NumPy - Iterando sobre array

O pacote NumPy contém um objeto iterador **numpy.nditer** . É um objeto iterador multidimensional eficiente com o qual é possível iterar sobre um array. Cada elemento de um array é visitado usando a interface Iterator padrão do Python.

Vamos criar um array 3X4 usando a função arange() e iterar sobre ele usando **nditer** .

Exemplo 1

Demonstração ao vivo

```
import numpy as np
a = np.arange(0,60,5)
a = a.reshape(3,4)

print 'Original array is:'
print a
print '\n'

print 'Modified array is:'
for x in np.nditer(a):
   print x,
```

A saída deste programa é a seguinte -

```
Original array is:
[[ 0 5 10 15]
 [20 25 30 35]
 [40 45 50 55]]

Modified array is:
0 5 10 15 20 25 30 35 40 45 50 55
```

Exemplo 2

A ordem da iteração é escolhida para corresponder ao layout de memória de um array, sem considerar uma ordem específica. Isso pode ser visto iterando a transposição da matriz acima.

Demonstração ao vivo

```
import numpy as np
a = np.arange(0,60,5)
a = a.reshape(3,4)

print 'Original array is:'
print a
print '\n'

print 'Transpose of the original array is:'
b = a.T
print b
print '\n'

print 'Modified array is:'
for x in np.nditer(b):
   print x,
```

A saída do programa acima é a seguinte -

```
Original array is:
[[ 0 5 10 15]
 [20 25 30 35]
 [40 45 50 55]]

Transpose of the original array is:
[[ 0 20 40]
 [ 5 25 45]
 [10 30 50]
 [15 35 55]]

Modified array is:
0 5 10 15 20 25 30 35 40 45 50 55
```

Ordem de iteração

Se os mesmos elementos forem armazenados usando a ordem do estilo F, o iterador escolhe a maneira mais eficiente de iterar em uma matriz.

Exemplo 1

```
import numpy as np
a = np.arange(0,60,5)
a = a.reshape(3,4)
print 'Original array is:'
print a
print '\n'

print 'Transpose of the original array is:'
b = a.T
print b
print '\n'

print 'Sorted in C-style order:'
c = b.copy(order = 'C')
print c
for x in np.nditer(c):
   print x,

print '\n'

print 'Sorted in F-style order:'
c = b.copy(order = 'F')
print c
for x in np.nditer(c):
   print x,
```

Sua saída seria a seguinte -

```
Original array is:
[[ 0 5 10 15]
 [20 25 30 35]
 [40 45 50 55]]

Transpose of the original array is:
[[ 0 20 40]
 [ 5 25 45]
 [10 30 50]
 [15 35 55]]

Sorted in C-style order:
[[ 0 20 40]
 [ 5 25 45]
 [10 30 50]
```

```
 [15 35 55]]
0 20 40 5 25 45 10 30 50 15 35 55

Sorted in F-style order:
[[ 0 20 40]
 [ 5 25 45]
 [10 30 50]
 [15 35 55]]
0 5 10 15 20 25 30 35 40 45 50 55
```

Exemplo 2

É possível forçar o objeto **nditer** a usar uma ordem específica mencionando-a explicitamente.

Demonstração ao vivo

```
import numpy as np
a = np.arange(0,60,5)
a = a.reshape(3,4)

print 'Original array is:'
print a
print '\n'

print 'Sorted in C-style order:'
for x in np.nditer(a, order = 'C'):
   print x,
print '\n'

print 'Sorted in F-style order:'
for x in np.nditer(a, order = 'F'):
   print x,
```

Sua saída seria -

```
Original array is:
[[ 0 5 10 15]
 [20 25 30 35]
 [40 45 50 55]]

Sorted in C-style order:
0 5 10 15 20 25 30 35 40 45 50 55
```

```
Sorted in F-style order:
0 20 40 5 25 45 10 30 50 15 35 55
```

Modificando Valores de Matriz

O objeto **nditer** possui outro parâmetro opcional chamado **op_flags** . Seu valor padrão é somente leitura, mas pode ser definido para modo leitura-gravação ou somente gravação. Isso permitirá a modificação de elementos da matriz usando este iterador.

Exemplo

Demonstração ao vivo

```
import numpy as np
a = np.arange(0,60,5)
a = a.reshape(3,4)
print 'Original array is:'
print a
print '\n'

for x in np.nditer(a, op_flags = ['readwrite']):
   x[...] = 2*x
print 'Modified array is:'
print a
```

Sua saída é a seguinte -

```
Original array is:
[[ 0 5 10 15]
 [20 25 30 35]
 [40 45 50 55]]

Modified array is:
[[ 0 10 20 30]
 [ 40 50 60 70]
 [ 80 90 100 110]]
```

Loop Externo

O construtor da classe nditer possui um parâmetro **'flags'** , que pode assumir os seguintes valores -

Sr. Não.	Parâmetro e Descrição
1	**índice_c** O índice C_order pode ser rastreado
2	**índice_f** O índice Fortran_order é rastreado
3	**multi-índice** Tipos de índices com um por iteração podem ser rastreados
4	**loop_externo** Faz com que os valores fornecidos sejam matrizes unidimensionais com vários valores em vez de matrizes de dimensão zero

Exemplo

No exemplo a seguir, as matrizes unidimensionais correspondentes a cada coluna são percorridas pelo iterador.

Demonstração ao vivo

```
import numpy as np
a = np.arange(0,60,5)
a = a.reshape(3,4)

print 'Original array is:'
print a
print '\n'

print 'Modified array is:'
for x in np.nditer(a, flags = ['external_loop'], order = 'F'):
   print x,
```

A saída é a seguinte -

```
Original array is:
[[ 0 5 10 15]
 [20 25 30 35]
 [40 45 50 55]]

Modified array is:
[ 0 20 40] [ 5 25 45] [10 30 50] [15 35 55]
```

Iteração de transmissão

Se duas matrizes forem **transmitidas** , um objeto **nditer** combinado será capaz de iterar sobre elas simultaneamente. Supondo que um array **a** tenha dimensão 3X4 e haja outro array **b** de dimensão 1X4, o iterador do seguinte tipo é usado (o array **b** é transmitido para o tamanho de **a**).

Exemplo

Demonstração ao vivo

```
import numpy as np
a = np.arange(0,60,5)
a = a.reshape(3,4)

print 'First array is:'
print a
print '\n'

print 'Second array is:'
b = np.array([1, 2, 3, 4], dtype = int)
print b
print '\n'

print 'Modified array is:'
for x,y in np.nditer([a,b]):
   print "%d:%d" % (x,y),
```

Sua saída seria a seguinte -

```
First array is:
[[ 0 5 10 15]
 [20 25 30 35]
 [40 45 50 55]]

Second array is:
[1 2 3 4]

Modified array is:
0:1 5:2 10:3 15:4 20:1 25:2 30:3 35:4 40:1 45:2 50:3 55:4
```