```
import numpy as np
```

```
#creating ndarray from list note ndarray and array are one and same thing in numpy
l1=[3,4,5,7,1,2,3]
arr=np.array(l1)
arr
```

```
array([3, 4, 5, 7, 1, 2, 3])
```

```
#creating multidimensional array using list remember to use[] as if you use arr=np.array(l1,l2) it will generate error
l2=[3,4,5,6,1,2,3]
arr=np.array([l1,l2])
arr
```

```
array([[3, 4, 5, 7, 1, 2, 3],
       [3, 4, 5, 6, 1, 2, 3]])
```

```
#methods of ndarray
print(arr.ndim)
print(arr.dtype)
arr.shape
```

```
2
int32
(2, 7)
```

```
arr=arr.reshape(7,2)
arr
```

```
array([[3, 4],
       [5, 7],
       [1, 2],
       [3, 3],
       [4, 5],
       [6, 1],
       [2, 3]])
```

```
arr.ndim
```

```
2
```

```
# creating array consisting of only zeroes and an empty array
print(np.zeros((3,3)))
np.empty((3,3))
```

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

```
# creating array using arange() takes lower limit , upper limit,step as argument

arr2=np.arange(3,10,2)
arr2
```

```
array([3, 5, 7, 9])
```

```
# creating array consisting of only ones
arr3=np.ones((3,3))
arr3
```

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

```
# numpy array eye and identity
arr4=np.eye(3)
```

```
arr4
```

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

```
from IPython.display import Image

# Display an image from a local file
Image(filename=r'.\WhatsApp Image 2024-09-08 at 21.36.20_4fa43a41.jpg')
```

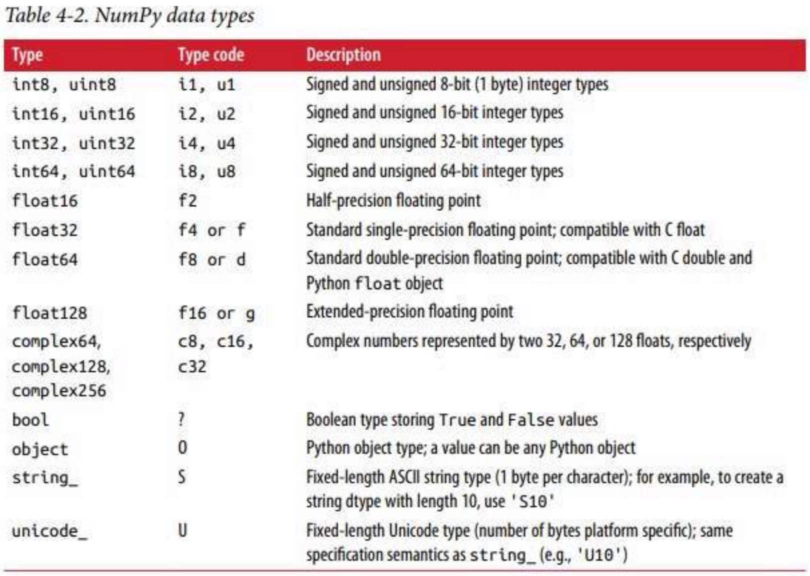


Table 4-2. NumPy data types

Type	Type code	Description
`int8, uint8`	`i1, u1`	Signed and unsigned 8-bit (1 byte) integer types
`int16, uint16`	`i2, u2`	Signed and unsigned 16-bit integer types
`int32, uint32`	`i4, u4`	Signed and unsigned 32-bit integer types
`int64, uint64`	`i8, u8`	Signed and unsigned 64-bit integer types
`float16`	`f2`	Half-precision floating point
`float32`	`f4 or f`	Standard single-precision floating point; compatible with C float
`float64`	`f8 or d`	Standard double-precision floating point; compatible with C double and Python `float` object
`float128`	`f16 or g`	Extended-precision floating point
`complex64, complex128, complex256`	`c8, c16, c32`	Complex numbers represented by two 32, 64, or 128 floats, respectively
`bool`	?	Boolean type storing `True` and `False` values
`object`	O	Python object type; a value can be any Python object
`string_`	S	Fixed-length ASCII string type (1 byte per character); for example, to create a string dtype with length 10, use `'S10'`
`unicode_`	U	Fixed-length Unicode type (number of bytes platform specific); same specification semantics as `string_` (e.g., `'U10'`)

```
from IPython.display import Image

# Display an image from a local file
Image(filename=r'.\WhatsApp Image 2024-09-08 at 21.58.28_38e647b9.jpg')
```

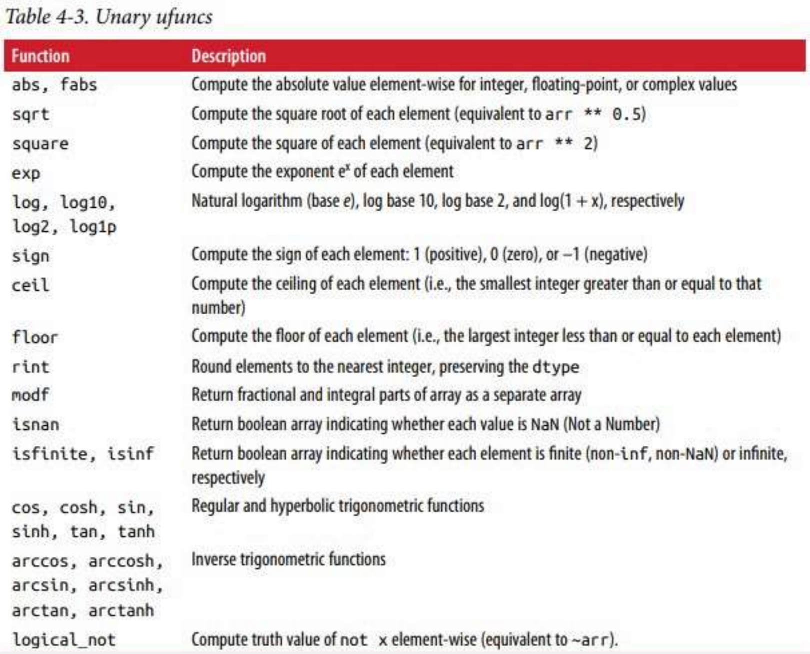


Table 4-3. Unary ufuncs

Function	Description
`abs, fabs`	Compute the absolute value element-wise for integer, floating-point, or complex values
`sqrt`	Compute the square root of each element (equivalent to `arr ** 0.5`)
`square`	Compute the square of each element (equivalent to `arr ** 2`)
`exp`	Compute the exponent e^x of each element
`log, log10, log2, log1p`	Natural logarithm (base *e*), log base 10, log base 2, and log(1 + x), respectively
`sign`	Compute the sign of each element: 1 (positive), 0 (zero), or –1 (negative)
`ceil`	Compute the ceiling of each element (i.e., the smallest integer greater than or equal to that number)
`floor`	Compute the floor of each element (i.e., the largest integer less than or equal to each element)
`rint`	Round elements to the nearest integer, preserving the `dtype`
`modf`	Return fractional and integral parts of array as a separate array
`isnan`	Return boolean array indicating whether each value is NaN (Not a Number)
`isfinite, isinf`	Return boolean array indicating whether each element is finite (non-`inf`, non-NaN) or infinite, respectively
`cos, cosh, sin, sinh, tan, tanh`	Regular and hyperbolic trigonometric functions
`arccos, arccosh, arcsin, arcsinh, arctan, arctanh`	Inverse trigonometric functions
`logical_not`	Compute truth value of `not x` element-wise (equivalent to `~arr`).

```
from IPython.display import Image

# Display an image from a local file
```

```
Image(filename=r'.\WhatsApp Image 2024-09-08 at 22.00.13_a473d6cf.jpg')
```

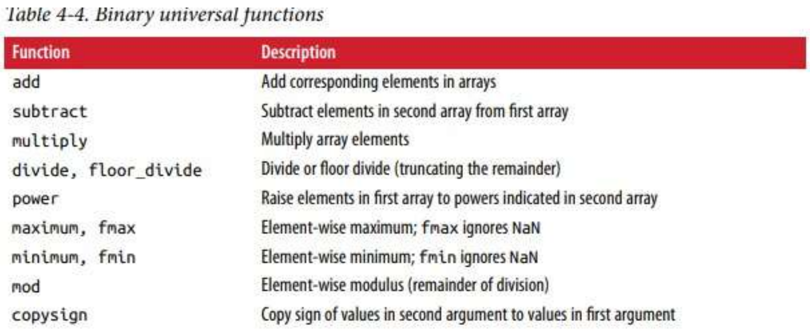


Table 4-4. Binary universal functions

Function	Description
add	Add corresponding elements in arrays
subtract	Subtract elements in second array from first array
multiply	Multiply array elements
divide, floor_divide	Divide or floor divide (truncating the remainder)
power	Raise elements in first array to powers indicated in second array
maximum, fmax	Element-wise maximum; fmax ignores NaN
minimum, fmin	Element-wise minimum; fmin ignores NaN
mod	Element-wise modulus (remainder of division)
copysign	Copy sign of values in second argument to values in first argument

```
from IPython.display import Image

# Display an image from a local file
Image(filename=r".\WhatsApp Image 2024-09-08 at 22.04.00_b6ce9469.jpg")
```

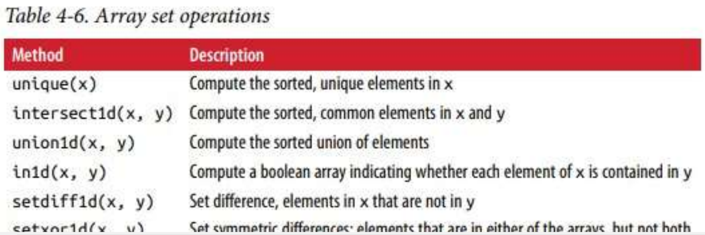
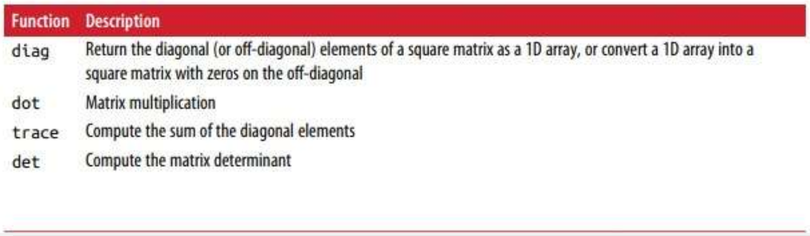


Table 4-6. Array set operations

Method	Description
unique(x)	Compute the sorted, unique elements in x
intersect1d(x, y)	Compute the sorted, common elements in x and y
union1d(x, y)	Compute the sorted union of elements
in1d(x, y)	Compute a boolean array indicating whether each element of x is contained in y
setdiff1d(x, y)	Set difference, elements in x that are not in y
setxor1d(x, y)	Set symmetric differences; elements that are in either of the arrays, but not both

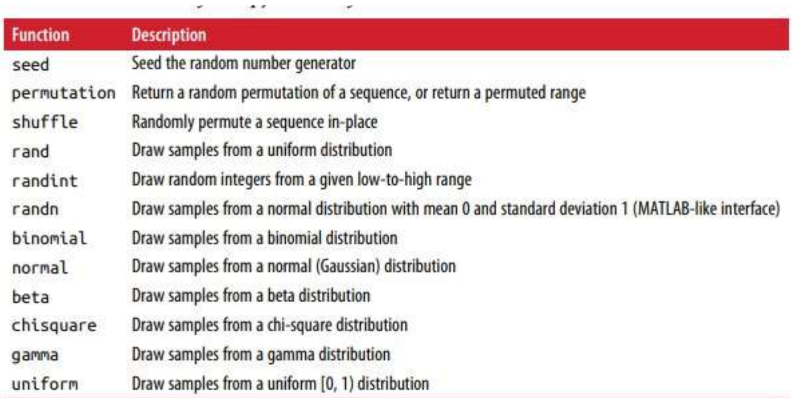
```
from IPython.display import Image

# Display an image from a local file
Image(filename=r".\WhatsApp Image 2024-09-08 at 22.04.29_8f31ba79.jpg")
```



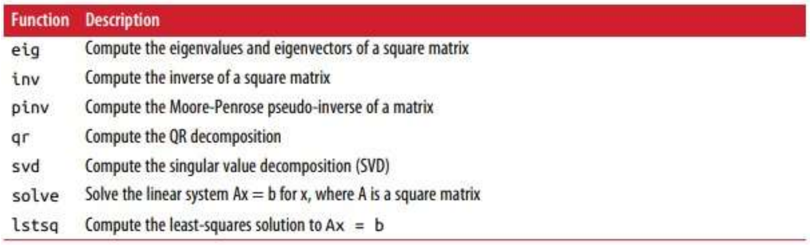
Function	Description
diag	Return the diagonal (or off-diagonal) elements of a square matrix as a 1D array, or convert a 1D array into a square matrix with zeros on the off-diagonal
dot	Matrix multiplication
trace	Compute the sum of the diagonal elements
det	Compute the matrix determinant

```
# Display an image from a local file
Image(filename=r".\WhatsApp Image 2024-09-08 at 22.05.23_7e1f09d5.jpg")
```



Function	Description
seed	Seed the random number generator
permutation	Return a random permutation of a sequence, or return a permuted range
shuffle	Randomly permute a sequence in-place
rand	Draw samples from a uniform distribution
randint	Draw random integers from a given low-to-high range
randn	Draw samples from a normal distribution with mean 0 and standard deviation 1 (MATLAB-like interface)
binomial	Draw samples from a binomial distribution
normal	Draw samples from a normal (Gaussian) distribution
beta	Draw samples from a beta distribution
chisquare	Draw samples from a chi-square distribution
gamma	Draw samples from a gamma distribution
uniform	Draw samples from a uniform [0, 1) distribution

```
# Display an image from a local file
Image(filename=r".\WhatsApp Image 2024-09-08 at 22.04.50_4ab76ef1.jpg")
```



Function	Description
eig	Compute the eigenvalues and eigenvectors of a square matrix
inv	Compute the inverse of a square matrix
pinv	Compute the Moore-Penrose pseudo-inverse of a matrix
qr	Compute the QR decomposition
svd	Compute the singular value decomposition (SVD)
solve	Solve the linear system Ax = b for x, where A is a square matrix
lstsq	Compute the least-squares solution to Ax = b

```
#saving arrays
np.save("some_arr",arr)
```

```
#loading arrays
np.load("some_arr.npy")
```

```
array([[3, 4],
       [5, 7],
       [1, 2],
       [3, 3],
       [4, 5],
       [6, 1],
       [2, 3]])
```

```
# slicing and boolean slicing
arr[arr>0]
```

```
array([3, 4, 5, 7, 1, 2, 3, 3, 4, 5, 6, 1, 2, 3])
```

```
#slicing
arr[1,:2]
```

```
array([5, 7])
```

Start coding or generate with AI.