Data Analysis con Python

Juan Felipe Acevedo Pérez

Monitor Junior

UIFCE
UNIDAD DE INFORMÁTICA

Curso libre:
Data Analysis con Python

Monitor encargado:
Juan Felipe Acevedo Pérez

Correo: uniic_bog@unal.edu.co
Tel: 3165000 ***Ext:*** 12301

Facultad de Ciencias Económicas
Unidad de Informática
Sede Bogotá

UNIVERSIDAD NACIONAL DE COLOMBIA

1

Sesión 1

Matrices y Computación Vectorizada

Correo: uniic_bog@unal.edu.co
Teléfono: 3165000 **ext** 12301

UIFCE
UNIDAD DE INFORMÁTICA

Contenido

Computación vectorizada

Numpy

Vectores y Matrices

Correo: uniic_bog@unal.edu.co
Teléfono: 3165000 **ext** 12301

Computación Vectorizada

La carga, el almacenamiento y la manipulación en memoria de datos, aparentemente se caracteriza por su *heterogeneidad*. Existe una amplia variedad en el tema:

- Diferentes fuentes y tipos de datos
- Gama de formatos (documentos, imágenes, sonidos, mediciones numéricas, etc.)

A pesar de esto, nos ayudará pensar que los datos se fundamentan en arreglos numéricos.

Correo: uniic_bog@unal.edu.co
Teléfono: 3165000 **ext** 12301

Pensemos de forma diferente

- **Imágenes:** pueden considerarse simplemente como dos matrices dimensionales de números que representan el brillo de píxeles en el área.

- **Clips de sonido:** Los clips de sonido se pueden considerar como matrices unidimensionales de intensidad frente al tiempo.

- **Texto:** El texto se puede convertir de varias maneras en representaciones numéricas, tal vez dígitos binarios que representan la frecuencia de ciertas palabras o pares de palabras.

¿Qué ocurre internamente?

- No importa cuáles sean los datos, el primer paso para hacerlos analizables será transformarlos en matrices de números.

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

- Por esta razón, el almacenamiento y la manipulación eficientes de matrices numéricas es absolutamente **fundamental** para el proceso de hacer ciencia de datos.

Numpy (Numerical Python)

- Numpy es la herramienta personalizada que se utiliza en Python para manejar los **arreglos (vectores) numéricos.**

- Los arreglos forman el **núcleo** de casi todo el **ecosistema** de herramientas de ciencia de datos en Python, por lo que el tiempo dedicado a aprender a usar NumPy de manera efectiva será valioso sin importar el aspecto de la ciencia de datos que más interese.

Correo: uniic_bog@unal.edu.co
Teléfono: 3165000 **ext** 12301

¿Qué hay en Numpy?

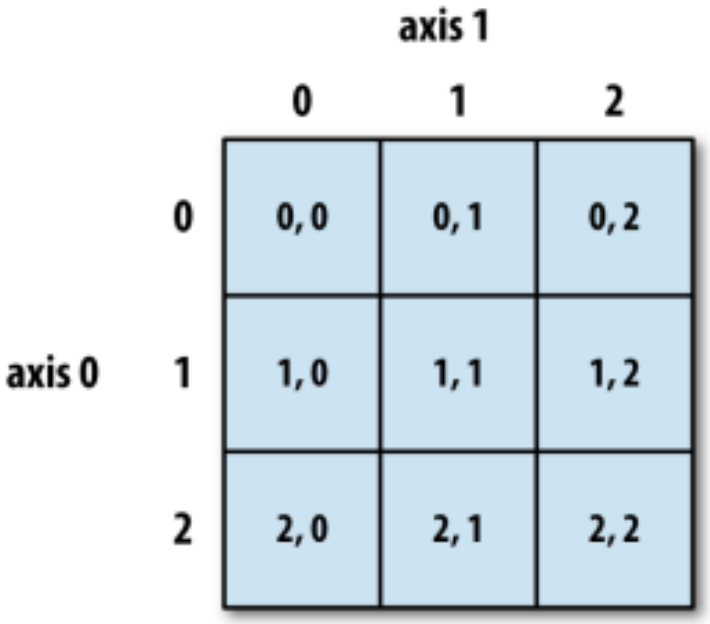
- ndarray, un arreglo multidimensional eficiente que proporciona operaciones aritméticas orientadas a computación vectorizada.
- Funciones matemáticas para operaciones rápidas en arreglos completos de datos sin tener que escribir bucles.
- Herramientas para leer/escribir datos de matriz en el disco y trabajar con archivos mapeados en memoria.
- Capacidades de álgebra lineal, generación de números aleatorios y transformada de Fourier.
- Una API de C para conectar NumPy con bibliotecas escritas en C, C++ o FORTRAN.

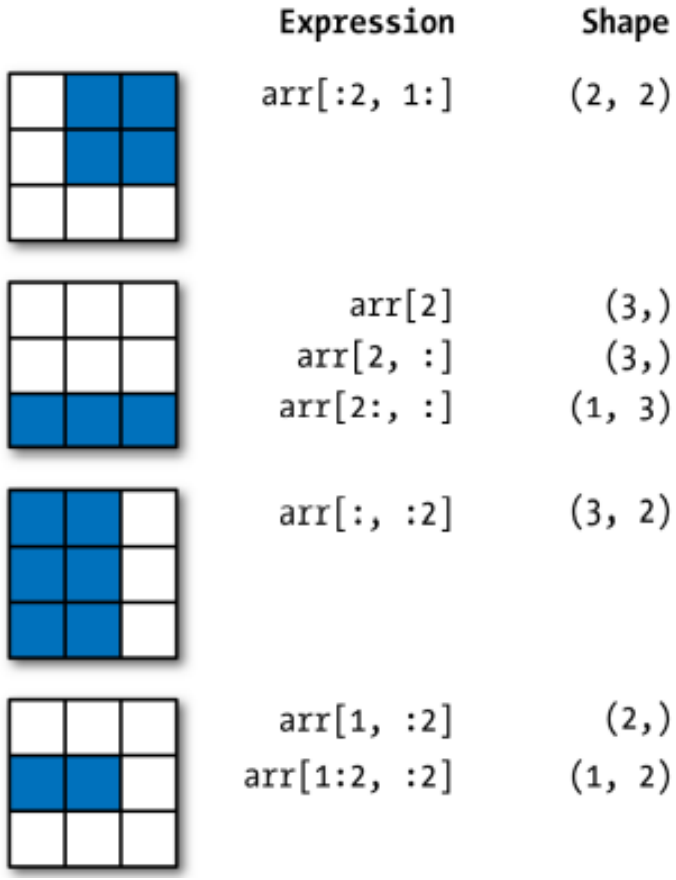
Tipos de datos en Numpy

Las matrices NumPy contienen valores de un solo tipo, por lo que es importante tener un conocimiento detallado de esos tipos y sus limitaciones. Debido a que NumPy está construido en C, los tipos serán familiares para los usuarios de C, Fortran y otros lenguajes relacionados.

Data type	Description
`bool_`	Boolean (True or False) stored as a byte
`int_`	Default integer type (same as C `long`; normally either `int64` or `int32`)
`intc`	Identical to C `int` (normally `int32` or `int64`)
`intp`	Integer used for indexing (same as C `ssize_t`; normally either `int32` or `int64`)
`int8`	Byte (–128 to 127)
`int16`	Integer (–32768 to 32767)
`int32`	Integer (–2147483648 to 2147483647)
`int64`	Integer (–9223372036854775808 to 9223372036854775807)
`uint8`	Unsigned integer (0 to 255)
`uint16`	Unsigned integer (0 to 65535)
`uint32`	Unsigned integer (0 to 4294967295)
`uint64`	Unsigned integer (0 to 18446744073709551615)
`float_`	Shorthand for `float64`
`float16`	Half-precision float: sign bit, 5 bits exponent, 10 bits mantissa
`float32`	Single-precision float: sign bit, 8 bits exponent, 23 bits mantissa
`float64`	Double-precision float: sign bit, 11 bits exponent, 52 bits mantissa
`complex_`	Shorthand for `complex128`
`complex64`	Complex number, represented by two 32-bit floats
`complex128`	Complex number, represented by two 64-bit floats

Indexación





Correo: uniic_bog@unal.edu.co
Teléfono: 3165000 **ext** 12301