La Librería Numpy

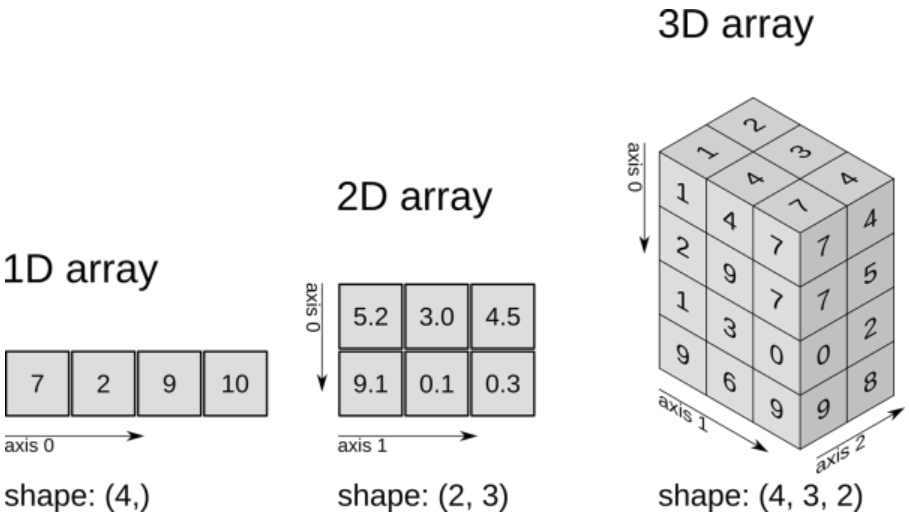
NumPy es una librería de Python especializada en el cálculo numérico y el análisis de datos, especialmente para un gran volumen de datos.

Incorpora una nueva clase de objetos llamados **arrays** que permite representar colecciones de datos de un mismo tipo en varias dimensiones, y funciones muy eficientes para su manipulación.

1. La clase de objetos `array`

Un array es una estructura de datos de un mismo tipo organizada en forma de tabla o cuadrícula de distintas dimensiones.

Las dimensiones de un array también se conocen como ejes.



2. Creación de arrays

Para crear un array se utiliza la siguiente función de **NumPy**

`np.array(lista)`: Crea un array a partir de la lista o tupla **`lista`** y devuelve una referencia a él. El número de dimensiones del array dependerá de las listas o tuplas anidadas en **`lista`**:

- Para una lista de valores se crea un array de una dimensión, también conocido como **vector**.
- Para una lista de listas de valores se crea un array de dos dimensiones, también conocido como **matriz**.
- Para una lista de listas de listas de valores se crea un array de tres dimensiones, también conocido como **cubo**.
- Y así sucesivamente. No hay límite en el número de dimensiones del array más allá de la memoria disponible en el sistema.

Los elementos de la lista o tupla deben ser del mismo tipo.

Ejercicio de Inducción: *Pruebe las siguientes líneas de código y verifique los resultados presentados:*

```
>>> # Array de una dimensión
>>> a1 = np.array([1, 2, 3])
>>> print(a1)
[1 2 3]
>>> # Array de dos dimensiones
>>> a2 = np.array([[1, 2, 3], [4, 5, 6]])
>>> print(a2)
[[1 2 3]
 [4 5 6]]
>>> # Array de tres dimensiones
>>> a3 = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])
>>> print(a3)
[[[ 1  2  3]
  [ 4  5  6]]

 [[ 7  8  9]
  [10 11 12]]]
```

Otras funciones útiles que permiten generar arrays son:

- **`np.empty(dimensiones)`**: Crea y devuelve una referencia a un array vacío con las dimensiones especificadas en la tupla **`dimensiones`**.
- **`np.zeros(dimensiones)`**: Crea y devuelve una referencia a un array con las dimensiones especificadas en la tupla **`dimensiones`** cuyos elementos son todos ceros.
- **`np.ones(dimensiones)`**: Crea y devuelve una referencia a un array con las dimensiones especificadas en la tupla **`dimensiones`** cuyos elementos son todos unos.
- **`np.full(dimensiones, valor)`**: Crea y devuelve una referencia a un array con las dimensiones especificadas en la tupla **`dimensiones`** cuyos elementos son todos valor.
- **`np.identity(n)`**: Crea y devuelve una referencia a la matriz identidad de dimensión **`n`**.
- **`np.arange(inicio, fin, salto)`**: Crea y devuelve una referencia a un array de una dimensión cuyos elementos son la secuencia desde **`inicio`** hasta **`fin`** tomando valores cada **`salto`**.
- **`np.linspace(inicio, fin, n)`**: Crea y devuelve una referencia a un array de una dimensión cuyos elementos son la secuencia de **`n`** valores equidistantes desde **`inicio`** hasta **`fin`**.
- **`np.random.random(dimensiones)`**: Crea y devuelve una referencia a un array con las dimensiones especificadas en la tupla **`dimensiones`** cuyos elementos son aleatorios.

Ejercicio de Inducción: *Pruebe las siguientes líneas de código y verifique los resultados presentados:*

```
>>> print(np.zeros(3,2))
[[0. 0. 0.]
 [0. 0. 0.]]
>>> print(np.idendity(3))
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
>>> print(np.arange(1, 10, 2))
[1 3 5 7 9]
>>> print(np.linspace(0, 10, 5))
[ 0.   2.5  5.   7.5 10. ]
```

3. Atributos de un array

Existen varios atributos y funciones que describen las características de un array.

- `a.ndi`: Devuelve el número de dimensiones del array `a`.
- `a.shape`: Devuelve una tupla con las dimensiones del array `a`.
- `a.size`: Devuelve el número de elementos del array `a`.
- `a.dtype`: Devuelve el tipo de datos de los elementos del array `a`.

4. Acceso a los elementos de un array

Para acceder a los elementos contenidos en un array se usan índices al igual que para acceder a los elementos de una lista, pero indicando los índices de cada dimensión separados por comas.

Al igual que para listas, los índices de cada dimensión comienzan en 0.

También es posible obtener subarrays con el operador dos puntos `:` indicando el índice inicial y el siguiente al final para cada dimensión, de nuevo separados por comas.

Ejercicio de Inducción: *Pruebe las siguientes líneas de código y verifique los resultados presentados:*

```
>>> a = np.array([[1, 2, 3], [4, 5, 6]])
>>> print(a[1, 0])  # Acceso al elemento de la fila 1 columna 0
4
>>> print(a[1][0])  # Otra forma de acceder al mismo elemento
4
>>> print(a[:, 0:2])
[[1 2]
 [4 5]]
```

---

5. Filtrado de elementos de un array

Una característica muy útil de los arrays es que es muy fácil obtener otro array con los elementos que cumplen una condición.

- `a[condicion]`: Devuelve una lista con los elementos del array a que cumplen la condición `condicion`.

Ejercicio de Inducción: *Pruebe las siguientes líneas de código y verifique los resultados presentados:*

```
>>> a = np.array([[1, 2, 3], [4, 5, 6]])
>>> print(a[(a %2 == 0)])
[2 4 6]
>>> print(a[(a %2 == 0) &  (a > 2)])
[2 4]
```

---

6. Operaciones matemáticas con arrays

Existen dos formas de realizar operaciones matemáticas con arrays: a nivel de elemento y a nivel de array.

Las operaciones a nivel de elemento operan los elementos que ocupan la misma posición en dos arrays. Se necesitan, por tanto, dos arrays con las mismas dimensiones y el resultado es una array de la misma dimensión.

Los operadores mamemáticos `+`, `-`, `*`, `/`, `%`, `**` se utilizan para la realizar suma, resta, producto, cociente, resto y potencia a nivel de elemento.

Ejercicio de Inducción: *Pruebe las siguientes líneas de código y verifique los resultados presentados:*

```
>>> a = np.array([[1, 2, 3], [4, 5, 6]])
>>> b = np.array([[1, 1, 1], [2, 2, 2]])
>>> print(a + b )
[[2 3 4]
 [6 7 8]]
>>> print(a / b)
[[1.  2.  3. ]
 [2.  2.5 3. ]]
>>> print(a ** 2)
[[ 1  4  9]
 [16 25 36]]
```

---

7. Operaciones matemáticas a nivel de array

Para realizar el producto matricial se utiliza el método

- `a.dot(b)`: Devuelve el array resultado del producto matricial de los arrays `a` y `b` siempre y cuando sus dimensiones sean compatibles.

Y para trasponer una matriz se utiliza el método

- `a.T`: Devuelve el array resultado de trasponer el array `a`.

Ejercicio de Inducción: *Pruebe las siguientes líneas de código y verifique los resultados presentados:*

```
>>> a = np.array([[1, 2, 3], [4, 5, 6]])
>>> b = np.array([[1, 1], [2, 2], [3, 3]])
>>> print(a.dot(b))
[[14 14]
 [32 32]]
>>> print(a.T)
[[1 4]
 [2 5]
 [3 6]]
```
