

NumPy is a Python package. It stands for 'Numerical Python'. It is a library consisting of multidimensional array objects and a collection of routines for processing of array.

The most important object defined in NumPy is an N-dimensional array type called ndarray. It describes the collection of items of the same type. Items in the collection can be accessed using a zero-based index. Every item in an ndarray takes the same size of block in the memory. Each element in ndarray is an object of data-type object (called dtype)

Standard Python distribution doesn't come bundled with NumPy module. A lightweight alternative is to install NumPy using popular Python package installer, pip.

```
pip install numpy
```

Numpy functions

```
import numpy as np
```

```
array
```

1. >>np.array(object, dtype = None, ndmin = 0)

object: Any object exposing the array interface method returns an array, or any (nested) sequence

dtype: Desired data type of array, optional

ndmin: Specifies minimum dimensions of resultant array

Example 1

```
import numpy as np
```

```
a = np.array([1,2,3])
```

```
print(a)
```

The output is as follows –

```
[1, 2, 3]
```

Example 2

```
# more than one dimensions
```

```
import numpy as np
```

```
a = np.array([[1, 2], [3, 4]])
```

```
print(a)
```

The output is as follows –

```
[[1, 2]
```

```
[3, 4]]
```

Example 3

```
# minimum dimensions
```

```
import numpy as np
```

```
a = np.array([1, 2, 3,4,5], ndmin = 2)
```

```
print(a)
```

The output is as follows –

```
[[1, 2, 3, 4, 5]]
```

Example 4

```
# dtype parameter
```

```
import numpy as np
```

```
a = np.array([1, 2, 3], dtype = complex)
```

```
print (a)
```

The output is as follows –

```
[ 1.+0.j,  2.+0.j,  3.+0.j]
```

2. >>ndarray.shape

This array attribute returns a tuple consisting of array dimensions. It can also be used to resize the array.

Example 1

```
import numpy as np
```

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

```
print (a.shape)
```

The output is as follows –

```
(2, 3)
```

Example 2

this resizes the ndarray

```
import numpy as np
```

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

```
a.shape = (3,2)
```

```
print (a)
```

The output is as follows –

```
[[1, 2]
```

```
[3, 4]
```

```
[5, 6]]
```

3. >>ndarray.reshape

Example 1

NumPy also provides a reshape function to resize an array.

```
import numpy as np
```

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

```
b = a.reshape(3,2)
```

```
print (b)
```

The output is as follows –

```
[[1, 2]
```

```
[3, 4]
```

```
[5, 6]]
```

4. >>np.arange(strt,end,step)

Array with float values

Example 1

an array of evenly spaced numbers

```
import numpy as np
```

```
a = np.arange(24)
```

```
print( a)
```

The output is as follows –

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

```
Arr = np.arange(1,20,2.4)
```

```
print(Arr)
```

The output is as follows –

```
array([ 1. , 3.4, 5.8, 8.2, 10.6, 13. , 15.4, 17.8])
```

now reshape it

```
b = a.reshape(2,4)
```

```
print( b)
```

Mathematical Functions

```
import numpy as np
```

```
a = np.array([0,30,45,60,90])
```

```
print ('Sine of different angles:' )
```

Convert to radians by multiplying with pi/180

```
print(np.sin(a*np.pi/180))
```

```
print ('Cosine values for angles in array:')
```

```
print (np.cos(a*np.pi/180) )
```

```
print ('Tangent values for given angles:')
```

```
print (np.tan(a*np.pi/180))
```

```
print('value of pi')
```

```
print(np.pi)
```

Functions for Rounding

```
numpy.around()
```

This is a function that returns the value rounded to the desired precision. The function takes the following parameters.

```
numpy.around(a,decimals)
```

The number of decimals to round to. Default is 0. If negative, the integer is rounded to position to the left of the decimal point

Example

```
import numpy as np
```

```
a = np.array([1.0,5.55, 123, 0.567, 25.532])
```

```
print ('Original array:')
```

```
print (a)
```

```
print ('After rounding:')
```

```
print (np.around(a) )
```

```
print (np.around(a, decimals = 1) )
```

numpy.floor()

This function returns the largest integer not greater than the input parameter. The floor of the scalar x is the largest integer i , such that $i \leq x$. Note that in Python, flooring always is rounded away from 0.

Example

```
import numpy as np  
a = np.array([-1.7, 1.5, -0.2, 0.6, 10])
```

```
print( 'The given array:')
```

```
print (a)
```

```
print ('The modified array:')
```

```
print (np.floor(a))
```

It produces the following output –

The given array:

```
[ -1.7  1.5 -0.2  0.6 10.]
```

The modified array:

```
[ -2.  1. -1.  0. 10.]
```

numpy.ceil()

The `ceil()` function returns the ceiling of an input value, i.e. the ceil of the scalar x is the smallest integer i , such that $i \geq x$.

Example

```
import numpy as np  
a = np.array([-1.7, 1.5, -0.2, 0.6, 10])
```

```
print( 'The given array:' )
```

```
print (a)
```

```
print( 'The modified array:' )
```

```
print( np.ceil(a))
```

It will produce the following output –

The given array:

```
[ -1.7  1.5 -0.2  0.6 10. ]
```

The modified array:

```
[ -1.  2. -0.  1. 10.]
```

NumPy - Arithmetic Operations

Input arrays for performing arithmetic operations such as `add()`, `subtract()`, `multiply()`, `dot()` and `divide()` must be either of the same shape or should conform to array broadcasting rules

Example

```
import numpy as np
```

```
a = np.arange(9, dtype = np.float_).reshape(3,3)
```

```
print ('First array:')
```

```
print (a)
```

```
print ('Second array:')
```

```
b = np.array([10,10,10])
```

```
print (b)
```

```
print( 'Add the two arrays:')
```

```
print (np.add(a,b) )
```

```
print ('Subtract the two arrays:')
```

```
print (np.subtract(a,b))
```

```
print ('Multiply the two arrays:')
```

```
print (np.multiply(a,b) )
```

```
print ('Divide the two arrays:')  
print (np.divide(a,b)  
print('Dot product two arrays:')  
print(np.dot(a,b))
```

```
np.product(A) : product at given axis  
np.sum(A) : sum  
np.max(A) : maximum  
np.min(A) : Minimum  
np.mean(A) : Returns the average of the array elements  
np.var() : Returns the variance of the array elements  
np.std() : standard deviations
```

Arrays with different functions

linspace() will create arrays with a specified number of elements, and spaced equally between the specified beginning and end values. For example:

```
>>> np.linspace(1., 4., 6)  
array([ 1. , 1.6, 2.2, 2.8, 3.4, 4. ])  
  
>>> np.zeros((2,3))  
array([[0., 0., 0.],  
       [0., 0., 0.]])  
  
>>> np.ones((2,3))  
array([[1., 1., 1.],  
       [1., 1., 1.]])  
  
>>> np.empty((2,4))  
array([[ 1. , 3.4, 5.8, 8.2],  
       [10.6, 13. , 15.4, 17.8]])
```



```
>>> np.full((2,2), 3)
```

```
array([[3, 3],
```

```
       [3, 3]])
```

```
>>> np.eye(3)
```

```
array([[1., 0., 0.],
```

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

```
       [0., 0., 1.]])
```