Recibe una cálida:

¡Bienvenida!

Te estábamos esperando 😁

alkemy

Clase grabada

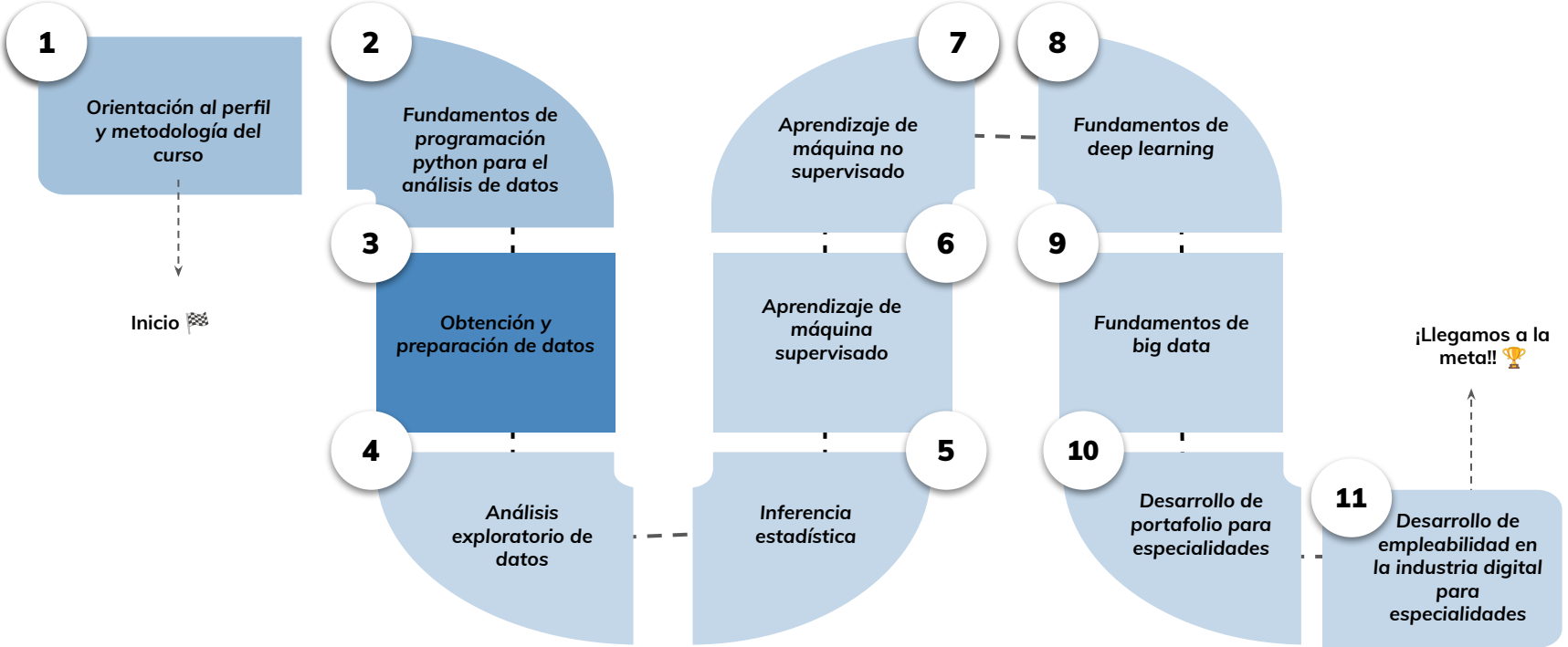
La librería numpy - Parte I

Aprendizaje Esperado: Manipular estructuras de datos vectoriales y matriciales utilizando biblioteca numpy para resolver un problema.

alkemy

Hoja de ruta

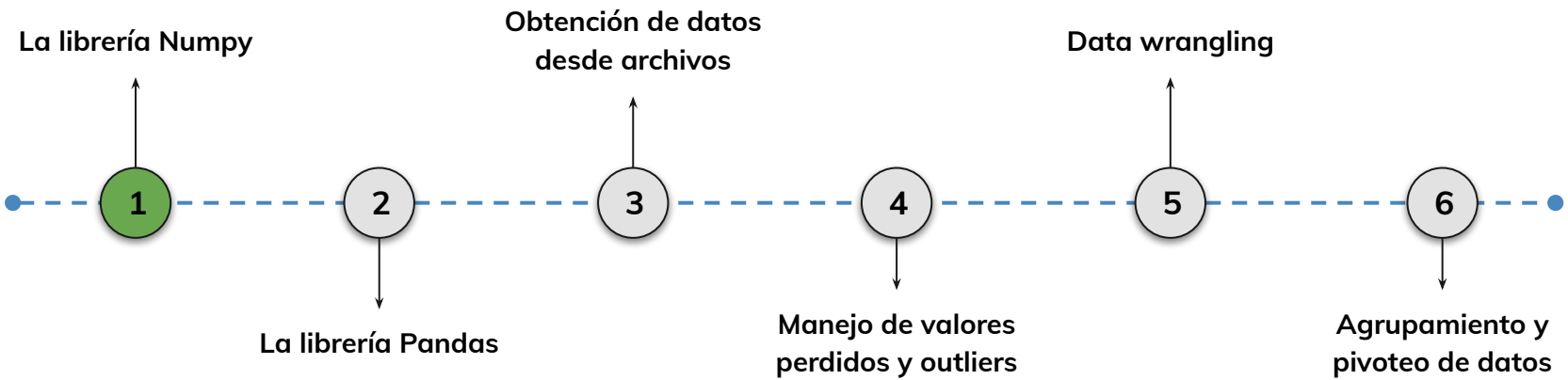
¿Cuáles skills conforman el programa? **Fundamentos de Ciencia de Datos**



alkemy

Roadmap de lecciones

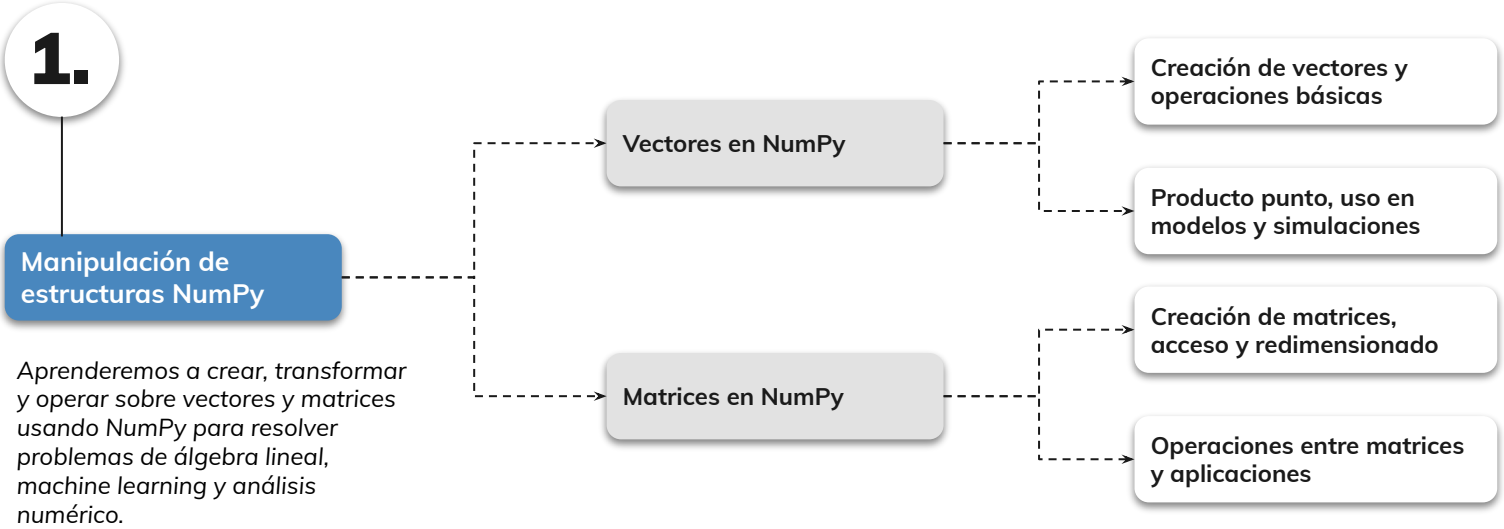
*¿Cuáles **lecciones** estaremos estudiando en este módulo?*



alkemy

Learning Path

¿Cuáles temas trabajaremos hoy?



alkemy

Objetivos de aprendizaje

¿Qué aprenderás?

- Manipular vectores y matrices con NumPy en Python
- Realizar operaciones matemáticas vectorizadas
- Crear arreglos con funciones como arange, zeros, ones, eye y random
- Aplicar funciones como reshape, dot, y linspace en problemas reales
- Entender la aplicación de NumPy en áreas como machine learning, álgebra lineal y visualización de datos

Repaso clase anterior

¿Quedó alguna duda?

En la clase anterior trabajamos 📚:

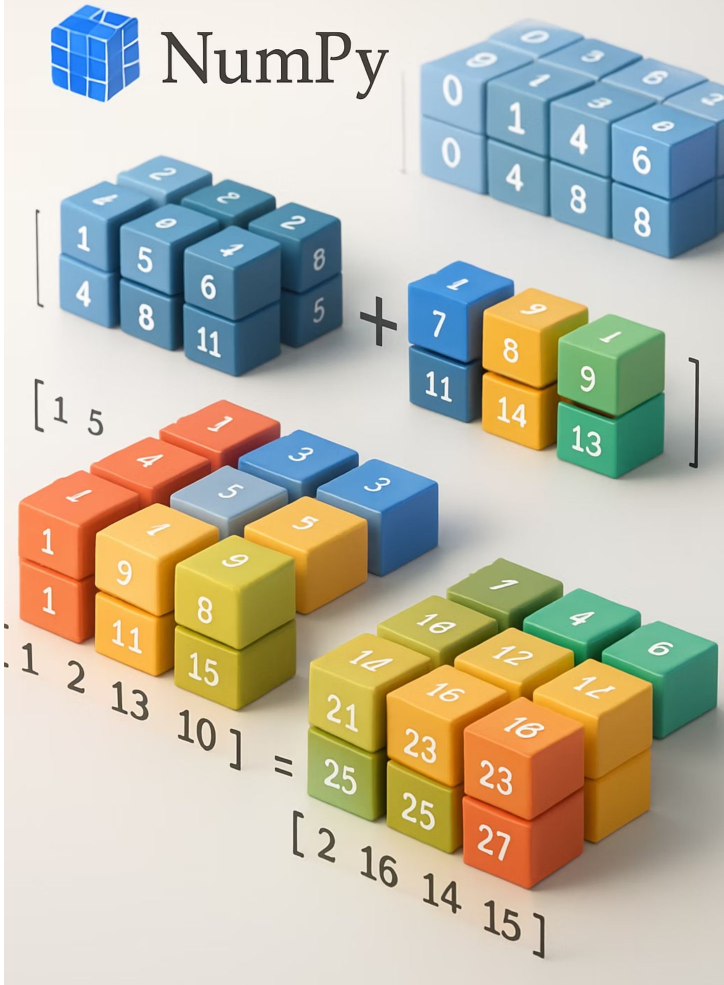
- → *Profundizamos en herencia, encapsulamiento y polimorfismo en Python*
- → *Implementamos clases con atributos privados y uso de @property*
- → *Aplicamos métodos especiales como __str__() y __add__()*
- → *Creamos clases abstractas y demostramos su utilidad como contrato*
- → *Modelamos jerarquías de empleados utilizando herencia múltiple y rutinas reutilizables*

La librería numpy

alkemy

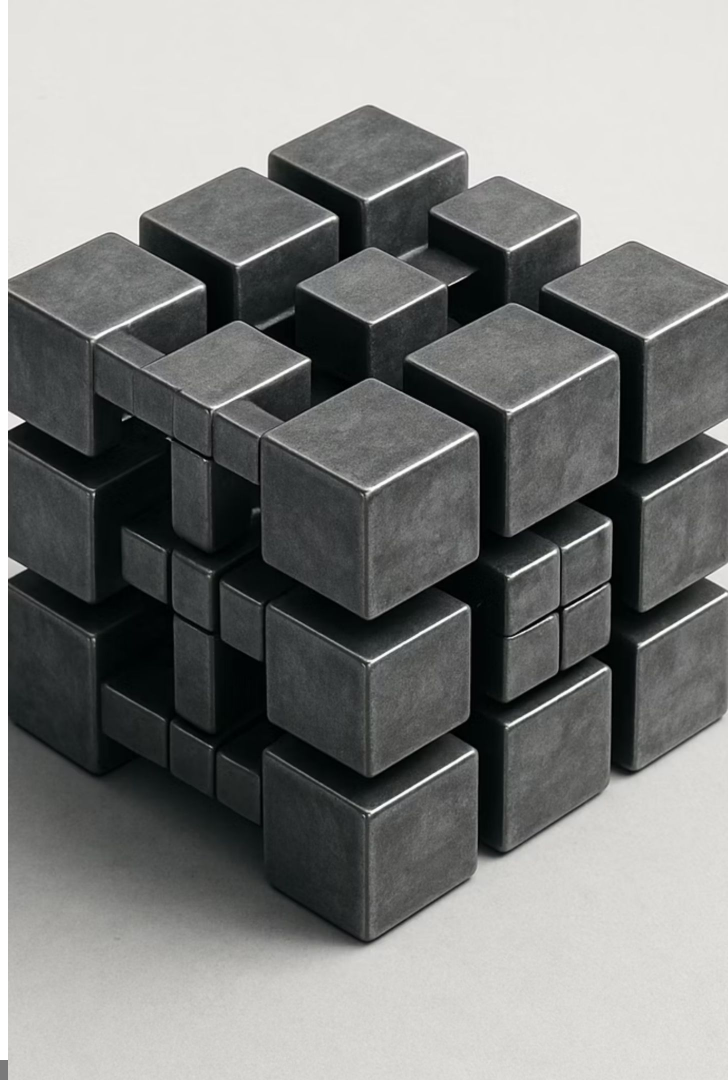
LA LIBRERÍA NUMPY

NumPy es una biblioteca esencial para el manejo eficiente de datos numéricos en Python. Su estructura optimizada y su capacidad para realizar operaciones matemáticas de manera eficiente la convierten en una herramienta indispensable en campos como la ciencia de datos, la inteligencia artificial y la computación científica.



Reseña de la Librería NumPy

NumPy (Numerical Python) es una de las bibliotecas más utilizadas en Python para el cálculo numérico. Proporciona una estructura de datos eficiente para manejar grandes volúmenes de datos numéricos en forma de arreglos multidimensionales y matrices. Su uso es fundamental en campos como la inteligencia artificial, la ciencia de datos y la estadística.



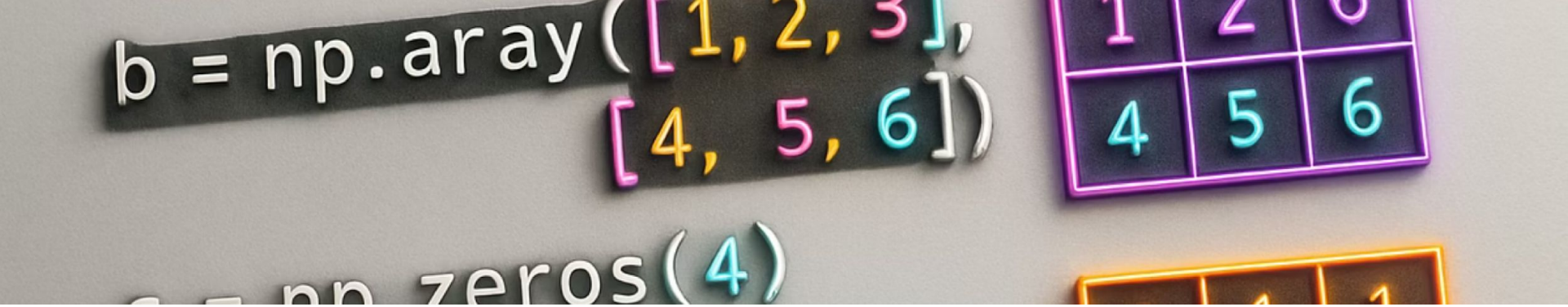
Características Principales de NumPy

Integración con Otras Bibliotecas

Una característica clave de NumPy es su integración con otras bibliotecas populares como Pandas, Matplotlib y Scikit-learn. Estas librerías dependen de NumPy para manejar datos de manera eficiente, lo que lo convierte en un pilar fundamental dentro del ecosistema de ciencia de datos en Python.

Operaciones Vectorizadas

Otra ventaja de NumPy es su capacidad para manejar operaciones vectorizadas. Esto significa que se pueden realizar cálculos matemáticos sobre conjuntos de datos completos sin necesidad de escribir bucles, lo que mejora la legibilidad del código y la eficiencia computacional.



Creación de Arreglos de NumPy

NumPy ofrece diversas formas de crear arreglos, desde vectores unidimensionales hasta matrices multidimensionales, con diferentes tipos de datos y estructuras.

Vectores en NumPy

Los vectores en NumPy son arreglos unidimensionales que pueden almacenar múltiples valores numéricos. Se pueden definir utilizando la función numpy.array() a partir de listas de Python.

Un vector puede ser de cualquier tipo de dato numérico, como enteros, flotantes o booleanos. NumPy permite la conversión automática del tipo de datos para optimizar la memoria utilizada en la ejecución de cálculos.

Aplicaciones de Vectores

Álgebra Lineal

Los vectores son especialmente útiles en el álgebra lineal, donde representan coordenadas, velocidades y fuerzas en el espacio.

Modelos Matemáticos

También se utilizan en modelos matemáticos y de machine learning.

Operaciones Eficientes

Podemos realizar operaciones matemáticas directamente sobre vectores sin necesidad de escribir bucles.

Operaciones con Vectores

También se pueden realizar operaciones entre dos vectores, como la suma, la resta y la multiplicación elemento a elemento.

```
import numpy as np
vector = np.array([1, 2, 3, 4, 5])
print(vector * 2)  # Multiplica cada elemento por 2
```

Operaciones entre Vectores

```
vector1 = np.array([1, 2, 3])
vector2 = np.array([4, 5, 6])
suma = vector1 + vector2
print(suma)  # Output: [5 7 9]
```

El producto punto entre dos vectores es una operación común en machine learning y álgebra lineal, y se puede calcular fácilmente con np.dot().

Matrices en NumPy

Definición

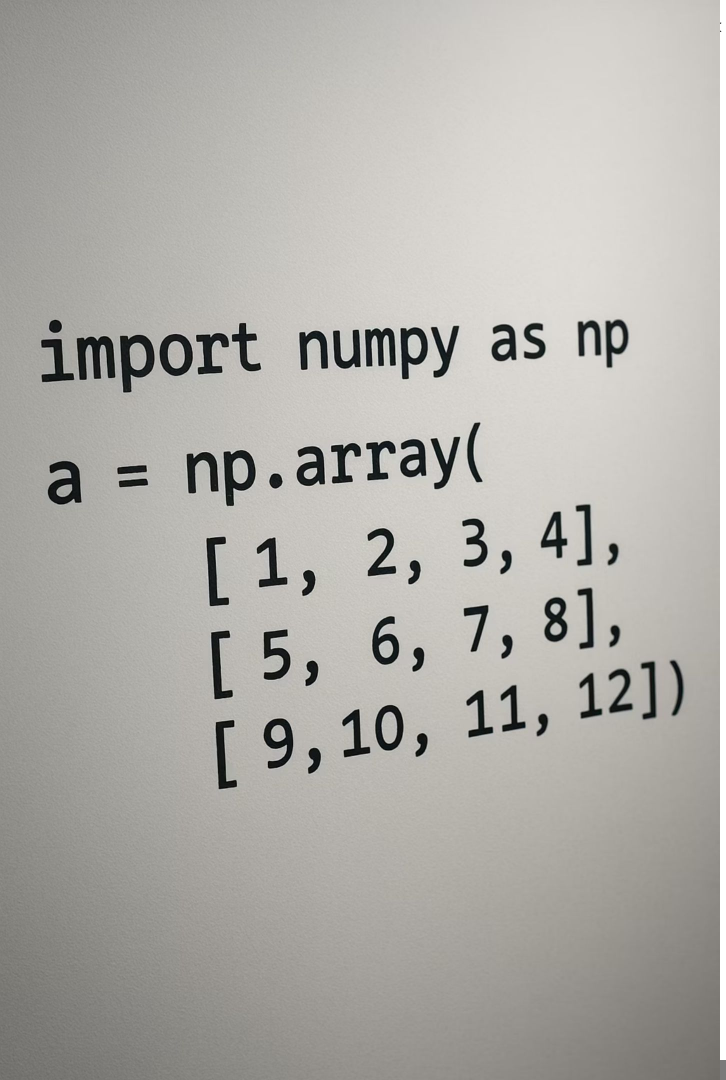
Las matrices en NumPy son arreglos bidimensionales que almacenan datos en filas y columnas. Se pueden definir como listas de listas dentro de numpy.array().

Aplicaciones

Son esenciales en muchas áreas, como la estadística, el procesamiento de imágenes y la simulación de datos.

Eficiencia

Las matrices permiten la aplicación de múltiples operaciones matemáticas de manera eficiente y rápida.



Creación de Matrices

Podemos crear una matriz de la siguiente forma:

```
import numpy as np

# Crear una matriz 3x3
matrix = np.array([
 [1, 2, 3],
 [4, 5, 6],
 [7, 8, 9]
])
```

alkemy

Acceso a Elementos de una Matriz

Para acceder a un elemento dentro de la matriz, utilizamos índices.

valor = matrix[1, 2] # Elemento en la fila 1, columna 2 print(value) # Output: 6

Operaciones con Matrices

```
matriz_cuadrada = matriz ** 2
print(matriz_cuadrada)
```

Es posible realizar operaciones matemáticas directamente sobre matrices sin necesidad de iteraciones.

Multiplicación de Matrices

Las matrices también pueden multiplicarse entre sí utilizando la función np.dot() o el operador @.

```
matriz2 = np.array([[1, 0], [0, 1], [1, 1]])
resultado = np.dot(matriz, matriz2)
print(resultado)
```

Matrices en Machine Learning

Conjuntos de Datos

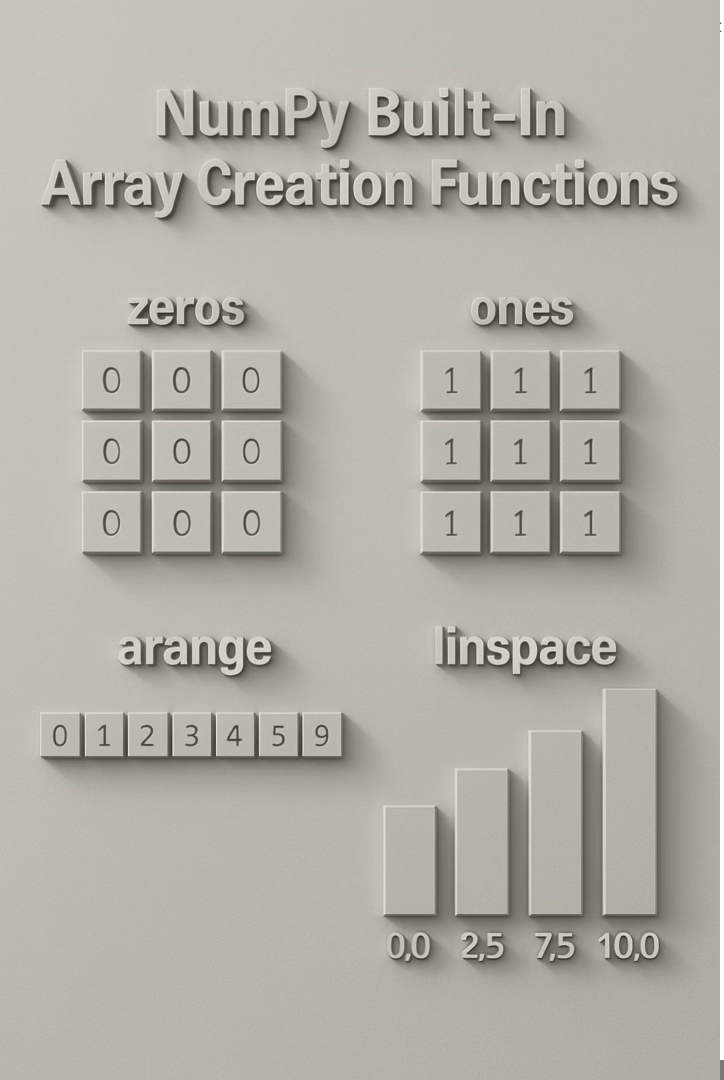
En machine learning, las matrices se utilizan para representar conjuntos de datos estructurados.

Imágenes

Las imágenes digitales se representan como matrices de píxeles con valores de intensidad.

Redes Neuronales

Las operaciones en redes neuronales se realizan mediante multiplicaciones de matrices.



Funciones Preconstruidas de Creación

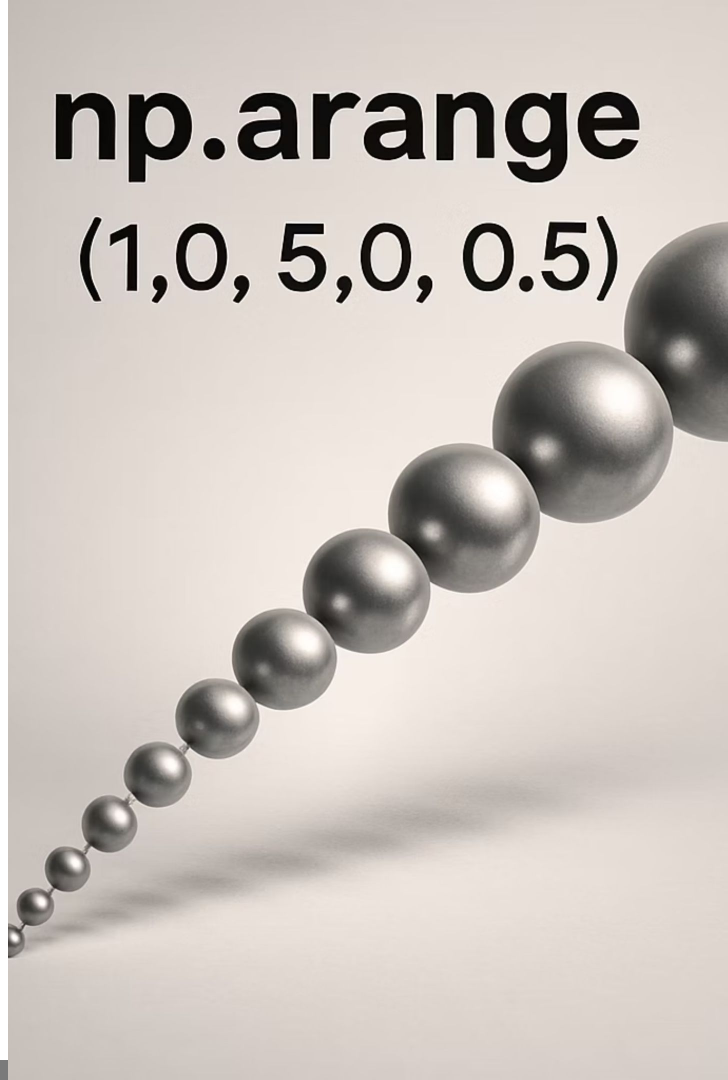
NumPy ofrece diversas funciones para crear arreglos con patrones específicos, facilitando la inicialización de estructuras de datos para diferentes aplicaciones.

alkemy

Arreglo con Valores (arange)

La función np.arange() genera secuencias de números con intervalos definidos. Es una alternativa eficiente a la función range() de Python.

```
secuencia = np.arange(0, 10, 2)  # Values de 0 a 10
print(secuencia)
```



Aplicaciones de np.arange()

Generación de Datos

Esta función es especialmente útil en la generación de datos de prueba para algoritmos y modelos.

Simulaciones

Se utiliza en simulaciones matemáticas donde se necesitan secuencias de valores con intervalos regulares.

Visualización

Facilita la creación de ejes para gráficos y visualizaciones de datos.

Matrices de Ceros y Unos

NumPy permite la creación de matrices de ceros y unos mediante np.zeros() y np.ones(), respectivamente.

```
# Matriz de ceros 3x3
zeros_matrix = np.zeros((3, 3))

# Matriz de unos 2x4
ones_matrix = np.ones((2, 4))
```

Aplicaciones de Matrices de Ceros y Unos

Inicialización

Estas matrices se utilizan frecuentemente en la inicialización de pesos en redes neuronales.

Algoritmos de Optimización

Son útiles en algoritmos de optimización donde se necesita partir de valores conocidos.

Máscaras

Las matrices de ceros y unos pueden servir como máscaras para operaciones de filtrado en procesamiento de imágenes.

Vector con Distribución de Puntos

La función linspace() genera un vector con valores equidistantes en un rango dado.

```
# 5 puntos equidistantes entre 0 y 1
points = np.linspace(0, 1, 5)
# Output: [0.   0.25 0.5  0.75 1.  ]
```

Aplicaciones de linspace()

Gráficos

Este tipo de vector es utilizado en la generación de datos para gráficos con puntos equidistantes en los ejes.

Modelos Matemáticos

Es útil en modelos matemáticos donde se necesita evaluar funciones en intervalos regulares.

Interpolación

Facilita la interpolación de valores entre puntos conocidos en análisis numérico.

Matriz Identidad

La matriz identidad es un concepto clave en álgebra lineal y se puede crear con np.eye().

```
# Matriz identidad 3x3
identity = np.eye(3)
# Output:
# [[1. 0. 0.]
#  [0. 1. 0.]
#  [0. 0. 1.]]
```

Aplicaciones de la Matriz Identidad

Transformaciones Lineales

La matriz identidad representa la transformación que no altera un vector, manteniendo su dirección y magnitud.

Sistemas de Ecuaciones

Es fundamental en la resolución de sistemas de ecuaciones lineales y en la inversión de matrices.

Regularización

Se utiliza en técnicas de regularización en machine learning para estabilizar modelos.

Matriz Aleatoria

Las matrices aleatorias se generan con np.random.rand() o np.random.randint(), dependiendo de si se necesitan valores flotantes o enteros.

```
matriz_aleatoria = np.random.rand(3, 3)
print(matriz_aleatoria)
```

Aplicaciones de Matrices Aleatorias

Inicialización de Modelos

Las matrices aleatorias son cruciales para inicializar pesos en redes neuronales y evitar la simetría.

Simulaciones

Se utilizan en simulaciones de Monte Carlo y otros métodos estocásticos.

Pruebas

Son útiles para generar datos de prueba aleatorios en el desarrollo de algoritmos.

Redimensionado de un Arreglo

La función reshape() permite cambiar la estructura de un arreglo sin modificar sus valores.

```
# Vector de 12 elementos
arr = np.arange(12)
# Redimensionar a matriz 3x4
reshaped = arr.reshape(3, 4)
```

Aplicaciones del Redimensionado

Preparación de Datos

El redimensionado es esencial en la preparación de datos para algoritmos de machine learning.

Procesamiento de Imágenes

Permite transformar imágenes entre diferentes representaciones (plana vs. matricial).

Reestructuración de Datos

Facilita la reorganización de datos para análisis y visualización.

Live Coding

¿En qué consistirá la Demo?

Vamos a trabajar con funciones de creación avanzada de arreglos NumPy y realizar transformaciones estadísticas básicas sobre los datos simulados. Esto permitirá comprender cómo se puede simular un conjunto de datos y procesarlo para análisis exploratorio, sin necesidad de bucles ni librerías adicionales.

1. *Importar NumPy*
2. *Crear un conjunto de datos aleatorios con np.random.rand()*
3. *Calcular la media por columna (promedio por característica)*
4. *Calcular la desviación estándar por columna*
5. *Estandarizar los datos (Z-score normalization)*
6. *Filtrar filas donde la primera característica sea mayor al promedio*
7. *Agregar una columna con el promedio por fila (nueva característica)*
8. *Obtener la matriz de correlación entre columnas*

Tiempo: 20 minutos

#Momentode Preguntas...

¿?

¿Para qué sirve la función reshape y en qué casos se usa?

¿?

¿Qué diferencia hay entre @ y np.dot()?

¿?

¿Qué ventajas tiene NumPy frente a listas tradicionales en Python?

¿?

¿Cómo afecta el uso de operaciones vectorizadas al rendimiento del código?

Momento:

Time-out!

⏰5 -10 min.

Ejercicio N° 1

Manipulando datos con NumPy sin bucles

Manipulando datos con NumPy sin bucles

Contexto: 🙌

Una empresa de análisis de datos necesita generar estructuras de prueba para un nuevo algoritmo de predicción. Para ello, debe crear arreglos numéricos representando variables simuladas y aplicar transformaciones sin utilizar bucles for, optimizando el rendimiento.

Consigna: ✍️

Utilizando la librería NumPy, crea un vector con valores del 0 al 99. A partir de él, genera una matriz de 10x10. Luego:

- Calcula su transpuesta.
- Multiplica cada valor por 2 sin utilizar bucles.
- Obtén la diagonal principal.
- Crea una matriz identidad del mismo tamaño.
- Realiza una multiplicación matricial con @ entre la original y la identidad.

Paso a paso: ⚙️

- Importar la librería NumPy.
- Generar el vector con np.arange(100).
- Transformarlo en matriz con .reshape(10, 10).
- Realizar la transpuesta con .T.
- Multiplicar por 2 con operaciones vectorizadas.
- Extraer la diagonal con np.diag().
- Crear la identidad con np.eye(10).
- Multiplicar matrices con @.

Tiempo 🕛**:** *45 minutos*

alkemy

¿Alguna consulta?

alkemy

Resumen

¿Qué logramos en esta clase?

- ✓ ***Entendimos qué es NumPy y por qué es fundamental***
- ✓ ***Creamos vectores y matrices utilizando NumPy***
- ✓ ***Aplicamos operaciones matemáticas sin bucles***
- ✓ ***Usamos funciones como reshape, dot, eye, linspace, arange, etc.***
- ✓ ***Exploramos aplicaciones reales en machine learning y álgebra lineal***

¡Ponte a prueba!

Momento de ejercitación

Te invitamos a aprovechar esta última sección del espacio sincrónico para realizar de manera individual las ***actividades disponibles en la plataforma.*** *Estas propuestas son clave para afianzar lo trabajado y* ***forman parte obligatoria del recorrido de aprendizaje.***

💪 Análisis de caso ——— 💪 Selección Múltiple

💪 Comprensión lectora

Si al resolverlas surge alguna duda, compartela o tráela al próximo encuentro sincrónico.

alkemy

¡Muchas gracias!

alkemy