

Matrizes e Objetos Multidimensionais

MC102

Matrices

Importância para computação

- Processamento de imagem:
 - Uma imagem pode ser vista como uma matriz de pixels.
 - Operações como blur, rotação e detecção de bordas usam vizinhanças da matriz.
- Mapas e geolocalização:
 - Grades de terreno, mapas de calor e distâncias em malhas são modeladas com matrizes.
- LLM e redes neurais:
 - Entradas, pesos e ativações são organizados como vetores e matrizes.
 - Multiplicação de matrizes aparece o tempo todo no treinamento e inferência.
- Jogos de tabuleiro:
 - Tabuleiros de xadrez, damas e sudoku são representados como matrizes.
 - Regras de movimento e validação consultam posições por linha e coluna.

Breve revisão de listas em Python

- Criando listas:

```
1 numeros = [10, 20, 30]
2 nomes = ["Ana", "Bruno", "Carla"]
3 misturada = [1, "abc", True]
```

- Acessando elementos (índices começam em 0):

```
1 numeros[0] # 10
2 numeros[2] # 30
3 numeros[-1] # último elemento
```

- Inserindo no final com `.append()`:

```
1 numeros.append(40)
2 # numeros agora é [10, 20, 30, 40]
```

Matrizes em Python

Em python, matrizes são representadas como **listas de listas**.

Isto é, podemos representa a matriz:

$$\begin{pmatrix} 7 & 0 & 2 & 3 \\ 3 & 1 & 4 & 2 \\ 0 & 3 & 2 & 7 \end{pmatrix}$$

como `m = [[7, 0, 2, 3], [3, 1, 4, 2], [0, 3, 2, 7]]`.

- Isto é uma linha de linhas da matriz.
- Onde cada linha é, também, uma lista.
- O elemento na linha `i` e coluna `j` é acessado por `m[i][j]`.

Entrada de matriz

- Como ler uma lista de números?

Entrada de matriz

- Como ler uma lista de números?

```
1 linha = input().split()
2 for i, elemento in enumerate(linha):
3     linha[i] = int(elemento)
```

Entrada de matriz

- Como ler uma lista de números?

```
1 linha = input().split()
2 for i, elemento in enumerate(linha):
3     linha[i] = int(elemento)
```

```
linha = [int(x) for x in input().split()]
```

Entrada de matriz

- Como ler uma lista de números?

```
1 linha = input().split()
2 for i, elemento in enumerate(linha):
3     linha[i] = int(elemento)
```

```
linha = [int(x) for x in input().split()]
```

- Se uma matriz é uma lista de listas (as linhas) e sabemos ler uma linha, como ler várias linhas?

Entrada de matriz

- Como ler uma lista de números?

```
1 linha = input().split()
2 for i, elemento in enumerate(linha):
3     linha[i] = int(elemento)
```

```
linha = [int(x) for x in input().split()]
```

- Se uma matriz é uma lista de listas (as linhas) e sabemos ler uma linha, como ler várias linhas?

```
1 n = int(input()) # número de linhas
2 matriz = []
3 for _ in range(n):
4     linha = [int(x) for x in input().split()]
5     matriz.append(linha)
```

Entrada de matriz

- Como ler uma lista de números?

```
1 linha = input().split()
2 for i, elemento in enumerate(linha):
3     linha[i] = int(elemento)
```

```
linha = [int(x) for x in input().split()]
```

- Se uma matriz é uma lista de listas (as linhas) e sabemos ler uma linha, como ler várias linhas?

```
1 n = int(input()) # número de linhas
2 matriz = []
3 for _ in range(n):
4     linha = [int(x) for x in input().split()]
5     matriz.append(linha)
```

```
1 n = int(input()) # número de linhas
2 matriz = [[int(x) for x in input().split()] for _ in range(n)]
```

Entrada de matriz

</> Code

Faça uma função que recebe um número n e retorna uma matriz $n \times n$ identidade.

Entrada de matriz

</> Code

Faça uma função que recebe um número n e retorna uma matriz $n \times n$ identidade.

```
1 def identidade(n: int) -> list[list[int]]:
```

Entrada de matriz

</> Code

Faça uma função que recebe um número n e retorna uma matriz $n \times n$ identidade.

```
1 def identidade(n: int) -> list[list[int]]:  
2     matriz = []
```

Entrada de matriz

</> Code

Faça uma função que recebe um número n e retorna uma matriz $n \times n$ identidade.

```
1 def identidade(n: int) -> list[list[int]]:  
2     matriz = []  
3     for i in range(n):
```

Entrada de matriz

</> Code

Faça uma função que recebe um número n e retorna uma matriz $n \times n$ identidade.

```
1 def identidade(n: int) -> list[list[int]]:  
2     matriz = []  
3     for i in range(n):  
4         linha = [0 for _ in range(n)] # Cria uma linha de zeros
```

Entrada de matriz

</> Code

Faça uma função que recebe um número n e retorna uma matriz $n \times n$ identidade.

```
1 def identidade(n: int) -> list[list[int]]:  
2     matriz = []  
3     for i in range(n):  
4         linha = [0 for _ in range(n)] # Cria uma linha de zeros  
5         linha[i] = 1                 # Coloca um 1 na posição diagonal
```

Entrada de matriz

</> Code

Faça uma função que recebe um número n e retorna uma matriz $n \times n$ identidade.

```
1 def identidade(n: int) -> list[list[int]]:  
2     matriz = []  
3     for i in range(n):  
4         linha = [0 for _ in range(n)] # Cria uma linha de zeros  
5         linha[i] = 1                 # Coloca um 1 na posição diagonal  
6         matriz.append(linha)
```

Entrada de matriz

</> Code

Faça uma função que recebe um número n e retorna uma matriz $n \times n$ identidade.

```
1 def identidade(n: int) -> list[list[int]]:  
2     matriz = []  
3     for i in range(n):  
4         linha = [0 for _ in range(n)] # Cria uma linha de zeros  
5         linha[i] = 1                 # Coloca um 1 na posição diagonal  
6         matriz.append(linha)  
7     return matriz
```

Entrada de matriz

</> Code

Faça uma função que recebe uma lista de listas e retorna True ou False se é uma matriz ou não.

Entrada de matriz

</> Code

Faça uma função que recebe uma lista de listas e retorna True ou False se é uma matriz ou não.

```
1 def eh_matriz(lista: list[list]) -> bool:
```

Entrada de matriz

</> Code

Faça uma função que recebe uma lista de listas e retorna True ou False se é uma matriz ou não.

```
1 def eh_matriz(lista: list[list]) -> bool:  
2     tamanho = len(lista[0])          # Tamanho da primeira linha
```

Entrada de matriz

</> Code

Faça uma função que recebe uma lista de listas e retorna True ou False se é uma matriz ou não.

```
1 def eh_matriz(lista: list[list]) -> bool:
2     tamanho = len(lista[0])      # Tamanho da primeira linha
3     for linha in lista:
```

Entrada de matriz

</> Code

Faça uma função que recebe uma lista de listas e retorna True ou False se é uma matriz ou não.

```
1 def eh_matriz(lista: list[list]) -> bool:
2     tamanho = len(lista[0])      # Tamanho da primeira linha
3     for linha in lista:
4         if len(linha) != tamanho: # Verifica se todas as linhas têm o mesmo tamanho
```

Entrada de matriz

</> Code

Faça uma função que recebe uma lista de listas e retorna True ou False se é uma matriz ou não.

```
1 def eh_matriz(lista: list[list]) -> bool:
2     tamanho = len(lista[0])      # Tamanho da primeira linha
3     for linha in lista:
4         if len(linha) != tamanho: # Verifica se todas as linhas têm o mesmo tamanho
5             return False
```

Entrada de matriz

</> Code

Faça uma função que recebe uma lista de listas e retorna True ou False se é uma matriz ou não.

```
1 def eh_matriz(lista: list[list]) -> bool:
2     tamanho = len(lista[0])      # Tamanho da primeira linha
3     for linha in lista:
4         if len(linha) != tamanho: # Verifica se todas as linhas têm o mesmo tamanho
5             return False
6     return True
```

Entrada de matriz

</> Code

Escreva uma função que imprime uma matriz de forma legível.

Entrada de matriz

</> Code

Escreva uma função que imprime uma matriz de forma legível.

```
1 def imprimir_matriz(matriz: list[list]):
```

Entrada de matriz

</> Code

Escreva uma função que imprime uma matriz de forma legível.

```
1 def imprimir_matriz(matriz: list[list]):  
2     for linha in matriz:
```

Entrada de matriz

</> Code

Escreva uma função que imprime uma matriz de forma legível.

```
1 def imprimir_matriz(matriz: list[list]):  
2     for linha in matriz:  
3         print(*linha)
```

Acesso a elementos

- Suponha a seguinte matriz:

Acesso a elementos

- Suponha a seguinte matriz:

```
1 m = [  
2   [7, 0, 2, 3],  
3   [3, 1, 4, 2],  
4   [0, 3, 2, 7],  
5   ]
```

Acesso a elementos

- Suponha a seguinte matriz:

```
1 m = [  
2   [7, 0, 2, 3],  
3   [3, 1, 4, 2],  
4   [0, 3, 2, 7],  
5   ]
```

- Acessando um elemento específico (`m[i][j]`):

Acesso a elementos

- Suponha a seguinte matriz:

```
1 m = [  
2   [7, 0, 2, 3],  
3   [3, 1, 4, 2],  
4   [0, 3, 2, 7],  
5 ]
```

- Acessando um elemento específico (`m[i][j]`):

```
1 m[1][2] # 4 (linha 1, coluna 2)  
2 m[0][0] # 7
```

Acesso a elementos

- Suponha a seguinte matriz:

```
1 m = [  
2   [7, 0, 2, 3],  
3   [3, 1, 4, 2],  
4   [0, 3, 2, 7],  
5 ]
```

- Acessando um elemento específico (`m[i][j]`):

```
1 m[1][2] # 4 (linha 1, coluna 2)  
2 m[0][0] # 7
```

- Acessando linha inteira e coluna inteira:

Acesso a elementos

- Suponha a seguinte matriz:

```
1 m = [  
2   [7, 0, 2, 3],  
3   [3, 1, 4, 2],  
4   [0, 3, 2, 7],  
5 ]
```

- Acessando um elemento específico (`m[i][j]`):

```
1 m[1][2] # 4 (linha 1, coluna 2)  
2 m[0][0] # 7
```

- Acessando linha inteira e coluna inteira:

```
1 linha_1 = m[1]  
2 # [3, 1, 4, 2]  
3 coluna_2 = [linha[2] for linha in m]  
4 # [2, 4, 2]
```

Acesso a elementos

- Suponha a seguinte matriz:

```
1 m = [  
2   [7, 0, 2, 3],  
3   [3, 1, 4, 2],  
4   [0, 3, 2, 7],  
5 ]
```

- Acessando um elemento específico (`m[i][j]`):

```
1 m[1][2] # 4 (linha 1, coluna 2)  
2 m[0][0] # 7
```

- Acessando linha inteira e coluna inteira:

```
1 linha_1 = m[1]  
2 # [3, 1, 4, 2]  
3 coluna_2 = [linha[2] for linha in m]  
4 # [2, 4, 2]
```

Acesso a elementos

</> Code

Escreva uma função que recebe uma matriz e retorna a transposta dela.

Acesso a elementos

</> Code

Escreva uma função que recebe uma matriz e retorna a transposta dela.

```
1 def transposta(matriz: list[list]) -> list[list]:
```

Acesso a elementos

</> Code

Escreva uma função que recebe uma matriz e retorna a transposta dela.

```
1 def transposta(matriz: list[list]) -> list[list]:  
2     n_linhas = len(matriz)
```

Acesso a elementos

</> Code

Escreva uma função que recebe uma matriz e retorna a transposta dela.

```
1 def transposta(matriz: list[list]) -> list[list]:  
2     n_linhas = len(matriz)  
3     m_colunas = len(matriz[0])
```

Acesso a elementos

</> Code

Escreva uma função que recebe uma matriz e retorna a transposta dela.

```
1 def transposta(matriz: list[list]) -> list[list]:  
2     n_linhas = len(matriz)  
3     m_colunas = len(matriz[0])  
4     transposta = [[0 for _ in range(n_linhas)] for _ in range(m_colunas)]
```

Acesso a elementos

</> Code

Escreva uma função que recebe uma matriz e retorna a transposta dela.

```
1 def transposta(matriz: list[list]) -> list[list]:  
2     n_linhas = len(matriz)  
3     m_colunas = len(matriz[0])  
4     transposta = [[0 for _ in range(n_linhas)] for _ in range(m_colunas)]
```

Acesso a elementos

</> Code

Escreva uma função que recebe uma matriz e retorna a transposta dela.

```
1 def transposta(matriz: list[list]) -> list[list]:
2     n_linhas = len(matriz)
3     m_colunas = len(matriz[0])
4     transposta = [[0 for _ in range(n_linhas)] for _ in range(m_colunas)]
5
6     for i in range(n_linhas):
```

Acesso a elementos

</> Code

Escreva uma função que recebe uma matriz e retorna a transposta dela.

```
1 def transposta(matriz: list[list]) -> list[list]:
2     n_linhas = len(matriz)
3     m_colunas = len(matriz[0])
4     transposta = [[0 for _ in range(n_linhas)] for _ in range(m_colunas)]
5
6     for i in range(n_linhas):
7         for j in range(m_colunas):
```

Acesso a elementos

</> Code

Escreva uma função que recebe uma matriz e retorna a transposta dela.

```
1 def transposta(matriz: list[list]) -> list[list]:
2     n_linhas = len(matriz)
3     m_colunas = len(matriz[0])
4     transposta = [[0 for _ in range(n_linhas)] for _ in range(m_colunas)]
5
6     for i in range(n_linhas):
7         for j in range(m_colunas):
8             transposta[j][i] = matriz[i][j]
```

Acesso a elementos

</> Code

Escreva uma função que recebe uma matriz e retorna a transposta dela.

```
1 def transposta(matriz: list[list]) -> list[list]:
2     n_linhas = len(matriz)
3     m_colunas = len(matriz[0])
4     transposta = [[0 for _ in range(n_linhas)] for _ in range(m_colunas)]
5
6     for i in range(n_linhas):
7         for j in range(m_colunas):
8             transposta[j][i] = matriz[i][j]
```

Acesso a elementos

</> Code

Escreva uma função que recebe uma matriz e retorna a transposta dela.

```
1 def transposta(matriz: list[list]) -> list[list]:
2     n_linhas = len(matriz)
3     m_colunas = len(matriz[0])
4     transposta = [[0 for _ in range(n_linhas)] for _ in range(m_colunas)]
5
6     for i in range(n_linhas):
7         for j in range(m_colunas):
8             transposta[j][i] = matriz[i][j]
9
10    return transposta
```

Acesso a elementos

</> Code

Escreva uma função que recebe duas matrizes e retorna a soma delas.

Acesso a elementos

</> Code

Escreva uma função que recebe duas matrizes e retorna a soma delas.

```
1 def soma_matrizes(m1: list[list[int]], m2: list[list[int]]) -> list[list[int]]:
```

Acesso a elementos

</> Code

Escreva uma função que recebe duas matrizes e retorna a soma delas.

```
1 def soma_matrizes(m1: list[list[int]], m2: list[list[int]]) -> list[list[int]]:  
2     n_linhas = len(m1)
```

Acesso a elementos

</> Code

Escreva uma função que recebe duas matrizes e retorna a soma delas.

```
1 def soma_matrizes(m1: list[list[int]], m2: list[list[int]]) -> list[list[int]]:  
2     n_linhas = len(m1)  
3     m_colunas = len(m1[0])
```

Acesso a elementos

</> Code

Escreva uma função que recebe duas matrizes e retorna a soma delas.

```
1 def soma_matrizes(m1: list[list[int]], m2: list[list[int]]) -> list[list[int]]:  
2     n_linhas = len(m1)  
3     m_colunas = len(m1[0])  
4     soma = [[0 for _ in range(m_colunas)] for _ in range(n_linhas)]
```

Acesso a elementos

</> Code

Escreva uma função que recebe duas matrizes e retorna a soma delas.

```
1 def soma_matrizes(m1: list[list[int]], m2: list[list[int]]) -> list[list[int]]:  
2     n_linhas = len(m1)  
3     m_colunas = len(m1[0])  
4     soma = [[0 for _ in range(m_colunas)] for _ in range(n_linhas)]
```

Acesso a elementos

</> Code

Escreva uma função que recebe duas matrizes e retorna a soma delas.

```
1 def soma_matrizes(m1: list[list[int]], m2: list[list[int]]) -> list[list[int]]:  
2     n_linhas = len(m1)  
3     m_colunas = len(m1[0])  
4     soma = [[0 for _ in range(m_colunas)] for _ in range(n_linhas)]  
5  
6     for i in range(n_linhas):
```

Acesso a elementos

</> Code

Escreva uma função que recebe duas matrizes e retorna a soma delas.

```
1 def soma_matrizes(m1: list[list[int]], m2: list[list[int]]) -> list[list[int]]:  
2     n_linhas = len(m1)  
3     m_colunas = len(m1[0])  
4     soma = [[0 for _ in range(m_colunas)] for _ in range(n_linhas)]  
5  
6     for i in range(n_linhas):  
7         for j in range(m_colunas):
```

Acesso a elementos

</> Code

Escreva uma função que recebe duas matrizes e retorna a soma delas.

```
1 def soma_matrizes(m1: list[list[int]], m2: list[list[int]]) -> list[list[int]]:  
2     n_linhas = len(m1)  
3     m_colunas = len(m1[0])  
4     soma = [[0 for _ in range(m_colunas)] for _ in range(n_linhas)]  
5  
6     for i in range(n_linhas):  
7         for j in range(m_colunas):  
8             soma[i][j] = m1[i][j] + m2[i][j]
```

Acesso a elementos

</> Code

Escreva uma função que recebe duas matrizes e retorna a soma delas.

```
1 def soma_matrizes(m1: list[list[int]], m2: list[list[int]]) -> list[list[int]]:  
2     n_linhas = len(m1)  
3     m_colunas = len(m1[0])  
4     soma = [[0 for _ in range(m_colunas)] for _ in range(n_linhas)]  
5  
6     for i in range(n_linhas):  
7         for j in range(m_colunas):  
8             soma[i][j] = m1[i][j] + m2[i][j]
```

Acesso a elementos

</> Code

Escreva uma função que recebe duas matrizes e retorna a soma delas.

```
1 def soma_matrizes(m1: list[list[int]], m2: list[list[int]]) -> list[list[int]]:  
2     n_linhas = len(m1)  
3     m_colunas = len(m1[0])  
4     soma = [[0 for _ in range(m_colunas)] for _ in range(n_linhas)]  
5  
6     for i in range(n_linhas):  
7         for j in range(m_colunas):  
8             soma[i][j] = m1[i][j] + m2[i][j]  
9  
10    return soma
```

Acesso a elementos

</> Code

Escreva uma função que recebe duas matrizes e retorna o produto delas.

Acesso a elementos

</> Code

Escreva uma função que recebe duas matrizes e retorna o produto delas.

```
1 def produto_matrizes(m1: list[list[int]], m2: list[list[int]]) -> list[list[int]]:
```

Acesso a elementos

</> Code

Escreva uma função que recebe duas matrizes e retorna o produto delas.

```
1 def produto_matrizes(m1: list[list[int]], m2: list[list[int]]) -> list[list[int]]:  
2     ... # Assumindo que eu tenho as dimensões das matrizes
```

Acesso a elementos

</> Code

Escreva uma função que recebe duas matrizes e retorna o produto delas.

```
1 def produto_matrizes(m1: list[list[int]], m2: list[list[int]]) -> list[list[int]]:  
2     ... # Assumindo que eu tenho as dimensões das matrizes  
3     produto = [[0 for _ in range(m_colunas_m2)] for _ in range(n_linhas_m1)]
```

Acesso a elementos

</> Code

Escreva uma função que recebe duas matrizes e retorna o produto delas.

```
1 def produto_matrizes(m1: list[list[int]], m2: list[list[int]]) -> list[list[int]]:  
2     ... # Assumindo que eu tenho as dimensões das matrizes  
3     produto = [[0 for _ in range(m_colunas_m2)] for _ in range(n_linhas_m1)]
```

Acesso a elementos

</> Code

Escreva uma função que recebe duas matrizes e retorna o produto delas.

```
1 def produto_matrizes(m1: list[list[int]], m2: list[list[int]]) -> list[list[int]]:  
2     ... # Assumindo que eu tenho as dimensões das matrizes  
3     produto = [[0 for _ in range(m_colunas_m2)] for _ in range(n_linhas_m1)]  
4  
5     for i in range(n_linhas_m1):
```

Acesso a elementos

</> Code

Escreva uma função que recebe duas matrizes e retorna o produto delas.

```
1 def produto_matrizes(m1: list[list[int]], m2: list[list[int]]) -> list[list[int]]:  
2     ... # Assumindo que eu tenho as dimensões das matrizes  
3     produto = [[0 for _ in range(m_colunas_m2)] for _ in range(n_linhas_m1)]  
4  
5     for i in range(n_linhas_m1):  
6         for j in range(m_colunas_m2):
```

Acesso a elementos

</> Code

Escreva uma função que recebe duas matrizes e retorna o produto delas.

```
1 def produto_matrizes(m1: list[list[int]], m2: list[list[int]]) -> list[list[int]]:  
2     ... # Assumindo que eu tenho as dimensões das matrizes  
3     produto = [[0 for _ in range(m_colunas_m2)] for _ in range(n_linhas_m1)]  
4  
5     for i in range(n_linhas_m1):  
6         for j in range(m_colunas_m2):  
7             for k in range(m_colunas_m1):
```

Acesso a elementos

</> Code

Escreva uma função que recebe duas matrizes e retorna o produto delas.

```
1 def produto_matrizes(m1: list[list[int]], m2: list[list[int]]) -> list[list[int]]:  
2     ... # Assumindo que eu tenho as dimensões das matrizes  
3     produto = [[0 for _ in range(m_colunas_m2)] for _ in range(n_linhas_m1)]  
4  
5     for i in range(n_linhas_m1):  
6         for j in range(m_colunas_m2):  
7             for k in range(m_colunas_m1):  
8                 produto[i][j] += m1[i][k] * m2[k][j]
```

Acesso a elementos

</> Code

Escreva uma função que recebe duas matrizes e retorna o produto delas.

```
1 def produto_matrizes(m1: list[list[int]], m2: list[list[int]]) -> list[list[int]]:  
2     ... # Assumindo que eu tenho as dimensões das matrizes  
3     produto = [[0 for _ in range(m_colunas_m2)] for _ in range(n_linhas_m1)]  
4  
5     for i in range(n_linhas_m1):  
6         for j in range(m_colunas_m2):  
7             for k in range(m_colunas_m1):  
8                 produto[i][j] += m1[i][k] * m2[k][j]
```

Acesso a elementos

</> Code

Escreva uma função que recebe duas matrizes e retorna o produto delas.

```
1 def produto_matrizes(m1: list[list[int]], m2: list[list[int]]) -> list[list[int]]:  
2     ... # Assumindo que eu tenho as dimensões das matrizes  
3     produto = [[0 for _ in range(m_colunas_m2)] for _ in range(n_linhas_m1)]  
4  
5     for i in range(n_linhas_m1):  
6         for j in range(m_colunas_m2):  
7             for k in range(m_colunas_m1):  
8                 produto[i][j] += m1[i][k] * m2[k][j]  
9  
10    return produto
```

Objetos Multidimensionais

Matrizes de matrizes

- Até agora, cada posição da matriz guardava apenas um número.

Matrizes de matrizes

- Até agora, cada posição da matriz guardava apenas um número.
- Mas podemos guardar estruturas maiores em cada posição, como listas, vetores ou até outras matrizes.

Matrizes de matrizes

- Até agora, cada posição da matriz guardava apenas um número.
- Mas podemos guardar estruturas maiores em cada posição, como listas, vetores ou até outras matrizes.
- Quando isso acontece, o objeto deixa de ser bidimensional e passa a ser multidimensional.

Matrizes de matrizes

- Até agora, cada posição da matriz guardava apenas um número.
- Mas podemos guardar estruturas maiores em cada posição, como listas, vetores ou até outras matrizes.
- Quando isso acontece, o objeto deixa de ser bidimensional e passa a ser multidimensional.
- Exemplo de um objeto 3D em Python:

Matrizes de matrizes

- Até agora, cada posição da matriz guardava apenas um número.
- Mas podemos guardar estruturas maiores em cada posição, como listas, vetores ou até outras matrizes.
- Quando isso acontece, o objeto deixa de ser bidimensional e passa a ser multidimensional.
- Exemplo de um objeto 3D em Python:

```
1 objeto = [  
2     [  
3         [1, 2],  
4         [3, 4],  
5     ],  
6     [  
7         [5, 6],  
8         [7, 8],  
9     ],  
10 ]
```

Matrizes de matrizes

- Até agora, cada posição da matriz guardava apenas um número.
- Mas podemos guardar estruturas maiores em cada posição, como listas, vetores ou até outras matrizes.
- Quando isso acontece, o objeto deixa de ser bidimensional e passa a ser multidimensional.
- Exemplo de um objeto 3D em Python:

```
1 objeto = [  
2     [  
3         [1, 2],  
4         [3, 4],  
5     ],  
6     [  
7         [5, 6],  
8         [7, 8],  
9     ],  
10 ]
```

- Para acessar um elemento específico, usamos um índice para cada nível:

Matrizes de matrizes

- Até agora, cada posição da matriz guardava apenas um número.
- Mas podemos guardar estruturas maiores em cada posição, como listas, vetores ou até outras matrizes.
- Quando isso acontece, o objeto deixa de ser bidimensional e passa a ser multidimensional.
- Exemplo de um objeto 3D em Python:

```
1 objeto = [  
2     [  
3         [1, 2],  
4         [3, 4],  
5     ],  
6     [  
7         [5, 6],  
8         [7, 8],  
9     ],  
10 ]
```

- Para acessar um elemento específico, usamos um índice para cada nível:

```
objeto[1][0][1] # 6
```

Matrizes de matrizes

- Até agora, cada posição da matriz guardava apenas um número.
- Mas podemos guardar estruturas maiores em cada posição, como listas, vetores ou até outras matrizes.
- Quando isso acontece, o objeto deixa de ser bidimensional e passa a ser multidimensional.
- Exemplo de um objeto 3D em Python:

```
1 objeto = [  
2     [  
3         [1, 2],  
4         [3, 4],  
5     ],  
6     [  
7         [5, 6],  
8         [7, 8],  
9     ],  
10 ]
```

- Para acessar um elemento específico, usamos um índice para cada nível:

```
objeto[1][0][1] # 6
```

- Leitura intuitiva: primeiro escolhemos a camada, depois a linha, e por fim a coluna ou posição interna.

Dúvidas?