NumPy - Funções Matemáticas

Compreensivelmente, o NumPy contém um grande número de diversas operações matemáticas. NumPy fornece funções trigonométricas padrão, funções para operações aritméticas, manipulação de números complexos, etc.

Funções trigonométricas

NumPy possui funções trigonométricas padrão que retornam razões trigonométricas para um determinado ângulo em radianos.

Exemplo

Demonstração ao vivo

```
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)
```

Aqui está o resultado -

```
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]
```

As funções arcsin, arcos e **arctan** retornam o inverso trigonométrico de sen, cos e tan do ângulo fornecido. O resultado dessas funções pode ser verificado pela **função numpy.degrees()** convertendo radianos em graus.

Exemplo

Demonstração ao vivo

```
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)
```

Sua saída é a seguinte -

```
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.]
```

Funções para arredondamento

numpy.around()

Esta é uma função que retorna o valor arredondado para a precisão desejada. A função usa os seguintes parâmetros.

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

Onde,

Sr. Não.	Parâmetro e Descrição
1	**a** Dados de entrada
2	**decimais** O número de casas decimais para arredondar. O padrão é 0. Se for negativo, o número inteiro será arredondado para a posição à esquerda da vírgula decimal

Exemplo

Demonstração ao vivo

```
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)
```

Ele produz a seguinte saída -

```
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()

Esta função retorna o maior número inteiro não maior que o parâmetro de entrada. O piso do **escalar x** é o maior **inteiro i** , tal que **i <= x** . Observe que em Python, o piso sempre é

arredondado de 0.

Exemplo

```
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)
```

Demonstração ao vivo

Ele produz a seguinte saída -

```
The given array:
[ -1.7  1.5  -0.2  0.6  10. ]

The modified array:
[ -2.  1.  -1.  0.  10.]
```

numpy.ceil()

A função ceil() retorna o teto de um valor de entrada, ou seja, o teto do **escalar x** é o menor **inteiro i** , tal que **i >= x.**

Exemplo

```
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)
```

Demonstração ao vivo

Ele produzirá a seguinte saída -

```
The given array:
[ -1.7   1.5  -0.2   0.6  10. ]

The modified array:
[ -1.   2.  -0.   1.  10.]
```