**NUMPY**

**Librerías:**
import numpy as np

**Arrays:** un array o matriz es una **estructura de datos** que permite almacenar un conjunto de elementos **del mismo tipo**.
Los elementos están dispuestos en una secuencia ordenada y **se accede** a ellos **mediante un índice**, comienza en 0.

**Métodos de arrays**
**.shape**: devuelve una tupla que representa la **forma** (dimensión), y el **tamaño** de una dimensión. Si es tridimensional dará 3 valores: número de arrays pequeños, número de filas, número de columnas.
**.ndim**: número de índices para acceder a un elemento del array. Unidimensional 1 (filas), bidimensional 2 (filas, columnas), multidimensional 3 (arrays, filas, columnas).
**.size**: número total de **elementos** del array.
**.dtype**: **tipo** de datos

**CREACIÓN:**
**np.array(lista)**: a partir de una lista o de una lista de listas.

**np.empty((shape), dtype=float)**: un array vacío especificando la forma. Por defecto es float, no hace falta especificar type.
En shape bidimensional se especifican 2 valores: filas y columnas. En shape tridimensional se especifican 3 valores: dimensiones, filas y columnas.

**np.zeros(shape)**: un array de ceros. En shape tridimensional se especifican 3 valores: dimensiones, filas y columnas.

**np.ones(shape)**: un array de unos. En shape tridimensional se especifican 3 valores: dimensiones, filas y columnas.

**np.arange(start, stop, step)**: un array con valores secuenciales. Start opcional por defecto 0. Stop excluye el valor. Step opcional por defecto es 1.

**MÓDULO RANDOM**
Para simular datos aleatorios y realizar experimentos numéricos.

**np.random.randint(low, high, (size))**: low y high como start y stop. Size es la forma, por defecto es unidimensional. Dtype por defecto int.
array_random = np.random.randint(0, 50, (2,3)) números aleatorios entre el 0 y el 50 (no incluido) de 2 filas y 3 columnas

**np.random.rand(shape)**: números aleatorios en el rango [0, 1), 0 incluido y 1 excluido.
**np.random.sample**: Genera números aleatorios en un **array** en el rango [0, 1).

**INDEXACIÓN:** para acceder a un elemento se accede por su índice, al igual que en las listas.

**Array unidimensional:**
array(i) → i elemento
array[0] accede al primer elemento
array[-3] accede a los 3 últimos
array[2:5] accede a los elementos del 3 al 5
array[:11:2] accede a los elementos del primero al onceavo incluido saltando de 2 en 2.

**Array bidimensional:**
array(i, j) → i filas, j columnas. Lo que va a la izquierda de la , son filas a la derecha columnas. Funciona como start, stop, step.
array[0] accede a la primera fila
array[0][0] accede al elemento de la primera fila, primera columna
array[:2,:] accede a las dos primeras filas, todas las columnas
array[:,-3:] accede a todas las filas, tres últimas columnas

**Array tridimensional:**
array(i, j, k) → i array, j fila, k columna. Lo que va a la izquierda de la , son filas a la derecha columnas.
array[0, 1, 3] accede al primer array, fila 2, columna 4
array[0, 0, :] accede al primer array, primera fila, todas las columnas

**FILTRADO**

Paso 1: creamos una máscara booleana mask = array < 0.6
Paso 2: aplicamos la máscara array[mask] o array[array < 0.6] devuelve array unidimensional solo con los valores por debajo de 0.6

**Condiciones:** & es como and, | es como or.
array [(array < 0.2) | (array >0.7)] devuelve los resultados menores de 0.2 o los mayores de 0.7.

**Filtrado con np.where()**
np.where(condición)
array = np.where(array > 0.8)
Devuelve un resultado con dos tuplas, la primera es el índice de las filas la segunda el índice de las columnas. Hay que combinar ambas tuplas.

np.where(condición, valor_si_verdadero, valor_si_falso)
array = np.where(array > 0.8, 'xxx', 'ooo')
Devuelve xxx cuando la condición es verdadera y ooo cuando es falsa.
np.where(array > 50 array + 500, array)
Devuelve el número+500 cuando es mayor de 50 y el número original cuando es menor de 50.

**OPERACIONES ARITMÉTICAS**
**Suma elementos** = np.sum() axis 0c/1f suma todos los elementos entre sí
np.add(): **suma**
np.substract(): **resta**
np.multiply(): **multiplicación**
np.divide(): **división**
np.power(): **potencia**
np.round(array, 2): **redondea** a 2 **decimales** los elementos del array
+, -, *, / también funcionan pero es más correcto los métodos NumPy
**Escalar**: array * 2 multiplicar cada elemento del array * 2
suma = np.add(array1, array2)
Suma los elementos del array1 con el correspondiente del array2.

**Mínimo** = np.min() el valor mínimo
Mínimo_columna = np.min(array, axis = 0)
Mínimo_fila = np.min(array, axis = 1)

**Máximo** = np.max() el valor máximo
Máximo_columna = np. max (array, axis = 0)
Máximo_fila = np. max (array, axis = 1)

**FUNCIONES ESTADÍSTICAS**
**Media** = np.mean() media de todos los elementos del array
media_columna = np.mean(array, axis = 0) axis 0 por columnas
media_fila = np.mean(array, axis = 1) axis 1 por filas

**Varianza =** np.var()
Indica cómo de dispersos están los **valores alrededor de su media**. Una varianza alta significa que los valores están dispersos, y baja que los valores están cercanos a la media. **Expresada en unidades al cuadrado.**
axis 0/1 por columnas/filas.

**Desviación** = np.std()
Indica cuánto varían los valores con respecto a la media. Una desviación alta significa que los valores están dispersos, y baja que los valores están cercanos a la media. **Expresada en las mismas unidades.**
axis 0/1 por columnas/filas.

**OTROS MÉTODOS**
np.sort(): **ordena de menor a mayor** por filas los **elementos**. Si queremos que ordene por columnas axis = 0.
Para ordenar **de mayor a menor** ponerle símbolo menos al método y al array → ordenar = -np.sort(-array)

np.transpose(array): bidimensional. Devuelve un **nuevo array** con las dimensiones intercambiadas, **cambia las filas por las columnas**.
np.transpose(arrays, filas, columnas) multidimensional. Devuelve un **nuevo array** con el número de dimensiones, filas y columnas de cada dimensión indicadas.

np.shape(array, (filas, columnas)): **cambia la forma** pero no los datos del array, cambia las filas por las columnas
np.swapaxes(array, axis1, axis2): **intercambia los ejes** especificados en axis1 y axis2.
.copy(): crea una copia en una nueva variable.
.flatten(): convierte un array multidimensional en uno unidimensional.