

Repaso de NumPy

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

Array

- Es el tipo de dato fundamental en NumPy
- Representa una “matriz” con un número arbitrario de dimensiones (rango)

```
▶ a = np.array([1, 2, 3]) # Array de rango 1 (vector)  
print(a)  
print(a.shape)
```

```
[1 2 3]  
(3,)
```

1	2	3
---	---	---

```
▶ b = np.array([[1,2,3], [4,5,6]]) # Array de rango 2 (matriz)  
print(b)  
print(b.shape)
```

```
[[1 2 3]  
 [4 5 6]]  
(2, 3)
```

1	2	3
4	5	6

Array

```
▶ x = np.array(2)    # Array de rango 0 (escalar)  
print(x)  
print(x.shape)
```

```
2  
()
```

2

```
▶ z = np.array([[[1,2,3], [4,5,6]], [[1,2,3], [4,5,6]]])    # Array de rango 3  
print(z)  
print(z.shape)
```

```
[[[1 2 3]  
  [4 5 6]]  
  
 [[1 2 3]  
  [4 5 6]]]  
(2, 2, 3)
```

1	2	3		
4	5	6		
			2	3
			4	6

Rango y forma

- El rango (dimensión, orden, grado) es el número de índices que necesitamos para seleccionar cada elemento del array
- La forma tiene en cuenta el número de elementos a lo largo de cada dimensión

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15



Rango = 2

Forma = 3 x 5

Funciones para crear arrays

<https://numpy.org/doc/stable/user/basics.creation.html#arrays-creation>



```
a = np.zeros((2,2))
```

```
b = np.ones((1,2))
```

```
c = np.full((2,2), 7)
```

```
d = np.eye(2)
```

```
e = np.random.random((2,2))
```

7	7
7	7

0	0
0	0

1	1
---	---

0.12	0.47
0.53	0.91

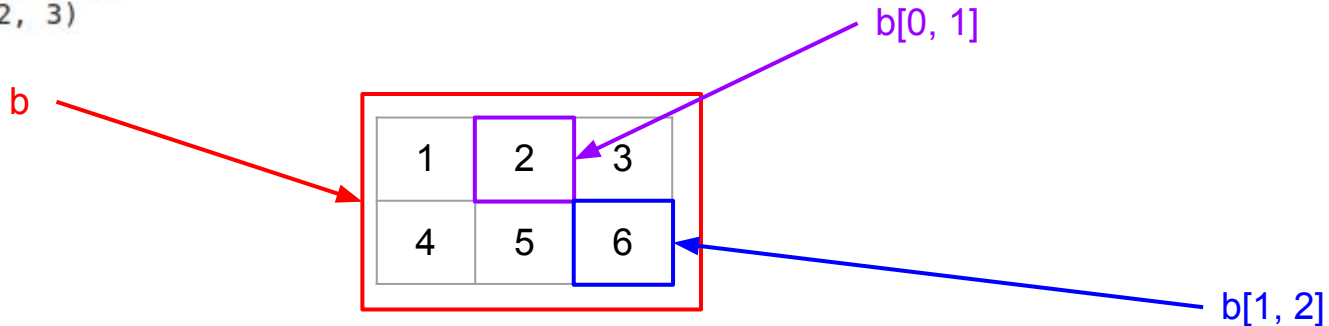
1	0
0	1

Indexación

- Para indexar un array es necesaria una tupla con tantos elementos como el rango del array

```
▶ b = np.array([[1,2,3], [4,5,6]])    # Array de rango 2 (matriz)  
print(b)  
print(b.shape)
```

```
[[1 2 3]  
 [4 5 6]]  
(2, 3)
```



Slicing

- Extraer una “rodaja” del array

```
▶ a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])  
  b = a[:2, 1:3]  
  print(b)
```

```
[[2 3]  
 [6 7]]
```

índice inicial, incluido

índice final, excluido

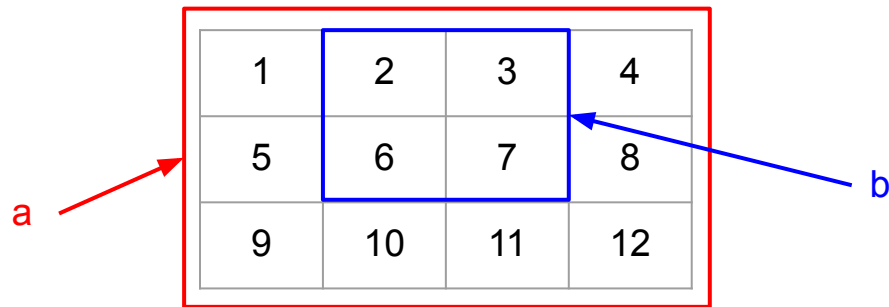
Slicing

- El array y el slice comparten memoria

```
▶ a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])  
  b = a[:2, 1:3]  
  print(b)
```

```
[[2 3]  
 [6 7]]
```

```
▶ print(a[0, 1])  
  b[0, 0] = 10  
  print(a[0, 1])
```



Indexación con enteros

```
▶ a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])  
  
x = a[1, :]  
print(x)  
print(x.shape)  
  
y = a[1:2, :]  
print(y)  
print(y.shape)  
  
[5 6 7 8]  
(4,)  
[[5 6 7 8]]  
(1, 4)
```

- Ojo a la forma de los arrays

Indexación con enteros



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



```
[[ 1  2  3  4]  
 [ 5  6  7  8]  
 [ 9 10 11 12]]  
[[ 3  1  4]  
 [ 7  5  8]  
 [11  9 12]]
```

Indexación booleana



```
a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])  
print(a)  
  
ix = (a > 2)  
print(ix)  
  
b = a[ix]  
print(b)
```

```
[[ 1  2  3  4]  
 [ 5  6  7  8]  
 [ 9 10 11 12]]  
[[False False  True  True]  
 [ True  True  True  True]  
 [ True  True  True  True]]  
[ 3  4  5  6  7  8  9 10 11 12]
```

Indexación booleana

- ¿Qué imprime el siguiente código?

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

Indexación booleana y con enteros

- Un fragmento de un array indexado con enteros o con booleanos, ¿comparte memoria con el array original?

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

```
[[ 1  2  3  4]  
 [ 5  6  7  8]  
 [ 9 10 11 12]]  
[[10  3]  
 [ 6  7]]  
[[ 1 10  3  4]  
 [ 5  6  7  8]  
 [ 9 10 11 12]]
```

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

```
▶ a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])  
print(a)  
  
b = a[a>5]  
b[0] = -9  
print(b)  
  
print(a)
```

Más detalles sobre indexación en Numpy

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

Matemática con arrays

- Todos los operadores habituales actúan “element-wise”

```
▶ x = np.array([[1,2],[3,4]])  
  y = np.array([[5,6],[7,8]])  
  
  print(x + y)  
  print(x - y)  
  print(x * y)
```

- El operador `*` representa el producto elemento a elemento
- El operador `@` (o la función `dot`) representa el producto de matrices

Funciones sobre arrays

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

Manipulación de arrays

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

Broadcasting

1	2	3
4	5	6

(2×3)

*

2

(1×1)

Broadcasting

$$\begin{array}{|c|c|c|} \hline 1 & 2 & 3 \\ \hline 4 & 5 & 6 \\ \hline \end{array} \quad (2 \times 3) \quad * \quad \begin{array}{|c|} \hline 2 \\ \hline \end{array} \quad (1 \times 1) = \begin{array}{|c|c|c|} \hline 1 & 2 & 3 \\ \hline 4 & 5 & 6 \\ \hline \end{array} \quad * \quad \begin{array}{|c|c|c|} \hline 2 & 2 & 2 \\ \hline 2 & 2 & 2 \\ \hline \end{array}$$

The diagram illustrates the concept of broadcasting in matrix operations. It shows a (2×3) matrix multiplied by a (1×1) matrix, which is then equated to the same (2×3) matrix multiplied by a (2×3) matrix where every element is 2. The second (2×3) matrix is drawn with red borders to highlight the broadcasted values.

Broadcasting

$$\begin{array}{|c|c|c|} \hline 1 & 2 & 3 \\ \hline 4 & 5 & 6 \\ \hline \end{array} \quad (2 \times 3) \quad * \quad \begin{array}{|c|} \hline 2 \\ \hline \end{array} \quad (1 \times 1) = \begin{array}{|c|c|c|} \hline 1 & 2 & 3 \\ \hline 4 & 5 & 6 \\ \hline \end{array} \quad * \quad \begin{array}{|c|c|c|} \hline 2 & 2 & 2 \\ \hline 2 & 2 & 2 \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline 2 & 4 & 6 \\ \hline 8 & 10 & 12 \\ \hline \end{array}$$

Broadcasting

1	2	3
4	5	6

(2x3)

*

2	2	2
3	3	3

(2x3)

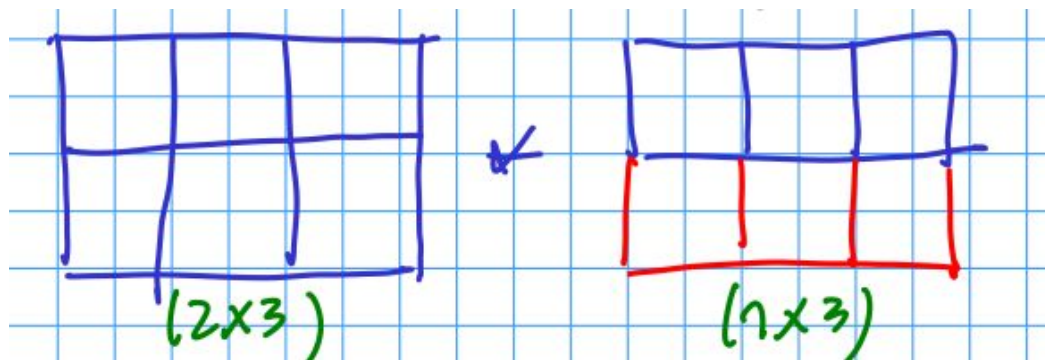
=

2	4	6
12	15	18

Broadcasting

A hand-drawn diagram on a blue grid background illustrating matrix multiplication. On the left is a 2×3 matrix, represented by a rectangle divided into six cells, with the label (2×3) written in green below it. To its right is a multiplication symbol $*$. Next is a 2×2 matrix, represented by a rectangle divided into four cells, with the label (2×2) written in green below it. To the right of this is an equals sign $=$. Finally, on the far right, is a circle containing a question mark $?$, indicating the result of the operation.

Broadcasting



Broadcasting

Diagram illustrating matrix broadcasting:

Matrix 1 (Dimensions: 2×1):

1
2

Matrix 2 (Dimensions: 1×3):

3	4	5
---	---	---

The operation is represented by an asterisk ($*$) between the two matrices.

Broadcasting

Diagram illustrating array broadcasting:

Array 1 (2x1):

1
2

(2x1)

Array 2 (1x3):

3	4	5
---	---	---

(1x3)

Result of first multiplication (2x3):

1	1	1
2	2	2

Array 3 (1x3):

3	4	5
---	---	---

(1x3)

Final Result (2x3):

3	4	5
9	10	10

Broadcasting

Para poder hacer broadcasting con 2 arrays A y B, estos deben ser compatibles:

- El rango de ambos es el mismo
- El tamaño en cada dimensión:
 - O bien es igual para los dos arrays
 - O bien es 1 para uno de ellos

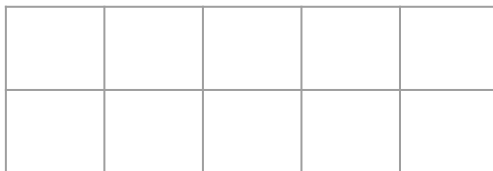


COMPATIBLES

Broadcasting

Para poder hacer broadcasting con 2 arrays A y B, estos deben ser compatibles:

- El rango de ambos es el mismo
- El tamaño en cada dimensión:
 - O bien es igual para los dos arrays
 - O bien es 1 para uno de ellos



INCOMPATIBLES

Broadcasting

Para poder hacer broadcasting con 2 arrays A y B, estos deben ser compatibles:

- El rango de ambos es el mismo
- El tamaño en cada dimensión:
 - O bien es igual para los dos arrays
 - O bien es 1 para uno de ellos

A.shape = (4, 2, 1)

B.shape = (1, 1, 2)

???