NumPy - Numerical Python

1. Create an array using Numpy Module

a) Method 1

In [1]:
```
1 import numpy
2 a=numpy.array([10,20,30])
3 print('The array a =',a)
```

```
The array a = [10 20 30]
```

In [2]:
```
1 import numpy as np
2 a=np.array([10,20,30])
3 print('The array a =',a)
```

```
The array a = [10 20 30]
```

b) Method 2 - from numpy import*

In [3]:
```
1 from numpy import*
2 a = array([10,20,30])
3 print('The array a =',a)
```

```
The array a = [10 20 30]
```

In [4]:
```
1 from numpy import*
2 a = array([10,20.2,30])
3 print('The array a =',a)
```

```
The array a = [10.  20.2 30. ]
```

2. Arrays of characters and strings

In [5]:
```
1 from numpy import*
2 a = array(['B','A','L','A'])
3 print('The array a =',a)
```

```
The array a = ['B' 'A' 'L' 'A']
```

In [6]:
```
1 from numpy import*
2 a = array(['BALA','NARASIMHA','PYTHON'])
3 print('The array a =',a)
```

```
The array a = ['BALA' 'NARASIMHA' 'PYTHON']
```

3. Array of Different data types

In [7]:
```
1 from numpy import*
2 a = array(['BALA','NARASIMHA',10,20.35,3+6J])
3 print('The array a =',a)
```

```
The array a = ['BALA' 'NARASIMHA' '10' '20.35' '(3+6j)']
```

4. Assigning Array to an another variable

In [8]:
```
from numpy import*
a = array([10,20,30])
print('The array a =',a)
b=array(a)
c=b
print('The array a =',a)
print('The array b =',b)
print('The array c =',c)
```

```
The array a = [10 20 30]
The array a = [10 20 30]
The array b = [10 20 30]
The array c = [10 20 30]
```

5. Methods of Creating arrays using Numpy

5.1 array() -

In [9]:
```
import numpy as np
a = np.array([10,20,30])
print('The array a =',a)
```

```
The array a = [10 20 30]
```

5.2 linspace(start,stop,number of parts) -evenly spaced elements

In [10]:
```
import numpy as np
a = linspace(0,10,3)
print('The array a =',a)
```

```
The array a = [ 0.  5. 10.]
```

5.3 logspace(start,stop,number of parts) - evenly spaced elements on a log scale

In [11]:
```
import numpy as np
a = logspace(0,10,3)
print('The array a =',a)
```

```
The array a = [1.e+00 1.e+05 1.e+10]
```

5.4 arange(start,stop,step size) - elements with step size

In [12]:
```
import numpy as np
a = arange(0,10,3)
print('The array a =',a)
```

```
The array a = [0 3 6 9]
```

5.5 zeros(n, datatype)

In [13]:
```
import numpy as np
a = zeros(5)
b = zeros(5,int)
c = zeros(5,float)
print('The array a =',a)
print('The array b =',b)
print('The array c =',c)
```

```
The array a = [0. 0. 0. 0. 0.]
The array b = [0 0 0 0 0]
The array c = [0. 0. 0. 0. 0.]
```

5.6 ones(n, datatype)

In [14]:
```
import numpy as np
a = ones(5)
b = ones(5,int)
c = ones(5,float)
print('The array a =',a)
print('The array b =',b)
print('The array c =',c)
```

```
The array a = [1. 1. 1. 1. 1.]
The array b = [1 1 1 1 1]
The array c = [1. 1. 1. 1. 1.]
```

6. Mathematical Operations on Arrays

In [15]:
```
import numpy as np
a = np.array([2,5,6])
print("Array \'a\' before addition :",a)
a = a+5                                  #Adding '5' to the all elements in the array
print("Array \'a\' after addition :",a)
b= np.array([4,2,3])
print("Array \'b\' before addition :",b)
b = b*10                                 #Multiplying '10' to the all elements in the
print("Array \'a\' after addition :",b)
c = b-a #subract array b from array a. #Subtracting array 'b' from array 'a'.
print("Array \'a\' after addition :",c)
```

```
Array 'a' before addition : [2 5 6]
Array 'a' after addition : [ 7 10 11]
Array 'b' before addition : [4 2 3]
Array 'a' after addition : [40 20 30]
Array 'a' after addition : [33 10 19]
```

7. Mathematical Functions on Arrays

7.1 add(arr1,arr2....) - Add array1 elements to array2.

In [16]:
```
import numpy as np
a = np.array([2,5,6])
b= np.array([4,2,3])
print("Addition of elements in array \'a\' with array \'b\' :",np.add(a,b))
```

```
Addition of elements in array 'a' with array 'b' : [6 7 9]
```

In [17]:
```
from numpy import*
a = array([2,5,6])
b= array([4,2,3])
print("Addition of elements in array \'a\' with array \'b\' :",add(a,b))
```

```
Addition of elements in array 'a' with array 'b' : [6 7 9]
```

7.2 subtract(arr1,arr2,...) - subtract array1 elements from array 2

In [18]:
```
import numpy as np
a = np.array([2,5,6])
b= np.array([4,2,3])
print("Subtraction of elements from array \'a\' and array \'b\' :",np.subtract(a,b))
```

```
Subtraction of elements from array 'a' and array 'b' : [-2  3  3]
```

In [19]:
```
from numpy import*
a = array([2,5,6])
b = array([4,2,3])
print("Subtraction of elements from array \'a\' and array \'b\' :",subtract(a,b))
```

```
Subtraction of elements from array 'a' and array 'b' : [-2  3  3]
```

7.3 multiply(arr1,arr2....) - Multiply array1 elements with array2.

In [20]:
```
import numpy as np
a = np.array([2,5,6])
b = np.array([4,2,3])
print("Product of elements in array \'a\' with array \'b\' :",np.multiply(a,b))
```

```
Product of elements in array 'a' with array 'b' : [ 8 10 18]
```

In [21]:
```
from numpy import*
a = array([2,5,6])
b= array([4,2,3])
print("Product of elements in array \'a\' with array \'b\' :",multiply(a,b))
```

```
Product of elements in array 'a' with array 'b' : [ 8 10 18]
```

7.4 divide(arr1,arr2,..) - Divide Array 1 elements with array2 elements.

In [22]:
```
import numpy as np
a = np.array([20,50,60])
b = np.array([3,6,9])
print("Division of elements in array \'a\' by elements in array \'b\' :",np.divide(a
```

```
Division of elements in array 'a' by elements in array 'b' : [6.66666667 8.33333333 6.6
6666667]
```

In [23]:
```
from numpy import*
a = array([20,50,60])
b = array([3,6,9])
print("Division of elements in array \'a\' by elements in array \'b\' :",divide(a,b)
```

```
Division of elements in array 'a' by elements in array 'b' : [6.66666667 8.33333333 6.6
6666667]
```

7.5 true_divide(arr1,arr2...) - Returns true division

In [24]:
```
import numpy as np
a = np.array([20,50,60])
b = np.array([3,6,9])
print("True Division of elements in array \'a\' by array \'b\' :",np.true_divide(a,b
```

```
True Division of elements in array 'a' by array 'b' : [6.66666667 8.33333333 6.6666666
7]
```

In [25]:
```
from numpy import*
a = array([20,50,60])
b = array([3,6,9])
print("True Division of elements in array \'a\' by array \'b\' :",true_divide(a,b))
```

```
True Division of elements in array 'a' by array 'b' : [6.66666667 8.33333333 6.6666666
7]
```

7.6 floor_divide(arr1,arr2...) - Returns floor division

In [26]:
```
import numpy as np
a = np.array([20,50,60])
b= np.array([3,6,9])
print("Floor Division of elements in array \'a\' by array \'b\' :",np.floor_divide(a
```

```
Floor Division of elements in array 'a' by array 'b' : [6 8 6]
```

In [27]:
```
from numpy import*
a = np.array([20,50,60])
b= np.array([3,6,9])
print("Floor Division of elements in array \'a\' by array \'b\' :",floor_divide(a,b)
```

```
Floor Division of elements in array 'a' by array 'b' : [6 8 6]
```

7.7 remainder(arr1,arr2,...) - Returns remainder

In [28]:
```
import numpy as np
a = np.array([20,50,60])
b = np.array([3,6,9])
print("Remainder of array \'a\' elements divided by array \'b\' :",np.remainder(a,b)
```

```
Remainder of array 'a' elements divided by array 'b' : [2 2 6]
```

In [29]:
```
from numpy import*
a = array([20,50,60])
b = array([3,6,9])
print("Remainder of array \'a\' elements divided by array \'b\' :",remainder(a,b))
```

```
Remainder of array 'a' elements divided by array 'b' : [2 2 6]
```

7.8 mod(arr1,arr2,...) - Returns remainder after division

In [30]:
```
import numpy as np
a = np.array([20,50,60])
b = np.array([3,6,9])
print("Remainder of array \'a\' elements divided by array \'b\' :",np.mod(a,b))
```

```
Remainder of array 'a' elements divided by array 'b' : [2 2 6]
```

In [31]:
```
from numpy import*
a = array([20,50,60])
b = np.array([3,6,9])
print("Remainder of array \'a\' elements divided by array \'b\' :",mod(a,b))
```

```
Remainder of array 'a' elements divided by array 'b' : [2 2 6]
```

7.9 fmod(arr1,arr2,...) - Returns floor of the remainder after division

In [32]:
```
import numpy as np
a = np.array([20,50,60])
b = np.array([3,6,9])
print("Remainder of array \'a\' elements divided by array \'b\' :",np.fmod(a,b))
```

```
Remainder of array 'a' elements divided by array 'b' : [2 2 6]
```

In [33]:
```
from numpy import*
a = array([20,50,60])
b = array([3,6,9])
print("Remainder of array \'a\' elements divided by array \'b\' :",fmod(a,b))
```

```
Remainder of array 'a' elements divided by array 'b' : [2 2 6]
```

7.10 divmod(arr1,arr2,..) - Returns quotient and Remainder

In [34]:
```
import numpy as np
a = np.array([20,50,60])
b = np.array([3,6,9])
print("Quotient and Remainder of array \'a\' divided by array \'b\' :",np.divmod(a,b
```

```
Quotient and Remainder of array 'a' divided by array 'b' : (array([6, 8, 6]), array([2,
2, 6]))
```

In [35]:
```
from numpy import*
a = array([20,50,60])
b = array([3,6,9])
print("Quotient and Remainder of array \'a\' divided by array \'b\' :",divmod(a,b))
```

```
Quotient and Remainder of array 'a' divided by array 'b' : (array([6, 8, 6]), array([2,
2, 6]))
```

7.11 power(arr1,arr2....) - array1 elements raised to the power of array 2 elements

In [36]:
```
import numpy as np
a = np.array([2,5,6])
b = np.array([4,2,3])
print("elements of array \'a\' raised to array \'b\' is:",np.power(a,b))
```

```
elements of array 'a' raised to array 'b' is: [ 16  25 216]
```

In [37]:
```
from numpy import*
a = array([2,5,6])
b = array([4,2,3])
print("elements of array \'a\' raised to array \'b\' is:",power(a,b))
```

```
elements of array 'a' raised to array 'b' is: [ 16  25 216]
```

7.12 float_power(arr1,arr2...) - Usable for negative powers

In [38]:
```
import numpy as np
a = np.array([2,5,6])
b= np.array([-4,2,-3])
print("elements of array \'a\' raised to array \'b\' is:",np.float_power(a,b))
```

```
elements of array 'a' raised to array 'b' is: [6.25000000e-02 2.50000000e+01 4.62962963
e-03]
```

In [39]:
```
from numpy import*
a = array([2,5,6])
b= array([-4,-2,-3])
print("elements of array \'a\' raised to array \'b\' is:",float_power(a,b))
```

```
elements of array 'a' raised to array 'b' is: [0.0625     0.04       0.00462963]
```

7.13 reciprocal(arr) - returns reciprocal of the elements in an array

In [40]:
```
import numpy as np
a = np.array([2.0,5.0,6.0])
b= np.array([4.0,2.0,3.0])
print("Reciprocal of elements in array \'a\'",np.reciprocal(a))
print("Reciprocal of elements in array \'b\'",np.reciprocal(b))
```

```
Reciprocal of elements in array 'a' [0.5        0.2        0.16666667]
Reciprocal of elements in array 'b' [0.25       0.5        0.33333333]
```

In [41]:
```
from numpy import*
a = array([2.0,5.0,6.0])
b = array([4.0,2.0,3.0])
print("Reciprocal of elements in array \'a\'",reciprocal(a))
print("Reciprocal of elements in array \'b\'",reciprocal(b))
```

```
Reciprocal of elements in array 'a' [0.5        0.2        0.16666667]
Reciprocal of elements in array 'b' [0.25       0.5        0.33333333]
```

7.14 negative(arr) - Converts positive elements to negative and viceversa.

In [42]:
```
import numpy as np
a = np.array([2.0,5.0,6.0])
b= np.array([4.0,2.0,3.0])
print("Negative of a:",np.negative(a))
print("Negative of b:",np.negative(b))
```

```
Negative of a: [-2. -5. -6.]
Negative of b: [-4. -2. -3.]
```

In [43]:
```
from numpy import*
a = array([2.0,5.0,6.0])
b = array([4.0,2.0,3.0])
print("Negative of a:",negative(a))
print("Negative of b:",negative(b))
```

```
Negative of a: [-2. -5. -6.]
Negative of b: [-4. -2. -3.]
```

7.15 sum(arr) - Returns sum of all elements in an array.

In [44]:
```
import numpy as np
a = np.array([16,25,256])
print("Sum of all elements in array a:",np.sum(a))
```

```
Sum of all elements in array a: 297
```

In [45]:
```
from numpy import*
a = array([16,25,256])
print("Sum of all elements in array a:",sum(a))
```

```
Sum of all elements in array a: 297
```

7.16 prod(arr) - Returns product of all elements in an array

In [46]:
```
import numpy as np
a = np.array([16,25,256])
print("Product of all elements in array a:",np.prod(a))
```

```
Product of all elements in array a: 102400
```

In [47]:
```
from numpy import*
a = array([16,25,256])
print("Product of all elements in array a:",prod(a))
```

```
Product of all elements in array a: 102400
```

7.17 mean (arr) - Returns the mean/average of all elements in an array

In [48]:
```
import numpy as np
a = np.array([16,25,256])
print("Mean of all elements in array a:",np.mean(a))
```

```
Mean of all elements in array a: 99.0
```

In [49]:
```
from numpy import*
a = array([16,25,256])
print("Mean of all elements in array a:",mean(a))
```

```
Mean of all elements in array a: 99.0
```

7.18 median(arr) - Returns the median of the elements in an array

In [50]:
```
import numpy as np
a = np.array([16,25,256])
print("Median of all elements in array a:",np.median(a))
```

```
Median of all elements in array a: 25.0
```

In [51]:
```
from numpy import*
a = array([16,25,256])
print("Median of all elements in array a:",median(a))
```

```
Median of all elements in array a: 25.0
```

7.19 var(arr) - Returns the variance of elements in an array

In [52]:
```
import numpy as np
a = np.array([16,25,256])
print("Variance of all elements in array a:",np.var(a))
```

```
Variance of all elements in array a: 12338.0
```

In [53]:
```
from numpy import*
a = array([16,25,256])
print("Variance of all elements in array a:",var(a))
```

```
Variance of all elements in array a: 12338.0
```

7.20 cov(arr) - Returns the covariance of elements in an array

In [54]:
```
import numpy as np
a = np.array([16,25,256])
print("Covariance of all elements in array a:",np.cov(a))
```

```
Covariance of all elements in array a: 18507.0
```

In [55]:
```
from numpy import*
a = array([16,25,256])
print("Covariance of all elements in array a:",cov(a))
```

```
Covariance of all elements in array a: 18507.0
```

7.21 std(arr) - Returns the standard deviation of all elements in array

In [56]:
```
import numpy as np
a = np.array([16,25,256])
print("standard deviation of all elements in array a:",np.std(a))
```

```
standard deviation of all elements in array a: 111.07655018049489
```

In [57]:
```
from numpy import*
a = array([16,25,256])
print("standard deviation of all elements in array a:",std(a))
```

```
standard deviation of all elements in array a: 111.07655018049489
```

8. Trigonometric Functions on Arrays

8.1 sin(arr) - Returns trigonometric sine of an element in arrays

In [58]:
```
from numpy import*
a = array([0,30,45,60,90])
print("sine value of all elements in array a:",sin(a))
```

```
sine value of all elements in array a: [ 0.         -0.98803162  0.85090352 -0.30481062
0.89399666]
```

8.2 cos(arr) - Returns trigonometric cosine of an element in arrays

In [59]:
```
from numpy import*
a = array([0,30,45,60,90])
print("cosine value of all elements in array a:",cos(a))
```

```
cosine value of all elements in array a: [ 1.          0.15425145  0.52532199 -0.952412
98 -0.44807362]
```

8.3 tan(arr) - Returns trigonometric tan of an element in arrays

In [60]:
```
from numpy import*
a = array([0,30,45,60,90])
print("tan value of all elements in array a:",tan(a))
```

```
tan value of all elements in array a: [ 0.         -6.4053312   1.61977519  0.32004039
-1.99520041]
```

8.4 arcsin(arr) - Returns trigonometric inverse sine of an element in arrays

In [61]:
```
from numpy import*
a = array([-1,1])
print("inverse sine value of all elements in array a:",arcsin(a))
```

```
inverse sine value of all elements in array a: [-1.57079633  1.57079633]
```

8.5 arccos(arr) - Returns trigonometric inverse cos of an element in arrays

In [62]:
```
from numpy import*
a = array([-1,1])
print("inverse cosine values of all elements in array a:",arccos(a))
```

```
inverse cosine values of all elements in array a: [3.14159265 0.        ]
```

8.6 arctan(arr) - Returns trigonometric inverse tan of an element in arrays

In [63]:
```
from numpy import*
a = array([0,1,0])
print("Inverse tan values of all elements in array a:",arctan(a))
```

```
Inverse tan values of all elements in array a: [0.         0.78539816 0.        ]
```

8.7 hypot(arr1,arr2) - Calculates hypotenuse length

In [64]:
```
from numpy import*
a = array([3,4,5])
b = array([4,12,13])
print("Inverse tan values of all elements in array a:",hypot(a,b))
```

```
Inverse tan values of all elements in array a: [ 5.         12.64911064 13.92838828]
```

8.8 degrees(arr) - converts radians to degrees

In [65]:
```
from numpy import*
a = array([0,1,0])
print("Converting radians to degrees of elements in array \'a\':",degrees(a))
```

```
Converting radians to degrees of elements in array 'a': [ 0.         57.29577951  0.
]
```

8.9 rad2deg(arr) - converts radians to degrees

In [66]:
```
from numpy import*
a = array([0,1,0])
print("Converting radians to degrees of elements in array \'a\':",rad2deg(a))
```

```
Converting radians to degrees of elements in array 'a': [ 0.         57.29577951  0.
]
```

8.10 radians(arr) - Converts degrees to radians

In [67]:
```
from numpy import*
a = array([0,1,0])
print("Converting degrees to radians of elements in array \'a\':",radians(a))
```

```
Converting degrees to radians of elements in array 'a': [0.         0.01745329 0.
]
```

9. Logarithmic and exponential Functions

9.1 exp() - Returns exponential of an input array element wise

In [68]:
```
import numpy as np
a = np.array([1,2,3])
print("exponential value of the elements in array a:",np.exp(a))
```

```
exponential value of the elements in array a: [ 2.71828183  7.3890561  20.08553692]
```

9.2 expm1() - Returns exponential exp(x)-1 of an input array element wise

In [69]:
```
import numpy as np
a = np.array([1,2,3])
print("exponential value of the elements  -1 in array a:",np.expm1(a))
```

```
exponential value of the elements  -1 in array a: [ 1.71828183  6.3890561  19.08553692]
```

9.3 exp2() - Returns exponential 2**x of all elements in an array

In [70]:
```
import numpy as np
a = np.array([1,2,3])
print("2 to the power of elements in array a:",np.exp2(a))
```

```
2 to the power of elements in array a: [2. 4. 8.]
```

9.4 log(arr) - Returns natural log of an element in array.

In [71]:
```
1 import numpy as np
2 a = np.array([1,2,3])
3 print("Logarithmic value of all elements in array a:",log(a))
```

```
Logarithmic value of all elements in array a: [0.         0.69314718 1.09861229]
```

9.5 log10(arr) - Returns log base 10 of an input array element wise

In [72]:
```
1 import numpy as np
2 a = np.array([1,2,3])
3 print("Logarithmic base 10 value of all elements in array a:",log10(a))
```

```
Logarithmic base 10 value of all elements in array a: [0.         0.30103    0.4771212
5]
```

9.6 log2(arr) - Returns log base 2 of an input array element wise

In [73]:
```
1 import numpy as np
2 a = np.array([16,25,256])
3 print("standard deviation of all elements in array a:",log2(a))
```

```
standard deviation of all elements in array a: [4.         4.64385619 8.        ]
```

9.7 logaddexp(arr) - calculate Logarithm of the sum of exponentiations of the inputs.

In [74]:
```
1 import numpy as np
2 a = np.array([1,2,3])
3 b = np.array([2,4,6])
4 print("loagrithmic sum of exponentation of elements in arrays :",logaddexp(a,b))
```

```
loagrithmic sum of exponentation of elements in arrays : [2.31326169 4.12692801 6.04858
735]
```

9.8 logaddexp2(arr) - Returns logarithm of the sum of exponentiations of the inputs in base 2

In [75]:
```
1 import numpy as np
2 a = np.array([1,2,3])
3 b = np.array([2,4,6])
4 print("loagrithmic sum of exponentation of elements in arrays :",logaddexp2(a,b))
```

```
loagrithmic sum of exponentation of elements in arrays : [2.5849625  4.32192809 6.16992
5  ]
```

10. Rounding Functions

10.1 around(arr,decimal) - Rounds the elements of an input array upto given decimal places

In [76]:
```
1 import numpy as np
2 a = np.array([16.234567,25.26789543,256.334567644])
3 print("After rounding the elements in array a into decimal places:",around(a,3))
```

```
After rounding the elements in array a into decimal places: [ 16.235  25.268 256.335]
```

10.2 round_(arr,decimal)

In [77]:
```
1 import numpy as np
2 a = np.array([16.234567,25.26789543,256.334567644])
3 print("After rounding the elements in array a into decimal places:",round_(a,3))
```

```
After rounding the elements in array a into decimal places: [ 16.235  25.268 256.335]
```

10.3 rint(arr)- Round the elements of an array to the next highest integer towards zero

In [78]:
```
1 import numpy as np
2 a = np.array([16.794567,25.26789543,256.334567644])
3 print("After rounding of all elements in array a:",rint(a))
```

```
After rounding of all elements in array a: [ 17.  25. 256.]
```

10.4 fix(arr) - Round the elements of an array to the next lowest integer towards zero

In [79]:
```
1 import numpy as np
2 a = np.array([16.794567,25.99789543,256.934567644])
3 print("After rounding of all elements in array a:",fix(a))
```

```
After rounding of all elements in array a: [ 16.  25. 256.]
```

10.5 floor(arr) - Returns floor of an array element

In [80]:
```
1 import numpy as np
2 a = np.array([16.794567,25.99789543,256.934567644])
3 print("after rounding of all elements in array a:",floor(a))
```

```
after rounding of all elements in array a: [ 16.  25. 256.]
```

10.6 ceil(arr) - Returns ceiling of an array element wise

In [81]:
```
1 import numpy as np
2 a = np.array([16.294567,25.99789543,256.934567644])
3 print("After rounding of all elements in array a:",ceil(a))
```

```
After rounding of all elements in array a: [ 17.  26. 257.]
```

10.7 trunc(arr) - Return the truncated value of an array element wise

In [82]:
```
import numpy as np
a = np.array([16.994567,25.99789543,256.994567644])
print("Truncated values of all elements in array a:",trunc(a))
```

```
Truncated values of all elements in array a: [ 16.  25. 256.]
```

11. Miscellaneous Functions of arrays

11.1 sqrt(arr) - Returns the square root of the elements in an array.

In [83]:
```
import numpy as np
a = np.array([16,25.2,256])
print("Square root of a:",np.sqrt(a))
```

```
Square root of a: [ 4.          5.01996016 16.        ]
```

In [84]:
```
from numpy import*
a = array([16,25,256])
print("Square root of a:",sqrt(a))
```

```
Square root of a: [ 4.  5. 16.]
```

11.2 cbrt(arr) - Returns the cube root of the elements in an array.

In [85]:
```
import numpy as np
a = np.array([27,125,1000])
print("Cube root of a:",np.cbrt(a))
```

```
Cube root of a: [ 3.  5. 10.]
```

In [86]:
```
from numpy import*
a = array([27,125,1000])
print("Cube root of a:",cbrt(a))
```

```
Cube root of a: [ 3.  5. 10.]
```

11.3 abs(arr) - Returns absolue values of the elements in an array

In [87]:
```
import numpy as np
a = np.array([-2.56,5.34,-6.27])
print("Absolute value of the elements of array a:",np.abs(a))
```

```
Absolute value of the elements of array a: [2.56 5.34 6.27]
```

In [88]:
```
from numpy import*
a = array([-2.56,5.34,-6.27])
print("Absolute value of the elements of array a:",abs(a))
```

```
Absolute value of the elements of array a: [2.56 5.34 6.27]
```

11.4 max(arr) - Returns the maximum value of elements in an array

In [89]:
```
import numpy as np
a = np.array([16,25,256])
print("Maximum of all elements in array a:",np.max(a))
```

```
Maximum of all elements in array a: 256
```

In [90]:
```
from numpy import*
a = array([16,25,256])
print("Maximum of all elements in array a:",max(a))
```

```
Maximum of all elements in array a: 256
```

11.5 maximum(arr1,arr2) - Returns the maximum value of elements in an arrays in column wise.

In [91]:
```
import numpy as np
a = np.array([16,25,256])
b = np.array([1,50,3])
print("Maximum element in array a and b in column wise:",np.maximum(a,b))
```

```
Maximum element in array a and b in column wise: [ 16  50 256]
```

11.6 min(arr) - Returns minimum value of the elements in an array

In [92]:
```
import numpy as np
a = np.array([16,25,256])
print("Minimum of all elements in array a:",np.min(a))
```

```
Minimum of all elements in array a: 16
```

In [93]:
```
from numpy import*
a = array([16,25,256])
print("Minimum of all elements in array a:",min(a))
```

```
Minimum of all elements in array a: 16
```

11.7 minimum(arr1,arr2) - Returns the minimum value of elements in an arrays in column wise

In [94]:
```
import numpy as np
a = np.array([16,25,256])
b = np.array([1,26,3])
print("Minimum element in array a and b in column wise:",np.minimum(a,b))
```

```
Minimum element in array a and b in column wise: [ 1 25  3]
```

11.8 interp(arr,xp,fp) - Returns the one-dimensional piecewise linear interpolant to a function with given discrete data points (xp, fp), evaluated at x.

In [95]:
```
import numpy as np
x = [0, 1, 2.5, 2.72, 3.14]
xp = [2, 4, 6]
fp = [1, 3, 5]

a = np.interp(x, xp, fp)

print (a)
```

```
[1.   1.   1.5  1.72 2.14]
```

11.9 convolve(arr,v) - Returns the discrete, linear convolution of two one-dimensional sequences.

In [96]:
```
import numpy as np
a = [2, 4, 6]
v = [1, 3, 5]

b = np.convolve(a, v)

print (b)
```

```
[ 2 10 28 38 30]
```

11.10 clip(arr,arrmin,arrmax) - Clip (limit) the values in an array.

In [97]:
```
import numpy as np
a = array([1,2,3,4,5,6,7,8,9,10])
b=clip(a,3,7)

print ('After clipping the elements are:',b)
```

```
After clipping the elements are: [3 3 3 4 5 6 7 7 7 7]
```

11.11 sort(arr) - sorts the elements in ascending order

In [98]:
```
import numpy as np
a = array([100,25,3,46,5,65,17,86,9,10])
b=sort(a)

print ('The elemnets of array a sorted in ascending order are:',b)
```

```
The elemnets of array a sorted in ascending order are: [  3   5   9  10  17  25  46  65
86 100]
```

11.12 unique(arr) - Returns the unique elements in the list

In [99]:

```
import numpy as np
a = array([100,100,100,250,36,46,56,36])
b=unique(a)

print ('The Unique values from the array a are:',b)
```

```
The Unique values from the array a are: [ 36  46  56 100 250]
```

11.13 argmin(arr) - returns indices of the min element of the array in a particular axis.

In [100]:

```
import numpy as np
a = array([100,100,100,250,36,46,56,36])
b=argmin(a)

print ('The index position of the minimum element in array a is',b)
```

```
The index position of the minimum element in array a is 4
```

11.14 argmax(arr) - returns indices of the max element of the array in a particular axis.

In [101]:

```
import numpy as np
a = array([100,100,100,250,36,46,56,36])
b=argmax(a)

print ('The index position of the maximum element in array a is',b)
```

```
The index position of the maximum element in array a is 3
```