numpyAll

January 28, 2026

[1]:
```
import numpy as np

# Check version
print(np.__version__)
```

```
1.24.3
```

[2]:
```
# Creating arrays
# From list
a = np.array([1, 2, 3])
b = np.array([[1, 2], [3, 4]])

# With datatype
c = np.array([1, 2, 3], dtype=float)

# Special arrays
zeros = np.zeros((2, 3))
ones = np.ones((3, 3))
full = np.full((2, 2), 7)

# Ranges
ar1 = np.arange(0, 10, 2)
ar2 = np.linspace(0, 1, 5)

# Identity
eye = np.eye(3)

print(a, b, zeros, ar1, ar2, eye)
```

```
[1 2 3] [[1 2]
 [3 4]] [[0. 0. 0.]
 [0. 0. 0.]] [0 2 4 6 8] [0.   0.25 0.5  0.75 1.  ] [[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

[3]:
```
# Array attributes
arr = np.array([[1, 2, 3], [4, 5, 6]])

print(arr.ndim)  # dimensions
print(arr.shape)  # rows, cols
print(arr.size)  # total elements
print(arr.dtype)  # data type
print(arr.itemsize)  # bytes per element
```

```
2
(2, 3)
6
int32
4
```

[4]:
```
# Indexing & Slicing
x = np.array([10, 20, 30, 40, 50])

print(x[0])
print(x[-1])
print(x[1:4])
print(x[::-1])  # reverse

y = np.array([[1, 2, 3], [4, 5, 6]])
print(y[0, 1])
print(y[:, 1])  # column
print(y[1, :])  # row
```

```
10
50
[20 30 40]
[50 40 30 20 10]
2
[2 5]
[4 5 6]
```

[5]:
```
# Boolean Indexing & where
a = np.array([10, 15, 20, 25, 30])

print(a[a > 20])

# where
res = np.where(a % 2 == 0, "even", "odd")
print(res)
```

```
[25 30]
['even' 'odd' 'even' 'odd' 'even']
```

[6]:
```
# reshaping & view vs copy
a = np.arange(6)

b = a.reshape((2, 3))
c = a.view()
```

```
d = a.copy()

a[0] = 100
print(b)  # affected
print(c)  # affected
print(d)  # not affected
```

```
[[100   1   2]
 [  3   4   5]]
[100   1   2   3   4   5]
[0 1 2 3 4 5]
```

[7]:
```
# Flattening
arr = np.array([[1, 2], [3, 4]])

print(arr.flatten())  # copy
print(arr.ravel())  # view (if possible)
```

```
[1 2 3 4]
[1 2 3 4]
```

[8]:
```
# Mathematical operations
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])

print(a + b)
print(a - b)
print(a * b)
print(a / b)
print(a**2)

print(np.add(a, b))
print(np.sqrt(a))
print(np.exp(a))
print(np.log(a))
```

```
[5 7 9]
[-3 -3 -3]
[ 4 10 18]
[0.25 0.4  0.5 ]
[1 4 9]
[5 7 9]
[1.         1.41421356 1.73205081]
[ 2.71828183  7.3890561  20.08553692]
[0.         0.69314718 1.09861229]
```

[9]:
```
# Aggregate Functions
arr = np.array([[1, 2, 3], [4, 5, 6]])
```

```
print(arr.sum())
print(arr.sum(axis=0))  # column-wise
print(arr.sum(axis=1))  # row-wise

print(arr.min())
print(arr.max())
print(arr.mean())
print(arr.std())
print(arr.var())
```

```
21
[5 7 9]
[ 6 15]
1
6
3.5
1.707825127659933
2.9166666666666665
```

[11]:
```
# sorting & searching
a = np.array([5, 2, 9, 1])

print(np.sort(a))
print(np.argsort(a))

# Searching
print(np.argmax(a))
print(np.argmin(a))
print(np.where(a > 3))
```

```
[1 2 5 9]
[3 1 0 2]
2
3
(array([0, 2], dtype=int64),)
```

[13]:
```
# unique,count
a = np.array([1, 2, 2, 3, 3, 3])

print(np.unique(a))
print(np.unique(a, return_counts=True))

# Stack, Split, Concatenate
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])

print(np.concatenate((a, b)))
print(np.vstack((a, b)))
print(np.hstack((a, b)))

# Split
print(np.split(b, 3))
```

```
[1 2 3]
(array([1, 2, 3]), array([1, 2, 3], dtype=int64))
[1 2 3 4 5 6]
[[1 2 3]
 [4 5 6]]
[1 2 3 4 5 6]
[array([4]), array([5]), array([6])]
```

[16]:
```
# Missing Values
a = np.array([1, np.nan, 3])

print(np.isnan(a))
print(np.nanmean(a))
print(np.nan_to_num(a))

# comparisons & logical operations
a = np.array([1, 2, 3])

print(a > 2)
print(np.any(a > 2))
print(np.all(a > 0))



# Type Casting
a = np.array([1.2, 2.8, 3.5])

print(a.astype(int))
print(a.astype(str))
```

```
[False  True False]
2.0
[1. 0. 3.]
[False False  True]
True
True
[1 2 3]
['1.2' '2.8' '3.5']
```

[18]:
```
# Linear algebra
A = np.array([[1, 2], [3, 4]])
B = np.array([[5, 6], [7, 8]])

print(np.dot(A, B))
print(A @ B)

print(np.linalg.det(A))
print(np.linalg.inv(A))
print(np.linalg.eig(A))


# random module
np.random.seed(42)

print(np.random.rand(3))
print(np.random.randn(3))
print(np.random.randint(1, 10, size=5))
print(np.random.choice([10, 20, 30], size=4))
```

```
[[19 22]
 [43 50]]
[[19 22]
 [43 50]]
-2.0000000000000004
[[-2.   1. ]
 [ 1.5 -0.5]]
(array([-0.37228132,  5.37228132]), array([[-0.82456484, -0.41597356],
       [ 0.56576746, -0.90937671]]))
[0.37454012 0.95071431 0.73199394]
[-1.11188012  0.31890218  0.27904129]
[8 3 6 5 2]
[20 20 20 10]
```

[ ]:
```
# Broadcasting
a = np.array([[1, 2, 3], [4, 5, 6]])
b = np.array([10, 20, 30])

print(a + b)  # broadcasting
```

```
[[11 22 33]
 [14 25 36]]
```