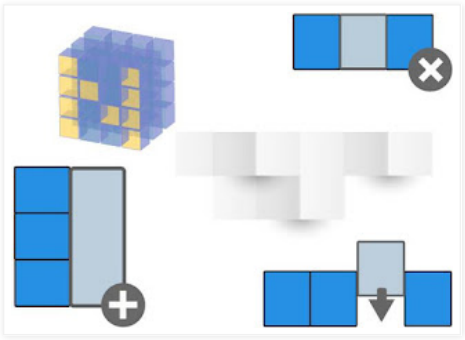
Python 3 para impacientes

"Simple es mejor que complejo" (Tim Peters)

sábado, 9 de noviembre de 2019

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 utiliza la referencia a su posición dentro del array mediante uno o más **índices** según se trate de un array unidimensional o de más dimensiones, escribiendo el índice o índices entre corchetes "[]" y separando con comas "," cuando sean más de uno: [1], [5, 23], [-1, 1, 10].

En cada **eje** o dimensión el índice 0 hace referencia al primer elemento, el 1 al segundo y así sucesivamente hasta el último. También es posible referenciar comenzando desde el último elemento utilizando números negativos: en ese caso el índice -1 hace referencia al último elemento, el -2 al penúltimo y así sucesivamente.

Cuando se hace referencia a un índice inexistente en un array Numpy genera la excepción **IndexError**.

```
# Acceder a los elementos de un array.

# Declarar 3 arrays con distinto número de dimensiones:

a = np.array([1, 2, 3, 4, 5])
b = np.array([[1, 2, 3], [4, 5, 6]])
c = np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]])
print(a)

# [1 2 3 4 5]

print(b)

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

print(c)

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

# Acceder al primer elemento de cada array:
```

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...

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...

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...

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...

Tkinter: Tipos de ventanas
Ventanas de aplicación y de diálogos En la entrada anterior tratamos los distintos gestores de geometría que se utilizan para di...

El módulo random
El módulo random de la librería estándar de Python incluye un conjunto de funciones

```
print(a[0])

# 1

print(b[0,0])

# 1

print(c[0,0,0])

# 1

# Acceder a elementos de cada array:

print(a[2])

# 3

print(b[1,0])

# 4

print(c[1,0,1])

# 6

# Acceder al último elemento de cada array:

print(a[-1])

# 5

print(b[-1,-1])

# 6

print(c[-1,-1,-1])

# 8
```

[:], [::], ...

Acceder a varios elementos de un array.

Para acceder a varios elementos se utiliza el mismo procedimiento que se sigue con listas y tuplas en Python: si los elementos son consecutivos se expresa un rango con el índice del elemento inicial seguido de dos puntos ':' y el índice del elemento final:

- array[0:3] : Acceder a los 3 primeros elementos (desde el índice 0 al 2).
- array[1:-2] : Acceder a los elementos desde el índice 1 al antepenúltimo.

También se puede acceder a elementos separados entre sí por un número fijo de elementos (paso). Para estos casos a la expresión anterior hay que añadir dos puntos ':' seguido del valor que indica el paso o separación entre elementos:

- array[0:6:2] : Acceder a los elementos con el indice 0, 2 y 4.
- array[-1:-4:-2] : Acceder a los elementos que ocupan la última y antepenúltima posición.

Si se omite en la expresión el índice del elemento inicial la selección comenzará en el primer elemento. En contraposición, si se omite el ordinal del elemento final el acceso terminará en el último elemento:

- array[:4] : Acceder a los cuatro primeros elementos (desde el índice 0 al 3).
- array[2:] : Acceder a los elementos desde el índice 2 al último.
- array[:] : Acceder a todos los elementos.

En referencias de arrays multidimensionales se utilizará el mismo sistema para el acceso a los elementos en cada uno de sus ejes:

- array[0:2, 0:2] : Accede a los elementos 0,0; 0,1; 1,0 y 1,1 de un array 2D.
- array[:, 1:3] : Accede a todos los elementos del eje 0 pero con índice 1 y 2 en el eje 1.

Cuando se asigna un rango de un array a una variable en realidad se está asignando un array con los elementos seleccionados, en definitiva, un nuevo objeto **ndarray**.

```
# Acceder a varios elementos de un array.
```

que permiten obtener de distintos modos números a...

Archivo

noviembre 2019 (3)

python.org

pypi.org

Sitios

- ActivePython
- Anaconda
- Bpython
- Django
- Flask
- Ipython
- IronPython
- Matplotlib
- MicroPython
- Numpy
- Pandas
- Pillow
- PortablePython
- PyBrain
- PyCharm
- PyDev
- PyGame
- Pypi
- PyPy
- Pyramid
- Python.org
- PyTorch
- SciPy.org
- Spyder
- Tensorflow
- TurboGears

```
# Declarar 3 arrays con distinto número de dimensiones:

a = np.array([1, 2, 3, 4, 5])
b = np.array([[1, 2, 3], [4, 5, 6]])
c = np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]])

# Acceder a los cuatro primeros elementos (del índice 0 al 3):

print(a[0:4])

# [1 2 3 4]

# Acceder a los elementos del 3º hasta el penúltimo (del
# índice 2 al 4):

print(a[2:-1])

# [3 4]

# Acceder a los elementos impares del array:

print(a[::2])

# [1 3 5]

# Acceder a los elementos pares del array:

print(a[1::2])

# [2 4]

# Acceder a los elementos con índices 0, 3 y 4:

print(a[[0, 3, 4]])

# [1, 4, 5]

# Acceder a los últimos elementos en el eje 0 (filas) y
# a todos en el eje 1 (columnas):

print(b[-1, :])

# [4 5 6]

# Acceder a los últimos elementos en eje 0 y a los del
# índice 1 hasta el final en eje 1:

print(b[-1, 1:])

# [5 6]

# Acceder a todos los elementos en eje 0, en eje 1 y a los
# del índice 1 en adelante en eje 2:

print(c[:,:,1:])

# [[[2]
#   [4]]
#
#  [[6]
#   [8]]]
```

Asignar elementos a una variable.

var = []

Asignar un elemento a una variable Numpy.

```
a = np.array([1, 2, 3, 4, 5])
b = np.array([[1, 2, 3], [4, 5, 6]])
c = np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]])
var1 = a[3]
var2 = b[1,2]
var3 = c[0,1,0]
print(var1)

# 4

print(var2)
```

```
# 6

print(var3)

# 3

# Obtener el tipo de una variable Numpy:

a = np.array([1, 2, 3, 4, 5])
var1 = a[3]
type(var1)

# numpy.int64
```

var = [:]

Asignar una selección de elementos.

```
a = np.array([1, 2, 3, 4, 5])
b = np.array([[1, 2, 3], [4, 5, 6]])
c = np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]])
var1 = a[3:]
var2 = b[:,-1]
var3 = c[:1,:,:]
print(var1)

# [4 5]

print(var2)

# [3 6]

print(var3)

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

item()

Asignar un elemento a una variable Python.

```
# Asignar el valor del elemento 1 a una variable Python:

var1 = a.item(1)
print(var1)

# 2

# Asignar el valor del elemento (1,0) a una variable Python:

var2 = b.item(1,0)
print(var2)

# 4

# Obtener el tipo de una variable Python:

type(var1)

# int
```

[condición]

Asignar los elementos que cumplan una condición.

```
a = np.array([[1, 2, 3, 4], [5, 6, 7, 8]], dtype=np.int8)
b = a[a > 4]
print(b)

# [5 6 7 8]
```

Añadir, insertar y suprimir elementos.

append()

Añadir elementos al final de un array.

```
# Añadir 2 elementos al final de un array:

a = np.array([1, 2, 3, 4, 5])
a = np.append(a, [6, 7])
print(a)

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

insert()

Insertar elementos en un array.

```
# Insertar en la posición 3 el valor 0:

a = np.array([1, 2, 3, 4, 5, 6, 7])
a = np.insert(a, 3, 0)
print(a)

# [1 2 3 0 4 5 6 7]

# Insertar en la posición 1 los valores -1 y -2:

a = np.insert(a, 1, [-1, -2])
print(a)

# [ 1 -1 -2  2  3  0  4  5  6  7]

# Insertar columna (eje 1) con valores 0 en la columna
# con índice 2:

b = np.array([[1, 2, 3], [4, 5, 6]])
b = np.insert(b, 2, 0, axis=1)
print(b)

# [[1 2 0 3]
#  [4 5 0 6]]

# Insertar dos columnas (eje 1) con valores -1 y -2
# repetidos al comienzo del array:

b = np.insert(b, 0, [[-1], [-2]], axis=1)
print(b)

# [[-1 -2  1  2  0  3]
#  [-1 -2  4  5  0  6]]

# Insertar columna (eje 1) con valores -4 y -5 al
# comienzo del array:

b = np.insert(b, [0], [[-4], [-5]], axis=1)
print(b)

# [[-4 -1 -2  1  2  0  3]
#  [-5 -1 -2  4  5  0  6]]

# Insertar dos filas (eje 0) con valores 0 en las
# posiciones 0 y 2:

b = np.insert(b, [0, 2], 0, axis=0)
print(b)

# [[ 0  0  0  0  0  0  0]
#  [-4 -1 -2  1  2  0  3]
#  [-5 -1 -2  4  5  0  6]
#  [ 0  0  0  0  0  0  0]]
```

delete()

Suprimir elementos de un array.

```
# Suprimir el elemento 2 del array:

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

# [1 2 4 5]

# Suprimir la primera fila (eje 0) del array:

b = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])
b = np.delete(b, 0, axis=0)
print(b)

# [[ 5  6  7  8]
#  [ 9 10 11 12]]

Suprimir las columnas (eje 1) 0 y 2 del array:

b = np.delete(b, [0, 2], axis=1)
print(b)

# [[ 6  8]
#  [10 12]]
```

delete() y s_[]]

Suprimir un rango de elementos.

```
# Suprimir las 2 primeras columnas (eje 1) del array:

b = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])
b = np.delete(b, np.s_[0:2], axis=1)
print(b)

# [[ 3  4]
#  [ 7  8]
#  [11 12]]

# Suprimir las columnas (eje 1) pares del array:

b = np.delete(b, np.s_[::2], axis=1)
print(b)

# [[ 4]
#  [ 8]
#  [12]]
```

vstack()

Añadir filas a un array.

```
# Añadir una fila (eje 0).

b = np.array([[1, 2, 3], [4, 5, 6]])
b = np.vstack([b, [7, 8, 9]])
print(b)

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

hstack()

Añadir columnas a un array.

```
# Añadir una columna (eje 1):

b = np.hstack([b, ([-1], [-2], [-3])])
print(b)
```

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

Modificar elementos.

array[indice/s] = valor | itemset()

Modificar un elemento.

```
# Cambiar el valor del elemento 4 por 0:

a = np.array([1, 2, 3, 4, 5])
a[4] = 0  # equivalente a a.itemset(4, 0)
print(a)

# [1 2 3 4 0]

# Cambiar el valor del elemento (1,1) por -1:

b = np.array([[1, 2, 3], [4, 5, 6]])
b[1, 1] = -1  # equivalente a b.itemset((1, 1), -1)
print(b)

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

Operaciones con múltiples elementos.

take()

Crear un array a partir de referencias a los elementos de otro array.

```
a = np.array([1, 2, 3, 4, 5])
indices = [0, 2, 4]
b = a.take(indices)
print(b)

# [1 3 5]
```

put()

Sustituir valores de un array por nuevos a partir de sus referencias.

```
a = np.array([1, 2, 3, 4, 5])
indices = [1, 3, 4]
nuevos = [10, 30, 40]
a.put(indices, nuevos)
print(a)

# [ 1 10  3 30 40]
```

putmask()

Modificar los elementos del array que cumplan una condición.

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

[[ 1  2  3]
 [ 8 10 12]]
```

repeat()

Crear un array con elementos repetidos.

```
# Crear un array con el mismo elemento repetido:

a = np.repeat(3, 4)
print(a)

# [3 3 3 3]

# Crear un array con varios elementos repetidos:

a = np.repeat([1,2,3], 2)
print(a)

# [1 1 2 2 3 3]

# Crear un array bidimensional repitiendo filas 2 veces:

a = np.repeat([[1,2],[3,4]], 2, axis=0)
print(a)

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

# Crear un array bidimensional repitiendo columnas 2 veces:

a = np.repeat([[1,2],[3,4]], 2, axis=1)
print(a)

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

compress()

Seleccionar por su posición elementos de un array.

```
# Seleccionar elementos de "a" con valor 1 en la misma posición:

a = np.array([99, 98, 97, 88, 87, 86, 77, 76, 75])
b = a.compress([0, 0, 1, 0, 0, 1])
print(b)

# [97 86]

# Obtener segmentos de "a" con valor 1 en la misma posición:

a = np.array([[99, 98, 97], [88, 87, 86], [77, 76, 75]])
b = a.compress([1,0,1], axis=0)
print(b)

# [[99 98 97]
#  [77 76 75]]
```

diagonal()

Obtener la diagonal de un array.

```
a = np.arange(9).reshape(3,3)
print(a)

# [[0 1 2]
#  [3 4 5]
#  [6 7 8]]

print(a.diagonal())    # [0 4 8]
print(a.diagonal(-1))  # [3 7]
print(a.diagonal(1))   # [1 5]
```

choose()

Crear un array seleccionando elementos de varios arrays.

```
# Crear un array seleccionando los elementos del segundo array:

a = np.array([1, 2, 3, 4])
b = np.array([5, 6, 7, 8])
matrices = [a, b]
c = np.choose([1], matrices)
print(c)

# [5 6 7 8]

# Crear un array seleccionando elementos de varios arrays.

# Seleccionar el primer elemento del array a; el segundo elemento
# del array b y el tercer elemento del array b.

# matrices es un lista que contiene los arrays a, b, c que ocupan
# las posiciones 0, 1 y 2, respectivamente.

# En el método choose() el primer argumento es una lista que indica
# los elementos que hay que seleccionar. La posición de sus valores
# en dicha lista señalan la posición del elemento a seleccionar y
# el valor el array de donde hay que seleccionar:

a = np.array([10, 20, 30])
b = np.array([40, 50, 60])
c = np.array([70, 80, 90])
matrices = [a, b, c]
d = np.choose([0, 1, 1], matrices)
print(d)

# [10 50 60]


# Seleccionar el primer elemento del array c; el segundo del
# array b y el tercero del array a:

a = np.array([10, 20, 30])
b = np.array([40, 50, 60])
c = np.array([70, 80, 90])
matrices = [a, b, c]
d = np.choose([2, 1, 0], matrices)
print(d)

# [70 50 30]
```

for/in, enumerate(), range()

Recorrer todos los elementos de un array.

```
# Recorrer todos los elementos del vector a con enumerate().

a = np.array([1, 2, 3, 4, 5])
for indice, elemento in enumerate(a):
    print(f'{indice}: {elemento}')

# 0: 1
# 1: 2
# 2: 3
# 3: 4
# 4: 5

# Recorrer todos los elementos de un array 2D con range():

b = np.array([[1, 2, 3], [4, 5, 6]])
filas, columnas = b.shape
for fila in range(filas):
    for columna in range(columnas):
        print(f'{fila},{columna}: {b[fila, columna]}')

# 0,0: 1
# 0,1: 2
# 0,2: 3
# 1,0: 4
# 1,1: 5
# 1,2: 6
```

[for in]] | [for in if]

Listas de comprensión.

```
# Generar lista con elementos elevados al cuadrado:

a = np.array([1, 2, 3, 4, 5])
lista1 = [valor**2 for valor in a]
print(lista1)

# [1, 4, 9, 16, 25]

# Generar lista con elementos pares elevados al cuadrado:

lista2 = [valor**2 for valor in a if valor % 2 == 0]
print(lista2)

# [4, 16]
```

Publicado por Pherkad en 1:43

Etiquetas: Numpy

Entrada más reciente Inicio Entrada antigua