NumPy - Cópias e Visualizações

Ao executar as funções, algumas delas retornam uma cópia do array de entrada, enquanto outras retornam a visualização. Quando o conteúdo é armazenado fisicamente em outro local, chama-se **Copiar** . Se, por outro lado, for fornecida uma visão diferente do mesmo conteúdo da memória, chamamos-lhe **View** .

Sem cópia

Atribuições simples não fazem a cópia do objeto array. Em vez disso, ele usa o mesmo id() do array original para acessá-lo. O **id()** retorna um identificador universal do objeto Python, semelhante ao ponteiro em C.

Além disso, quaisquer alterações em um deles são refletidas no outro. Por exemplo, a mudança na forma de um também mudará a forma do outro.

Exemplo

Demonstração ao vivo

```
import numpy as np
a = np.arange(6)

print 'Our array is:'
print a

print 'Applying id() function:'
print id(a)

print 'a is assigned to b:'
b = a
print b

print 'b has same id():'
print id(b)

print 'Change shape of b:'
b.shape = 3,2
print b

print 'Shape of a also gets changed:'
print a
```

Ele produzirá a seguinte saída -

```
Our array is:
[0 1 2 3 4 5]

Applying id() function:
139747815479536

a is assigned to b:
[0 1 2 3 4 5]
b has same id():
139747815479536

Change shape of b:
[[0 1]
 [2 3]
 [4 5]]

Shape of a also gets changed:
[[0 1]
 [2 3]
 [4 5]]
```

Visualizar ou cópia superficial

NumPy possui o método **ndarray.view()** que é um novo objeto de array que analisa os mesmos dados do array original. Ao contrário do caso anterior, a alteração nas dimensões do novo array não altera as dimensões do original.

Exemplo

Demonstração ao vivo

```
import numpy as np
# To begin with, a is 3X2 array
a = np.arange(6).reshape(3,2)

print 'Array a:'
print a

print 'Create view of a:'
b = a.view()
print b
```

```
print 'id() for both the arrays are different:'
print 'id() of a:'
print id(a)
print 'id() of b:'
print id(b)

# Change the shape of b. It does not change the shape of a
b.shape = 2,3

print 'Shape of b:'
print b

print 'Shape of a:'
print a
```

Ele produzirá a seguinte saída -

```
Array a:
[[0 1]
 [2 3]
 [4 5]]

Create view of a:
[[0 1]
 [2 3]
 [4 5]]

id() for both the arrays are different:
id() of a:
140424307227264
id() of b:
140424151696288

Shape of b:
[[0 1 2]
 [3 4 5]]

Shape of a:
[[0 1]
 [2 3]
 [4 5]]
```

Uma fatia de um array cria uma visualização.

Exemplo

```
import numpy as np
a = np.array([[10,10], [2,3], [4,5]])

print 'Our array is:'
print a

print 'Create a slice:'
s = a[:, :2]
print s
```

Demonstração ao vivo

Ele produzirá a seguinte saída -

```
Our array is:
[[10 10]
 [ 2 3]
 [ 4 5]]

Create a slice:
[[10 10]
 [ 2 3]
 [ 4 5]]
```

Cópia profunda

A função **ndarray.copy()** cria uma cópia profunda. É uma cópia completa do array e de seus dados e não é compartilhada com o array original.

Exemplo

```
import numpy as np
a = np.array([[10,10], [2,3], [4,5]])

print 'Array a is:'
print a

print 'Create a deep copy of a:'
b = a.copy()
print 'Array b is:'
```

Demonstração ao vivo

```
print b

#b does not share any memory of a
print 'Can we write b is a'
print b is a

print 'Change the contents of b:'
b[0,0] = 100

print 'Modified array b:'
print b

print 'a remains unchanged:'
print a
```

Ele produzirá a seguinte saída -

```
Array a is:
[[10 10]
 [ 2 3]
 [ 4 5]]

Create a deep copy of a:
Array b is:
[[10 10]
 [ 2 3]
 [ 4 5]]
Can we write b is a
False

Change the contents of b:
Modified array b:
[[100 10]
 [ 2 3]
 [ 4 5]]

a remains unchanged:
[[10 10]
 [ 2 3]
 [ 4 5]]
```