Vetores e Matrizes em Python Puro:

Dominando Arrays Multidimensionais sem Bibliotecas Externas

Uma introdução didática para iniciantes e estudantes de programação.

Por que Aprender Sem NumPy?

Fundamentos Sólidos

Compreender a lógica por trás da manipulação de dados é crucial. Aprender "na unha" fortalece seu entendimento de estruturas de dados e algoritmos.

Independência

Desenvolva soluções sem depender de bibliotecas externas, o que é útil em ambientes restritos ou para depurar problemas em nível fundamental.

Base para o Futuro

Entender como NumPy e outras bibliotecas funcionam internamente. Você saberá o "porquê" das coisas e poderá usá-las de forma mais eficiente.

Lógica de Programação

Exercita o pensamento lógico e a resolução de problemas, habilidades transferíveis para qualquer linguagem ou framework.

O que são Vetores em Python?

Em Python puro, um vetor é simplesmente uma lista **unidimensional**. É uma coleção ordenada e mutável de itens que podem ser de diferentes tipos, mas para fins de vetores e matrizes, geralmente trabalhamos com tipos numéricos homogêneos.

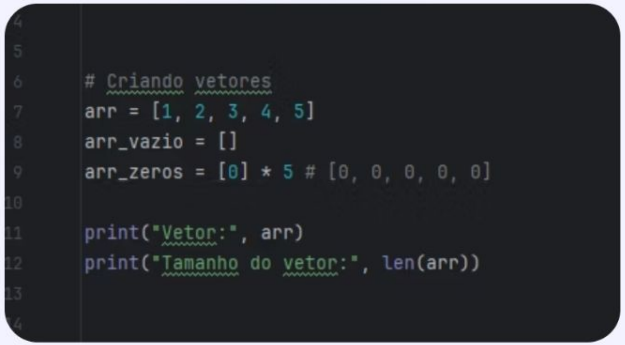
São a estrutura mais básica para armazenar sequências de dados e realizar operações nelas.

```
# Criando vetores
vetor = [1, 2, 3, 4, 5]
vetor_vazio = []
vetor_zeros = [0] * 5 # [0, 0, 0, 0, 0]

print("Vetor:", vetor)
print("Tamanho do vetor:", len(vetor))
```

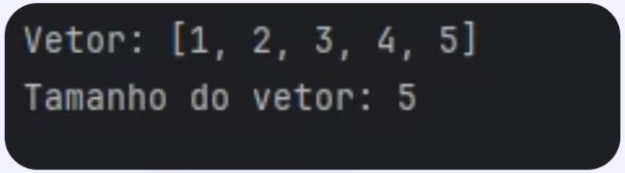
Saída:

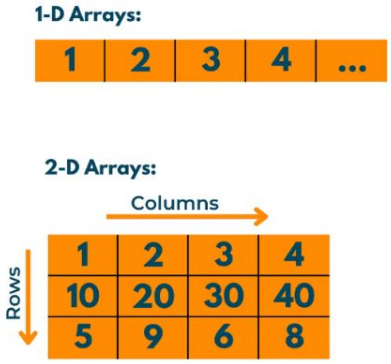
Vetor: [1, 2, 3, 4, 5]

Tamanho do vetor: 5



Visualização de Vetor





Operações Básicas com Vetores

Mesmo sem bibliotecas, podemos realizar operações comuns em vetores usando compreensões de lista e a função `zip()`. Isso nos permite aplicar transformações ou combinar vetores elemento a elemento.

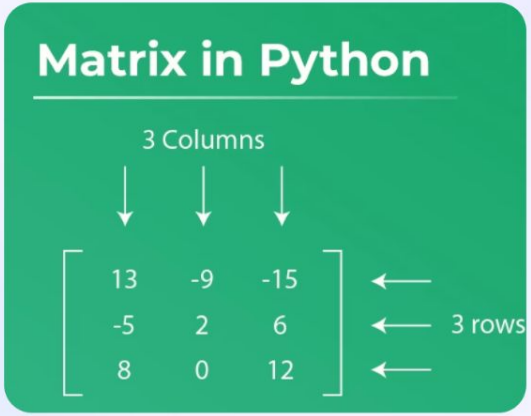
```
vetor_a = [1, 2, 3]
vetor_b = [4, 5, 6]

# Adição elemento a elemento
soma = [a + b for a, b in zip(vetor_a, vetor_b)]
print("Soma:", soma) # [5, 7, 9]

# Multiplicação por escalar
multiplicado = [x * 2 for x in vetor_a]
print("Multiplicado por 2:", multiplicado) # [2, 4, 6]
```

Saída:

Soma: [5, 7, 9]

Multiplicado por 2: [2, 4, 6]

O que são Matrizes em Python?



Visualização de Matriz 3x3

Uma matriz em Python puro é implementada como uma **lista de listas**, ou seja, uma estrutura bidimensional. Cada sublista representa uma linha da matriz.

É fundamental que todas as sublistas (linhas) tenham o mesmo número de elementos (colunas) para formar uma matriz retangular válida.

```
# Criando matrizes
matriz_2x2 = [[1, 2], [3, 4]]
matriz_3x3 = [[1, 2, 3],
              [4, 5, 6],
              [7, 8, 9]]
matriz_zeros = [[0] * 3 for _ in range(3)] # 3x3 de zeros

print("Matriz 3x3:")
for linha in matriz_3x3:
    print(linha)
```

Saída:

Matriz 3x3:
[1, 2, 3]
[4, 5, 6]
[7, 8, 9]

Acessando Elementos em Matrizes

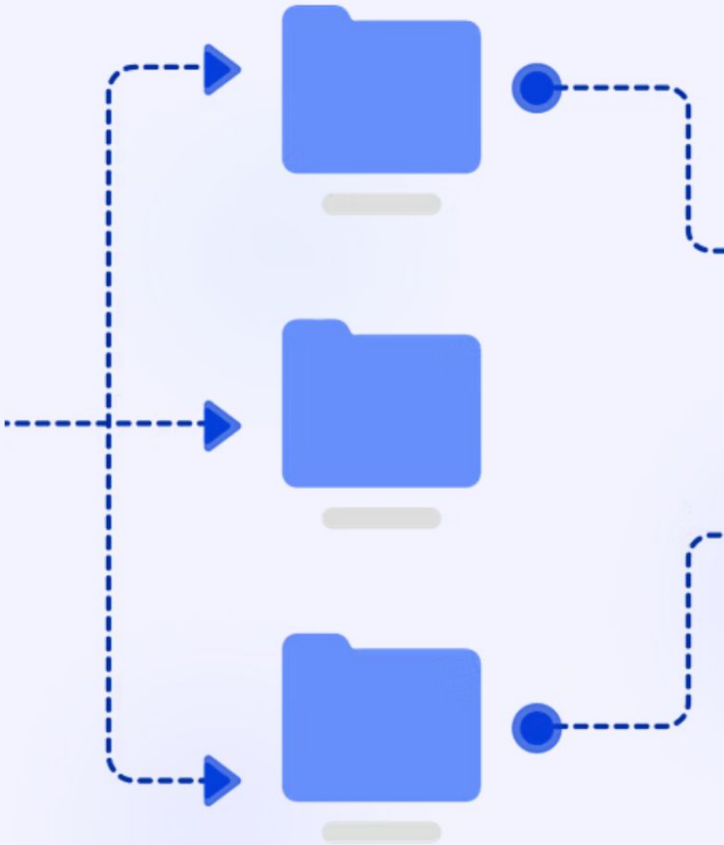
O acesso aos elementos de uma matriz segue a lógica de listas aninhadas: primeiro o índice da linha, depois o índice da coluna. Python utiliza indexação baseada em zero.

Podemos acessar elementos individuais ou linhas inteiras da matriz.

```
matriz = [[1, 2, 3],
          [4, 5, 6],
          [7, 8, 9]]

# Acessando elementos
primeiro_elemento = matriz[0][0] # 1
elemento_central = matriz[1][1] # 5
ultimo_elemento = matriz[2][2] # 9

# Acessando linhas inteiras
primeira_linha = matriz[0] # [1, 2, 3]
segunda_linha = matriz[1] # [4, 5, 6]

print("Primeiro elemento:", primeiro_elemento)
print("Elemento central:", elemento_central)
print("Último elemento:", ultimo_elemento)
print("Primeira linha:", primeira_linha)
```

Saída:

Primeiro elemento: 1
Elemento central: 5
Último elemento: 9
Primeira linha: [1, 2, 3]



Função para Visualizar Matrizes

Para facilitar a leitura e depuração, é útil ter uma função que imprima a matriz de forma formatada, alinhando os elementos e destacando as linhas.

```
def print_matriz(matriz):
    """Função para imprimir matriz de forma formatada"""
    for linha in matriz:
        print("[" + " ".join(f"{elem:3d}" for elem in linha) + "]")

# Exemplo de uso
matriz_exemplo = [[1, 2, 3],
                  [4, 5, 6],
                  [7, 8, 9]]
print_matriz(matriz_exemplo)
```

Saída:

[1 2 3]
[4 5 6]
[7 8 9]

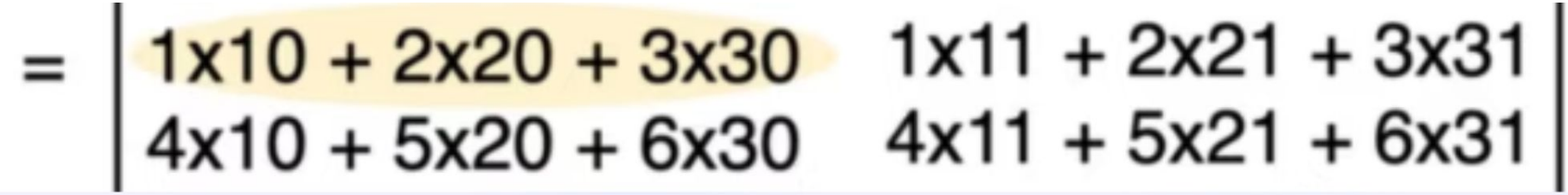
Operações com Matrizes: Soma

A soma de duas matrizes é realizada somando-se os elementos correspondentes em cada posição. É fundamental que as matrizes tenham as mesmas dimensões para que a operação seja válida. Caso contrário, um erro deve ser gerado.

```
def soma_matrizes(A, B):
    """Soma duas matrizes de mesma dimensão"""
    if len(A) != len(B) or len(A[0]) != len(B[0]):
        raise ValueError("Matrizes devem ter mesma dimensão")

    resultado = []
    for i in range(len(A)):
        linha = []
        for j in range(len(A[0])):
            linha.append(A[i][j] + B[i][j])
        resultado.append(linha)
    return resultado

# Exemplo
A = [[1, 2], [3, 4]]
B = [[5, 6], [7, 8]]
C = soma_matrizes(A, B) # [[6, 8], [10, 12]]
print_matriz(C) # Usando a função do slide anterior
```

Saída:

[6 8]
[10 12]



Multiplicação de Matriz por Escalar

Multiplicar uma matriz por um escalar significa multiplicar cada elemento da matriz por esse valor escalar. Esta operação não altera a dimensão da matriz.

```
def multiplica_escalar(matriz, escalar):
    """Multiplica matriz por escalar"""
    resultado = []
    for linha in matriz:
        nova_linha = [elemento * escalar for elemento in linha]
        resultado.append(nova_linha)
    return resultado

# Exemplo
matriz_original = [[1, 2],
                   [3, 4]]
escalar_valor = 3
resultado = multiplica_escalar(matriz_original, escalar_valor) # [[3, 6], [9, 12]]
print_matriz(resultado) # Usando a função print_matriz
```

Saída:

[3 6]
[9 12]

Multiplicação de Matrizes

A multiplicação de matrizes é uma operação mais complexa, onde o elemento `(i, j)` da matriz resultante é o produto escalar da i-ésima linha da primeira matriz pela j-ésima coluna da segunda matriz.

A condição para multiplicar duas matrizes A (m x n) e B (p x q) é que `n` deve ser igual a `p` (número de colunas de A = número de linhas de B). A matriz resultante terá dimensão `m x q`.

```
def multiplica_matrizes(A, B):
    """Multiplica duas matrizes"""
    if len(A[0]) != len(B):
        raise ValueError("Número de colunas de A deve igual número de linhas de B")

    n_linhas_A = len(A)
    n_colunas_B = len(B[0])
    n_colunas_A = len(A[0]) # ou len(B)

    resultado = [[0] * n_colunas_B for _ in range(n_linhas_A)]

    for i in range(n_linhas_A):
        for j in range(n_colunas_B):
            for k in range(n_colunas_A):
                resultado[i][j] += A[i][k] * B[k][j]
    return resultado

# Exemplo
A = [[1, 2], [3, 4]]
B = [[5, 6], [7, 8]]
C = multiplica_matrizes(A, B) # [[19, 22], [43, 50]]
print_matriz(C) # Usando a função print_matriz
```

Saída:

[19 22]
[43 50]