NumPy: Processamento de Dados Estruturados Multidimensionais

M6 - Python para processamento de dados

Tutoria 16/07/2025

Objetivos

- Entender o que é o NumPy e por que é utilizado
- Aprender a manipular arrays (1D, 2D, 3D)
- Comparar arrays NumPy com listas Python
- Realizar operações matemáticas com arrays
- Utilizar funções universais (ufuncs)
- Gerar gráficos com o Matplotlib
- Praticar com exemplos interativos no PyCharm

Instalação :

- Verificar se todos têm o NumPy e o Matplotlib
- Verificando a instalação do *NumPy*
- Na IDE PyCharm tem 2 maneiras de importar essa biblioteca

prompt

```
pip install numpy matplotlib
```

Python

```
import numpy as np

print("NumPy versão:", np.__version__
```

Como funciona :

- Biblioteca fundamental para computação científica com Python
- Suporte para arrays n-dimensionais e operações vetoriais rápidas
- Base para outras bibliotecas: Pandas, SciPy, Scikit-learn

Python

```python
import numpy as np

# Criando um array simples

a = np.array([1, 2, 3])

print(a) # Saída: [1 2 3]

print(type(a)) # <class 'numpy.ndarray'>
```

Python

```python
a = np.array([1, 2, 3, 4, 5])

print("Array a:", a)

b = a * 2

print("Array b (a multiplicado por 2):", b)
```

Vetores, Matrizes e Tensors :

- Vetor = Array 1D (ex: [1, 2, 3])
- Matriz = Array 2D (ex: [[1,2],[3,4]])
- Tensor = Array 3D+ (ex: imagem colorida)

Python

```python
# Vetor (1D)
vetor = np.array([1, 2, 3])
print("Vetor:", vetor)
print("Shape:", vetor.shape)  # (3,)

# Matriz (2D)
matriz = np.array([[1, 2], [3, 4]])
print("Matriz:", matriz)
print("Shape:", matriz.shape)  # (2, 2)

# Tensor (3D)
tensor = np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]])
print("Tensor:", tensor)
print("Shape:", tensor.shape)  # (2, 2, 2)
```

Operações com Arrays :

- Soma, subtração, multiplicação
- Transposição, fatiamento, indexação
- Operações de arrays e escalar

Python

```python
a = np.array([1, 2, 3])

b = np.array([4, 5, 6])

c = a + b

print("Adição:", c) # Saída esperada: Adição: [5 7 9]

d = b - a

print("Subtração:", d) # Saída esperada: Subtração: [3 3 3]

e = a * b

print("Multiplicação:", e) # Saída esperada: Multiplicação:

[ 4 10 18]

f = b / a

print("Divisão:", f) # Saída esperada: Divisão: [4. 2.5 2. ]
```

Operações com Arrays :

- Soma, subtração, multiplicação
- Transposição, fatiamento, indexação
- Operações de arrays e escalar

Python

```python
arr = np.array([[1, 2, 3], [4, 5, 6]])

print(arr + 10) # Soma escalar

print(arr - 1) # Subtração escalar

print(arr * 2) # Multiplicação escalar

print(arr / 2) # Divisão escalar

print(arr.T) # Transposição

print(arr[0, 1]) # Elemento linha 0, coluna 1

print(arr[:, 1]) # Segunda coluna: [2 5]
```

Funções Universais (ufunc):

- Funções matemáticas aplicadas elemento a elemento
- Exemplos: np.mean, np.sum, np.sqrt, np.exp, np.sin
- np.arange(start, stop, step)
- np.linspace(start, stop, num)

Python

```python
arr = np.array([1, 2, 3, 4, 5])

print("Média:", np.mean(arr))         # 3.0

print("Mediana:", np.median(arr))     # 3.0

print("Desvio padrão:", np.std(arr))  # 1.41...

print(np.zeros((2, 3))) # Matriz 2x3 de zeros

print(np.ones((3, 2))) # Matriz 3x2 de uns

print(np.arange(0, 10, 2)) # [0 2 4 6 8]

print(np.linspace(0, 1, 5)) # [0.   0.25 0.5  0.75 1. ]

m1 = np.array([[1, 2], [3, 4]])

m2 = np.array([[5, 6], [7, 8]])

print(np.dot(m1, m2)) # Produto matricial

print(np.linalg.det(m1))  # Determinante

print(np.linalg.eigvals(m1))  # Autovalores

print(m1.shape, m1.ndim, m1.dtype) # Forma, dimensões e tipo dos dados
```

Indexação e Slicing :

- Técnicas para acessar e manipular partes do array

Python

```python
arr = np.array([10, 20, 30, 40, 50])

print(arr[1:4]) # Slicing: [20 30 40]

print(arr[::2]) # Slicing com step: [10 30 50]

print(arr[[0, 2, 4]]) # Indexação por lista: [10 30 50]

print(arr[arr > 30]) # Indexação booleana: [40 50]

cond = np.where(arr > 30, 1, 0)  # Condição booleana

print(cond)  # [0 0 0 1 1]
```

Visualização com Matplotlib :

- Permite criar gráficos simples e avançados
- Importante para exploração visual de dados

Python

```python
import matplotlib.pyplot as plt

x = np.linspace(0, 10, 100)  # Gera 100 valores

igualmente espaçados entre 0 e 10

y = np.sin(x) # Aplica a função seno a cada valor de x

plt.plot(x, y, label="seno") # Plota linha

plt.title("Funço gráfico

plt.xlabel("x") # Rótulo do eixo x

plt.ylabel("sen(x)") # Rótulo do eixo y

plt.grid(True) # Adiciona grade

plt.legend() # Mostra legenda

plt.show() # Exibe o gráfico
```

Desafio Prático :

- Calcular médias por linha
- Mostrar gráfico de barras

Documentação: https://numpy-org.translate.goog/devdocs/user/absolute_beginners.html?_x_tr_sl=en&_x_tr_tl=pt&_x_tr_hl=pt&_x_tr_pto=tc

Obrigada