Aula 02

Trabalhando com Matriz (Lista 2D) no Python

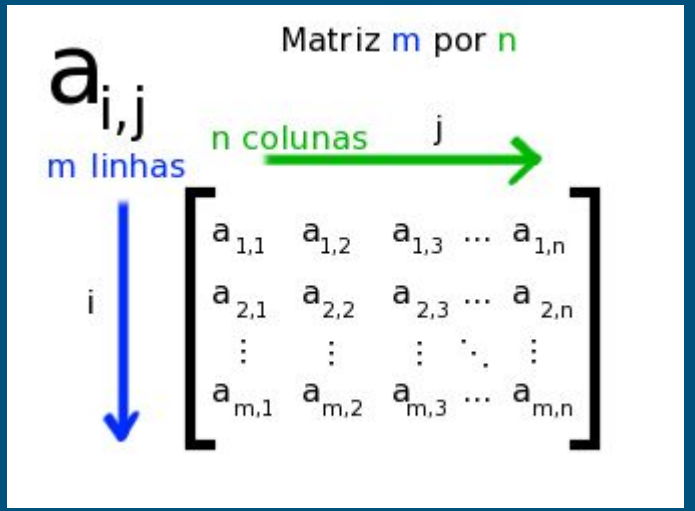
Tópicos

- Introdução às matrizes em Python
- Criação de matrizes utilizando listas aninhadas
- Acesso a elementos de uma matriz
- Operações básicas com matrizes (adição, subtração, multiplicação)
- Iteração sobre elementos de uma matriz
- Indexação e fatiamento de matrizes
- Operações de linha e coluna (adição, remoção, alteração)
- Aplicações práticas de matrizes em problemas do mundo real

Introdução às matrizes em Python

Introdução às matrizes em Python

- Uma matriz é uma estrutura matemática composta por elementos organizados em linhas e colunas. É amplamente utilizada na representação de dados tabulares e multidimensionais em diversos campos, como matemática, ciência da computação, engenharia e estatística. Cada elemento em uma matriz pode ser identificado por sua posição única, definida pela combinação da linha e da coluna em que está localizado. Matrizes são fundamentais em operações algébricas, transformações lineares, análise de sistemas e muito mais.



Introdução às matrizes em Python

- Em Python, uma matriz é frequentemente representada como uma lista de listas, onde cada lista interna representa uma linha da matriz e os elementos são acessados por meio de índices de linha e coluna.

```
[1]  # Criando uma matriz 2x3
     matriz = [[1, 2, 3],
               [4, 5, 6]]
```

```
[3]  # Criando uma matriz 3x3
     matriz = [[1, 2, 3],
               [4, 5, 6],
               [7, 8, 9]]
```

```
[[1, 2, 3], [4, 5, 6]]
```

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

Criação de matrizes utilizando listas aninhadas

Criação de matrizes (lista 2D)

- Em Python, as matrizes, também conhecidas como listas 2D, são criadas utilizando listas aninhadas. Uma lista aninhada é uma lista que contém outras listas como seus elementos. Cada lista interna representa uma linha da matriz, e os elementos individuais podem ser acessados por meio de índices de linha e coluna.
- Por exemplo, veja os dois exemplos ao lado, o primeiro apresenta uma matriz de tamanho 2x3, o segundo uma matriz de tamanho 3x3.

```
[1]  # Criando uma matriz 2x3
     matriz = [[1, 2, 3],
               [4, 5, 6]]
     print(matriz)
```

```
[[1, 2, 3], [4, 5, 6]]
```

```
[3]  # Criando uma matriz 3x3
     matriz = [[1, 2, 3],
               [4, 5, 6],
               [7, 8, 9]]
     print(matriz)
```

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

Acesso a elementos de uma matriz

Acesso a elementos de uma matriz

- O acesso a elementos de uma matriz em Python é feito utilizando índices de linha e coluna. Para acessar um elemento específico, basta indicar o índice da linha e o índice da coluna correspondentes. Por exemplo, para acessar o elemento na linha 2 e coluna 3 de uma matriz, utilizamos matriz[1][2], considerando que os índices em Python começam em 0. Se tivermos a seguinte matriz:

```
[6]  # Acessando um elemento específico da matriz
     matriz = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
     print("Elemento na linha 2, coluna 3:", matriz[1][2])

     Elemento na linha 2, coluna 3: 6
```

Operações básicas com matrizes

Operações básicas com matrizes

- Soma de Matrizes: A soma de duas matrizes é feita somando os elementos correspondentes de cada matriz.

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \quad B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

$$A + B = \begin{bmatrix} 1+5 & 2+6 \\ 3+7 & 4+8 \end{bmatrix} = \begin{bmatrix} 6 & 8 \\ 10 & 12 \end{bmatrix}$$

Operações básicas com matrizes

- No exemplo ao lado são utilizados dois “for” aninhados, para percorrer toda a matriz, realizar a operação e adicionar em uma nova matriz.

```
[13] # Duas matrizes para somar
     A = [[1, 2], [3, 4]]
     B = [[5, 6], [7, 8]]

     soma = []

     for i in range(len(A)):
         linha_soma = []
         for j in range(len(A[0])):
             elemento_soma = A[i][j] + B[i][j]
             linha_soma.append(elemento_soma)
         soma.append(linha_soma)

     print('Matriz A    -> ',A)
     print('Matriz B    -> ',B)
     print('Matriz Soma -> ',soma)
```

```
Matriz A    ->  [[1, 2], [3, 4]]
Matriz B    ->  [[5, 6], [7, 8]]
Matriz Soma ->  [[6, 8], [10, 12]]
```

Operações básicas com matrizes

- Aqui é uma outra forma de somar matrizes, agora com a aplicação compreensão de lista .

```
[18] A = [[1, 2],[3, 4]]
     B = [[5, 6],[7, 8]]

     # Adição de matrizes
     soma = [[A[i][j] + B[i][j] for j in range(len(A[0]))] for i in range(len(A))]

     print('Matriz A    -> ',A)
     print('Matriz B    -> ',B)
     print('Matriz Soma -> ',soma)
```

```
Matriz A    ->  [[1, 2], [3, 4]]
Matriz B    ->  [[5, 6], [7, 8]]
Matriz Soma ->  [[6, 8], [10, 12]]
```

Operações básicas com matrizes

- Para as outras operações é necessário apenas alterar o operador.

```
[19] A = [[1, 2],[3, 4]]
     B = [[5, 6],[7, 8]]

     # Adição de matrizes
     soma = [[A[i][j] - B[i][j] for j in range(len(A[0]))] for i in range(len(A))]

     print('Matriz A    -> ',A)
     print('Matriz B    -> ',B)
     print('Matriz Soma -> ',soma)
```

```
Matriz A    ->  [[1, 2], [3, 4]]
Matriz B    ->  [[5, 6], [7, 8]]
Matriz Soma ->  [[-4, -4], [-4, -4]]
```

```
[20] A = [[1, 2],[3, 4]]
     B = [[5, 6],[7, 8]]

     # Adição de matrizes
     soma = [[A[i][j] * B[i][j] for j in range(len(A[0]))] for i in range(len(A))]

     print('Matriz A    -> ',A)
     print('Matriz B    -> ',B)
     print('Matriz Soma -> ',soma)
```

```
Matriz A    ->  [[1, 2], [3, 4]]
Matriz B    ->  [[5, 6], [7, 8]]
Matriz Soma ->  [[5, 12], [21, 32]]
```

Interação sobre elementos de uma matriz

Interação sobre elementos de uma matriz

- A interação sobre elementos de uma matriz refere-se ao processo de percorrer cada elemento da matriz e realizar alguma operação ou ação com ele. Isso é feito geralmente usando loops, como “for”, para iterar sobre as linhas e colunas da matriz.

- Aqui está um exemplo de como iterar sobre os elementos de uma matriz em Python e imprimir cada elemento:

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

     print(matriz)
     # Iterando sobre os elementos da matriz e imprimindo-os
     for linha in matriz:
         for elemento in linha:
             print(elemento, end=' ')
         print()  # Imprime uma nova linha após cada linha da matriz
```

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

Indexação e fatiamento de matrizes

Indexação e fatiamento de matrizes

- Indexação e fatiamento de matrizes referem-se ao processo de acessar elementos individuais ou subconjuntos de uma matriz em Python. A indexação é usada para acessar um elemento específico da matriz, enquanto o fatiamento é usado para acessar uma parte específica da matriz.

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

     # Indexação para acessar um elemento específico
     elemento = matriz[1][2]  # Acessando o elemento na segunda linha e terceira coluna
     print("Elemento na posição (1, 2):", elemento)

     Elemento na posição (1, 2): 6
```

Indexação e fatiamento de matrizes

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

     # Indexação para acessar um elemento específico
     elemento = matriz[1][2]  # Acessando o elemento na segunda linha e terceira coluna
     print("Elemento na posição (1, 2):", elemento)  # Saída: 6

     # Fatiamento para acessar uma parte específica da matriz
     submatriz = matriz[0:2][0:2]  # Fatiando para obter as duas primeiras linhas e colunas
     print("Submatriz:")
     for linha in submatriz:
         print(linha)
```

```
Elemento na posição (1, 2): 6
Submatriz:
[1, 2, 3]
[4, 5, 6]
```

Indexação e fatiamento de matrizes

- Também é possível criar submatriz com o recurso da compreensão de lista. Veja o exemplo abaixo:

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

     # Acesso a uma submatriz
     submatriz = [linha[1:] for linha in matriz]

     print(matriz)
     print(submatriz)
```

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

Operações de linha e coluna

Operações de linha e coluna

- Operações de linha e coluna em uma matriz (lista 2D) em Python referem-se às operações que podem ser realizadas para adicionar, remover ou alterar elementos em uma determinada linha ou coluna da matriz.
- Aqui temos um exemplo da operação de adição de uma linha inteira na matriz.

```
[36] matriz = [[1, 2, 3], [4, 5, 6]]

     print(matriz)

     # Adição de uma nova linha
     nova_linha = [7, 8, 9]
     matriz.append(nova_linha)

     print(matriz)
```

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

Operações de linha e coluna

- Aqui temos um exemplo da operação de remoção de uma linha inteira da matriz.

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

     print(matriz)

     # Remoção de uma linha
     del matriz[1]
     print(matriz)
```

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

Operações de linha e coluna

- Aqui temos um exemplo da operação de alteração de um elemento da matriz.

```
[39] matriz = [[1, 2, 3], [4, 5, 6]]

     # Alteração de um elemento
     matriz[0][0] = 10
     print(matriz)

     [[10, 2, 3], [4, 5, 6]]
```

Aplicações práticas de matrizes em problemas do mundo real

Análise de Dados Financeiros

- Suponha que você trabalhe em uma empresa financeira e precisa analisar dados de investimentos. Você pode representar os retornos diários de diferentes ações ou ativos financeiros em uma matriz, onde cada linha representa um ativo e cada coluna representa o retorno em um dia específico. Com essa matriz, você pode calcular métricas como a média de retorno, o desvio padrão e a correlação entre diferentes ativos, auxiliando na tomada de decisões de investimento.

Sistemas de Transporte Público

- Em um sistema de transporte público de uma cidade, é possível representar as rotas e horários dos ônibus ou trens em uma matriz de adjacência, onde cada linha e coluna representam uma estação e os elementos indicam se há uma conexão direta entre as estações. Com essa matriz, é possível calcular rotas mais eficientes, determinar a frequência dos veículos em cada linha e identificar estações de transferência importantes, contribuindo para a otimização do sistema de transporte público.

Grade de horário

- Neste exemplo, usamos uma matriz para representar a disponibilidade de salas em uma grade de horários ao longo da semana. Cada linha da matriz representa um dia da semana, e cada coluna representa um horário. Mostramos como verificar se uma sala específica está disponível em um determinado dia e horário, ajudando na organização e agendamento de eventos.

Controle de estoque

- Neste exemplo, utilizamos uma matriz para representar o estoque de diferentes produtos ao longo da semana. Cada linha da matriz representa um produto, e cada coluna representa um dia da semana. Demonstramos como calcular a média de estoque de cada produto ao longo da semana e como atualizar o estoque de um produto com base em uma entrega recebida.

Vamos praticar!

Link de acesso