numpy.ndarray

```
import numpy as np
# creating arrays
arr = np.arr ay([1, 2, 3]) # 1D
array, shape (3,)
arr = np.arr ay([[1, 2], [3,
4]]) # 2D array, shape (2, 2)
arr = np.zer os( (rows, cols)) #
2D array of zeros
arr = np.emp ty( (rows, cols,
channels)) # 3D array uninit -
ialized
# inspecting arrays
arr.shape # dimensions e.g.
(480, 640, 3)
arr.dtype # data type e.g.
uint8, float32
arr.ndim # number of dimensions
```

The core data structure in NumPy — a multi-dimensional array of elements all of the same type.
More efficient than Python lists for numerical data as elements are stored contiguously in memory.
Shape describes the dimensions — (rows, cols) for 2D, (rows, cols, channels) for 3D.

numpy.dtype

```
import numpy as np
# common dtypes
np.dty pe( np.u int8) # unsigned
8-bit integer (0 to 255)
np.dty pe( np.i nt8) # signed 8-
bit integer (-128 to 127)
np.dty pe( np.u int16) #
unsigned 16-bit integer (0 to
65535)
np.dty pe( np.i nt16) # signed
16-bit integer (-32768 to 32767)
np.dty pe( np.i nt32) # signed
32-bit integer
np.dty pe( np.f lo at32) # 32-
bit floating point
np.dty pe( np.f lo at64) # 64-
bit floating point
# checking dtype
arr = np.arr ay([1, 2, 3],
dtype= np.f lo at32)
arr.dtype # float32
arr.it emsize # bytes per
element → 4
```

Describes the data type of elements stored in a numpy.ndarray.
Common types are uint8 for images (0-255), float32 for floating point, and int32 for integers.
The dtype determines how many bytes each element takes and how values are interpreted.

numpy.uint8

```
import numpy as np
arr = np.arr ay([0, 128, 255],
dtype= np.u int8) # valid values
arr.dtype # uint8
arr.it emsize # 1 byte per
element
# wrapping behaviour — values
overflow silently
np.uin t8(255) + np.uin t8(1) #
returns 0, not 256
# common usage — converting to
uint8 for image saving
arr = np.cli p(arr, 0, 255).a -
sty pe( np.u int8) # safe
conversion
```

An unsigned 8-bit integer type commonly used for image data.
Can hold values from 0 to 255 — exactly the range of a single colour channel in an image.
Values that exceed 255 wrap around rather than raising an error — a common source of bugs.

By **blakecromar**

cheatography.com/blakecromar/

Not published yet.
Last updated 3rd July, 2026.