NumPy - Arithmetic Operations

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

Example

Live Demo

```
import numpy as np

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


print 'First array:'

print a

print '\n'


print 'Second array:'

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

print b

print '\n'


print 'Add the two arrays:'

print np.add(a,b)

print '\n'


print 'Subtract the two arrays:'

print np.subtract(a,b)

print '\n'
```

```
print 'Multiply the two arrays:'

print np.multiply(a,b)

print '\n'



print 'Divide the two arrays:'

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

It will produce the following output –

```
First array:
[[ 0. 1. 2.]
 [ 3. 4. 5.]
 [ 6. 7. 8.]]

Second array:
[10 10 10]

Add the two arrays:
[[ 10. 11. 12.]
 [ 13. 14. 15.]
 [ 16. 17. 18.]]

Subtract the two arrays:
[[-10. -9. -8.]
 [ -7. -6. -5.]
 [ -4. -3. -2.]]

Multiply the two arrays:
[[ 0. 10. 20.]
 [ 30. 40. 50.]
 [ 60. 70. 80.]]

Divide the two arrays:
[[ 0. 0.1 0.2]
 [ 0.3 0.4 0.5]
 [ 0.6 0.7 0.8]]
```

Let us now discuss some of the other important arithmetic functions available in NumPy.

numpy.reciprocal()

This function returns the reciprocal of argument, element-wise. For elements with absolute values larger than 1, the result is always 0 because of the way in which Python handles integer division. For integer 0, an overflow warning is issued.

Example

Live Demo

```
import numpy as np

a = np.array([0.25, 1.33, 1, 0, 100])


print 'Our array is:'

print a

print '\n'


print 'After applying reciprocal function:'

print np.reciprocal(a)

print '\n'


b = np.array([100], dtype = int)

print 'The second array is:'

print b

print '\n'


print 'After applying reciprocal function:'

print np.reciprocal(b)
```

It will produce the following output –

```
Our array is:
[   0.25    1.33    1.      0.    100.  ]

After applying reciprocal function:
main.py:9: RuntimeWarning: divide by zero encountered in reciprocal
  print np.reciprocal(a)
[ 4.         0.7518797  1.               inf  0.01     ]

The second array is:
[100]

After applying reciprocal function:
[0]
```

numpy.power()

This function treats elements in the first input array as base and returns it raised to the power of the corresponding element in the second input array.

Live Demo

```
import numpy as np

a = np.array([10,100,1000])


print 'Our array is:'

print a

print '\n'


print 'Applying power function:'

print np.power(a,2)

print '\n'


print 'Second array:'

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

print b

print '\n'


print 'Applying power function again:'

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

It will produce the following output –

```
Our array is:
[  10  100 1000]

Applying power function:
[    100   10000 1000000]

Second array:
[1 2 3]
```

```
Applying power function again:
[        10      10000 1000000000]
```

numpy.mod()

This function returns the remainder of division of the corresponding elements in the input array. The function **numpy.remainder()** also produces the same result.

Live Demo

```
import numpy as np

a = np.array([10,20,30])

b = np.array([3,5,7])


print 'First array:'

print a

print '\n'


print 'Second array:'

print b

print '\n'


print 'Applying mod() function:'

print np.mod(a,b)

print '\n'


print 'Applying remainder() function:'

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

It will produce the following output –

```
First array:
[10 20 30]
```

```
Second array:
[3 5 7]

Applying mod() function:
[1 0 2]

Applying remainder() function:
[1 0 2]
```

The following functions are used to perform operations on array with complex numbers.

- **numpy.real()** – returns the real part of the complex data type argument.
- **numpy.imag()** – returns the imaginary part of the complex data type argument.
- **numpy.conj()** – returns the complex conjugate, which is obtained by changing the sign of the imaginary part.
- **numpy.angle()** – returns the angle of the complex argument. The function has degree parameter. If true, the angle in the degree is returned, otherwise the angle is in radians.

Live Demo

```
import numpy as np

a = np.array([-5.6j, 0.2j, 11. , 1+1j])


print 'Our array is:'

print a

print '\n'


print 'Applying real() function:'

print np.real(a)

print '\n'


print 'Applying imag() function:'

print np.imag(a)

print '\n'
```

```
print 'Applying conj() function:'

print np.conj(a)

print '\n'


print 'Applying angle() function:'

print np.angle(a)

print '\n'


print 'Applying angle() function again (result in degrees)'

print np.angle(a, deg = True)
```

It will produce the following output –

```
Our array is:
[ 0.-5.6j 0.+0.2j 11.+0.j 1.+1.j ]

Applying real() function:
[ 0. 0. 11. 1.]

Applying imag() function:
[-5.6 0.2 0. 1. ]

Applying conj() function:
[ 0.+5.6j 0.-0.2j 11.-0.j 1.-1.j ]

Applying angle() function:
[-1.57079633 1.57079633 0. 0.78539816]

Applying angle() function again (result in degrees)
[-90. 90. 0. 45.]
```