



NumPy (Manejo de Arrays Numéricicos)

- ◇ **NumPy** es una librería para trabajar con **matrices** y operaciones matemáticas eficientes.

1 Instalar NumPy

```
pip install numpy
```

2 Importar NumPy

```
import numpy as np
```

3 Crear un Array

```
a = np.array([1, 2, 3, 4, 5]) # Array 1D
b = np.array([[1, 2, 3], [4, 5, 6]]) # Array 2D (Matriz)
```

4 Operaciones con Arrays

```
print(a + 10) # Sumar 10 a cada elemento
print(b * 2) # Multiplicar cada elemento por 2
print(np.mean(a)) # Calcular el promedio
print(np.sum(b)) # Sumar todos los elementos de la matriz
print(b.shape) # Tamaño de la matriz (filas, columnas)
```

la **estructura de datos principal** en NumPy es el **array multidimensional** llamado **ndarray (N-dimensional array)**.

Los ndarray son **más eficientes** que las listas de Python porque:

- Ocupan **menos memoria**.
- Son **más rápidos** en operaciones matemáticas.
- Permiten operaciones vectorizadas **sin necesidad de bucles**.



REGISTRO NACIONAL DE ASOCIACIONES N°611922
DECLARADA ENTIDAD DE UTILIDAD PÚBLICA ESTATAL
AGENCIA DE COLOCACIÓN: ID 0100000017

CENTRO EN MÁLAGA
C/DOS ACERAS 23, 29012
MÁLAGA | (+34) 952 300 500
ARRABAL@ARRABALEMPLEO.ORG

CENTRO EN CÁDIZ
TR.^a ALAMEDA DE SOLANO, 32, 11130
CHICLANA DE LA FRONTERA | (+34) 956 900 312
CHICLANA@ARRABALEMPLEO.ORG



💡 Tipos de Arrays en NumPy

1 Array Unidimensional (Vector)

```
import numpy as np
a = np.array([1, 2, 3, 4, 5])
print(a) # [1 2 3 4 5]
print(a.shape) # (5,) -> Tiene 5 elementos en una sola dimensión
```

2 Array Bidimensional (Matriz)

```
b = np.array([[1, 2, 3], [4, 5, 6]])
print(b)
```

- Esto representa una **matriz de 2 filas y 3 columnas**:

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

```
print(b.shape) # (2, 3) -> 2 filas, 3 columnas
```

3 Array Multidimensional (3D o más)

```
c = np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]])
print(c.shape) # (2, 2, 2) -> 2 matrices de 2x2
```



❖ Operaciones Básicas con Arrays

◇ Operaciones Aritméticas

```
a = np.array([1, 2, 3, 4])
print(a * 2) # Multiplica cada elemento por 2 -> [2 4 6 8]
print(a + 10) # Suma 10 a cada elemento -> [11 12 13 14]
```

◇ Funciones Matemáticas

```
print(np.mean(a)) # Promedio
print(np.sum(a)) # Suma total
print(np.max(a)) # Máximo valor
print(np.min(a)) # Mínimo valor
```

◇ Indexación y Slicing

```
print(a[0]) # Primer elemento
print(a[1:3]) # Elementos del índice 1 al 2 -> [2 3]
```

❖ Resumen

Concepto	Descripción
np.array()	Crea un array
.shape	Tamaño del array
a * 2	Operaciones vectorizadas
np.mean(a)	Promedio de los valores
a[1:3]	Slicing (subconjunto del array)

🔥 NumPy es clave para cálculos numéricos rápidos en Python





Pandas (Manejo de Datos)

- ◇ Pandas se usa para manejar datos en **tablas (DataFrames)**.

1 Instalar Pandas

```
pip install pandas
```

2 Importar Pandas

```
import pandas as pd
```

3 Crear un DataFrame

```
datos = {
    'Nombre': ['Ana', 'Luis', 'Pedro'],
    'Edad': [25, 30, 22],
    'Salario': [2500, 3200, 2800]
}

df = pd.DataFrame(datos) # Crear DataFrame
print(df)
```

4 Leer Datos desde un Archivo

```
df = pd.read_csv("archivo.csv") # Leer un archivo CSV
print(df.head()) # Mostrar las primeras filas
```

5 Filtrar y Ordenar Datos

```
df_filtrado = df[df["Edad"] > 25] # Filtrar empleados mayores de 25 años
df_ordenado = df.sort_values("Salario", ascending=False) # Ordenar por salario
```





❖ ¿Qué es un DataFrame en Pandas?

Un **DataFrame** en Pandas es una estructura de datos **tabular** (como una **hoja de cálculo** o una **tabla de base de datos**), que contiene **filas y columnas**. Cada columna puede contener diferentes tipos de datos, como números, cadenas de texto, fechas, etc.

Es una de las estructuras de datos más poderosas de Pandas porque permite hacer **operaciones complejas** con facilidad, como filtrar, agrupar, combinar y transformar datos.

❖ ¿Qué otros tipos de datos se usan en Pandas?

Series: Una **Serie** es una columna individual de un DataFrame. Es un array unidimensional con etiquetas (índices).

```
s = pd.Series([1, 2, 3, 4])
print(s) # Índices automáticos: 0, 1, 2, 3
```

Categorical: Tipo de dato para representar **variables categóricas** que tienen un número limitado de posibles valores.

```
cat = pd.Categorical(['a', 'b', 'a', 'b', 'a'])
```

Datetime: Pandas también tiene tipos de datos especiales para manejar **fechas y horas** de manera eficiente.

```
fechas = pd.to_datetime(['2021-01-01', '2022-02-01'])
```





❖ Principales Operaciones con DataFrame

1 Crear un DataFrame

```
import pandas as pd

# Crear un DataFrame a partir de un diccionario
data = {'Nombre': ['Ana', 'Luis', 'Pedro'],
        'Edad': [25, 30, 22],
        'Salario': [2500, 3200, 2800]}

df = pd.DataFrame(data)
print(df)
```

2 Leer datos desde un archivo (CSV, Excel, etc.)

```
df = pd.read_csv('empleados.csv') # Leer un archivo CSV
```

3 Acceder a columnas y filas

```
print(df['Nombre']) # Acceder a la columna 'Nombre'
print(df.iloc[0]) # Acceder a la primera fila (por índice)
```

4 Filtrar datos

```
df_filtrado = df[df['Edad'] > 25] # Filtrar empleados mayores de 25 años
print(df_filtrado)
```

5 Operaciones estadísticas

```
print(df['Salario'].mean()) # Promedio de salarios
print(df['Edad'].max()) # Edad máxima
```

6 Ordenar datos

```
df_ordenado = df.sort_values('Salario', ascending=False) # Ordenar por salario
print(df_ordenado)
```





7 Agrupar datos

```
df_grouped = df.groupby('Edad').mean() # Agrupar por edad y calcular el promedio
print(df_grouped)
```

8 Modificar valores

```
df['Salario'] = df['Salario'] * 1.1 # Aumentar un 10% a los salarios
```

9 Eliminar columnas o filas

```
df = df.drop('Edad', axis=1) # Eliminar la columna 'Edad'
```

Resumen

- **DataFrame:** Estructura de datos tabular (como una tabla) con filas y columnas.
- **Series:** Columna individual de un DataFrame.
- **Operaciones:**
 - Crear, leer y acceder a datos.
 - Filtrar, ordenar, agrupar.
 - Realizar estadísticas y modificar valores.

 **Pandas es ideal para trabajar con datos tabulares y hacer análisis y manipulaciones complejas de forma sencilla.**



Matplotlib (Generar Gráficos)

- ◇ **Matplotlib** permite hacer gráficos como **barras, líneas y pastel**.

1 Instalar Matplotlib

```
pip install matplotlib
```

2 Importar Matplotlib

```
import matplotlib.pyplot as plt
```

3 Crear un Gráfico de Barras

```
plt.bar(df["Nombre"], df["Salario"], color="skyblue")
plt.title("Salarios de Empleados")
plt.xlabel("Empleados")
plt.ylabel("Salario")
plt.show()
```

4 Crear un Gráfico de Pastel

```
plt.pie(df["Edad"].value_counts(), labels=df["Edad"].unique(), autopct="%1.1f%%")
plt.title("Distribución de Edades")
plt.show()
```

Resumen Final

Librería	Uso Principal
NumPy	Cálculo numérico y matrices eficientes
Pandas	Manejo de datos en tablas (DataFrames)
Matplotlib	Visualización de datos (gráficos)





💡 ¿Qué es plt en Matplotlib?

En **Matplotlib**, **plt** es el alias comúnmente utilizado para la librería **matplotlib.pyplot**, que es un módulo que contiene funciones para crear gráficos de manera sencilla y rápida.

¿Por qué usar plt?

Usamos plt porque **pyplot** proporciona una interfaz similar a **MATLAB** que hace que crear gráficos en Python sea fácil, sin necesidad de manejar configuraciones más complejas.

Principales Funciones de plt en Matplotlib

Aquí te dejo las funciones más utilizadas para crear gráficos.

1 Crear un gráfico de barras

```
import matplotlib.pyplot as plt

# Datos
categorias = ['A', 'B', 'C', 'D']
valores = [3, 7, 5, 2]

# Crear gráfico de barras
plt.bar(categorias, valores)
plt.title("Gráfico de Barras")
plt.xlabel("Categorías")
plt.ylabel("Valores")
plt.show()
```

2 Crear un gráfico de líneas

```
# Datos
x = [1, 2, 3, 4, 5]
y = [1, 4, 9, 16, 25]

# Crear gráfico de Líneas
plt.plot(x, y, color='green') # Línea verde
plt.title("Gráfico de Líneas")
plt.xlabel("X")
plt.ylabel("Y")
plt.show()
```



REGISTRO NACIONAL DE ASOCIACIONES N°611922
DECLARADA ENTIDAD DE UTILIDAD PÚBLICA ESTATAL
AGENCIA DE COLOCACIÓN: ID 0100000017

CENTRO EN MÁLAGA
C/DOS ACERAS 23, 29012
MÁLAGA | (+34) 952 300 500
ARRABAL@ARRABALEMPLEO.ORG

CENTRO EN CÁDIZ
TR.^a ALAMEDA DE SOLANO, 32, 11130
CHICLANA DE LA FRONTERA | (+34) 956 900 312
CHICLANA@ARRABALEMPLEO.ORG



3 Crear un gráfico de pastel

```
# Datos
sizes = [40, 30, 20, 10]
labels = ['A', 'B', 'C', 'D']

# Crear gráfico de pastel
plt.pie(sizes, labels=labels, autopct='%1.1f%%')
plt.title("Gráfico de Pastel")
plt.show()
```

4 Crear un histograma

```
import numpy as np

# Datos
data = np.random.randn(1000)

# Crear histograma
plt.hist(data, bins=30, color='orange')
plt.title("Histograma")
plt.xlabel("Valores")
plt.ylabel("Frecuencia")
plt.show()
```

5 Configuración del gráfico

```
# Títulos y etiquetas
plt.title("Mi gráfico")
plt.xlabel("Eje X")
plt.ylabel("Eje Y")

# Líneas de cuadrícula
plt.grid(True)
```



Funciones Importantes de plt

- **plt.plot():** Crear un gráfico de líneas.
- **plt.bar():** Crear un gráfico de barras.
- **plt.pie():** Crear un gráfico de pastel.
- **plt.hist():** Crear un histograma.
- **plt.title():** Establecer el título del gráfico.
- **plt.xlabel() y plt.ylabel():** Establecer las etiquetas de los ejes.
- **plt.show():** Mostrar el gráfico.
- **plt.savefig():** Guardar el gráfico como imagen

❖ Resumen

- **plt** es una forma de acceder a **matplotlib.pyplot** y crear gráficos con funciones simples.
- Las funciones más usadas son **plot()**, **bar()**, **pie()**, y **hist()** para diferentes tipos de gráficos.

En **Matplotlib** hay otras formas de trabajar con gráficos además de **plt**. Aunque **plt** es la forma más común y sencilla de crear gráficos, **Matplotlib** tiene una interfaz orientada a objetos que permite un control más preciso y avanzado sobre los gráficos.

1 Interfaz Orientada a Objetos (OO)

Matplotlib ofrece una **interfaz orientada a objetos** usando las clases **Figure** y **Axes** para crear gráficos más complejos y personalizables.

Estructura Principal: Figure, Axes, y Axis

- **Figure:** Es el contenedor principal para el gráfico. Puede contener uno o más ejes (gráficos).
- **Axes:** Es el área del gráfico donde se dibujan los datos. Un gráfico puede tener varios ejes, como un gráfico de líneas y uno de barras.
- **Axis:** Son los ejes de un gráfico (X, Y, Z).



2 Crear Gráficos Usando la Interfaz OO

Aquí te muestro cómo se usaría esta estructura sin la ayuda de plt:

Ejemplo de creación de un gráfico de barras con la interfaz OO:

```
import matplotlib.pyplot as plt

# Crear la figura y los ejes
fig, ax = plt.subplots() # Crea una figura y un eje

# Datos para el gráfico
categorias = ['A', 'B', 'C', 'D']
valores = [3, 7, 5, 2]

# Crear el gráfico de barras en el eje (ax)
ax.bar(categorias, valores)

# Personalizar el gráfico
ax.set_title("Gráfico de Barras") # Título del gráfico
ax.set_xlabel("Categorías") # Etiqueta del eje X
ax.set_ylabel("Valores") # Etiqueta del eje Y

# Mostrar el gráfico
plt.show()
```

3 Principales Clases en la Interfaz OO

- **plt.subplots():** Crea una figura y uno o más ejes. Es equivalente a **plt.figure()** y **plt.add_subplot()** pero más flexible.
- **Figure:** El contenedor principal de todos los elementos de un gráfico.
- **Axes:** Es el área donde se grafican los datos. Puedes tener varios ejes dentro de una figura.
- **Axis:** Es el objeto que representa los ejes (como X, Y, Z).



Ejemplo de usar múltiples ejes dentro de una figura:

```
fig, (ax1, ax2) = plt.subplots(1, 2) # Crea una figura con 2 ejes (Lado a Lado)

# Gráfico 1 (Barras)
ax1.bar(categorias, valores)
ax1.set_title("Gráfico de Barras")

# Gráfico 2 (Líneas)
ax2.plot(categorias, valores)
ax2.set_title("Gráfico de Líneas")

plt.show()
```

4 ¿Cuándo Usar la Interfaz OO?

- **Control Avanzado:** Cuando necesitas un control más preciso sobre la disposición de los gráficos, ejes, y elementos.
- **Subgráficos:** Si deseas crear varias visualizaciones dentro de una misma figura.
- **Personalización Completa:** Si necesitas personalizar los elementos del gráfico más allá de las capacidades de plt.

Resumen

- **plt** es la interfaz sencilla y más común para crear gráficos rápidamente.
- **Interfaz Orientada a Objetos (OO)** usa **Figure**, **Axes**, y **Axis** para una mayor flexibilidad y control.
- **subplots()** es la forma más usada para crear gráficos con la interfaz OO, permitiendo trabajar con múltiples ejes dentro de una figura.

 Si bien plt es más fácil de usar, la interfaz OO es muy útil para proyectos más complejos o cuando necesitas más control.