Introducción a NumPy

Fundamentos, Utilidad y Funciones Principales

Autor: Juan Camilo Hernandez Perez 20251020029

26 de febrero de 2026

Índice

1. ¿Qué es NumPy? **2**
1.1. Característica Principal 2

2. ¿Para qué sirve NumPy? **2**

3. Funciones más importantes **2**
3.1. Creación de arreglos 2
3.2. Manipulación de arreglos 3
3.3. Funciones estadísticas 3
3.4. Funciones matemáticas 3

4. Álgebra Lineal (np.linalg) **4**

5. Funciones útiles adicionales **4**
5.1. Indexación avanzada 4
5.2. Operaciones acumulativas 4
5.3. Valores únicos y filtrado 4

6. Números Aleatorios **4**

7. Conclusión **5**

1. ¿Qué es NumPy?

NumPy (Numerical Python) es una biblioteca fundamental para la computación científica en Python. Proporciona soporte para arreglos multidimensionales eficientes y una gran colección de funciones matemáticas de alto rendimiento.

Es la base de muchas otras bibliotecas como pandas, matplotlib, scikit-learn y TensorFlow.

1.1. Característica Principal

Su estructura central es el **ndarray** (N-dimensional array), que permite almacenar datos de forma contigua en memoria, lo que lo hace mucho más rápido que las listas tradicionales de Python.

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

2. ¿Para qué sirve NumPy?

NumPy es utilizado en:

- Ciencia de datos
- Inteligencia artificial
- Machine Learning
- Álgebra lineal
- Simulaciones numéricas
- Procesamiento de señales

Su principal ventaja es la vectorización, que permite realizar operaciones matemáticas sin necesidad de usar ciclos `for`.

3. Funciones más importantes

3.1. Creación de arreglos

- `np.array()` – Crea un arreglo a partir de listas o datos existentes.
- `np.zeros()` – Genera un arreglo lleno de ceros.
- `np.ones()` – Genera un arreglo lleno de unos.
- `np.arange()` – Crea un arreglo con valores en un rango específico.
- `np.linspace()` – Genera valores equidistantes entre dos números.
- `np.eye()` – Crea una matriz identidad.

- `np.full()` – Crea un arreglo lleno con un valor específico.
- `np.empty()` – Crea un arreglo sin inicializar (valores aleatorios en memoria).

3.2. Manipulación de arreglos

- `reshape()` – Cambia la forma o dimensiones del arreglo.
- `transpose()` o `.T` – Intercambia filas por columnas.
- `flatten()` – Convierte el arreglo en una dimensión (copia).
- `ravel()` – Convierte el arreglo en una dimensión (vista si es posible).
- `concatenate()` – Une varios arreglos.
- `split()` – Divide un arreglo en partes.
- `vstack()` – Apila arreglos verticalmente.
- `hstack()` – Apila arreglos horizontalmente.

3.3. Funciones estadísticas

- `mean()` – Calcula el promedio.
- `median()` – Calcula la mediana.
- `std()` – Calcula la desviación estándar.
- `var()` – Calcula la varianza.
- `sum()` – Suma los elementos.
- `min()` / `max()` – Obtiene el valor mínimo o máximo.
- `percentile()` – Calcula un percentil específico.

3.4. Funciones matemáticas

- `sqrt()` – Calcula la raíz cuadrada.
- `exp()` – Calcula la exponencial.
- `log()` – Logaritmo natural.
- `log10()` – Logaritmo en base 10.
- `sin()`, `cos()`, `tan()` – Funciones trigonométricas.
- `abs()` – Valor absoluto.
- `round()` – Redondea valores.

4. Álgebra Lineal (np.linalg)

- `np.linalg.inv()` – Calcula la inversa de una matriz.
- `np.linalg.det()` – Calcula el determinante.
- `np.linalg.eig()` – Obtiene valores y vectores propios.
- `np.linalg.solve()` – Resuelve sistemas de ecuaciones lineales.
- `np.dot()` – Producto punto entre vectores o matrices.
- `np.matmul()` – Multiplicación matricial.

5. Funciones útiles adicionales

5.1. Indexación avanzada

- `where()` – Devuelve elementos que cumplen una condición.
- `argmax()` / `argmin()` – Devuelve la posición del valor mayor o menor.
- `argsort()` – Devuelve los índices que ordenan el arreglo.
- `nonzero()` – Devuelve índices de elementos distintos de cero.

5.2. Operaciones acumulativas

- `cumsum()` – Suma acumulativa.
- `cumprod()` – Producto acumulativo.

5.3. Valores únicos y filtrado

- `unique()` – Devuelve valores únicos.
- `clip()` – Limita valores dentro de un rango.
- `astype()` – Cambia el tipo de dato del arreglo.

6. Números Aleatorios

- `np.random.rand()` – Genera números aleatorios uniformes.
- `np.random.randn()` – Genera números aleatorios con distribución normal estándar.
- `np.random.randint()` – Genera enteros aleatorios en un rango.
- `np.random.choice()` – Selecciona elementos aleatorios.
- `np.random.normal()` – Genera números con distribución normal.

7. Conclusión

NumPy es una herramienta esencial para cualquier persona que trabaje con datos numéricos en Python. Su eficiencia, estructura de memoria optimizada y enorme cantidad de funciones matemáticas lo convierten en la base del ecosistema científico en Python.

Dominar NumPy significa comprender cómo funcionan los arreglos, la vectorización y las operaciones matemáticas de alto rendimiento.