PythonNumpy

January 19, 2026

0.1 PAO25-26- NumPy

Gabriel Salguero

0.2 NumPy ndarray

- NumPy (**Num** erical **Py** thon) es la de facto librería estándar para análisis numérico en Python
- Estructura de datos multidimensional eficiente (escrita en C): ndarray
- Colección de funciones para álgebra lineal, estadística descriptiva
- Ayuda al procesamiento de datos (np.where)

[2]:
```
#importar libreria
import numpy as np
```

0.2.1 Performance Sample, NumPy vs Plain Python

[3]:
```
my_list = list(range(10000000)) # Utilizando Listas Python
%time my_list = [x * 2 for x in my_list]
```

```
CPU times: total: 1.36 s
Wall time: 1.41 s
```

[4]:
```
my_arr = np.arange(10000000) # Utilizando NumPy Arrays
%time my_arr2 = my_arr * 2
```

```
CPU times: total: 31.2 ms
Wall time: 36.7 ms
```

0.2.2 Formas de crear ndarrays

[5]:
```
array = np.array([ [2,71,0,34], [2,171,-35,34] ]) # 2D
print(array)
```

```
[[  2  71   0  34]
 [  2 171 -35  34]]
```

[6]:
```
shopping_list = [ ['onions', 'carrots', 'celery'], ['apples',␣
 ↪'oranges','grapes'] ] # 2D
print(shopping_list)
array = np.array(shopping_list) # convertir una lista en una ndarray
print(array)
print(type(array))
```

```
[['onions', 'carrots', 'celery'], ['apples', 'oranges', 'grapes']]
[['onions' 'carrots' 'celery']
 ['apples' 'oranges' 'grapes']]
<class 'numpy.ndarray'>
```

[7]:
```
print(np.array([100,10,1])) # array a partir de una lista
print(np.arange(2,10,2.1)) # array a partir de una secuencia(range) 1D
```

```
[100  10   1]
[2.  4.1 6.2 8.3]
```

[8]:
```
# Generar arrays con valores fijos
print(np.zeros(2)) # 1D
print(np.ones((4,3))) # matrix 2x2
print(np.empty((2,2,2))) # matrix 2x2x2, no inicializados
print(np.full((5,2),2)) # matrix 5x2
print(np.eye(3)) # matrix identity
```

```
[0. 0.]
[[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]
[[[9.88e-324 3.51e-322]
  [0.00e+000 1.68e-322]]

 [[9.88e-324 8.45e-322]
  [      nan 1.68e-322]]]
[[2 2]
 [2 2]
 [2 2]
 [2 2]
 [2 2]]
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

0.2.3 Obtener información sobre ndarray

[9]:
```
my_ndarray = np.array([2,71,0,34])
print(my_ndarray)
```

```
[ 2 71  0 34]
```

[10]:
```
print(my_ndarray.argmax())  # índice del valor más alto
print(np.amax(my_ndarray))

print(my_ndarray.argmin())  # índice del valor más bajo
print(np.amin(my_ndarray))

print(my_ndarray.nonzero()) # retorna los índices de los elementos distintos a 0
```

```
1
71
2
0
(array([0, 1, 3]),)
```

[11]:
```
my_ndarray = np.full((5, 2,2), 2) # matrix 5x2x2
print(my_ndarray)

print(my_ndarray.size)   # número de elementos
print(my_ndarray.shape)  # dimensiones
print(my_ndarray.ndim)   # número de dimensiones
```

```
[[[2 2]
  [2 2]]

 [[2 2]
  [2 2]]

 [[2 2]
  [2 2]]

 [[2 2]
  [2 2]]

 [[2 2]
  [2 2]]]
20
(5, 2, 2)
3
```

[12]:
```
my_ndarray = np.array([2,71,0,34])

print(my_ndarray.size)   # número de elementos
print(my_ndarray.shape)  # dimensiones
print(my_ndarray.ndim)   # número de dimensiones
```

```
4
(4,)
1
```

Redimensionar los datos

[13]:
```
my_ndarray = np.array([[0, 71, 21, 19, 213, 412, 111, 98]]) # matrix 1x8
print(my_ndarray)
print(my_ndarray.shape)
print(my_ndarray.ndim)
```

```
[[  0  71  21  19 213 412 111  98]]
(1, 8)
2
```

[14]:
```
new_dims = my_ndarray.reshape(2, 4) # cambiar las dimensiones a 2x4
print(new_dims)
print(new_dims.ndim)
```

```
[[  0  71  21  19]
 [213 412 111  98]]
2
```

[15]:
```
new_dims = my_ndarray.reshape(4, 2) # cambiar las dimensiones a 4x2
print(new_dims)
print(new_dims.ndim)
```

```
[[  0  71]
 [ 21  19]
 [213 412]
 [111  98]]
2
```

[16]:
```
x = np.array([100, 10, 1]).reshape(3,1)
print(x.shape)
print(x)
```

```
(3, 1)
[[100]
 [ 10]
 [  1]]
```

[17]:
```
# reducir a 1D
new_dims = np.arange(9).reshape(3,3)
print(new_dims)
print(new_dims.flatten())
```

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

[18]:
```
# reducir
new_dims = np.arange(16).reshape(4,4)
print(new_dims)
print(new_dims.reshape(2,-1)) # el segundo valor se infiere del primero
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]
[[ 0  1  2  3  4  5  6  7]
 [ 8  9 10 11 12 13 14 15]]
```

0.2.4 Manipulación de ndarrays

[19]:
```
# Cambiar ejes
new_dims = np.arange(16).reshape(8,2)
print(new_dims)
print(new_dims.swapaxes(0,1))
```

```
[[ 0  1]
 [ 2  3]
 [ 4  5]
 [ 6  7]
 [ 8  9]
 [10 11]
 [12 13]
 [14 15]]
[[ 0  2  4  6  8 10 12 14]
 [ 1  3  5  7  9 11 13 15]]
```

[20]:
```
# Flip
new_dims = np.arange(16).reshape(4,4)
print(new_dims)
print(np.flip(new_dims, axis = None))
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]
[[15 14 13 12]
 [11 10  9  8]
 [ 7  6  5  4]
 [ 3  2  1  0]]
```

[21]:
```
# ordenar
array1 = np.array([10,2,9,17])
array1.sort()
print(array1)
```

```
[ 2  9 10 17]
```

[22]:
```
# Juntar dos arrays
a1 = [0,0,0,0,0,0]
a2 = [1,1,1,1,1,1]

print(np.hstack((a1,a2))) # una al lado de la otra
print(np.hstack((a1,a2)).shape)

print(np.vstack((a1,a2))) # una encima de la otra
print(np.vstack((a1,a2)).shape)
```

```
[0 0 0 0 0 0 1 1 1 1 1 1]
```

```
(12,)
[[0 0 0 0 0 0]
 [1 1 1 1 1 1]]
(2, 6)
```

[23]:
```
array1 = np.array([[2, 4], [6, 8]])
array2 = np.array([[3, 5], [7, 9]])

print(np.concatenate((array1, array2), axis = 0))
```

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

[24]:
```
# dividir un array
array = np.arange(16).reshape(4,4)
print(array)
print(np.array_split(array, 2, axis = 0)) # divide array en 2 partes iguales
print(np.array_split(array, 3)) # divide array en 3 partes "iguales"
```

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

[25]:
```
# dividir un array
array = np.arange(16).reshape(4,4)
print(array)
print(np.array_split(array, [3], axis = 1)) # divide array, por la fila 3
```

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

[26]:
```
split = np.array_split(array, [3], axis = 1)
print(split[0][3][0])
```

```
12
```

[27]:
```
array = np.arange(64).reshape(8,8)
print(array)
print(np.array_split(array, [1, 3], axis = 0)) # divide array por la fila 1 y 3

print(np.array_split(array, [1, 3], axis = 1)) # divide array por la columna 1
 ↪y 3
# axis = n_dimensions- 1
```

```
[[ 0  1  2  3  4  5  6  7]
 [ 8  9 10 11 12 13 14 15]
 [16 17 18 19 20 21 22 23]
 [24 25 26 27 28 29 30 31]
 [32 33 34 35 36 37 38 39]
 [40 41 42 43 44 45 46 47]
 [48 49 50 51 52 53 54 55]
 [56 57 58 59 60 61 62 63]]
[array([[0, 1, 2, 3, 4, 5, 6, 7]]), array([[ 8,  9, 10, 11, 12, 13, 14, 15],
       [16, 17, 18, 19, 20, 21, 22, 23]]), array([[24, 25, 26, 27, 28, 29, 30,
31],
       [32, 33, 34, 35, 36, 37, 38, 39],
       [40, 41, 42, 43, 44, 45, 46, 47],
       [48, 49, 50, 51, 52, 53, 54, 55],
       [56, 57, 58, 59, 60, 61, 62, 63]])]
[array([[ 0],
       [ 8],
       [16],
       [24],
       [32],
       [40],
       [48],
       [56]]), array([[ 1,  2],
       [ 9, 10],
       [17, 18],
       [25, 26],
       [33, 34],
       [41, 42],
       [49, 50],
       [57, 58]]), array([[ 3,  4,  5,  6,  7],
       [11, 12, 13, 14, 15],
       [19, 20, 21, 22, 23],
       [27, 28, 29, 30, 31],
       [35, 36, 37, 38, 39],
       [43, 44, 45, 46, 47],
```

```
       [51, 52, 53, 54, 55],
       [59, 60, 61, 62, 63]])]
```

0.3 Índices y slicing

- De manera análoga a índices y slicing para listas

[28]:
```
array = np.arange(10, 16)
print(array)
print(array[-1]) # acceso unidimensional
print(array[:-1]) # slice
```

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

[29]:
```
# bidimensional
array = np.arange(16).reshape(4,4)
print(array)
print(array[2, 2]) # fila, columna
print(array[3][2]) # fila, columna
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]
10
14
```

[30]:
```
# multidimensional
array = np.arange(36).reshape(2,3,6)
print(array)
print(array[0,2,4])
print(array[1][2][4])
```

```
[[[ 0  1  2  3  4  5]
  [ 6  7  8  9 10 11]
  [12 13 14 15 16 17]]

 [[18 19 20 21 22 23]
  [24 25 26 27 28 29]
  [30 31 32 33 34 35]]]
16
34
```

NumPy es row major

[31]:
```
row_major = np.array([1, 2, 3, 4, 5, 6])
row_major = row_major.reshape(2,3) # by default, row major
print(row_major)
```

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

[32]:
```
row_major = np.array([1, 2, 3, 4, 5, 6])
row_major = row_major.reshape(2,3,order='C') # by default, row major
print(row_major)

col_major = np.array([1, 2, 3, 4, 5, 6])
col_major = col_major.reshape(2,3,order='F') # col major
print(col_major)
```

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

Slice

[33]:
```
# se puede hacer slice por fila o columna
array = np.arange(81).reshape(9,9)
print(array)
print(array[1:3, 2::2])
```

```
[[ 0  1  2  3  4  5  6  7  8]
 [ 9 10 11 12 13 14 15 16 17]
 [18 19 20 21 22 23 24 25 26]
 [27 28 29 30 31 32 33 34 35]
 [36 37 38 39 40 41 42 43 44]
 [45 46 47 48 49 50 51 52 53]
 [54 55 56 57 58 59 60 61 62]
 [63 64 65 66 67 68 69 70 71]
 [72 73 74 75 76 77 78 79 80]]
[[11 13 15 17]
 [20 22 24 26]]
```

[34]:
```
# se puede hacer slice por fila o columna
array = np.arange(9).reshape(3,3)
print(array)
print(array[0,:])
```

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

Diferencias entre índices y slicing en listas y ndarrays:

- ¡Retornan una referencia, no una copia!

[35]:
```
# uso de slice en listas, retorna copia
lista = [0,1,2,3,4,5,6,7,8,9]
my_copy = lista[0:2] # devuelve una copia
print(my_copy)
my_copy[0] = 222
print(lista)
print(my_copy)
```

```
[0, 1]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[222, 1]
```

[36]:
```
# slicing retorna una referencia en ndarrays
array = np.arange(10)
my_copy = array[2:5] # devuelve una referencia
print(array)
print(my_copy)
my_copy[0] = 222
print(array)
print(my_copy)
```

```
[0 1 2 3 4 5 6 7 8 9]
[2 3 4]
[  0   1 222   3   4   5   6   7   8   9]
[222   3   4]
```

[37]:
```
# copiar ndarrays
array = np.arange(10)
my_copy = array[2:5].copy() # hay que hacer la copia explicitamente
my_copy[0] = 222
print(array)
print(my_copy)
```

```
[0 1 2 3 4 5 6 7 8 9]
[222   3   4]
```

- Permite asignar valores a un corte

[38]:
```
array = np.arange(10)
print(array)

print(array[0:3])

array[0:3] =-1
print(array)
```

```
[0 1 2 3 4 5 6 7 8 9]
[0 1 2]
[-1 -1 -1  3  4  5  6  7  8  9]
```

- Slicing condicionales (mask)

[39]:
```
# slicing condicional
array = np.arange(-3,4)
print(array)

mask = array %2 == 0
print(mask)
print(array[mask])

print(array[array %2 == 0])

array[mask] = 0
print(array)
```

```
[-3 -2 -1  0  1  2  3]
[False  True False  True False  True False]
[-2  0  2]
[-2  0  2]
[-3  0 -1  0  1  0  3]
```

[40]:
```
array = np.arange(-3,4)
print(array)

array[array < 0] = 0
print(array)
```

```
[-3 -2 -1  0  1  2  3]
[0 0 0 0 1 2 3]
```

[41]:
```
# slicing condicional
array = np.arange(-3,4)
print(array)

mask = array % 2 == 0
print(mask)

array_par = array[mask] # devuelve una copia
print(array_par)

array_par[0] =-10
print(array)
print(array_par)
```

```
[-3 -2 -1  0  1  2  3]
[False  True False  True False  True False]
[-2  0  2]
[-3 -2 -1  0  1  2  3]
[-10   0   2]
```

- Acceso a múltiples valores con listados de índices

[42]:
```
array = np.arange(0,16).reshape(4,4)
print(array)
print(array[[1,2]]) #selecting rows
print(array[:,[1,2]]) #selecting columns

# cross product of indexes
print(array[[1,2],:][:,[1,2]])

print(array[np.ix_([1,2],[1,2])]) # built in ix_

ar = array[[2,3,1]] # slice, copia
print(ar)
ar[0] = 999
print(ar)
print(array)
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]
[[ 4  5  6  7]
 [ 8  9 10 11]]
[[ 1  2]
 [ 5  6]
 [ 9 10]
 [13 14]]
[[ 5  6]
 [ 9 10]]
[[ 5  6]
 [ 9 10]]
[[ 8  9 10 11]
 [12 13 14 15]
 [ 4  5  6  7]]
[[999 999 999 999]
 [ 12  13  14  15]
 [  4   5   6   7]]
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]
```

0.4 Funciones matemáticas

- Aplicar operaciones entre escalares y arrays o entre arrays

[43]:
```
# listas y escalares
lista = [0,1,2,3]
lista *= 3
print(lista)

lista += [1]
print(lista)
```

```
[0, 1, 2, 3, 0, 1, 2, 3, 0, 1, 2, 3]
[0, 1, 2, 3, 0, 1, 2, 3, 0, 1, 2, 3, 1]
```

[44]:
```
# ndarray hace las operaciones sobre los elementos
array = np.ones(4, dtype = np.int8)
print(array)
array *= 2
print(array)
array += 10
print(array)

print(array.dtype) # check data type of array
```

```
[1 1 1 1]
[2 2 2 2]
[12 12 12 12]
int8
```

[45]:
```
# operaciones entre arrays es como operaciones entre matrices
array_1 = np.ones(4) * 2
array_2 = np.arange(4) + 1

print(array_1)
print(array_2)

print(array_1 + array_2)
print(array_1- array_2)
print(array_1 * array_2)
print(array_2 / array_1)
```

```
[2. 2. 2. 2.]
[1 2 3 4]
[3. 4. 5. 6.]
[ 1.  0. -1. -2.]
[2. 4. 6. 8.]
[0.5 1.  1.5 2. ]
```

0.4.1 Broadcasting

Si las arrays no tienen las mismas dimensiones, se hace broadcast del pequeño al grande-> broadcasting

0.4.2 Operadores unarios y binarios

Tipo	Operación	Descripción
Unario	abs	Valor absoluto de cada elemento
Unario	sqrt	Raíz cuadrada de cada elemento
Unario	exp	e^x, siendo x cada elemento
Unario	log / log10 / log2	Logaritmos en distintas bases de cada elemento
Unario	sign	Retorna el signo de cada elemento (-1 negativo, 0 o 1 positivo)
Unario	ceil	Redondea cada elemento por arriba
Unario	floor	Redondea cada elemento por abajo
Unario	isnan	Retorna si cada elemento es NaN
Unario	cos / sin / tan	Operaciones trigonométricas en cada elemento
Unario	arccos / arcsin / arctan	Inversas de operaciones trigonométricas
Binario	add	Suma de dos arrays
Binario	subtract	Resta de dos arrays
Binario	multiply	Multiplicación de dos arrays
Binario	divide	División de dos arrays
Binario	maximum / minimum	Retorna el valor máximo/mínimo de cada pareja de elementos
Binario	equal / not_equal	Comparación de igualdad o diferencia

Tipo	Operación	Descripción
Binario	greater / greater_equal / less / less_equal	Comparaciones >, >=, <, <= respectivamente) de cada pareja de elementos

[46]:
```
# ejemplos de operadores
array = np.arange(5)
print(array)
np.sqrt(array)
```

```
[0 1 2 3 4]
```

[46]:
```
array([0.        , 1.        , 1.41421356, 1.73205081, 2.        ])
```

[47]:
```
# ejemplos de operadores
array1 = np.arange(4)
array2 = np.array([0,-1,2,-3])

print(array1)
print(array2)

print(np.greater(array1,array2))
print(np.less(array2, array1))
```

```
[0 1 2 3]
[ 0 -1  2 -3]
[False  True False  True]
[False  True False  True]
```

0.5 Estadística descriptiva

- Importante saber la naturaleza de los datos
- Valores máximos, mínimos, distribución, etc.

Función	Descripción
sum(arr)	Suma de todos los elementos de arr
mean(arr)	Media aritmética de los elementos de arr
std(arr)	Desviación estándar de los elementos de arr
cumsum(arr)	Devuelve un array con la suma acumulada de cada elemento con todos los anteriores
cumprod(arr)	Devuelve un array con el producto acumulado de cada elemento con todos los anteriores
min(arr), max(arr)	Mínimo y máximo de arr
any(arr)	En arrays booleanos, retorna True si algún elemento es True

Función	Descripción
all(arr)	En arrays booleanos, retorna True si todos los elementos son True (o 0 en valores numéricos)
unique(arr)	Devuelve un array con valores únicos
in1d(arr1, arr2)	Devuelve un array booleano indicando si cada elemento de arr1 está en arr2
union1d(arr1, arr2)	Devuelve la unión de ambos arrays
intersect1d(arr1, arr2)	Devuelve la intersección de ambos arrays

[48]:
```
# sum != cumsum
array1 = np.arange(9)
print(array1)
print(array1.sum())
print(array1.cumsum())
```

```
[0 1 2 3 4 5 6 7 8]
36
[ 0  1  3  6 10 15 21 28 36]
```

[49]:
```
# any vs all
array1 = np.arange(-8,2).reshape(2,5)
print(array1)
print(array1.any()) # any para saber si hay elementos True o valores <> 0
print(array1.all()) # all para saber si TODOS los elementos son True o valores␣
 ↪<> 0
```

```
[[-8 -7 -6 -5 -4]
 [-3 -2 -1  0  1]]
True
False
```

[50]:
```
# todas las funciones aceptan parametro 'axis'
# 0 para columnas, 1 para filas, 2 para profundidad...
array1 = np.arange(10).reshape(2,5)
print(array1)
print(array1.all(axis=0))
print(array1.all(axis=1))
print(array1.all())
```

```
[[0 1 2 3 4]
 [5 6 7 8 9]]
[False  True  True  True  True]
[False  True]
False
```

0.6 Álgebra lineal

- numpy.linalg contiene funciones para álgebra lineal

- dot product, multiplicación de matrices, cálculo del determinante, factorizaciones…

Función	Descripción
dot(mat1, mat2)	Devuelve el producto escalar entre dos arrays. Si son matrices 2D, es equivalente a la multiplicación de ambas
matmul(mat1, mat2)	Devuelve el producto entre dos matrices
trace(mat)	Suma de los elementos de la diagonal principal de la matriz
det(mat)	Devuelve el determinante de la matriz
eig(mat)	Computa los autovalores y autovectores de la matriz cuadrada mat
inv(mat)	Devuelve la inversa de la matriz
qr(mat)	Computa la factorización QR de mat
solve(A, b)	Resuelve el sistema lineal de ecuaciones Ax = b, cuando A es una matriz cuadrada
transpose(mat)	Devuelve la transpuesta de la matriz

[51]:
```
A = np.arange(12).reshape(4,3)
B = np.arange(6).reshape(3,2)

print(A)
print(B)

print(np.matmul(A, B))
print(np.dot(A, B))
print(A @ B)
```

```
[[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]]
[[0 1]
 [2 3]
 [4 5]]
[[10 13]
 [28 40]
 [46 67]
 [64 94]]
[[10 13]
 [28 40]
 [46 67]
 [64 94]]
[[10 13]
 [28 40]
 [46 67]
 [64 94]]
```

[52]:
```
import numpy.linalg as lg
import math

# resolver sistema de ecuaciones
# 3x + 2y + 5z = 12
# x- 3y-z = 2
# 2x-y +12z = 32
# Ax = b--> resolver para x

a = np.array([ [3,2,5], [1,-3,-1], [2,-1,12] ])
b = np.array([12,2,32])

print(a)
print(b)
x,y,z = lg.solve(a,b)
print(f'x={x} y={y} z={z}')

print(math.isclose(3*x + 2*y +5*z, 12))
print(math.isclose(x- 3*y- z, 2))
print(math.isclose(2*x-y + 12*z, 32))
```

```
[[ 3  2  5]
 [ 1 -3 -1]
 [ 2 -1 12]]
[12  2 32]
x=0.7543859649122807 y=-1.2280701754385965 z=2.43859649122807
True
True
True
```

[53]:
```
#matriz identidad
I = np.identity(4)
print(I)

A = np.arange(16).reshape(4,4)
print(A)

print(np.matmul(A,I))
```

```
[[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]
[[ 0.  1.  2.  3.]
```

```
[ 4.  5.  6.  7.]
[ 8.  9. 10. 11.]
[12. 13. 14. 15.]]
```

0.7 Filtrado de datos

- Filtrar y modificar datos numéricos con np.where
- Retorna A o B en función de una condición en una array

[54]:
```
prices = np.array([0.99, 14.49, 19.99, 20.99, 0.49])
mask = prices < 1
print(prices[prices < 1]) #slicing condicional

# mask con los elementos menores de 1
mask = np.where(prices < 1, True, False)
print(mask)
print(prices[mask])

prices[mask] = 1.0
print(prices)

print(np.where(prices < 1, 1.0, prices))
```

```
[0.99 0.49]
[ True False False False  True]
[0.99 0.49]
[ 1.   14.49 19.99 20.99  1.  ]
[ 1.   14.49 19.99 20.99  1.  ]
```

[55]:
```
# Se puede usar para sustituir datos fuera de rango
bank_transfers_values = np.array([0.99,-1.49, 19.99, 20.99,-0.49, 12.1]).␣
 ↪reshape(2, 3)
print(bank_transfers_values)
clean_data = np.where(bank_transfers_values > 0, bank_transfers_values, 0)
print(clean_data)
```

```
[[ 0.99 -1.49 19.99]
 [20.99 -0.49 12.1 ]]
[[ 0.99  0.   19.99]
 [20.99  0.   12.1 ]]
```

[56]:
```
# limpiar NaN (sustitución)
data = [10, 12,-143, np.nan, 1,-3] # np.nan--> NotANumber, elemento especial
data_clean = np.where(np.isnan(data), 0, data)
print(data_clean)

print(id(data))
print(id(data_clean))
```

```
[  10.   12. -143.    0.    1.   -3.]
2391704024896
2391703941168
```

[57]:
```
# np.where puede seleccionar elementos
bank_transfers_values = np.array([0.99,-1.49, 19.99, 20.99,-0.49, 12.1]).
 ↪reshape(2, 3)
print(bank_transfers_values)
credits = np.where(bank_transfers_values > 0) # array de índices para los que␣
 ↪la condición es True

print(credits)
print(bank_transfers_values[credits]) # es igual que␣
 ↪bank_transfers_values[bank_transfers_values > 0]
print(bank_transfers_values[bank_transfers_values > 0])
```

```
[[ 0.99 -1.49 19.99]
 [20.99 -0.49 12.1 ]]
(array([0, 0, 1, 1]), array([0, 2, 0, 2]))
[ 0.99 19.99 20.99 12.1 ]
[ 0.99 19.99 20.99 12.1 ]
```

[58]:
```
# valor de array dependiendo del valor en array referencia
rewards_default = np.arange(6)
print(rewards_default)

rewards_upgrade = np.arange(6) * 10
print(rewards_upgrade)

daily_points = np.array([0, 0, 1, 6, 0, 2])
print(daily_points)

final_rewards = np.where(daily_points > 1, rewards_upgrade, rewards_default)
print(final_rewards)
```

```
[0 1 2 3 4 5]
[ 0 10 20 30 40 50]
[0 0 1 6 0 2]
[ 0  1  2 30  4 50]
```

0.8 Números aleatorios

- Python módulo random

[59]:
```
import random

print(random.random()) # número entre 0 y 1
print(random.randint(0,5)) # integral entre dos valores
print(random.uniform(0,5)) # real entre dos valores
```

```
0.8534118836465702
2
2.344610163996577
```

[60]:
```
# elegir un elemento al azar dentro de una colección
names = ['Pikachu','Eevee','Charmander']
random.choice(names)
```

[60]:
```
'Eevee'
```

0.8.1 NumPy.random

- Generar fácilmente listas con valores aleatorios

[61]:
```
print(np.random.rand()) # número entre 0 y 1
print(np.random.randint(0,5)) # integral entre dos valores
print(np.random.randint(5, size=(2, 4))) # parametro 'size' para indicar las␣
 ↪dimensiones)
```

```
0.24782780642357605
0
[[1 1 1 0]
 [4 4 2 2]]
```

[62]:
```
# array de elementos aleatorios (distribucion normal gaussiana, media 0,␣
 ↪desviación 1)
print(np.random.randn(4,2))
```

```
[[ 1.81341842  1.01135592]
 [-1.01342687 -0.56240034]
 [-0.59814063 -0.47337214]
 [ 0.18946842  0.89653896]]
```

[63]:
```
# otras distribuciones
print(np.random.binomial(n=5, p=0.3)) # 'n' intentos, 'p' probabilidad
print(np.random.uniform(low=0, high=10)) # distribución uniforme
print(np.random.poisson(lam=2, size=(2,2))) # 'lam' número de ocurrencias␣
 ↪esperadas, retornar 'size' valores
```

```
1
7.77888691646209
[[3 1]
 [1 5]]
```

0.8.2 Random seed

- Números pseudo-aleatorios
- Ordenadores generan números a partir de ecuaciones
- Basadas en un número inicial (seed)
- Bueno para reproducibilidad de experimentos

[64]:
```
i = 5
for _ in range(5):
    np.random.seed(i) # misma semilla
    print(np.random.rand())
    print(np.random.rand())

print()

for i in range(6):
    np.random.seed(i) # diferentes semillas
    print(np.random.rand())
    print(np.random.rand())
```

```
0.22199317108973948
0.8707323061773764
0.22199317108973948
0.8707323061773764
0.22199317108973948
0.8707323061773764
0.22199317108973948
0.8707323061773764
0.22199317108973948
0.8707323061773764

0.5488135039273248
0.7151893663724195
0.417022004702574
0.7203244934421581
0.43599490214200376
0.025926231827891333
0.5507979025745755
0.7081478226181048
0.9670298390136767
0.5472322491757223
0.22199317108973948
0.8707323061773764
```

0.9 Escritura y lectura de ndarrays a archivos

- Para futuro acceso
- Compartir datos con otros
- Formato *.npy

[65]:
```
# escribir a archivo
import os
ruta = os.path.join("res" ,"o_values_array.npy")
values = np.random.randint(9, size=(3,3))
np.save(ruta, values)
```

```
print(values)
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In[65], line 5
      3 ruta = os.path.join("res" ,"o_values_array.npy")
      4 values = np.random.randint(9, size=(3,3))
----> 5 np.save(ruta, values)
      6 print(values)

File E:\anaconda3\envs\machines2026\Lib\site-packages\numpy\lib\_npyio_impl.py:
 ↪568, in save(file, arr, allow_pickle)
    566     if not file.endswith('.npy'):
    567         file = file + '.npy'
--> 568     file_ctx = open(file,     )
    570 with file_ctx as fid:
    571     arr = np.asanyarray(arr)

FileNotFoundError: [Errno 2] No such file or directory: 'res\\o_values_array.np '
```

[ ]:
```
# leer de archivo
old_values = np.load(ruta)
print(old_values)
```

[ ]:
```
# escribir y cargar múltiples arrays
ruta = os.path.join("res" ,"o_array_group.npz")

values1 = np.random.randint(9,size=(3,3))
values2 = np.random.randint(4,size=(2,2))
values3 = np.random.randint(16,size=(4,4))
np.savez(ruta,values1,values2,values3) # añade la extensión .npz
```

[ ]:
```
# leer los archivos (retorna un diccionario con las arrays)
npzfile = np.load(ruta)

print(type(npzfile.files))
print(npzfile.files)

for key in npzfile.files: # array.files contiene el nombre de las arrays␣
 ↪(arr_0, arr_1...)
    print(npzfile[key])
```

0.9.1 ¿Por qué molestarse con .npy?

- Bases de datos suelen estar en CSV o txt, que requieren lectura con streams (parsing)
- ¿Para qué escribir los datos de nuevo?

[66]:
```
import os
ruta_csv = os.path.join("res" ,"fdata.csv")
with open(ruta_csv,'r') as f:
    data_str = f.read()
data = data_str.split(',')
data_array = np.array(data, dtype = np.int8).reshape(1000,1000)
# print(data_array)

ruta_npy = os.path.join("res" ,"o_fdata.npy")
np.save(ruta_npy, data_array)
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In[66], line 3
      1 import os
      2 ruta_csv = os.path.join("res" ,"fdata.csv")
----> 3 with open(ruta_csv,   ) as f:
      4     data_str = f.read()
      5 data = data_str.split(',')

File E:
 ↪\anaconda3\envs\machines2026\Lib\site-packages\IPython\core\interactiveshell.
 ↪py:343, in _modified_open(file, *args, **kwargs)
    336 if file in {0, 1, 2}:
    337     raise ValueError(
    338         f"IPython won't let you open fd={file} by default "
    339         "as it is likely to crash IPython. If you know what you are␣
 ↪doing, "
    340         "you can use builtins' open."
    341     )
--> 343 return io_open(file, *args, **kwargs)

FileNotFoundError: [Errno 2] No such file or directory: 'res\\fdata.csv'
```

[67]:
```
%%timeit # magic function jupyter
with open(ruta_csv,'r') as f:
    data_str = f.read()
data = data_str.split(',')
data_array = np.array(data, dtype = np.int8).reshape(1000,1000)
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In[67], line 1
----> 1␣
 ↪get_ipython().run_cell_magic(        ,                              ,
```

```
File E:
 ↪\anaconda3\envs\machines2026\Lib\site-packages\IPython\core\interactiveshell.
 ↪py:2565, in InteractiveShell.run_cell_magic(self, magic_name, line, cell)
   2563 with self.builtin_trap:
   2564     args = (magic_arg_s, cell)
-> 2565     result = fn(*args, **kwargs)
   2567 # The code below prevents the output from being displayed
   2568 # when using magics with decorator @output_can_be_silenced
   2569 # when the last Python token in the expression is a ';'.
   2570 if getattr(fn, magic.MAGIC_OUTPUT_CAN_BE_SILENCED, False):

File E:
 ↪\anaconda3\envs\machines2026\Lib\site-packages\IPython\core\magics\execution.
 ↪py:1227, in ExecutionMagics.timeit(self, line, cell, local_ns)
   1225 for index in range(0, 10):
   1226     number = 10 ** index
-> 1227     time_number = timer.timeit(number)
   1228     if time_number >= 0.2:
   1229         break

File E:
 ↪\anaconda3\envs\machines2026\Lib\site-packages\IPython\core\magics\execution.
 ↪py:184, in Timer.timeit(self, number)
    182 gc.disable()
    183 try:
--> 184     timing = self.inner(it, self.timer)
    185 finally:
    186     if gcold:

File <magic-timeit>:1, in inner(_it, _timer)

File E:
 ↪\anaconda3\envs\machines2026\Lib\site-packages\IPython\core\interactiveshell.
 ↪py:343, in _modified_open(file, *args, **kwargs)
    336 if file in {0, 1, 2}:
    337     raise ValueError(
    338         f"IPython won't let you open fd={file} by default "
    339         "as it is likely to crash IPython. If you know what you are␣
 ↪doing, "
    340         "you can use builtins' open."
    341     )
--> 343 return io_open(file, *args, **kwargs)

FileNotFoundError: [Errno 2] No such file or directory: 'res\\fdata.csv'
```

[68]:
```
%%timeit
data = np.load(ruta_npy)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[68], line 1
----> 1 get_ipython().run_cell_magic(        ,    ,                          \n

File E:
 ↪\anaconda3\envs\machines2026\Lib\site-packages\IPython\core\interactiveshell.
 ↪py:2565, in InteractiveShell.run_cell_magic(self, magic_name, line, cell)
   2563 with self.builtin_trap:
   2564     args = (magic_arg_s, cell)
-> 2565     result = fn(*args, **kwargs)
   2567 # The code below prevents the output from being displayed
   2568 # when using magics with decorator @output_can_be_silenced
   2569 # when the last Python token in the expression is a ';'.
   2570 if getattr(fn, magic.MAGIC_OUTPUT_CAN_BE_SILENCED, False):

File E:
 ↪\anaconda3\envs\machines2026\Lib\site-packages\IPython\core\magics\execution.
 ↪py:1227, in ExecutionMagics.timeit(self, line, cell, local_ns)
   1225 for index in range(0, 10):
   1226     number = 10 ** index
-> 1227     time_number = timer.timeit(number)
   1228     if time_number >= 0.2:
   1229         break

File E:
 ↪\anaconda3\envs\machines2026\Lib\site-packages\IPython\core\magics\execution.
 ↪py:184, in Timer.timeit(self, number)
    182 gc.disable()
    183 try:
--> 184     timing = self.inner(it, self.timer)
    185 finally:
    186     if gcold:

File <magic-timeit>:1, in inner(_it, _timer)

NameError: name 'ruta_npy' is not defined
```

[69]:
```
%%timeit # magic function jupyter
result = np.fromfile(ruta_csv, dtype=np.int8, sep=",")
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In[69], line 1
----> 1␣
 ↪get_ipython().run_cell_magic(        ,                          ,
```

```
File E:
 ↪\anaconda3\envs\machines2026\Lib\site-packages\IPython\core\interactiveshell.
 ↪py:2565, in InteractiveShell.run_cell_magic(self, magic_name, line, cell)
   2563 with self.builtin_trap:
   2564     args = (magic_arg_s, cell)
-> 2565     result = fn(*args, **kwargs)
   2567 # The code below prevents the output from being displayed
   2568 # when using magics with decorator @output_can_be_silenced
   2569 # when the last Python token in the expression is a ';'.
   2570 if getattr(fn, magic.MAGIC_OUTPUT_CAN_BE_SILENCED, False):

File E:
 ↪\anaconda3\envs\machines2026\Lib\site-packages\IPython\core\magics\execution.
 ↪py:1227, in ExecutionMagics.timeit(self, line, cell, local_ns)
   1225 for index in range(0, 10):
   1226     number = 10 ** index
-> 1227     time_number = timer.timeit(number)
   1228     if time_number >= 0.2:
   1229         break

File E:
 ↪\anaconda3\envs\machines2026\Lib\site-packages\IPython\core\magics\execution.
 ↪py:184, in Timer.timeit(self, number)
    182 gc.disable()
    183 try:
--> 184     timing = self.inner(it, self.timer)
    185 finally:
    186     if gcold:

File <magic-timeit>:1, in inner(_it, _timer)

FileNotFoundError: [Errno 2] No such file or directory: 'res\\fdata.csv'
```

0.9.2 velocidad + simplicidad vs memoria

0.10 Ejercicios

- Ejercicios para practicar NumPy: https://github.com/rougier/numpy-100/blob/master/100_Numpy_exercises.ipynb

[70]:
```
''' 2. Print the numpy version and the configuration '''

import numpy as np
print(np.__version__)
np.show_config()
```

```
2.4.0
Build Dependencies:
```

```
  blas:
    detection method: pkgconfig
    found: true
    include directory: C:/Users/runneradmin/AppData/Local/Temp/cibw-
run-5tdgda_u/cp311-win_amd64/build/venv/Lib/site-
packages/scipy_openblas64/include
    lib directory: C:/Users/runneradmin/AppData/Local/Temp/cibw-
run-5tdgda_u/cp311-win_amd64/build/venv/Lib/site-packages/scipy_openblas64/lib
    name: scipy-openblas
    openblas configuration: OpenBLAS 0.3.30  USE64BITINT DYNAMIC_ARCH
NO_AFFINITY
      Haswell MAX_THREADS=24
    pc file directory: D:/a/numpy-release/numpy-release/.openblas
    version: 0.3.30
  lapack:
    detection method: pkgconfig
    found: true
    include directory: C:/Users/runneradmin/AppData/Local/Temp/cibw-
run-5tdgda_u/cp311-win_amd64/build/venv/Lib/site-
packages/scipy_openblas64/include
    lib directory: C:/Users/runneradmin/AppData/Local/Temp/cibw-
run-5tdgda_u/cp311-win_amd64/build/venv/Lib/site-packages/scipy_openblas64/lib
    name: scipy-openblas
    openblas configuration: OpenBLAS 0.3.30  USE64BITINT DYNAMIC_ARCH
NO_AFFINITY
      Haswell MAX_THREADS=24
    pc file directory: D:/a/numpy-release/numpy-release/.openblas
    version: 0.3.30
Compilers:
  c:
    commands: cl
    linker: link
    name: msvc
    version: 19.44.35222
  c++:
    commands: cl
    linker: link
    name: msvc
    version: 19.44.35222
  cython:
    commands: cython
    linker: cython
    name: cython
    version: 3.2.3
Machine Information:
  build:
    cpu: x86_64
    endian: little
```

```
    family: x86_64
    system: windows
  host:
    cpu: x86_64
    endian: little
    family: x86_64
    system: windows
Python Information:
  path: C:\Users\runneradmin\AppData\Local\Temp\build-
env-20ylnh9g\Scripts\python.exe
  version: '3.11'
SIMD Extensions:
  baseline:
  - X86_V2
  found:
  - X86_V3
```

[71]:
```
''' 4. How to find the memory size of any array '''

Z = np.zeros((10,10))
print("%d bytes" % (Z.size * Z.itemsize))
```

```
800 bytes
```

[72]:
```
''' 6. Create a null vector of size 10 but the fifth value which is 1 '''

Z = np.zeros(10)
Z[4] = 1
print(Z)
```

```
[0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
```

[73]:
```
''' 8. Reverse a vector (first element becomes last) '''

Z = np.arange(10)
Z = Z[::-1]
print(Z)
```

```
[9 8 7 6 5 4 3 2 1 0]
```

[74]:
```
''' 10. Find indices of non-zero elements from [1,2,0,0,4,0] '''

nz = np.nonzero([1,2,0,0,4,0])
print(nz)
```

```
(array([0, 1, 4]),)
```

[75]:
```
''' 12. Create a 3x3x3 array with random values '''

Z = np.random.random((3,3,3))
print(Z)
```

```
[[[0.99082093 0.80828168 0.64945888]
  [0.81947279 0.24254041 0.76437261]
  [0.11090076 0.20415475 0.11909536]]

 [[0.87790307 0.52367529 0.492136  ]
  [0.7318711  0.01458075 0.09336303]
  [0.82655425 0.83349274 0.89241102]]

 [[0.9580135  0.56133712 0.0905487 ]
  [0.99598891 0.47740172 0.68491561]
  [0.84337526 0.61612483 0.56318751]]]
```

[76]:
```
''' 14. Create a random vector of size 30 and find the mean value '''

Z = np.random.random(30)
m = Z.mean()
print(m)
```

```
0.46344434547642693
```

[77]:
```
''' 16. How to add a border (filled with 0's) around an existing array? '''

ones = np.ones((3,3))
Z = np.pad(ones, pad_width=1, mode='constant', constant_values=0)
print(Z)
```

```
[[0. 0. 0. 0. 0.]
 [0. 1. 1. 1. 0.]
 [0. 1. 1. 1. 0.]
 [0. 1. 1. 1. 0.]
 [0. 0. 0. 0. 0.]]
```

[78]:
```
''' 18. Create a 5x5 matrix with values 1,2,3,4 just below the diagonal '''

Z = np.diag(1+np.arange(4),k=-1)
print(Z)
```

```
[[0 0 0 0 0]
 [1 0 0 0 0]
 [0 2 0 0 0]
 [0 0 3 0 0]
 [0 0 0 4 0]]
```

[79]:
```
''' 20. Consider a (6,7,8) shape array, what is the index (x,y,z) of the 100th␣
↪element? '''

print(np.unravel_index(99,(6,7,8)))
```

```
(np.int64(1), np.int64(5), np.int64(3))
```

0.11 Referencias

GitHub: https://github.com/reromash1/machines2026.git

[ ]:
```

```