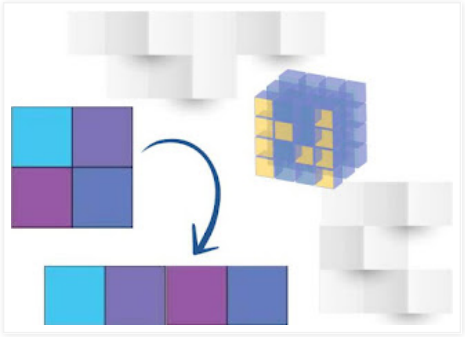
Python 3 para impacientes

"Simple es mejor que complejo" (Tim Peters)

sábado, 2 de noviembre de 2019

Estructura de un array Numpy



Propiedades de los arrays Numpy

Las propiedades permiten obtener información de las dimensiones de un array **Numpy**, el número y tipo de elementos que pueden contener y sobre la memoria ocupada.

ndim

Obtener el número de dimensiones de un array.

```
import numpy as np
a = np.array([1, 2, 3], dtype=np.int16)
b = np.array([(1, 2, 3), (4, 5, 6)], dtype=np.int8)
c = np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]], dtype=np.uint8)
print(a.ndim)  # 1 dimensión
print(b.ndim)  # 2 dimensiones
print(c.ndim)  # 3 dimensiones
```

shape

Obtener las dimensiones de un array (en una tupla).

```
print(a.shape)  # (3,)        -> 3
print(b.shape)  # (2, 3)      -> 2x3
print(c.shape)  # (2, 2, 2)   -> 2x2x2
```

size

Obtener el tamaño o número de elementos de un array.

```
print(a.size)  # 3
print(b.size)  # 6
print(c.size)  # 8
```

type()

Obtener el tipo de objeto de un array Numpy.

Buscar

Buscar

Python para impacientes

Python
IPython
EasyGUI
Tkinter
JupyterLab
Numpy

Anexos

Guía urgente de MySQL
Guía rápida de SQLite3

Entradas + populares

Dar color a las salidas en la consola
En Python para dar color a las salidas en la consola (o en la terminal de texto) existen varias posibilidades. Hay un método basado ...

Instalación de Python, paso a paso
Instalación de Python 3.6 A finales de 2016 se produjo el lanzamiento de Python 3.6 . El propósito de esta entrada es mostrar, pas...

Añadir, consultar, modificar y suprimir elementos en Numpy
Acceder a los elementos de un array. [], [,], ... Acceder a un elemento de un array. Para acceder a un elemento se utiliz...

Cálculo con arrays Numpy
Numpy ofrece todo lo necesario para obtener un buen rendimiento cuando se trata de hacer cálculos con arrays. Por como está concebid...

Variables de control en Tkinter
Variables de control Las variables de control son objetos especiales que se asocian a los widgets para almacenar sus valore...

Operaciones con fechas y horas. Calendarios
Los módulos datetime y calendar amplían las posibilidades del módulo time que provee funciones para manipular expresiones de ti...

Tkinter: interfaces gráficas en Python
Introducción Con Python hay muchas posibilidades para programar una interfaz gráfica de usuario (GUI) pero Tkinter es fácil d...

Convertir, copiar, ordenar, unir y dividir arrays Numpy
Esta entrada trata sobre algunos métodos que se utilizan en Numpy para convertir listas en arrays y viceversa; para copiar arrays d...

Threading: programación con hilos (I)
En programación, la técnica que permite que una aplicación ejecute simultáneamente varias operaciones en el mismo espacio de proceso se...

Tkinter: Tipos de ventanas
Ventanas de aplicación y de diálogos En la entrada anterior tratamos los distintos

```
print(type(a))  # class numpy.ndarray
print(type(b))  # class numpy.ndarray
print(type(c))  # class numpy.ndarray
```

isinstance()

Comprobar si un objeto es un array Numpy.

```
print(isinstance(a, np.ndarray))  # True
print(isinstance(b, np.ndarray))  # True
print(isinstance(c, np.ndarray))  # True
```

dtype

Obtener el tipo de los datos de un array.

```
print(a.dtype)  # int16
print(b.dtype)  # int8
print(c.dtype)  # uint8
```

itemsize

Obtener el tamaño en bytes que ocupa cada elemento en un array.

```
print(a.itemsize)  # 2 bytes
print(b.itemsize)  # 1 byte
print(c.itemsize)  # 1 byte
```

nbytes

Obtener el tamaño total en bytes que ocupa un array.

```
print(a.nbytes)  # 6 bytes
print(b.nbytes)  # 6 bytes
print(c.nbytes)  # 8 bytes
```

flags

Obtener información sobre la memoria ocupada por un array.

```
print(a.flags)

#  C_CONTIGUOUS : True  Los datos comparten mismo segmento (C)
#  F_CONTIGUOUS : True  Los datos comparten mismo segmento (Fortram)
#  OWNDATA : True       Array propietario de espacio, no comparte
#  WRITEABLE : True     Es posible escribir en el área de datos
#  ALIGNED : True       Datos alineados adecuadamente para hardware
#  WRITEBACKIFCOPY : False    Indica si el array es copia de otro
#  UPDATEIFCOPY : False       (No se utiliza)
```

Métodos para cambiar la estructura de un array

Los métodos permiten cambiar el tipo de datos de los elementos de un array, modificar su forma y tamaño, transponer un array, intercambiar sus ejes, convertir a vectores y crear vistas.

astype()

Cambiar el tipo de los datos de un array.

```
a = np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]], dtype=np.uint8)
print(a.dtype)  # uint8
print(a.nbytes)  # 8 bytes (Son 8 elementos x 1 byte = 8 bytes)
a = a.astype(np.float32)  # Convertir el tipo a float32
```

gestores de geometría que se utilizan para di...

```
print(a.dtype)  # float32
print(a.nbytes)  # 32 bytes (Son 8 elmentos x 4 bytes = 32 bytes)
print(a)

# [[[1. 2.]
#   [3. 4.]]
#
#  [[5. 6.]
#   [7. 8.]]]
```

shape

Cambiar la forma de un array.

```
# Cambiar la forma de un array de (3x2) a (2x3).

a = np.ones((3, 2))
a.shape = (2, 3)
print(a)

# [[1. 1. 1.]
#  [1. 1. 1.]]

# Cambiar la forma de un array de (3x2) a (1x6).

a = np.ones((3, 2))
a.shape = (6)
print(a)

# [1. 1. 1. 1. 1. 1.]
```

reshape()

Cambiar la forma de un array.

```
# Cambiar la forma de un array de (1x6) a (3x2).

a = np.arange(1, 7)
print(a)

# [1 2 3 4 5 6]

a = a.reshape(3, 2)
print(a)

# [[1 2]
#  [3 4]
#  [5 6]]
```

resize()

Redimensionar un array rellenando los elementos nuevos.

```
# Redimensionar array de (1x9) a (4x4) rellenando
# los elementos nuevos.

a = np.arange(1, 10)
print(a)

# [1 2 3 4 5 6 7 8 9]

a = np.resize(a, (4, 4))
print(a)

# [[1 2 3 4]
#  [5 6 7 8]
#  [9 1 2 3]
#  [4 5 6 7]]

# Redimensionar un array de (1x9) a (2x2) eliminando elementos.

a = np.arange(1, 10)
print(a)
```

```
# [1 2 3 4 5 6 7 8 9]

a = np.resize(a, (2, 2))
print(a)

# [[1 2]
#  [3 4]]
```

transpose()

Transponer un array.

```
# Transponer un array de (2x3).

a = np.array([(1, 2, 3), (4, 5, 6)])
print(a)

# [[1 2 3]
#  [4 5 6]]

a = np.transpose(a)
print(a)

# [[1 4]
#  [2 5]
#  [3 6]]
```

swapaxes()

Intercambiar los ejes de un array.

```
# Intercambiar los ejes de un array de (2x3).

a = np.array([(10, 11, 12), (13, 14, 15)])
print(a)

# [[10 11 12]
#  [13 14 15]]

a = a.swapaxes(0, 1)
print(a)

# [[10 13]
#  [11 14]
#  [12 15]]
```

flatten()

Convertir un array 2D en un vector.

```
# Convertir un array (2x3) en un vector de 6 elementos.

a = np.array([(1, 2, 3), (4, 5, 6)])
print(a)

# [[1 2 3]
#  [4 5 6]]

a = a.flatten()
print(a)

# [1 2 3 4 5 6]

# Convertir un array 2D en vector ordenando los
# elementos por columnas:

a = np.array([(1, 2, 3), (4, 5, 6)])
print(a)

# [[1 2 3]
#  [4 5 6]]
```

```
a = a.flatten('F')
print(a)

# [1 4 2 5 3 6]
```

ravel()

Crear una vista con forma de vector de un array 2D.

El método ravel() devuelve una vista de los datos siemque que sea posible, no como flatten() que devuelve una copia siempre. Esto hace a ravel() a menudo más rápido pero hay que tener cuidado con las modificaciones en el array que devuelve.

```
# Crear vista con forma de vector ordenando por columnas (eje 1).

a = np.array([(1, 2, 3), (4, 5, 6)])
print(a)

# [[1 2 3]
#  [4 5 6]]

print(a.ravel('F'))

# [1 4 2 5 3 6]

# Crear vista con forma de vector ordenando por filas (eje 0).

a = np.array([(1, 2, 3), (4, 5, 6)])
print(a)

# [[1 2 3]
#  [4 5 6]]

print(a.ravel('C'))

# [1 2 3 4 5 6]
```

Publicado por Pherkad en 12:48

Etiquetas: Numpy

Entrada más reciente Inicio Entrada antigua