

03-11-2025

Complete Numpy Documentation

1.Array Creation Functions

```
In [1]: import numpy as np
```

```
In [2]: a = np.array([1, 2, 3])
print("Array a:", a)
```

Array a: [1 2 3]

```
In [3]: b = np.arange(0, 10, 2)
print("Array b:", b)
```

Array b: [0 2 4 6 8]

```
In [4]: d = np.zeros((2, 3),)
print("Array d:\n", d)
```

Array d:
[[0. 0. 0.]
 [0. 0. 0.]]

```
In [5]: d = np.zeros((2, 3),dtype=int)
print("Array d:\n",d)
```

Array d:
[[0 0 0]
 [0 0 0]]

```
In [6]: e = np.ones((3, 2))
print("Array e:\n", e)
```

Array e:
[[1. 1.]
 [1. 1.]
 [1. 1.]]

```
In [7]: f = np.eye(4)
print("Identity matrix f:\n", f)
```

Identity matrix f:
[[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]

2.Array Manipulation Functions

```
In [8]: a1 = np.array([1, 2, 3])
        reshaped = np.reshape(a1, (1, 3))
        print("Reshaped array:", reshaped)
```

Reshaped array: `[[1 2 3]]`

```
In [9]: a1 = np.array([1, 2, 3])
        reshaped = np.reshape(a1, (3, 1))
        print("Reshaped array:", reshaped)
```

Reshaped array: `[[1]
[2]
[3]]`

```
In [11]: f1 = np.array([[1, 2], [3, 4], [5, 6], [7, 8]])
        flattened = np.ravel(f1)
        print("Flattened array:", flattened)
```

Flattened array: `[1 2 3 4 5 6 7 8]`

```
In [12]: e1 = np.array([[1, 2], [3, 4]])
        transposed = np.transpose(e1)
        print("Transposed array:\n", transposed)
```

Transposed array:
`[[1 3]
[2 4]]`

```
In [13]: a2 = np.array([1, 2])
        b2 = np.array([3, 4])
        stacked = np.vstack([a2, b2])
        print("Stacked arrays:\n", stacked)
```

Stacked arrays:
`[[1 2]
[3 4]]`

3. Mathematical Functions

```
In [14]: g = np.array([1, 2, 3, 4])
        added = np.add(g, 2)
        print("Added 2 to g:", added)
```

Added 2 to g: `[3 4 5 6]`

```
In [15]: squared = np.power(g, 2)
        print("Squared g:", squared)
```

Squared g: `[1 4 9 16]`

```
In [16]: sqrt_val = np.sqrt(g)
        print("Square root of g:", sqrt_val)
```

Square root of g: `[1. 1.41421356 1.73205081 2.]`

```
In [17]: print(a1)
        print(g)
```

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

```
In [18]: print(a)
         print(a1)
```

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

```
In [19]: a3 = np.array([1, 2, 3])
         dot_product = np.dot(a1, a)
         print("Dot product of a1 and a:", dot_product)
```

Dot product of a1 and a: 14

4. Statistical Functions

```
In [20]: s = np.array([1, 2, 3, 4])
         mean = np.mean(s)
         print("Mean of s:", mean)
```

Mean of s: 2.5

```
In [21]: std_dev = np.std(s)
         print("Standard deviation of s:", std_dev)
```

Standard deviation of s: 1.118033988749895

```
In [22]: minimum = np.min(s)
         print("Min of s:", minimum)
```

Min of s: 1

```
In [23]: maximum = np.max(s)
         print("Max of s:", maximum)
```

Max of s: 4

5. Linear Algebra Functions

```
In [24]: matrix = np.array([[1, 2], [3, 4]])
```

6. Random Sampling Functions

```
In [25]: random_vals = np.random.rand(3)
         print("Random values:", random_vals)
```

Random values: [0.98954587 0.3137986 0.68888632]

```
In [27]: np.random.seed(0)

         random_vals = np.random.rand(3)
         print("Random values:", random_vals)
```

Random values: [0.5488135 0.71518937 0.60276338]

```
In [28]: rand_ints = np.random.randint(0, 10, size=5)
print("Random integers:", rand_ints)
```

Random integers: [3 7 9 3 5]

```
In [29]: np.random.seed(0)

rand_ints = np.random.randint(0, 10, size=5)
print("Random integers:", rand_ints)
```

Random integers: [5 0 3 3 7]

7. Boolean & Logical Functions

```
In [30]: logical_test = np.array([True, False, True])
all_true = np.all(logical_test)
print("All elements True:", all_true)
```

All elements True: False

```
In [31]: logical_test = np.array([True, False, True])
all_true = np.all(logical_test)
print("All elements True:", all_true)
```

All elements True: False

```
In [32]: logical_test = np.array([False, False, False])
all_true = np.all(logical_test)
print("All elements True:", all_true)
```

All elements True: False

```
In [33]: any_true = np.any(logical_test)
print("Any elements True:", any_true)
```

Any elements True: False

8. Set Operations

```
In [34]: set_a = np.array([1, 2, 3, 4])
set_b = np.array([3, 4, 5, 6])
intersection = np.intersect1d(set_a, set_b)
print("Intersection of a and b:", intersection)
```

Intersection of a and b: [3 4]

```
In [35]: union = np.union1d(set_a, set_b)
print("Union of a and b:", union)
```

Union of a and b: [1 2 3 4 5 6]

9. Array Attribute Functions

```
In [36]: a = np.array([1, 2, 3])
shape = a.shape
size = a.size
```

```
dimensions = a.ndim
dtype = a.dtype

print("Shape of a:", shape)
print("Size of a:", size)
print("Number of dimensions of a:", dimensions)
print("Data type of a:", dtype)
```

Shape of a: (3,)
Size of a: 3
Number of dimensions of a: 1
Data type of a: int32

10. Other Functions

```
In [37]: a = np.array([1, 2, 3])
        copied_array = np.copy(a)
        print("Copied array:", copied_array)
```

Copied array: [1 2 3]

```
In [38]: array_size_in_bytes = a.nbytes
        print("Size of a in bytes:", array_size_in_bytes)
```

Size of a in bytes: 12

```
In [39]: shared = np.shares_memory(a, copied_array)
        print("Do a and copied_array share memory?", shared)
```

Do a and copied_array share memory? False