Guía para Principiantes de NumPy

Tiempo Estimado : 10 Minutos

Objetivo:

En esta lectura, aprenderás:

- Conceptos básicos de NumPy
- Cómo crear arreglos de NumPy
- Atributos de los arreglos e indexación
- Operaciones básicas como suma y multiplicación

¿Qué es NumPy?

NumPy, abreviatura de **Num**erical **Py**thon, es una biblioteca fundamental para la computación numérica y científica en Python. Proporciona soporte para grandes arreglos y matrices multidimensionales, junto con una colección de funciones matemáticas de alto nivel para operar en estos arreglos. NumPy sirve como la base para muchas bibliotecas de ciencia de datos y aprendizaje automático, lo que lo convierte en una herramienta esencial para el análisis de datos y la investigación científica en Python.

Aspectos clave de NumPy en Python:

- **Estructuras de datos eficientes**: NumPy introduce estructuras de arreglos eficientes, que son más rápidas y más eficientes en memoria que las listas de Python. Esto es crucial para manejar grandes conjuntos de datos.
- **Arreglos multidimensionales**: NumPy te permite trabajar con arreglos multidimensionales, lo que facilita la representación de matrices y tensores. Esto es especialmente útil en la computación científica.
- **Operaciones elemento a elemento**: NumPy simplifica las operaciones matemáticas elemento a elemento en arreglos, lo que facilita realizar cálculos en conjuntos de datos completos de una sola vez.
- **Generación de números aleatorios**: Proporciona una amplia gama de funciones para generar números aleatorios y datos aleatorios, lo cual es útil para simulaciones y análisis estadístico.
- **Integración con otras bibliotecas**: NumPy se integra sin problemas con otras bibliotecas de ciencia de datos como SciPy, Pandas y Matplotlib, mejorando su utilidad en varios dominios.
- **Optimización del rendimiento**: Las funciones de NumPy están implementadas en lenguajes de bajo nivel como C y Fortran, lo que aumenta significativamente su rendimiento. Es una opción preferida cuando la velocidad es esencial.

Instalación

Si aún no has instalado NumPy, puedes hacerlo usando `pip`:

```
pip install numpy
```

Creando arreglos NumPy

Puedes crear arreglos NumPy a partir de listas de Python. Estos arreglos pueden ser unidimensionales o multidimensionales.

Creando un array 1D

```
import numpy as np
```

import numpy as np: En esta línea, se importa la biblioteca NumPy y se le asigna un alias `np` para facilitar su referencia en el código.

```
# Creating a 1D array
arr_1d = np.array([1, 2, 3, 4, 5]) # **np.array()** is used to create NumPy arrays.
```

arr_1d = np.array([1, 2, 3, 4, 5]): En esta línea, se crea un arreglo unidimensional de NumPy llamado `arr_1d`. Utiliza la función `np.array()` para convertir una lista de Python [1, 2, 3, 4, 5] en un arreglo de NumPy. Este arreglo contiene cinco elementos, que son 1, 2, 3, 4 y 5. `arr_1d` es un arreglo 1D porque tiene una única fila de elementos.

Creando un arreglo 2D

```
import numpy as np
```

import numpy as np: En esta línea, se importa la biblioteca NumPy y se le asigna un alias `np` para facilitar su referencia en el código.

```
# Creating a 2D array
arr_2d = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
```

arr_2d = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]]): En esta línea, se crea un array de NumPy bidimensional llamado `arr_2d`. Utiliza la función `np.array()` para convertir una lista de listas en un array 2D de NumPy. La lista exterior contiene tres listas interiores, cada una de las cuales representa una fila de elementos. Así, `arr_2d` es un array 2D con tres filas y tres columnas. Los elementos en este array forman una matriz con valores del 1 al 9, organizados en una cuadrícula de 3x3.

Atributos de arreglos

Los arreglos de NumPy tienen varios atributos útiles:

```
# Array attributes
print(arr_2d.ndim)  # ndim : Represents the number of dimensions or "rank" of the array.
# output : 2
print(arr_2d.shape)  # shape : Returns a tuple indicating the number of rows and columns in the array.
# Output : (3, 3)
print(arr_2d.size) # size: Provides the total number of elements in the array.
# Output : 9
```

Indexación y corte

Puedes acceder a los elementos de un arreglo de NumPy utilizando indexación y corte:

En esta línea, se accede al tercer elemento (índice 2) del arreglo unidimensional `arr_1d`.

```
# Indexing and slicing
print(arr_1d[2])          # Accessing an element (3rd element)
```

En esta línea, se accede al elemento en la 2ª fila (índice 1) y 3ª columna (índice 2) del arreglo 2D `arr_2d`.

```
print(arr_2d[1, 2])       # Accessing an element (2nd row, 3rd column)
```

En esta línea, se accede a la 2ª fila (índice 1) del array 2D `arr_2d`.

```
print(arr_2d[1])          # Accediendo a una fila (2ª fila)
```

```
In this line, the 2nd column (index 1) of the 2D array `arr_2d` is accessed.
```python
print(arr_2d[:, 1])       # Accessing a column (2nd column)
```

Operaciones básicas

NumPy simplifica las operaciones básicas en arreglos:

Operaciones aritméticas elemento a elemento:

Suma, resta, multiplicación y división de arreglos con escalares u otros arreglos.

Suma de arreglos

```
# Array addition
array1 = np.array([1, 2, 3])
array2 = np.array([4, 5, 6])
result = array1 + array2
print(result)  # [5 7 9]
```

Multiplicación escalar

```
# Scalar multiplication
array = np.array([1, 2, 3])
result = array * 2 # each element of an array is multiplied by 2
print(result)  # [2 4 6]
```
```

Multiplicación elemento a elemento (Producto de Hadamard)

```
# Element-wise multiplication (Hadamard product)
array1 = np.array([1, 2, 3])
array2 = np.array([4, 5, 6])
result = array1 * array2
print(result)  # [4 10 18]
```

Multiplicación de matrices

```
# Matrix multiplication
matrix1 = np.array([[1, 2], [3, 4]])
matrix2 = np.array([[5, 6], [7, 8]])
result = np.dot(matrix1, matrix2)
print(result)
# [[19 22]
#  [43 50]]
```

NumPy simplifica estas operaciones, haciéndolas más fáciles y eficientes que las listas tradicionales de Python.

Operaciones con NumPy

Aquí está la lista de operaciones que se pueden realizar utilizando Numpy

Operación	**Descripción**	**Ejemplo**
Creación de Arreglos	Creando un arreglo de NumPy.	`arr = np.array([1, 2, 3, 4, 5])`
Aritmética Elemento por Elemento	Suma, resta, etc. elemento por elemento.	`result = arr1 + arr2`
Aritmética Escalar	Suma, resta, etc. escalar.	`result = arr * 2`
Funciones Elemento por Elemento	Aplicando funciones a cada elemento.	`result = np.sqrt(arr)`
Suma y Media	Calculando la suma y la media de un arreglo.	`total = np.sum(arr) average = np.mean(arr)`
Valores Máximos y Mínimos	Encontrando los valores máximos y mínimos.	`max_val = np.max(arr) min_val = np.min(arr)`
Cambio de Forma	Cambiando la forma de un arreglo.	`reshaped_arr = arr.reshape(2, 3)`
Transposición	Transponiendo un arreglo multidimensional.	`transposed_arr = arr.T`
Multiplicación de Matrices	Realizando multiplicación de matrices.	`result = np.dot(matrix1, matrix2)`

Conclusión

NumPy es una biblioteca fundamental para la ciencia de datos y los cálculos numéricos. Esta guía cubre lo básico de NumPy, y hay mucho más por explorar. Visita numpy.org para más información y ejemplos.

Autor

Akansha Yadav