3.17805383 3.09104245 3.80666249 2.39789527 3.33220451
 2.77258872 3.17805383 2.89037176]
Log of all elements with base 10 is:
 [1.11394335 1.38021124 1.34242268 1.65321251 1.04139269 1.44715803
 1.20411998 1.38021124 1.25527251]
Exponential of the array elements is:
 [4.42413392e+05 2.64891221e+10 3.58491285e+09 3.49342711e+19
 5.98741417e+04 1.44625706e+12 8.88611052e+06 2.64891221e+10
 6.56599691e+07]
```

It is possible to use basic mathematical operators and relational operators on 1-D array

In [9]:
```
#Mathematical operations on one-dimensional array.
myarray14= numpy.arange(25,34)
print ("Original array is: \n",myarray14)
print ("Adding an element to array elements is: \n",myarray14+45)
print ("Subtracting an element to array elements is: \n",myarray14-10)
print ("Multiplying an element to array elements is: \n",myarray14*3)
print ("Dividing an element from array elements is: \n", myarray14/4)
print ("Using modulus operator for the array elements: \n", myarray14%4)
print ("Using integer division for the array elements: \n", myarray14//4)
```

```
Original array is:
 [25 26 27 28 29 30 31 32 33]
Adding an element to array elements is:
 [70 71 72 73 74 75 76 77 78]
Subtracting an element to array elements is:
 [15 16 17 18 19 20 21 22 23]
Multiplying an element to array elements is:
 [75 78 81 84 87 90 93 96 99]
Dividing an element from array elements is:
 [6.25 6.5  6.75 7.   7.25 7.5  7.75 8.   8.25]
Using modulus operator for the array elements:
 [1 2 3 0 1 2 3 0 1]
Using integer division for the array elements:
 [6 6 6 7 7 7 7 8 8]
```

In [11]:
```
import numpy as np
#Creating three integer arrays.
myarray15=np.array([11,22,33], int)
myarray16=np.array([40, 50, 60],int)
myarray17=np.array([7,8,9], int)
#Performing addition and subtraction on all elements of an array.
myarray18=myarray17+myarray16-myarray15
print ("Addition and subtraction on multiple arrays:\n",myarray18)
#Performing multiplication and division on all elements of an array.
myarray19=(myarray15*myarray16)/myarray17
print ("Multiplication and division on multiple arrays: \n",myarray19)
#Using modulus operator (%) on all elements of the array.
myarray20=myarray16%myarray17
print ("Using modulus operator on two arrays:\n",myarray20)
#Using integer division (//) on all elements of the array.
myarray21=myarray16//myarray17
print ("Using integer division on two arrays: \n",myarray21)
```

```
Addition and subtraction on multiple arrays:
 [36 36 36]
Multiplication and division on multiple arrays:
 [ 62.85714286 137.5        220.        ]
Using modulus operator on two arrays:
 [5 2 6]
Using integer division on two arrays:
 [5 6 6]
```

In [15]:
```
myarray22=np.array([215,323,520,636,717,281,162,411], int)
myarray23=np.array([311,453,711,362,230,453,672, 891], int)
#Using greater than or equal to (>=) relational operator.
print ("Are elements of first array>= corresponding elements of second array?\n")
print (myarray22>=myarray23)
#Using less than or equal to (<=) relational operator.
print ("Are elements of first array<= corresponding elements of second array?\n",
       myarray22<=myarray23)
#Using equals (=) relational operator.
print ("Are elements of first array = corresponding elements of second array?\n",
       myarray22==myarray23)
#Using not equal to (!=) relational operator.
```

```
print ("Are elements of first array != corresponding elements of second array?\n",
       myarray22!=myarray23)
```

```
Are elements of first array>= corresponding elements of second array?

[False False False  True  True False False False]
Are elements of first array<= corresponding elements of second array?
 [ True  True  True False False  True  True  True]
Are elements of first array = corresponding elements of second array?
 [False False False False False False False False]
Are elements of first array != corresponding elements of second array?
 [ True  True  True  True  True  True  True  True]
```

Multidimensional Arrays

Creating a Multidimensional Array

The simplest form of multidimensional array is the two-dimensional array.

A two dimensional array has rows and columns.

It can be also considered as a list of one-dimensional arrays.

A two-dimensional array of size [m][n] can be considered as a tabular form having "m" number of rows and "n" number of columns.

Thus, a matrix is considered to be a specialized two-dimensional array.

A multidimensional array can be created using array(), reshape() , matrix(), zeros() or ones() function.

The array() function uses square bracket outside all elements to denote one row of the array.

A matrix() function uses semi colon (;) to separate a row from other.

The reshape() function converts the single dimensional array into multidimensional array on the basis of specified argument for rows and columns.

The zeros() and ones() function creates multidimensional array of specified dimensions and fills them with zeros and ones respectively.

In [18]:
```
#Program to show creation of multidimensional array.
import numpy as np
#Creating a two-dimensional array using array() function.
mymultarray1=np.array([[100,200, 300, 400], [600,700, 800, 90011]])
print ("Multidimensional array using array function is: \n",mymultarray1)
#Creating a two-dimensional array using matrix() function.
mymultarray2=np.matrix ('11 22; 33 44; 55 66')
print ("Multidimensional array using matrix function is: \n", mymultarray2)
#Creating a two-dimensional array using reshape() function.
```

```
newarray= np.array([10, 20, 30, 40, 50, 60])
mymultarray3=np.reshape(newarray, (3,2))
print ("Multidimensional array using reshape function is: \n",mymultarray3)
#Creating array using zeros () function.
mymultarray4=np.zeros((4,2))
print ("Multidimensional array using zeros function is: \n", mymultarray4)
#Creating array using Ones () function.
mymultarray5=np.ones ( (2,3))
print ("Multidimensional array using ones function is: \n",mymultarray5)
```

```
Multidimensional array using array function is:
 [[  100   200   300   400]
 [  600   700   800 90011]]
Multidimensional array using matrix function is:
 [[11 22]
 [33 44]
 [55 66]]
Multidimensional array using reshape function is:
 [[10 20]
 [30 40]
 [50 60]]
Multidimensional array using zeros function is:
 [[0. 0.]
 [0. 0.]
 [0. 0.]
 [0. 0.]]
Multidimensional array using ones function is:
 [[1. 1. 1.]
 [1. 1. 1.]]
```

Accessing Elements in Multidimensional Array

An element in a two-dimensional array is accessed by using the subscripts, i.e., row index and column index of the array.

Every element in the array is identified by an element name of the form myarray[x][y], where 'myarray' is the name of the array, and 'x' and 'y' are the subscripts corresponding to index of row and column that uniquely identify each element in 'myarray'.

In [20]:
```
#Program to show indexing in multidimensional arrays.
import numpy as np
mymultarray6=np.reshape(range (100, 400, 50), (3,2))
print ("The array is: \n",mymultarray6)
#Performing indexing.
print ("Element of first row and second column is: ", mymultarray6[0][1])
print ("Element of second row and first column is:", mymultarray6[1][0])
print ("Element of second row and second column is: ",mymultarray6[1][1])
print ("Element of third row and first column is: ", mymultarray6[2][0])
print ("Element of third row and second column is:", mymultarray6[2][1])
```

```
The array is:
 [[100 150]
 [200 250]
 [300 350]]
Element of first row and second column is:  150
Element of second row and first column is: 200
Element of second row and second column is:  250
Element of third row and first column is:  300
Element of third row and second column is: 350
```

In [21]:
```
mymultarray7=np.reshape(range (100, 500, 20), (4,5))
print("The multidimensional array is: \n", mymultarray7[:, :])
#Slicing by specifying only the row indexes.
print ("First two rows are: \n", mymultarray7 [0:2,])
print("Third row is: \n", mymultarray7 [2,])
#Slicing by specifying both the row and column indexes.
print ("Elements in first and second row and column are:\n",mymultarray7 [0:2, 0:2]
print ("Elements in second row and second to fourth column are:\n", mymultarray7 [1
```

```
The multidimensional array is:
 [[100 120 140 160 180]
 [200 220 240 260 280]
 [300 320 340 360 380]
 [400 420 440 460 480]]
First two rows are:
 [[100 120 140 160 180]
 [200 220 240 260 280]]
Third row is:
 [300 320 340 360 380]
Elements in first and second row and column are:
 [[100 120]
 [200 220]]
Elements in second row and second to fourth column are:
 [220 240 260]
```

Functions on Multidimensional Array

In [23]:
```
#Program for functions on multidimensional arrays. import numpy as np
mymultarray8=np.array([[10, 60, 40], [40,50,30], [50, 60, 40], [40, 50, 2011]])
#Displaying the matrix.
print ("The original matrix is: \n",mymultarray8)
#The transpose() function returns the transpose of a matrix.
print ("Transposed matrix is: \n",mymultarray8.transpose())
#The ndim determines dimension of the array.
print ("Dimension of the array is: ", mymultarray8.ndim)
#The shape determines shape of the array.
print ("Shape of the array is:", mymultarray8.shape)
#The flatten () flattens matrix and converts into one-dimensional.
print("Flattened matrix is: \n",mymultarray8.flatten())
#The sort() function with axis=1 (default) sorts the matrix on row basis.
print ("Sorted matrix on row basis (default): \n", np.sort(mymultarray8))
#The sort() function with axis=0 sorts the matrix on column basis.
print ("Sorted matrix on columnar basis: \n", np.sort(mymultarray8, axis=0))
```

```
#The diagonal () function helps to determine the diagonal elements.
print ("Diagonal elements are:\n", np.diagonal(mymultarray8))
```

```
The original matrix is:
 [[  10   60   40]
 [  40   50   30]
 [  50   60   40]
 [  40   50 2011]]
Transposed matrix is:
 [[  10   40   50   40]
 [  60   50   60   50]
 [  40   30   40 2011]]
Dimension of the array is:  2
Shape of the array is: (4, 3)
Flattened matrix is:
 [  10   60   40   40   50   30   50   60   40   40   50 2011]
Sorted matrix on row basis (default):
 [[  10   40   60]
 [  30   40   50]
 [  40   50   60]
 [  40   50 2011]]
Sorted matrix on columnar basis:
 [[  10   50   30]
 [  40   50   40]
 [  40   60   40]
 [  50   60 2011]]
Diagonal elements are:
 [10 50 40]
```

```
In [ ]:
```