DOCX

NumPy Array Copy Vs View

The Difference Between Copy and View

1) The main difference between a copy and a view of an array is that the copy is a new array, and the view is just a view of the original array.

2) The copy owns the data and any changes made to the copy will not affect original array, and any changes made to the original array will not affect the copy.

3) The view does not own the data and any changes made to the view will affect the original array, and any changes made to the original array will affect the view.

copy()

```
import numpy as np
arr = np.array([1,2,3,4,5])
x = arr.copy()
arr[0] = 42
print(arr)
print(x)
```

Output

```
[42 2 3 4 5]
[1 2 3 4 5]
```

view()

```
import numpy as np
arr = np.array([1,2,3,4,5])
x = arr.view()
arr[0] = 42
print(arr)
print(x)
```

Output

```
[42 2 3 4 5]
[42 2 3 4 5]
```

Check if Array Owns its Data

Print the value of the base attribute to check if an array own its data or not.

```
import numpy as np
arr = np.array([1,2,3,4,5])
x = arr.copy()
y = arr.view()
print(x.base)
print(y.base)
```

Output

```
None
[1 2 3 4 5]
```

1) The copy returns None.
2) The view returns the original array.

Numpy Array Shape

The shape of an array is the number of elements in each dimension.

Numpy arrays have an attribute called shape that returns a tuple with each index having the number of corresponding elements.

1) Print the shape of 2-D array.

```
import numpy as np
arr = np.array([[1,2,3,4],[5,6,7,8]])
print(arr.shape)
```

Output

(2,4)

. The output represents the array is an 2D array is an 2D array which contains 4 elements each.

2)

```
import numpy as np
arr = np.array([1,2,3,4], ndmin=5)
print(arr)
print('shape of array :', arr.shape)
```

Output

```
[[[[[1 2 3 4]]]]]
Shape of array: (1,1,1,1,4)
```

NumPy Array Reshaping

. Reshaping means changing the shape of an array.
The shape of an array is the number of elements in each dimension.
By reshaping we can add or remove dimensions or change number of elements in each dimensi

Reshape From 1-D to 2-D

```
import numpy as np
arr = np.array([1,2,3,4,5,6,7,8,9,10,11,12])
newarr = arr.reshape(4,3)
print(newarr)
```

Output

```
[[1 2 3]
 [4 5 6]
 [7 8 9]
 [10 11 12]]
```

Reshape From 1-D to 3-D

```
import numpy as np
arr = np.array([1,2,3,4,5,6,7,8,9,10,11,12])
newarr = arr.reshape(2,3,2)
print(newarr)
```

```
[[[1 2]
  [3 4]
  [5 6]]
 [[7 8]
  [9 10]
  [11 12]]]
```

Can we Reshape Into any Shape?

Yes, as long as the elements required for reshaping are equal in both shapes.

We can reshape an 8 elements 1D array into 4 elements in 2 rows 2D array but we cannot reshape it into a 3 elements 3 rows 2D array as that would require 3×3=9 elements.

Unknown Dimension

You are allowed to have one "unknown" dimension. Meaning that you do not have to specify an exact number for one of the dimensions in the reshape method.

Pass -1 as the value, and NumPy will calculate this number for you.

1)
```
import numpy as np
arr = np.array([1,2,3,4,5,6,7,8])
newarr = arr.reshape(2,2,-1)
print(newarr)
```

Output

```
[[[1 2]
  [3 4]]
 [[5 6]
  [7 8]]]
```