Clase Numpy AD

January 30, 2024

1 Clase de Numpy - Ademass (1 hora)

[1]:
```
import numpy as np
```

1.0.1 Dado el siguiente array bidimensional de 4x3.

[2]:
```
my_array = np.linspace(0, 11, 12, dtype=int).reshape(4,3)
```

[3]:
```
my_array
```

[3]:
```
array([[ 0,  1,  2],
       [ 3,  4,  5],
       [ 6,  7,  8],
       [ 9, 10, 11]])
```

1.0.2 Definimos las siguientes características:

Característica	Descripción	Valor para my_array
Axis	cada una de las dimensiones del array	tiene 2 (x,y)
Length	el número de valores que contiene una dimensión del array	x=3, y=4
Index	posición de un valor dentro del array	7 corresponde a (2,1)
Value	el valor que contiene el array en un index concreto	(1,2) tiene el valor 5
Rank	cuántas dimensiones tiene el array	2
Size	el número de elementos (celdas) que contiene el array	12
shape	el listado de AXIS con su longitud	4 x 3

1.0.3 Algunos ejemplos:

[4]:
```
# Array vacío de dimensión 4x4 (relleno con basura)
np.empty((4,4))
```

[4]:
```
array([[4.64319115e-310, 0.00000000e+000, 2.41907520e-312,
        2.14321575e-312],
       [2.46151512e-312, 2.31297541e-312, 2.35541533e-312,
        2.05833592e-312],
       [2.22809558e-312, 2.56761491e-312, 2.48273508e-312,
        2.05833592e-312],
       [2.05833592e-312, 2.29175545e-312, 2.07955588e-312,
        2.14321575e-312]])
```

[5]:
```
# Array vacío de dimensión 4x4 (relleno con zeros)
np.zeros((4,4), int)
```

[5]:
```
array([[0, 0, 0, 0],
       [0, 0, 0, 0],
       [0, 0, 0, 0],
```

```
       [0, 0, 0, 0]])
```

[6]:
```
# Array dimensión 2x3 relleno de "unos"
np.ones((2,3), int)
```

[6]:
```
array([[1, 1, 1],
       [1, 1, 1]])
```

[7]:
```
# Array de 5x4 relleno de ochos
np.full((5,4), 8)
```

[7]:
```
array([[8, 8, 8, 8],
       [8, 8, 8, 8],
       [8, 8, 8, 8],
       [8, 8, 8, 8],
       [8, 8, 8, 8]])
```

[8]:
```
# Array de 5x3 relleno con números aleatorios
np.random.rand(5,3)
```

[8]:
```
array([[0.50399253, 0.07466291, 0.1653578 ],
       [0.48787129, 0.35690243, 0.85998145],
       [0.64758805, 0.87975286, 0.06688693],
       [0.49216076, 0.43082918, 0.46064847],
       [0.20427743, 0.36472118, 0.02503452]])
```

[9]:
```
# Array de 4x3 relleno de números enteros aleatorios entre 0 y 9
arr = np.random.randint(0,10, (4,3))
arr
```

[9]:
```
array([[3, 2, 9],
       [5, 6, 8],
       [7, 7, 9],
       [9, 7, 0]])
```

[10]:
```
# El array anterior "reshaped" a 3x4
arr.reshape(3,4) # no modifica el original
```

[10]:
```
array([[3, 2, 9, 5],
       [6, 8, 7, 7],
       [9, 9, 7, 0]])
```

[11]:
```
# Otra forma de hacer "reshape"
np.reshape(arr, (2,6))
```

[11]:
```
array([[3, 2, 9, 5, 6, 8],
       [7, 7, 9, 9, 7, 0]])
```

[12]:
```
# podemos observar que el array original no cambia su "shape"
print(arr)
```

```
[[3 2 9]
 [5 6 8]
 [7 7 9]
 [9 7 0]]
```

[13]:
```
# convertir una lista en ndarray
lista = [22, 34, 1, -43, 0, 66]
np.array(lista).reshape(2,3)
```

[13]:
```
array([[ 22,  34,   1],
       [-43,   0,  66]])
```

[14]:
```
# Dieciséis números distribuidos entre 0 y 15 (le llama rangos)
np.linspace(0, 15, 16, dtype=int).reshape(4,4)
```

[14]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15]])
```

[15]:
```
# para saber el "shape" de un ndarray
arr.shape
```

[15]:
```
(4, 3)
```

[16]:
```
# para conocer el tamaño (cantidad de elemntos)
arr.size
```

[16]:
```
12
```

1.0.4 Operaciones con ndarrays

[17]:
```
# suma de arrays
arr1 = np.array([2,2,2])
arr2 = np.array([3,3,3])

arr1 + arr2
```

[17]:
```
array([5, 5, 5])
```

[18]:
```
# resta de arrays
arr1 - arr2
```

[18]:
```
array([-1, -1, -1])
```

[19]:
```
# multiplicación de arrays
arr1 * arr2
```

[19]:
```
array([6, 6, 6])
```

[20]:
```
# división de arrays
arr1 / arr2
```

[20]:
```
array([0.66666667, 0.66666667, 0.66666667])
```

1.0.5 Funciones especiales con arrays

Dado el siguiente array:

[21]:
```
arr
```

[21]:
```
array([[3, 2, 9],
       [5, 6, 8],
       [7, 7, 9],
       [9, 7, 0]])
```

Calculamos:

[22]:
```
# media aritmética
arr.mean()
```

[22]:
```
6.0
```

[23]:
```
# valor máximo
arr.max()
```

[23]:
```
9
```

[24]:
```
# valor mínimo
arr.min()
```

[24]:
```
0
```

[25]:
```
# suma todos sus elementos
arr.sum()
```

[25]:
```
72
```

1.0.6 Accediendo a los elementos del ndarray

[26]:
```
arr[2][2]
```

[26]:
```
9
```

[27]:
```
arr[0,2]
```

[27]:
```
9
```

[28]:
```
# submatriz 1,2 - 2,3
arr[0:2,1::]
```

[28]:
```
array([[2, 9],
       [6, 8]])
```

1.0.7 Desenvolviendo un array multidimensional

[29]:
```
arr.ravel()
```

[29]:
```
array([3, 2, 9, 5, 6, 8, 7, 7, 9, 9, 7, 0])
```