

Introducción a bibliotecas de datos en Python.

NumPy, Pandas y Matplotlib estas son tres de las bibliotecas más populares en Python para el trabajo con datos y visualización.

1. NumPy (Numerical Python)

NumPy es una biblioteca fundamental para la computación científica en Python. Ofrece soporte para **vectores** y **matrices multidimensionales**, y proporciona funciones matemáticas de alto nivel para operar con estos datos. Es especialmente útil para trabajar con grandes cantidades de datos numéricos.

Instalación:

```
pip install numpy
```

Conceptos clave:

- **Arreglos (arrays):** Los arrays de **NumPy** son similares a las listas en Python, pero más rápidos y eficientes.
- **Operaciones vectorizadas:** Puedes realizar operaciones en arrays completos sin la necesidad de bucles explícitos.

Ejemplo básico:

```
import numpy as np

# Crear un array de 1D
arr = np.array([1, 2, 3, 4, 5])
print(arr)

# Operaciones vectorizadas
arr2 = arr * 2 # Multiplica cada elemento por 2
print(arr2)

# Crear una matriz de 2D
matriz = np.array([[1, 2], [3, 4]])
print(matriz)

# Operaciones entre matrices
result = matriz + matriz
print(result)
```



Cofinanciado por
la Unión Europea



MINISTERIO
DE TRABAJO
Y ECONOMÍA SOCIAL



Fondos Europeos



Ayuda
en Acción
Impulsa Empleo Joven

Algunas funciones útiles de NumPy:

- **np.array()**: Crea arrays.
- **np.zeros()**, **np.ones()**: Crea arrays llenos de ceros o unos.
- **np.mean()**, **np.median()**, **np.std()**: Cálculos estadísticos.

2. Pandas

Pandas es una biblioteca para la manipulación y análisis de datos. Se usa principalmente para trabajar con **tablas de datos** (estructuras DataFrame y Series). Es ideal para importar, limpiar, transformar, explorar y analizar datos.

Instalación: `pip install pandas`

Conceptos clave:

- **DataFrame**: Una estructura tabular bidimensional, similar a una hoja de cálculo.
- **Series**: Es una columna en un **DataFrame**.

Ejemplo básico:

```
import pandas as pd

# Crear un DataFrame desde un diccionario
data = {'Nombre': ['Juan', 'Ana', 'Pedro'],
        'Edad': [28, 22, 35]}
df = pd.DataFrame(data)
print(df)

# Acceder a una columna
print(df['Nombre'])

# Operaciones de filtrado
print(df[df['Edad'] > 30])

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



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.º ALAMEDA DE SOLANO, 32, 11130
CHICLANA DE LA FRONTERA | (+34) 956 900 312
CHICLANA@ARRABALEMPLEO.ORG



Cofinanciado por
la Unión Europea



MINISTERIO
DE TRABAJO
Y ECONOMÍA SOCIAL



Fondos Europeos



Algunas funciones útiles de Pandas:

- **pd.DataFrame()**: Crea un DataFrame.
- **df.head()**, **df.tail()**: Muestra las primeras o últimas filas.
- **df.describe()**: Estadísticas descriptivas.
- **df['columna']**: Accede a una columna.
- **df.groupby()**: Agrupación de datos para resúmenes.

Aunque tanto **NumPy** como **Pandas** pueden manejar datos en estructuras bidimensionales, su propósito y enfoque son diferentes. Aquí te explico la diferencia clave:

NumPy

- Se enfoca en **operaciones matemáticas y científicas** de alto rendimiento.
- Usa **arrays multidimensionales** (n-dimensionales), que son eficientes en cuanto a memoria y velocidad.
- Soporta **operaciones vectorizadas** (suma, multiplicación, estadísticas, etc.), lo que lo hace ideal para cálculos numéricos.
- No tiene etiquetas de fila/columna, solo índices numéricos.

Ejemplo de una matriz en NumPy:

```
import numpy as np
matriz = np.array([[1, 2, 3], [4, 5, 6]])
print(matriz)
```

Pandas

Se especializa en **manejo y análisis de datos estructurados**.

- Usa **DataFrames** (tablas con filas y columnas etiquetadas), lo que facilita la manipulación de datos.
- Integra fácilmente fuentes de datos como archivos CSV, bases de datos, JSON, etc.
- Tiene funciones poderosas para **filtrar, agrupar y limpiar datos**.

Ejemplo de un DataFrame en Pandas:

```
import pandas as pd
datos = {'Nombre': ['Ana', 'Luis', 'Carlos'], 'Edad': [25, 30, 22]}
df = pd.DataFrame(datos)
print(df)
```



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.º ALAMEDA DE SOLANO, 32, 11130
CHICLANA DE LA FRONTERA | (+34) 956 900 312
CHICLANA@ARRABALEMPLEO.ORG

¿En qué se diferencian?

Característica	NumPy (ndarray)	Pandas (DataFrame)
Tipo de datos	Números y arrays rápidos	Tablas con datos estructurados
Etiquetas	Solo índices numéricos	Nombres de filas y columnas
Operaciones	Matemáticas y estadísticas rápidas	Manipulación y análisis de datos
Lectura de archivos	No directamente	Sí (CSV, Excel, SQL, JSON)
Uso principal	Cálculo numérico	Manejo de datos tabulares

NumPy es ideal para operaciones numéricas con **matrices y vectores**.

Pandas es mejor para manipular y analizar **datos en forma de tabla**.

*Si necesitas cálculos rápidos con matrices, usa **NumPy**.*

*Si trabajas con datos estructurados con etiquetas (como una tabla de Excel o una base de datos), usa **Pandas**.*

3. Matplotlib

Matplotlib es una biblioteca para crear gráficos y visualizaciones estáticas, animadas o interactivas. Es especialmente útil para crear gráficos como líneas, barras, dispersión y más.

Instalación: `pip install matplotlib`

Conceptos clave:

- **plt.plot()**: Para crear gráficos de líneas.
- **plt.scatter()**: Para gráficos de dispersión.
- **plt.bar()**: Para gráficos de barras.

Ejemplo básico:

```
import matplotlib.pyplot as plt

# Crear un gráfico de líneas
x = [1, 2, 3, 4, 5]
y = [1, 4, 9, 16, 25]

plt.plot(x, y) # Dibuja una línea
plt.title("Gráfico de líneas") # Título del gráfico
plt.xlabel("X") # Etiqueta eje X
plt.ylabel("Y") # Etiqueta eje Y
plt.show()

# Crear un gráfico de barras
plt.bar([1, 2, 3], [10, 20, 30])
plt.show()
```

Algunas funciones útiles de Matplotlib:

- **plt.plot()**: Crear un gráfico de líneas.
- **plt.scatter()**: Crear un gráfico de dispersión.
- **plt.title()**, **plt.xlabel()**, **plt.ylabel()**: Añadir títulos y etiquetas.
- **plt.show()**: Muestra el gráfico.

Resumen:

Biblioteca	Propósito	Funcionalidad clave
NumPy	Computación numérica eficiente con arrays.	Arrays multidimensionales, operaciones vectorizadas.
Pandas	Manipulación de datos en formato tabular.	DataFrames, Series, limpieza, transformación y análisis.
Matplotlib	Visualización de datos.	Gráficos de líneas, barras, dispersión y más.

¿Por qué son importantes?

- **NumPy** es esencial para realizar operaciones matemáticas y científicas de manera eficiente.
- **Pandas** facilita el manejo de grandes volúmenes de datos tabulares (por ejemplo, hojas de cálculo o bases de datos).
- **Matplotlib** es crucial para la visualización de datos y la creación de gráficos interpretables.