



arreglos NumPy

<https://numpy.org/doc/stable/>

NumPy es el paquete fundamental para la computación científica en Python.

Es una biblioteca de Python que proporciona objetos de arreglos multidimensionales, y una variedad de rutinas para operaciones rápidas en esos arreglos, que incluyen manipulación matemática, de formas, clasificación, selección, etc.

# Cargamos el modulo

```
import numpy as np
```

<https://numpy.org/doc/stable/reference/arrays.html>

NumPy proporciona un tipo de arreglo N-dimensional.

El `ndarray`, describe una colección de "elementos" del mismo tipo.


Los elementos se indexan utilizando, por ejemplo,  $n$  enteros.

En particular se pueden crear *vectores* (arreglos unidimensionales) de tamaño  $n$

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$


y *matrices* (arreglos bidimensionales) de tamaño  $m \times n$  ( $m$  renglones y  $n$  columnas)

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \dots & a_{m,n} \end{pmatrix}$$



Para generar arreglos, tenemos distintas alternativas con numpy

<https://numpy.org/doc/stable/reference/routines.array-creation.html>

- From shape or value
  - From existing data
  - Creating record arrays
  - Creating character arrays
  - Numerical ranges
  - Building matrices
  - The matrix class
- 

From existing data:

Para arreglos unidimensionales tenemos:

```
v = np.array([1,2,3,4,5,6,7,8,9,10,11,12,13])
```

```
>>> import numpy as np
>>> v = np.array([1,2,3,4,5,6,7,8,9,10,11,12,13])
>>> print(v)
[ 1  2  3  4  5  6  7  8  9 10 11 12 13]
```

El acceso a los elementos de los arreglos unidimensionales de numpy se puede hacer exactamente igual al acceso de los elementos un `str`.

De esta forma se pueden modificar sus valores mediante la referencia a un **elemento** o a un **subconjunto de elementos**.

```
>>> # Referencia a los elementos del vector
... print(v[0]) # Referencia al primer elemento
1
>>> print(v[1:3]) # Referencia a los elementos de la posición 1 a la 2
[2 3]
>>> print(v[:]) # Referencia a todos los elementos
[ 1  2  3  4  5  6  7  8  9 10 11 12 13]
>>>
```

## From shape or value:

`np.empty(shape)` Permite crear un arreglo n-dimensional sin inicialización de entradas del tamaño definido en *shape*.

`np.ones(shape)` Permite crear un arreglo n-dimensional con todas sus entradas *1* del tamaño definido por *shape*.

`np.zeros(shape)` Permite crear un arreglo n-dimensional con todas sus entradas *0* del tamaño definido por *shape*.

*Para crear una matriz de tamaño  $m \times n$ , el argumento *shape* puede ser una lista o una tupla que contenga el número de filas y columnas si tiene dos entradas  $(m,n)$  o un vector con *m* entradas si sólo contiene una  $(m)$  o  $(m,)$*

`np.eye(m[,n])` Permite crear un arreglo bidimensional donde los elementos son cero, excepto en la diagonal principal donde son 1.

`np.identity(n)` Permite crear un arreglo identidad bidimensional de tamaño *n*.

Para arreglos bidimensionales tenemos:

```
M = np.array([[1,2,3],[4,5,6]])
```

```
>>> import numpy as np
>>> m = np.array([[1,2,3],[4,5,6]])
>>> print(m)
[[1 2 3]
 [4 5 6]]
```

El acceso a los elementos de los arreglos bidimensionales de numpy se puede hacer exactamente igual al acceso de los unidimensionales pero haciendo doble referencia.

```
>>> print(m[0][1]) # referencia al primer renglón, segunda columna
2
>>> print(m[1][:]) # Segundo renglon de m
[4 5 6]
>>> # subarray bidimensional, ambos renglones, segunda y tercer columna
>>> print(m[:,1:])
[[2 3]
 [5 6]]
```

<https://numpy.org/doc/stable/reference/routines.array-manipulation.html>

Hay métodos definidos para ndarray, estos son sólo algunos ejemplos:

`.diagonal()` Extrae la diagonal principal.

`.sum()` Permite calcular la suma de todos los elementos en el arreglo. Si es un arreglo n-dimensional, permite hacer la suma sobre alguna dimensión específica obteniendo un arreglo (n-1)-dimensional.

`.prod()` Permite calcular el producto de todos los elementos en el arreglo. Si es un arreglo n-dimensional, permite hacer el producto sobre alguna dimensión específica obteniendo un arreglo (n-1)-dimensional.

También se tiene acceso a ciertos atributos que nos dan información sobre el arreglo.

`.shape` Permite saber la forma de un arreglo. Si el arreglo es una matriz arroja una tupla con el número de filas y columnas y si es un vector, regresa una tupla con el número de elementos.

`.size` Permite saber el número de elementos que hay en el arreglo.