Librería numérica de Python: NumPy

NumPy - Guía de referencia

Importar NumPy

```
import numpy as np   #Se usa np.array()
from numpy import *  #Se usa array()
```

np.array - Arreglos de NumPy

```
arr = np.array([1,2,3]) #1 dimensión
arr = np.array([[1,0,0],#2 dimensiones
                [0,1,1]])
arr = np.array( colección,
                dtype = float64]
```

Atributos de arreglos

arr.shape	Dimensiones del arreglo en cada eje
arr.ndim	Cantidad de dimensiones
arr.size	Número de elementos en el arreglo
arr.itemsize	Tamaño en bytes del tipo de dato del arreglo
arr.T	Matriz transpuesta
arr.real	Parte real de la matriz
arr.imag	Parte imaginaria de la matriz

Números aleatorios - np.random

np.random.rand(n,m)

Crea un arreglo de tamaño n x m de número aleatorios con distribución uniforme entre 0 y 1.

np.random.randn(n,m)

Crea un arreglo de tamaño n x m de valores de la distribución normal estándar.

np.random.randint(a,b,shape)

Crea un arreglo de números enteros aleatorios entre a y b con dimensiones shape.

Copias y vistas

#Valor	#Referencia
arr.copy()	arr.view()

Map

La manera más eficiente de aplicar funciones a un arreglo de NumPy es aplicándola directamente.

```
>foo = lambda a : a**2
>arr_2 = foo(arr)
```

Generar arreglos comunes

np.arange(a,b,step)

Arreglo separado uniformemente en el intervalo a y b con pasos de tamaño *step*.

np.linspace(a,b,n)

Arreglo separado uniformemente en el intervalo a y b con n muestras.

np.zeros((n,m))

Matriz de ceros de tamaño n x m

np.ones((n,m))

Matriz de unos de tamaño n x m

np.eye(n,m)

Matriz de tamaño n x m con unos en la diagonal y ceros en el resto de elementos

np.full((n,m), x)

Matriz de tamaño n x m con x en todos los elementos

np.empty((n,m))

Matriz de tamaño n x m sin inicializar

Tipos de dato dtypes

int8	complex64
int16	complex128
int32	complex256
int64	bool
uint8	object
uint16	byte
uint32	str
uint64	unicode
float16	single
float32	double
float64	longdouble
float128	NaN

Indexado

a[x,y]	Índices separados por comas
a[a:b,c:d]	Rangos por cada eje
a[l_a, l_b]	Arreglo de índices en las posiciones l_a[i],l_b[i]
a[i,...]	Índice en una dimensión específica

Selección condicional

a > x, a < x	Operadores relacionales y lógicos con arreglos.
a[a > x]	Retorna un arreglo de booleanos al evaluar la expresión con cada elemento.
a[a > x & a < y]	Se pueden usar arreglos de este tipo para indexar un arreglo.

Operaciones en arreglos

> a + b **> np.add(a,b)**	Suma de matrices
> a - b **> np.subtract(a,b)**	Resta de matrices
> a * b **> np.multiply(a,b)**	Multiplicación entre elementos de matrices
> a / b **> np.divide(a,b)**	División elementos de matrices
> a.dot(b)	Multiplicación matricial
> a @ b	Multiplicación matricial (otra forma)

a.max()	**a.mean()**	**a.std()**	**np.exp(a)**	**np.sin(a)**
a.min()	**a.sum()**	**a.var()**	**np.log(a)**	**np.cos(a)**

Transformación de arreglos

a.reshape(n,m)

Crea un arreglo de dimensión n x m a partir de un arreglo de n*m elementos.

a.ravel()

Crea un arreglo de 1 dimensión de los elementos del arreglo a.

Combinación de arreglos

np.concatenate((a,b), axis)

Concatena los valores de los arreglos a y b sobre el eje axis

np.vstack((a,b))

Apila verticalmente los arreglos a y b

np.hstack((a,b))

Apila horizontalmente los arreglos a y b

Separación de arreglos

np.hsplit(a, i)

Separa horizontalmente el arreglo a desde el índice *i*

np.vsplit(a, i)

Separa verticalmente el arreglo a desde el índice *i*