

# Aula 2 - NumPy Básico

Arrays e operações matemáticas

# Objetivos

Introduzir  
NumPy

Criar arrays

Operações  
matemáticas

# Aplicação

---

Cálculo de médias de temperatura em secadores de grãos

## Exercício

- 
- Use NumPy para calcular média e desvio padrão de temperaturas.

# Ambientes de Desenvolvimento Recomendados

## VISUAL STUDIO CODE (VS CODE)

- Editor de código versátil e leve, muito usado para programação em Python.
- Suporta extensões específicas para desenvolvimento científico, como Jupyter Notebooks integrados.
- Permite edição de código modular, controle de versão integrado e depuração avançada.
- Ideal para projetos locais e desenvolvimento offline com ambiente configurável.

## GOOGLE COLAB

- Ambiente baseado na nuvem para execução interativa de notebooks Jupyter.
- Não requer instalação local; acessível via navegador.
- Permite uso gratuito de GPUs para computação acelerada.
- Facilita compartilhamento e colaboração em tempo real

# Por que Python em Ciência e Dados?



**Facilidade e simplicidade:** Python possui sintaxe clara e fácil de aprender, ideal para estudantes com conhecimento básico em programação.



**Ampla comunidade e recursos:** Grande quantidade de bibliotecas específicas para ciência de dados, como pandas, NumPy, Matplotlib, Scikit-Learn e TensorFlow.



**Versatilidade:** Pode ser usado desde análise exploratória e visualização até modelagem estatística e aprendizado de máquina.



**Integração com diversas ferramentas:** Funciona bem com ambientes como Jupyter, VS Code, Google Colab, facilitando o desenvolvimento e a colaboração.



**Performance adequada:** Com bibliotecas otimizadas, Python oferece bom desempenho para grandes volumes de dados e cálculos complexos.

# O que é NumPy?

---

## O que é?

Biblioteca fundamental para computação numérica em Python, focada em arrays multidimensionais.

---

## Estrutura principal:

Arrays Numpy, que são vetores/matrizes eficientes para armazenar dados homogêneos.

---

## Vantagens:

Operações matemáticas vetorizadas (rápidas e concisas), consumo menor de memória comparado aos listas Python.

---

## Funcionalidades principais:

Criação, manipulação e operações aritméticas com arrays; funções para álgebra linear, estatística e geração de números aleatórios.

---

## Exemplo prático:

Criação de array, soma de vetores, multiplicação elemento a elemento, etc.

# Criando arrays

- **Função principal:** `np.array()` cria arrays a partir de listas ou tuplas Python.
- **Exemplos básicos:**
  - 1D: `np.array([1, 2, 3])` — vetor unidimensional
  - 2D: `np.array([[1, 2], [3, 4]])` — matriz bidimensional
- **Arrays especiais:**
  - `np.zeros(shape)` cria array cheio de zeros
  - `np.ones(shape)` cria array cheio de uns
  - `np.arange(start, stop, step)` cria array com valores em sequência
  - `np.linspace(start, stop, num)` cria array com valores igualmente espaçados
- **Importância:** arrays NumPy são eficientes para armazenar e manipular dados numéricos em ciência de dados.

# Operações vetorizadas

- **O que são?**  
Operações aplicadas diretamente em arrays, elemento a elemento, sem a necessidade de loops explícitos.
- **Vantagens:**  
Código mais simples, legível e com grande ganho de desempenho devido à execução otimizada.
- **Exemplos comuns:**
  - Soma, subtração, multiplicação e divisão entre arrays
  - Operações escalares (ex: multiplicar todos os elementos por um número)
  - Funções universais (ufuncs) como `np.sin()`, `np.exp()`, `np.sqrt()`



# Exemplo prático: dados de umidade do solo

```
import numpy as np

# Dados simulados de umidade do solo (%) em diferentes pontos de uma
plantação
umidade_manha = np.array([30, 45, 50, 40, 35])
umidade_tarde = np.array([35, 50, 55, 45, 38])

# Cálculo da variação de umidade entre manhã e tarde (elemento a
elemento)
variacao_umidade = umidade_tarde - umidade_manha

# Multiplicação por um fator de correção (vetorizada)
correcao = variacao_umidade * 1.1

print("Variação de umidade:", variacao_umidade)
print("Correção aplicada:", correcao)
```