NumPy - Matriz de dados existentes

Neste capítulo, discutiremos como criar um array a partir de dados existentes.

numpy.asarray

Esta função é semelhante a numpy.array, exceto pelo fato de possuir menos parâmetros. Esta rotina é útil para converter a sequência Python em ndarray.

```
numpy.asarray(a, dtype = None, order = None)
```

O construtor usa os seguintes parâmetros.

Sr. Não.	Parâmetro e Descrição
1	**a** Insira dados em qualquer formato, como lista, lista de tuplas, tuplas, tupla de tuplas ou tupla de listas
2	**tipo d** Por padrão, o tipo de dados de entrada é aplicado ao ndarray resultante
3	**ordem** C (linha maior) ou F (coluna maior). C é o padrão

Os exemplos a seguir mostram como você pode usar a função **asarray** .

Exemplo 1

Demonstração ao vivo

```
# convert list to ndarray
import numpy as np

x = [1,2,3]
a = np.asarray(x)
print a
```

Sua saída seria a seguinte -

```
[1  2  3]
```

Exemplo 2

Demonstração ao vivo

```
# dtype is set
import numpy as np

x = [1,2,3]
a = np.asarray(x, dtype = float)
print a
```

Agora, a saída seria a seguinte -

```
[ 1.  2.  3.]
```

Exemplo 3

Demonstração ao vivo

```
# ndarray from tuple
import numpy as np

x = (1,2,3)
a = np.asarray(x)
print a
```

Sua saída seria -

```
[1  2  3]
```

Exemplo 4

Demonstração ao vivo

```
# ndarray from list of tuples
import numpy as np

x = [(1,2,3),(4,5)]
a = np.asarray(x)
print a
```

Aqui, a saída seria a seguinte -

```
[(1, 2, 3) (4, 5)]
```

numpy.frombuffer

Esta função interpreta um buffer como um array unidimensional. Qualquer objeto que exponha a interface do buffer é usado como parâmetro para retornar um **ndarray** .

```
numpy.frombuffer(buffer, dtype = float, count = -1, offset = 0)
```

O construtor usa os seguintes parâmetros.

Sr. Não.	Parâmetro e Descrição
1	**amortecedor** Qualquer objeto que exponha a interface do buffer
2	**tipo d** Tipo de dados do ndarray retornado. O padrão é flutuar
3	**contar** O número de itens a serem lidos, o padrão -1 significa todos os dados
4	**desvio** A posição inicial a partir da qual ler. O padrão é 0

Exemplo

Os exemplos a seguir demonstram o uso da função **frombuffer** .

Demonstração ao vivo

```
import numpy as np
s = 'Hello World'
a = np.frombuffer(s, dtype = 'S1')
print a
```

Aqui está o resultado -

```
['H'  'e'  'l'  'l'  'o'  ' '  'W'  'o'  'r'  'l'  'd']
```

numpy.fromiter

Esta função cria um objeto **ndarray** a partir de qualquer objeto iterável. Uma nova matriz unidimensional é retornada por esta função.

```
numpy.fromiter(iterable, dtype, count = -1)
```

Aqui, o construtor usa os seguintes parâmetros.

Sr. Não.	Parâmetro e Descrição
1	**iterável** Qualquer objeto iterável
2	**tipo d** Tipo de dados da matriz resultante
3	**contar** O número de itens a serem lidos do iterador. O padrão é -1, o que significa que todos os dados serão lidos

Os exemplos a seguir mostram como usar a função interna **range()** para retornar um objeto de lista. Um iterador desta lista é usado para formar um objeto **ndarray** .

Exemplo 1

```
# create list object using range function
import numpy as np
list = range(5)
print list
```

Demonstração ao vivo

Sua saída é a seguinte -

```
[0, 1, 2, 3, 4]
```

Exemplo 2

```
# obtain iterator object from list
import numpy as np
list = range(5)
it = iter(list)
```

Demonstração ao vivo

```
# use iterator to create ndarray
x = np.fromiter(it, dtype = float)
print x
```

Agora, a saída seria a seguinte -

```
[0.  1.  2.  3.  4.]
```