NumPy - Mathematical Functions

Quite understandably, NumPy contains a large number of various mathematical operations. NumPy provides standard trigonometric functions, functions for arithmetic operations, handling complex numbers, etc.

Trigonometric Functions

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

Example

Live Demo

```
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'


print 'Tangent values for given angles:'

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

Here is its output −

```
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]

Tangent values for given angles:
[  0.00000000e+00   5.77350269e-01   1.00000000e+00   1.73205081e+00
   1.63312394e+16]
```

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

Live Demo

```
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'


print 'arccos and arctan functions behave similarly:'

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

```
print cos

print '\n'


print 'Inverse of cos:'

inv = np.arccos(cos)

print inv

print '\n'


print 'In degrees:'

print np.degrees(inv)

print '\n'


print 'Tan function:'

tan = np.tan(a*np.pi/180)

print tan

print '\n'


print 'Inverse of tan:'

inv = np.arctan(tan)

print inv

print '\n'


print 'In degrees:'

print np.degrees(inv)
```

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.]

arccos and arctan functions behave similarly:
[  1.00000000e+00   8.66025404e-01   7.07106781e-01   5.00000000e-01
   6.12323400e-17]

Inverse of cos:
[ 0.          0.52359878  0.78539816  1.04719755  1.57079633]

In degrees:
[  0.  30.  45.  60.  90.]

Tan function:
[  0.00000000e+00   5.77350269e-01   1.00000000e+00   1.73205081e+00
   1.63312394e+16]

Inverse of tan:
[ 0.          0.52359878  0.78539816  1.04719755  1.57079633]

In 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

Live Demo

```
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

Live Demo

```
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

Live Demo

```
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.]
```