Introducción a Numpy y Álgebra Lineal

Matemática y Tecnología II
Maestría en Matemática

Prof. Robert Munoz

Escuela de Matemática,
Universidad Autónoma de Santo Domingo (UASD)

2025

Tabla de Contenido

Librerías de Python

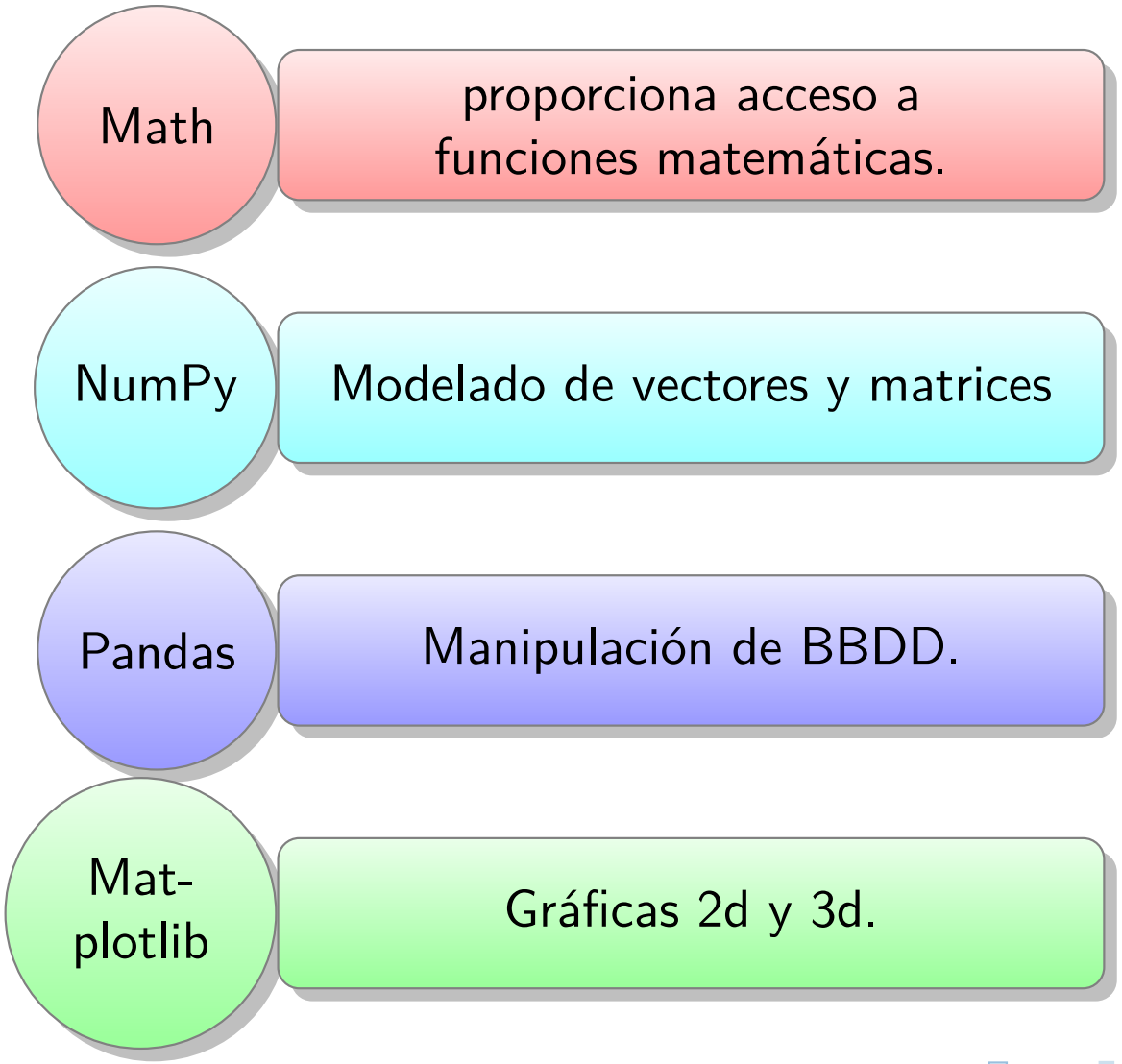
Instalar **Python** por sí sólo no representa ninguna ventaja, como cualquier software libre su fortaleza radica en la instalación e importación librerías desarrolladas por la comunidad.

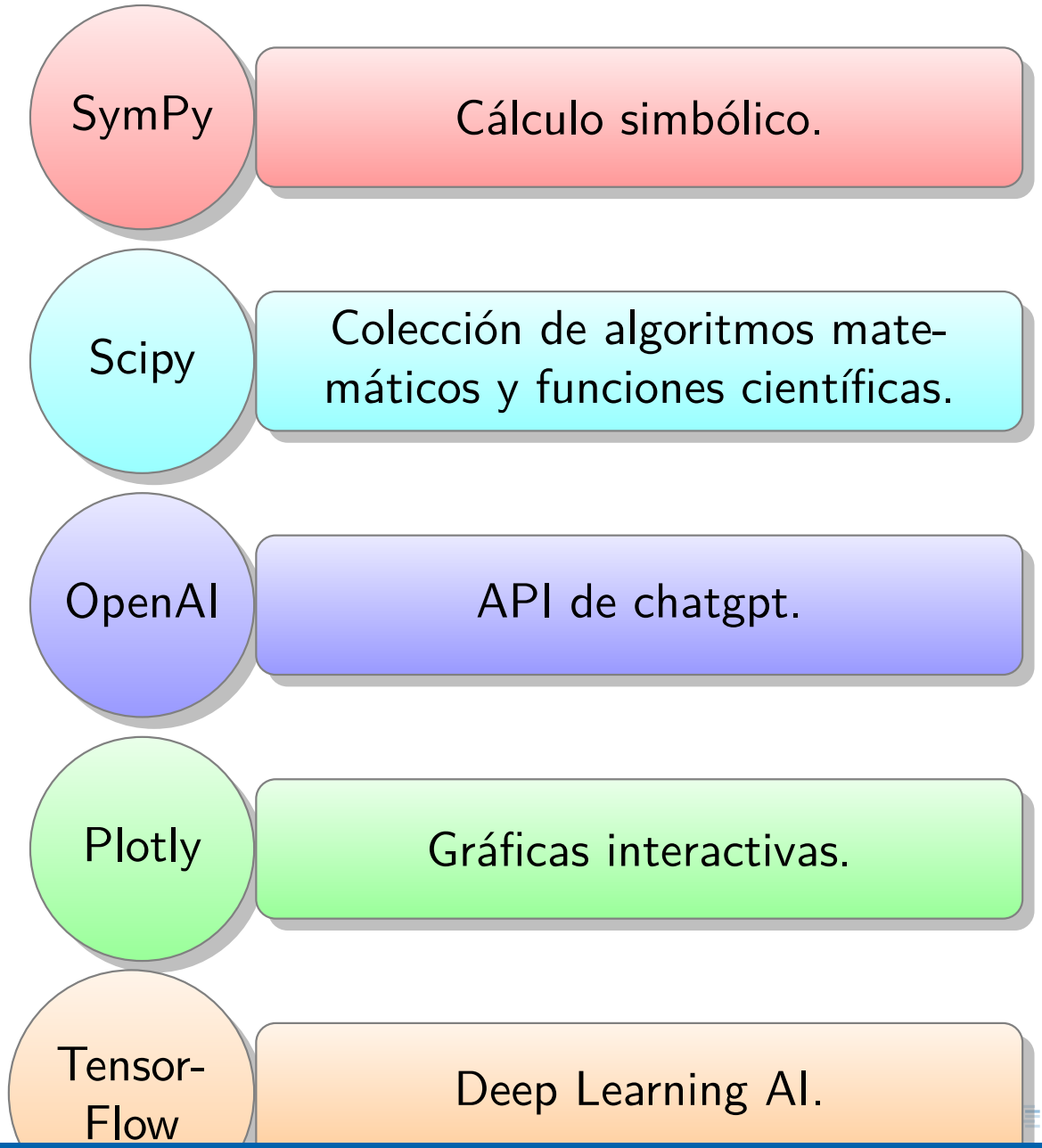
Librerías de Python

Instalar **Python** por sí sólo no representa ninguna ventaja, como cualquier software libre su fortaleza radica en la instalación e importación librerías desarrolladas por la comunidad.

Algunas librerías vienen instaladas y otras amerita una instalación. Las librerías o paquetes solo se instalan una vez, ya sea en nuestra pc o máquina virtual o cloud.

Algunas Librerías




SymPy
Cálculo simbólico.
Scipy
Colección de algoritmos matemáticos y funciones científicas.
OpenAI
API de chatgpt.
Plotly
Gráficas interactivas.
Tensor-Flow
Deep Learning AI.

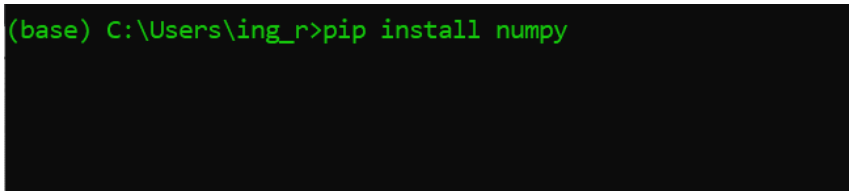
Instalación de una librería

En COLAB (Notebook en Google Drive) o en Jupyter (Notebook en nuestra pc): Por ejemplo si desea instalar numpy, escribir dentro de una celda y ejecutar el código:

```
1 !pip install numpy
```

Otras formas de instalar

Instalación en terminal de Anaconda:

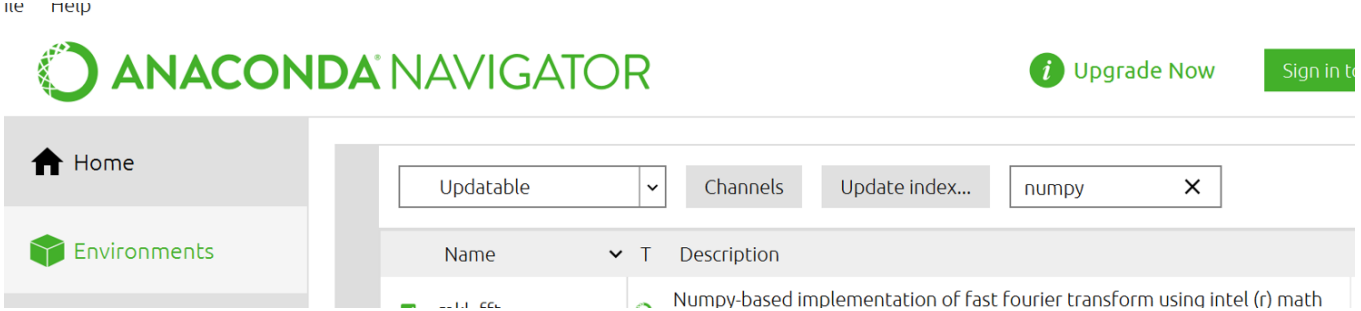


Otras formas de instalar

Instalación en terminal de Anaconda:

```
(base) C:\Users\ing_r>pip install numpy
```

Instalación usando Anaconda Navigator:



Importación de Librerías

Luego que una librería es instalada, para importarla, se pueden utilizar los siguientes comandos: *Por ejemplo, si se desea importar Numpy simplemente escribir:*

Importación de Librerías

Luego que una librería es instalada, para importarla, se pueden utilizar los siguientes comandos: *Por ejemplo, si se desea importar Numpy simplemente escribir:*

- `import numpy`

Importación de Librerías

Luego que una librería es instalada, para importarla, se pueden utilizar los siguientes comandos: *Por ejemplo, si se desea importar Numpy simplemente escribir:*

- `import numpy`
- `import numpy as np`

Importación de Librerías

Luego que una librería es instalada, para importarla, se pueden utilizar los siguientes comandos: *Por ejemplo, si se desea importar Numpy simplemente escribir:*

- `import numpy`
- `import numpy as np`
  Ésta es la más recomendada para evitar ambigüedad con otra librería.
  np es una abreviatura para facilitar las llamadas dentro del código.

Importación de Librerías

Luego que una librería es instalada, para importarla, se pueden utilizar los siguientes comandos: *Por ejemplo, si se desea importar Numpy simplemente escribir:*

- `import numpy`
- `import numpy as np`
  Ésta es la más recomendada para evitar ambigüedad con otra librería.
  np es una abreviatura para facilitar las llamadas dentro del código.
- `from numpy import *`
  Se importan todas las funciones, sin necesidad de hacer referencia a librería.

Importación de Librerías

Luego que una librería es instalada, para importarla, se pueden utilizar los siguientes comandos: *Por ejemplo, si se desea importar Numpy simplemente escribir:*

- `import numpy`
- `import numpy as np`
  Ésta es la más recomendada para evitar ambigüedad con otra librería.
  np es una abreviatura para facilitar las llamadas dentro del código.
- `from numpy import *`
  Se importan todas las funciones, sin necesidad de hacer referencia a librería.
- `from numpy import 'specific function'`

Importación de Librerías

Luego que una librería es instalada, para importarla, se pueden utilizar los siguientes comandos: *Por ejemplo, si se desea importar Numpy simplemente escribir:*

- `import numpy`
- `import numpy as np`
  Ésta es la más recomendada para evitar ambigüedad con otra librería.
  np es una abreviatura para facilitar las llamadas dentro del código.
- `from numpy import *`
  Se importan todas las funciones, sin necesidad de hacer referencia a librería.
- `from numpy import 'specific function'`

Algunos ejemplos con Math y numpy.

Numpy

NumPy es la librería fundamental para la computación científica en Python y proporciona un objeto de matriz multidimensional, varios objetos derivados (como matrices y matrices enmascaradas).

Numpy

NumPy es la librería fundamental para la computación científica en Python y proporciona un objeto de matriz multidimensional, varios objetos derivados (como matrices y matrices enmascaradas).

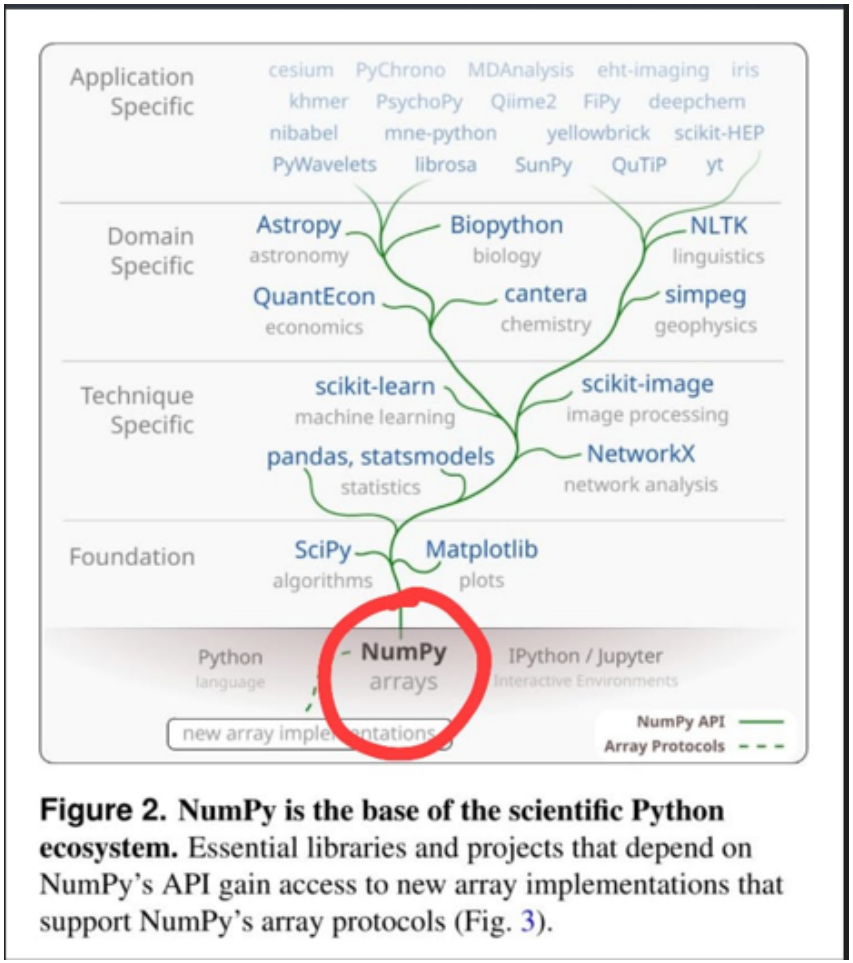


Figure 2. NumPy is the base of the scientific Python ecosystem. Essential libraries and projects that depend on NumPy's API gain access to new array implementations that support NumPy's array protocols (Fig. 3).

Características de Numpy

- Matrices multidimensionales.
- Operaciones de matrices integradas.
- Arreglo simplificado pero poderoso.
- Integración de otros lenguajes(Fortran, C, C++).
  Eso permite que sea más rápido que los objetos propios de python.

Ejemplo Prueba Lista vs Array

```
1  import numpy as np
2  import time
3  n = 1000000
4  numpy_array = np.arange(n)
5  python_list = list(range(n))
6  start_time = time.time()
7  numpy_sum = np.sum(numpy_array)
8  numpy_time = time.time() - start_time
9  start_time = time.time()
10 python_sum = sum(python_list)
11 python_time = time.time() - start_time
12 print(f"tiempo:{numpy_time}")
13 print(f"tiempo:{python_time}")
14 print(f"rapidez: {python_time/numpy_time}")
15 >tiempo: 0.0011866092681884766 segundos
16 >tiempo: 0.01525568962097168 segundos
17 >relacion rapidez : 12.856540084388186
```

Creación de un array

Un array 1D es una lista unidimensional de elementos. En NumPy, se puede crear un arreglo 1D utilizando la función numpy.array() o np.array(), pasando una lista como argumento. Por ejemplo:

```
array_1d = np.array(
[1, 2, 3]
)
```

Creación de un array

Un array 1D es una lista unidimensional de elementos. En NumPy, se puede crear un arreglo 1D utilizando la función numpy.array() o np.array(), pasando una lista como argumento. Por ejemplo:

```
array_1d = np.array(
[1, 2, 3]
)
```

Un array 2D es una matriz con filas y columnas. En NumPy, se puede crear un array 2D utilizando la función numpy.array(), pasando una lista de listas como argumento. Por ejemplo:

```
array_2d = np.array(
[[1, 2, 3],
[4, 5, 6],
[7, 8, 9]])
```

Comandos ndim, size y shape

son fundamentales para entender y manipular las dimensiones y el tamaño de los arrays.

- **ndim**: Dimensión del array, 1D, 2D, 3D, etc
- **size**: Para obtener el número total de elementos en el arreglo.
- **shape**: Para obtener una tupla con el orden (mxn) del arreglo.

Ejemplo

```
# Crear un array 2D con NumPy
array_2d = np.array(
[[1, 2, 3],
[4, 5, 6],
[7, 8, 9]])
#dimension
array_2d.ndim

#tamano, cantidad de elementos
array_2d.size

#forma del array (numero de filas y columnas)
array_2d.shape
```

Indexing en Numpy

Podemos acceder a los elementos del array utilizando indexing. El indexing en un arreglo 2D de NumPy se hace especificando la fila y la columna del elemento deseado. Por ejemplo:

Indexing en Numpy

Podemos acceder a los elementos del array utilizando indexing. El indexing en un arreglo 2D de NumPy se hace especificando la fila y la columna del elemento deseado. Por ejemplo:

```
# Crear un array 2D con NumPy
array_2d = np.array(
[[1, 2, 3],
[4, 5, 6],
[7, 8, 9]])

# Acceder a elementos del array 2D
print("Elemento fila 0, columna 1:", array_2d[0, 1])
print("Primera fila:", array_2d[0, :])
print("Primera columna:", array_2d[:, 0])
```

Slicing en un array o submatriz

an_array =

	0	1	2	3
0	11	12	13	14
1	21	22	23	24
2	31	32	33	34

a_slice = an_array[:2, 1:3]

Modificar elementos

También podemos modificar los elementos de un array. Por ejemplo:

```
array_2d = np.arange(1,10).reshape(3,3)

# Modificar un elemento especifico
#(fila 1, columna 2)
array_2d[1, 2] = 10
print("Array_modificado:\n", array_2d)

# Modificar una fila completa
array_2d[0, :] = [10, 20, 30]
```

Boolean Arrays

Un arreglo booleano en NumPy es simplemente un arreglo que contiene valores booleanos (True o False).

Boolean Arrays

Un arreglo booleano en NumPy es simplemente un arreglo que contiene valores booleanos (True o False).

```
1  array_2d =  np.arange(1,10).reshape(3,3)
2
3  # Mascara booleana para elementos > 5
4  array_booleano= array_2d > 5
5  print("Array boolean", array_booleano)
6
7  # Usar la mascara booleana
8  #para seleccionar elementos mayores que 5
9  elemento_filtrados = array_2d[array_booleano]
10 print("elemento_filtrados:", elemento_filtrados)
```

Arrays predefinidos

NumPy ofrece muchas formas de crear arreglos predefinidos con valores específicos para realizar operaciones matemáticas y estadísticas.

Arrays predefinidos

NumPy ofrece muchas formas de crear arreglos predefinidos con valores específicos para realizar operaciones matemáticas y estadísticas.

- `np.zeros(shape, dtype=float)`: Array de Ceros.
- `np.ones(shape, dtype=float)`: Array de Unos.
- `np.full(shape, fill_value, dtype=None)`: Array full.
- `np.arange(start, stop, step, dtype=None)`
- `np.linspace(start, stop, num=50)`
- `np.random.randint(low, high, size, dtype=int)`
- `np.random.randn(d0, d1, ..., dn)`
- `np.diag(v)`: Matriz diagonal.
- `np.eye(N, M, k, dtype=float)`: Matriz identidad.

Append

La función append agrega un elemento o varios elementos al final de un arreglo existente.

Append

La función append agrega un elemento o varios elementos al final de un arreglo existente.

1D array

```
1 array_1d = np.array([1, 2, 3])
2 print("Original:", array_1d)
3 # Agregar elementos al array 1D
4 array_1d_new = np.append(array_1d, [5, 6, 7])
5 print("Elementos_anexos:", array_1d_new)
6 >Original: [1 2 3]
7 >Elementos_anexos: [1 2 3 5 6 7]
```

Concatenate

Se combinan dos arrays existentes.

2D array

```
1  # Crear dos arrays 2D
2  array_2d_1 = np.arange(1, 7).reshape(2, 3)
3  array_2d_2 = np.arange(7, 13).reshape(2, 3)
4
5  # Concatenar los arrays en filas (axis=0)
6  array_2d_rows = np.concatenate(
7  (array_2d_1, array_2d_2), axis=0)
8  print("Concatenados_x_filas :\n", array_2d_rows)
9
10 # Concatenar los arrays en columnas (axis=1)
11 array_2d_cols = np.concatenate(
12 (array_2d_1, array_2d_2), axis=1)
13 print("Concatenados_x_columnas:\n", array_2d_cols)
```

Operaciones

```python
import numpy as np

# Creación de matrices
matriz_a = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
matriz_b = np.array([[9, 8, 7], [6, 5, 4], [3, 2, 1]])

# Suma de matrices
suma_matriz = matriz_a + matriz_b

# Resta de matrices
resta_matriz = matriz_a - matriz_b

# Multiplicación de matrices
producto_matriz = np.dot(matriz_a, matriz_b)

# Transposición de matriz
transpuesta_matriz = matriz_a.T

# Determinante de matriz
determinante_matriz = np.linalg.det(matriz_a)

# Inversa de matriz
inversa_matriz = np.linalg.inv(matriz_a)
```

Broadcasting

El broadcasting permite que NumPy realice operaciones en arrays de diferentes formas al "estirar" uno de ellos para que coincida con la forma del otro.

Broadcasting

El broadcasting permite que NumPy realice operaciones en arrays de diferentes formas al "estirar" uno de ellos para que coincida con la forma del otro.

	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12

A

+

0	1	2	3
0	1	0	2

B

Tratar en el notebook...

Solución Sistema de Ecuaciones Lineales

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\
a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\
&\vdots \\
a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n
\end{aligned}
$$

Se puede simplificar a la forma $Ax = b$, donde:

$$
A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \quad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}
$$

Ejemplo SEL

```
1  # array de coeficientes
2  A = np.array([[2, 3], [4, 1]])
3
4  #  array de terminos independientes
5  b = np.array([5, 11])
6
7  # Resolver el sistema de ecuaciones
8  x = np.linalg.solve(A, b)
9
10 print("solucion:", x)
11 >solucion: [ 2.8 -0.2]
```