NumPy Array Properties

Sunil Kunwar

October 25, 2024

1 Introduction

NumPy arrays come with several important properties that help in understanding and manipulating them effectively. Below are the key properties of a NumPy array along with examples.

2 Properties of NumPy Arrays

2.1 ndim (Number of Dimensions)

This property returns the number of dimensions (or axes) of the array.

```
arr = np.array([[1, 2, 3], [4, 5, 6]])
print(arr.ndim)  # Output: 2
```

2.2 shape (Shape of the Array)

Returns a tuple representing the shape of the array.

```
print(arr.shape)  # Output: (2, 3)
```

2.3 size (Total Number of Elements)

Returns the total number of elements in the array.

```
print(arr.size)  # Output: 6
```

2.4 dtype (Data Type of Elements)

Returns the data type of the elements in the array.

```
print(arr.dtype)  # Output: dtype('int64')
```

2.5 itemsize (Size of Each Element in Bytes)

Returns the size of each element in the array, in bytes.

```
print(arr.itemsize)  # Output: 8
```

2.6 nbytes (Total Bytes Consumed by the Array)

Returns the total number of bytes used by the array.

```
print(arr.nbytes)  # Output: 48
```

2.7 T (Transpose of the Array)

Returns the transpose of the array.

```
print(arr.T)  # Output: [[1, 4], [2, 5], [3, 6]]
```

2.8 flat (Flat Iterator)

Provides an iterator to iterate over all elements of the array as if it were 1D.

```
for item in arr.flat:
    print(item)  # Output: 1 2 3 4 5 6
```

2.9 real and imag (Real and Imaginary Parts)

For arrays with complex numbers, these return the real and imaginary parts of each element.

```
complex_arr = np.array([1+2j, 3+4j])
print(complex_arr.real)  # Output: [1. 3.]
print(complex_arr.imag)  # Output: [2. 4.]
```

2.10 base (Base Object if the Array is a View)

Returns the original array if the array is a view of another array.

```
arr_view = arr[0:2, :]
print(arr_view.base is arr)  # Output: True
```

2.11 strides (Tuple of Bytes to Step in Each Dimension)

Returns the number of bytes that need to be stepped to move to the next element along each dimension.

```
print(arr.strides)  # Output: (24, 8)
```

2.12 ctypes (Interface to C-types)

Provides a pointer to the array data for interfacing with C or C++ code.

```
print(arr.ctypes.data)  # Output: memory address of the array
```

2.13 flags (Memory Layout Information)

Provides detailed information about the memory layout of the array.

```
print(arr.flags)
# Output:
#   C_CONTIGUOUS : True
#   F_CONTIGUOUS : False
#   OWNDATA : True
#   WRITEABLE : True
#   ALIGNED : True
#   UPDATEIFCOPY : False
```

2.14 copy (Copy of the Array)

Creates a copy of the array.

```
arr_copy = arr.copy()
print(arr_copy)
```

2.15 view (View of the Array)

Creates a view (or shallow copy) of the array.

```
arr_view = arr.view()
print(arr_view)
```

3 Example: Displaying All Properties of an Array

Here is a complete example that demonstrates all the properties discussed above:

```
import numpy as np

arr = np.array([[1, 2, 3], [4, 5, 6]])

print("Array:", arr)
print("ndim:", arr.ndim)
print("shape:", arr.shape)
print("size:", arr.size)
print("dtype:", arr.dtype)
print("itemsize:", arr.itemsize)
print("nbytes:", arr.nbytes)
print("T (transpose):", arr.T)
print("flags:", arr.flags)
```