NUMPY

1. NumPy is a Python package that stands for ‘Numerical Python’.
2. Used to deal with multi-dimensional data.
3. The core of Numpy is the “ndarray” object.

Ndarray and its Attributes

ndarray is the collection of items of the same type. Items in the collection can be accessed using a zero-base 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).

Syntax

```
numpy.array(object, dtype = None, copy = True, order = None,
subok = False, ndmin = 0)
```

Sr.No.	Parameter & Description
1	object: Any object exposing the array interface method returns an array, or any (nested) sequence.
2	dtype: Desired data type of array, optional
3	copy:Optional. By default (true), the object is copied

4	order:C (row major) or F (column major) or A (any) (default)
5	subok:By default, returned array forced to be a base class array. If true, sub-classes passed through
6	ndmin:Specifies minimum dimensions of resultant array

Examples

```
1)
import numpy as np
a = np.array([1,2,3])
print a

Output- [1, 2, 3]

2)
# more than one dimensions
import numpy as np
a = np.array([[1, 2], [3, 4]])
print a

Output- [[1, 2]
          [3, 4]]

3)
# minimum dimensions
import numpy as np
a = np.array([1, 2, 3,4,5], ndmin = 2)
print a

Output- [[1, 2, 3, 4, 5]]


4)
# dtype parameter
import numpy as np
```

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

Output- [1.+0.j, 2.+0.j, 3.+0.j]

5)

```
import numpy as np
a = np.array([[1,2,3],[4,5,6]])
print a.shape
```

Output- (2, 3)

6)

```
# this resizes the ndarray
import numpy as np
a = np.array([[1,2,3],[4,5,6]])
a.shape = (3,2)
print a
```

```
Output- [[1, 2]
          [3, 4]
          [5, 6]]
```

Create a NumPy ndarray Object

NumPy is used to work with arrays. The array object in NumPy is called ndarray.

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
print(arr)
print(type(arr))
```

Dimensions in Arrays

0-D Arrays

0-D arrays, or Scalars, are the elements in an array.
Example:

```
import numpy as np
arr = np.array(42)
print(arr)
```

1-D Arrays

An array that has 0-D arrays as its elements is called a uni-dimensional or 1-D array.
Example:

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
print(arr)
```

2-D Arrays

An array that has 1-D arrays as its elements is called a 2-D array.
Example:

```
import numpy as np
arr = np.array([[1, 2, 3], [4, 5, 6]])
print(arr)
```

NumPy Array Indexing and Slicing

Slicing

Slicing in python means taking elements from one given index to another given index.

We pass slice instead of index like this: [*start*:*end*].

We can also define the step, like this: [*start*:*end*:*step*].

-If we don't pass start its considered 0

-If we don't pass end its considered length of array in that dimension

-If we don't pass step it’s considered 1

Example 1:

```
import numpy as np
```

```
arr = np.array([1, 2, 3, 4, 5, 6, 7])
print(arr[1:5]) --->  [2,3,4,5]
```

Example 2:

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5, 6, 7])
print(arr[4:])  ---> [5,6,7]
```

Example 3:

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5, 6, 7])
print(arr[::2]) --->[1,3,5,7]
```

Indexing

“Indexing” means referring to an element of an iterable by its position within the iterable.

```
a = ['spam', 'egg', 'bacon', 'tomato', 'ham', 'lobster']

a[5] --> 'lobster’

a[len(a)-1] --> 'lobster'

a[-1] --> ‘lobster’
```

NumPy - Mathematical Functions

Trigonometric Functions:

NumPy has standard trigonometric functions which return trigonometric ratios for a given angle in radians.

Example:

```
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 '\n'

print 'Cosine values for angles in array:'
print np.cos(a*np.pi/180)
print '\n'
```

<u>Here is its output −</u>

```
Sine of different angles:
[ 0.          0.5         0.70710678  0.8660254   1.        ]

Cosine values for angles in array:
[  1.00000000e+00   8.66025404e-01   7.07106781e-01   5.00000000e-01
   6.12323400e-17]
```

arcsin, arcos, and arctan functions return the trigonometric inverse of sin, cos, and tan of the given angle. The result of these functions can be verified by numpy.degrees() function by converting radians to degrees.

Example

```
import numpy as np

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

print 'Array containing sine values:'
sin = np.sin(a*np.pi/180)
print sin
print '\n'

print 'Compute sine inverse of angles. Returned values are in radians.'
inv = np.arcsin(sin)
print inv
print '\n'

print 'Check result by converting to degrees:'
print np.degrees(inv)
print '\n'
```

Its output is as follows −

```
Array containing sine values:
[ 0.          0.5         0.70710678  0.8660254   1.        ]
```

```
Compute sine inverse of angles. Returned values are in radians.
[ 0.         0.52359878  0.78539816  1.04719755  1.57079633]

Check result by converting to degrees:
[  0.  30.  45.  60.  90.]
```

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)

Where,

Sr.No.	Parameter & Description
1	a: Input data
2	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 '\n'

print 'After rounding:'
print np.around(a)
print np.around(a, decimals = 1)
print np.around(a, decimals = -1)
```

It produces the following output −

```
Original array:
[   1.       5.55   123.       0.567   25.532]

After rounding:
[   1.    6.   123.    1.   26. ]
[   1.    5.6  123.    0.6  25.5]
[   0.    10.  120.    0.   30. ]
```

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 <= 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 '\n'

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 >= 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 '\n'

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.]
```

It produces the following output −

```
Original array:
[   1.       5.55   123.       0.567   25.532]

After rounding:
[   1.    6.   123.    1.   26. ]
[   1.    5.6  123.    0.6  25.5]
[   0.    10.  120.    0.   30. ]
```