



**DigitalHouse** >  
Coding School

# **DATA SCIENCE**

MÓDULO 1

Numpy

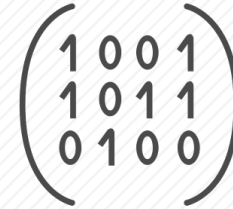
Ago 2017

# Numpy

1 **Manipular datos con Arrays de Numpy**

2 **Vectorización de operaciones y optimización**

3 **Acceder a los datos de forma eficiente**


$$\begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

Un arreglo es una colección de items de datos, llamados elementos, asociados a un único nombre de variable.

Diseñados para:

- Facilitar la programación y proveer buena performance.
- Almacenar y manipular grandes colecciones de datos.
- Simplificar la tarea de nombrar y referenciar ítems individuales en una colección de datos
- Permitir manipular una colección entera de datos con una simple sentencia.

Un **arreglo** es una estructura de datos que permite agrupar varios valores bajo un único nombre. Los elementos de un arreglo son todos del mismo tipo (a diferencia de las listas de Python).

Dimensions	Example	Terminology																																										
1	<table><tr><td>0</td><td>1</td><td>2</td></tr></table>	0	1	2	Vector																																							
0	1	2																																										
2	<table><tr><td>0</td><td>1</td><td>2</td></tr><tr><td>3</td><td>4</td><td>5</td></tr><tr><td>6</td><td>7</td><td>8</td></tr></table>	0	1	2	3	4	5	6	7	8	Matrix																																	
0	1	2																																										
3	4	5																																										
6	7	8																																										
3	<table><tr><td>0</td><td>1</td><td>2</td></tr><tr><td>3</td><td>4</td><td>5</td></tr><tr><td>6</td><td>7</td><td>8</td></tr></table>	0	1	2	3	4	5	6	7	8	3D Array (3 <sup>rd</sup> order Tensor)																																	
0	1	2																																										
3	4	5																																										
6	7	8																																										
N	<table><tr><td><table><tr><td>0</td><td>1</td><td>2</td></tr><tr><td>3</td><td>4</td><td>5</td></tr><tr><td>6</td><td>7</td><td>8</td></tr></table></td><td>...</td><td><table><tr><td>0</td><td>1</td><td>2</td></tr><tr><td>3</td><td>4</td><td>5</td></tr><tr><td>6</td><td>7</td><td>8</td></tr></table></td></tr><tr><td><table><tr><td>0</td><td>1</td><td>2</td></tr><tr><td>3</td><td>4</td><td>5</td></tr><tr><td>6</td><td>7</td><td>8</td></tr></table></td><td>...</td><td><table><tr><td>0</td><td>1</td><td>2</td></tr><tr><td>3</td><td>4</td><td>5</td></tr><tr><td>6</td><td>7</td><td>8</td></tr></table></td></tr></table>	<table><tr><td>0</td><td>1</td><td>2</td></tr><tr><td>3</td><td>4</td><td>5</td></tr><tr><td>6</td><td>7</td><td>8</td></tr></table>	0	1	2	3	4	5	6	7	8	...	<table><tr><td>0</td><td>1</td><td>2</td></tr><tr><td>3</td><td>4</td><td>5</td></tr><tr><td>6</td><td>7</td><td>8</td></tr></table>	0	1	2	3	4	5	6	7	8	<table><tr><td>0</td><td>1</td><td>2</td></tr><tr><td>3</td><td>4</td><td>5</td></tr><tr><td>6</td><td>7</td><td>8</td></tr></table>	0	1	2	3	4	5	6	7	8	...	<table><tr><td>0</td><td>1</td><td>2</td></tr><tr><td>3</td><td>4</td><td>5</td></tr><tr><td>6</td><td>7</td><td>8</td></tr></table>	0	1	2	3	4	5	6	7	8	ND Array
<table><tr><td>0</td><td>1</td><td>2</td></tr><tr><td>3</td><td>4</td><td>5</td></tr><tr><td>6</td><td>7</td><td>8</td></tr></table>	0	1	2	3	4	5	6	7	8	...	<table><tr><td>0</td><td>1</td><td>2</td></tr><tr><td>3</td><td>4</td><td>5</td></tr><tr><td>6</td><td>7</td><td>8</td></tr></table>	0	1	2	3	4	5	6	7	8																								
0	1	2																																										
3	4	5																																										
6	7	8																																										
0	1	2																																										
3	4	5																																										
6	7	8																																										
<table><tr><td>0</td><td>1</td><td>2</td></tr><tr><td>3</td><td>4</td><td>5</td></tr><tr><td>6</td><td>7</td><td>8</td></tr></table>	0	1	2	3	4	5	6	7	8	...	<table><tr><td>0</td><td>1</td><td>2</td></tr><tr><td>3</td><td>4</td><td>5</td></tr><tr><td>6</td><td>7</td><td>8</td></tr></table>	0	1	2	3	4	5	6	7	8																								
0	1	2																																										
3	4	5																																										
6	7	8																																										
0	1	2																																										
3	4	5																																										
6	7	8																																										

- Un vector es un arreglo unidimensional
- Una tabla o matriz es un arreglo bidimensional
- Por último un arreglo puede tener N dimensiones.

## — Creación:

```
import numpy as np  
a = np.array([1, 2, 3, 4, 5])  
b = np.array([5, 4, 3, 2, 1])
```

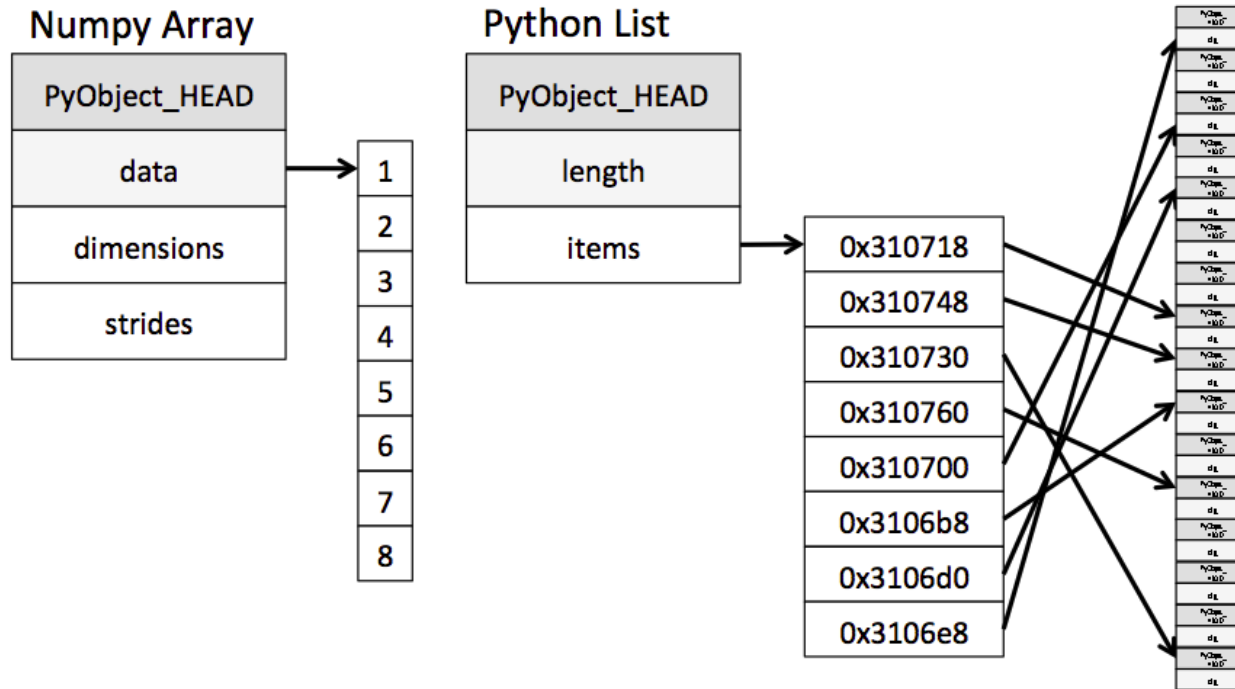
## — Acceso / Indexación:

```
a[0]
```

Los arreglos se indexan desde 0 a  $\text{len}(a) - 1$ :

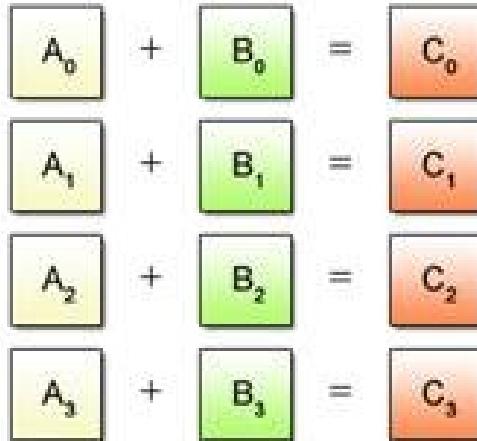
- $a[0]$  devuelve el valor de la primera posición del arreglo
- $a[\text{len}(a)-1]$  devuelve el valor de la última posición del arreglo

Los Arrays de Numpy son estructuras mucho más eficientes para operar sobre los datos que las listas de Python. Esto en parte se debe a que en un Numpy Array el tipo de datos es fijo.

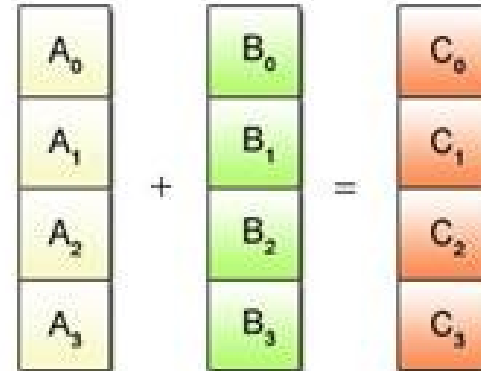


Las operaciones vectorizadas trabajan sobre los datos como un bloque. Por eso es necesario que los tipos sean homogéneos entre todos los elementos. En la operación vectorizada, no recorremos los elementos en orden.

**Operación escalar**



**Operación vectorizada**





## Tipos de datos en Numpy:

Data type	Description
<code>bool_</code>	Boolean (True or False) stored as a byte
<code>int_</code>	Default integer type (same as C long; normally either <code>int64</code> or <code>int32</code> )
<code>intc</code>	Identical to C <code>int</code> (normally <code>int32</code> or <code>int64</code> )
<code>intp</code>	Integer used for indexing (same as C <code>ssize_t</code> ; normally either <code>int32</code> or <code>int64</code> )
<code>int8</code>	Byte (-128 to 127)
<code>int16</code>	Integer (-32768 to 32767)
<code>int32</code>	Integer (-2147483648 to 2147483647)
<code>int64</code>	Integer (-9223372036854775808 to 9223372036854775807)
<code>uint8</code>	Unsigned integer (0 to 255)
<code>uint16</code>	Unsigned integer (0 to 65535)
<code>uint32</code>	Unsigned integer (0 to 4294967295)
<code>uint64</code>	Unsigned integer (0 to 18446744073709551615)
<code>float_</code>	Shorthand for <code>float64</code> .
<code>float16</code>	Half precision float: sign bit, 5 bits exponent, 10 bits mantissa
<code>float32</code>	Single precision float: sign bit, 8 bits exponent, 23 bits mantissa
<code>float64</code>	Double precision float: sign bit, 11 bits exponent, 52 bits mantissa
<code>complex_</code>	Shorthand for <code>complex128</code> .
<code>complex64</code>	Complex number, represented by two 32-bit floats
<code>complex128</code>	Complex number, represented by two 64-bit floats

Ejemplo de operaciones vectorizada:

```
In [7]: x = np.arange(4)
print("x      =", x)
print("x + 5 =", x + 5)
print("x - 5 =", x - 5)
print("x * 2 =", x * 2)
print("x / 2 =", x / 2)
print("x // 2 =", x // 2) # floor division
```

```
x      = [0 1 2 3]
x + 5 = [5 6 7 8]
x - 5 = [-5 -4 -3 -2]
x * 2 = [0 2 4 6]
x / 2 = [ 0.  0.5  1.  1.5]
x // 2 = [0 0 1 1]
```

Operator	Equivalent ufunc	Description
+	np.add	Addition (e.g., 1 + 1 = 2)
-	np.subtract	Subtraction (e.g., 3 - 2 = 1)
-	np.negative	Unary negation (e.g., -2)
*	np.multiply	Multiplication (e.g., 2 * 3 = 6)
/	np.divide	Division (e.g., 3 / 2 = 1.5)
//	np.floor_divide	Floor division (e.g., 3 // 2 = 1)
**	np.power	Exponentiation (e.g., 2 ** 3 = 8)
%	np.mod	Modulus/remainder (e.g., 9 % 4 = 1)

Numpy tiene un conjunto de reglas para aplicar operaciones miembro a miembro en Arrays de diferente tamaño. Se proyectan los valores de los arrays para poder operar sobre los mismos.

`np.arange(3)+5`

0	1	2
---	---	---

 + 

5	5	5
---	---	---

 = 

5	6	7
---	---	---

`np.ones((3,3))+np.arange(3)`

1	1	1
1	1	1
1	1	1

 + 

0	1	2
0	1	2
0	1	2

 = 

1	2	3
1	2	3
1	2	3

`np.arange(3).reshape((3,1))+np.arange(3)`

0	0	0
1	1	1
2	2	2

 + 

0	1	2
0	1	2
0	1	2

 = 

0	1	2
1	2	3
2	3	4

Llamamos “indexing” al proceso de acceder a los elementos de un array con algún criterio.

Existen tres tipos de indexing en Numpy:

- Slicing: cuando accedemos a los elementos con los parámetros *start, stop, step*.  
Por ejemplo `my_array[0:5:-1]`
- Fancy Indexing: Cuando creamos una lista de índices y la usamos para acceder a ciertos elementos del array:  
`my_array[[3,5,7,8]]`
- Boolean Indexing: Cuando creamos una “máscara booleana” (un array o lista de True y False) para acceder a ciertos elementos: `my_array[my_array > 4]`

# Práctica guiada

Utilizar los datasets `volumen_ventas_producto_sucursal.csv` y `precio_producto.csv` que indican las ventas en cantidad para cada producto en cada sucursal y el precio de cada producto respectivamente.

Ejercicios:

- Calcular la cantidad de unidades vendidas para cada sucursal
- Calcular la cantidad de unidades vendidas para cada producto
- ¿Cuál es la sucursal que más vendió?
- Calcular la facturación de cada sucursal