python para Ciencia de Datos: Hoja de Referencia

NumPy

Aprende Python para Ciencia de Datos en www.datademia.es

Datademia

NumPy

La biblioteca **NumPy** es la biblioteca central para la computación científica en Python. Proporciona un objeto de matriz multidimensional de alto rendimiento y herramientas para trabajar con estas matrices.

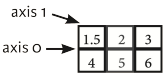
Usa la siguiente convención:

```
>>> import numpy as np
```

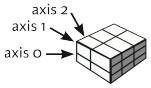
Matrizes NumPy

Matriz 1D

Matriz 2D



Matriz 3D



Crear Matrizes

```
>>> a = np.array([1,2,3])
>>> b = np.array([(1.5,2,3), (4,5,6)], dtype = float)
>>> c = np.array([[(1.5,2,3), (4,5,6)], [(3,2,1), (4,5,6)]],
                 dtype = float)
```

Marcadores de posición iniciales

```
>>> np.zeros((3,4))                  Crea una matriz de ceros
>>> np.ones((2,3,4),dtype=np.int16)  Crea una matriz de unos
>>> d = np.arange(10,25,5)           Crea una matriz de valores espaciados
                                     uniforme (por valor)
>>> np.linspace(0,2,9)               Crea una matriz de valores espaciados
                                     uniforme (por número de muestras)
>>> e = np.full((2,2),7)             Crea una matriz constante
>>> f = np.eye(2)                    Crea una matriz de identidad 2X2
>>> np.random.random((2,2))          Crea una matriz con valores aleatorios
>>> np.empty((3,2))                  Crea una matriz vacía
```

I/O

Guardando y cargando en disco

```
>>> np.save('my_array', a)
>>> np.savez('array.npz', a, b)
>>> np.load('my_array.npy')
```

Guarda y carga archivos de texto

```
>>> np.loadtxt("myfile.txt")
>>> np.genfromtxt("my_file.csv", delimiter=',')
>>> np.savetxt("myarray.txt", a, delimiter=" ")
```

Tipos de Datos

```
>>> np.int64       Numero entero de 64 bits
>>> np.float32     Flotante de 32 bits
>>> np.complex     Números complejos representados por 128 flotantes
>>> np.bool        Booleano que almacena valores VERDADERO y FALSO
>>> np.object      Tipo de objeto Python
>>> np.string_     Cadena de longitud fija
>>> np.unicode_    Unicode de longitud fija
```

Inspecciona tu Matriz

```
>>> a.shape        Dimensiones de la matriz
>>> len(a)         Longitud de la matriz
>>> b.ndim         Número de dimensiones de la matriz
>>> e.size         Número de elementos de la matriz
>>> b.dtype        Tipo de dato de los elementos de la matriz
>>> b.dtype.name   Nombre del tipo de dato
>>> b.astype(int)  Convertir matriz a otro tipo
```

Ayuda

```
>>> np.info(np.ndarray.dtype)
```

Matemáticas de Matrizes

Operaciones Aritméticas

```
>>> g = a - b                        Resta
  array([[-0.5, 0. , 0. ],
         [-3. , -3. , -3. ]])
>>> np.subtract(a,b)                 Resta
>>> b + a                            Suma
  array([[ 2.5, 4. , 6. ],
         [ 5. , 7. , 9. ]])
>>> np.add(b,a)                      Suma
>>> a / b                            División
  array([[ 0.66666667, 1. , 1. ],
         [ 0.25 , 0.4 , 0.5 ]])
>>> np.divide(a,b)                   División
>>> a * b                            Multiplicación
  array([[ 1.5, 4. , 9. ],
         [ 4. , 10. , 18. ]])
>>> np.multiply(a,b)                 Multiplicación
>>> np.exp(b)                        Exponenciación
>>> np.sqrt(b)                       Raíz cuadrada
>>> np.sin(a)                        Imprimir senos de una matriz
>>> np.cos(b)                        Coseno por elemento
>>> np.log(a)                        Logaritmo natural por elemento
>>> e.dot(f)                         Producto DOT
  array([[ 7., 7.],
         [ 7., 7.]])
```

Comparación

```
>>> a == b                           Comparación por elemento
  array([[False, True, True],
         [False, False, False]],
          dtype=bool)
>>> a < 2                            Comparación por elemento
  array([True, False, False],
        dtype=bool)
>>> np.array_equal(a, b)             Comparación por matriz
```

Funciones de Agregación

```
>>> a.sum()                          Suma
>>> a.min()                          Valor mínimo
>>> b.max(axis=0)                    Valor maximo por fila
>>> b.cumsum(axis=1)                 Suma cumulativa por elemento
>>> a.mean()                         Media
>>> b.median()                       Mediana
>>> a.corrcoef()                     Coeficiente de correlación
>>> np.std(b)                        Desviación estándar
```

Copiar Matrizes

```
>>> h = a.view()                     Vista de la matriz con los mismos datos
>>> np.copy(a)                       Crea una copia de la matriz
>>> h = a.copy()                     Crea una copia profunda de la matriz
```

Ordenar Matrizes

```
>>> a.sort()                         Ordena la matriz
>>> c.sort(axis=0)                   Ordena los elementos del eje de una matriz
```

Selecionar, Cortar y Indexar

Selecionar

```
>>> a[2] 1 2 3                       Selecciona el elemento en el 2do índice
3
>>> b[1,2]                           Selecciona el elemento en la fila 1 columna 2
1.5 2 3 6.0                          equivalente a b[1][2]
```

Cortar

```
>>> a[0:2]                           Selecciona elementos en los índices 0 y 1
  array([1, 2])
>>> b[0:2,1]                         Selecciona elementos en las filas 0 y 1 y en la
  array([ 2., 5.])                   columna 1
>>> b[:1]                            Selecciona todos los elementos en la fila 0
  array([[1.5, 2., 3.]])             (equivalente a b[0:1, :])
>>> c[1,...]
  array([[[3., 2., 1.],              Igual que [1,:,:]
          [4., 5., 6.]]])
>>> a[ : :-1]                        Matriz invertida a
  array([3, 2, 1])
```

Indexar

```
>>> a[a<2]                           Selecciona elementos de a menos de 2
  array([1])
```

Indexar Especial

```
>>> b[[1, 0, 1, 0],[0, 1, 2, 0]]    Selecciona elementos (1,0),(0,1),(1,2) y (0,0)
  array([4.,2.,6.,1.5])
>>> b[[1, 0, 1, 0]][:,[0,1,2,0]]    Selecciona parte de las columnas y filas de una
  array([[4.,5.,6.,4.],              matriz
         [1.5,2.,3.,1.5],
         [4.,5.,6.,4.],
         [1.5,2.,3.,1.5]])
```

Manipulación de Matrizes

Transponer

```
>>> i = np.transpose(b)              Permutar dimensiones de la matriz
>>> i.T
```

Cambiar

```
>>> b.ravel()                        Aplanar la matriz
>>> g.reshape(3,-2)                  Cambia la forma, pero no los datos
```

Añadir/Quitar Elementos

```
>>> h.resize((2,6))                  Devuelve una matriz con forma (2,6)
>>> np.append(h,g)                   Agregar elementos a una matriz
>>> np.insert(a, 1, 5)               Insertar elementos en una matriz
>>> np.delete(a,[1])                 Borrar elementos en una matriz
```

Combinar

```
>>> np.concatenate((a,d),axis=0)     Concatenar matrices
  array([ 1, 2, 3, 10, 15, 20])
>>> np.vstack((a,b))                 Apilar matrices verticalmente
  array([[ 1. , 2. , 3. ],           (por filas)
         [ 1.5, 2. , 3. ],
         [ 4. , 5. , 6. ]])
>>> np.r_[e,f]                       Apilar matrices verticalmente
>>> np.hstack((e,f))                 (por filas)
  array([[ 7., 7., 1., 0.],          Apilar matrices horizontalmente
         [ 7., 7., 0., 1.]])         (por columnas)
>>> np.column_stack((a,d))           Matrizes apiladas por columnas
  array([[ 1, 10],
         [ 2, 15],
         [ 3, 20]])
>>> np.c_[a,d]                       Matrizes apiladas por columnas
```

Dividir

```
>>> np.hsplit(a,3)                   Dividir la matriz horizontalmente en el tercer
 [array([1]),array([2]),             índice
  array([3])]
>>> np.vsplit(c,2)                   Dividir la matriz verticalmente en el segundo
  [array([[[ 1.5, 2. , 1. ],         índice
           [ 4. , 5. , 6. ]]]),
   array([[[ 3., 2., 3.],
           [ 4., 5., 6.]]])]
```

Datademia

Aprende Python para Ciencia de Datos en www.datademia.es