

# Algoritmos

---

Matrizes e Objetos Multidimensionais

Exercícios

# **Matrizes e Objetos** **Multidimensionais**

---

# Matrizes e Objetos Multidimensionais

- Matrizes e objetos multidimensionais são generalizações de objetos simples vistos anteriormente (listas e tuplas).
- Esses tipos de dados nos permitem armazenar informações mais complexas em uma única variável.
- Exemplo de informações/operações que podem ser armazenadas/manipuladas utilizando matrizes e objetos multidimensionais:
  - Matemática: operações com matrizes.
  - Processamento de imagem: cor de cada pixel presente na imagem.
  - Mapas e geolocalização: informação sobre o relevo em cada ponto do mapa.
  - Jogos de tabuleiro: Xadrez, Damas, Go, Batalha Naval, etc.

# Matrizes e Objetos Multidimensionais

- Uma lista pode conter elementos de tipos diferentes.
- Uma lista pode conter inclusive outras listas.
- Exemplo de declaração de uma lista:

```
1 obj = [  
2     7, 42, True, "Aula", 3.14,  
3     [0.1, 0.2, 0.3]  
4 ]
```

- Exemplo de declaração de um objeto multidimensional:

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

# Declaração de Matrizes

- Uma matriz é um objeto bidimensional, formada por listas, todas do mesmo tamanho.
- Sua representação é dada na forma de uma lista de listas (a mesma ideia pode ser aplicada para tuplas).
- Exemplo de declaração de uma matriz  $2 \times 2$ :

```
1 matriz = [  
2     [1, 2], # linha 1  
3     [3, 4]  # linha 2  
4 ]
```

- Exemplo de declaração de uma matriz  $3 \times 4$ :

```
1 matriz = [  
2     [11, 12, 13, 14], # linha 1  
3     [21, 22, 23, 24], # linha 2  
4     [31, 32, 33, 34]  # linha 3  
5 ]
```

# Declaração de Matrizes

- Podemos criar uma matriz com as informações fornecidas pelo usuário.
- Exemplo de como receber uma matriz de dimensões  $l \times c$  como entrada:

```
1 l = int(input("Entre com o número de linhas: "))
2 c = int(input("Entre com o número de colunas: "))
3 matriz = []
4
5 for i in range(l):
6     linha = []
7     for j in range(c):
8         linha.append(int(input())) # recebendo os dados
9     matriz.append(linha)
```

# Declaração de Matrizes

- Podemos ainda inicializar uma matriz com valores pré-definidos.
- Inicializando uma matriz de dimensões  $l \times c$  e atribuindo valor zero para todos os elementos:

```
1 l = int(input("Entre com o número de linhas: ")) # l = 3
2 c = int(input("Entre com o número de colunas: ")) # c = 4
3 matriz = []
4 for i in range(l):
5     linha = []
6     for j in range(c):
7         linha.append(0)
8     matriz.append(linha)
9 print(matriz)
10 # [[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]
11
12 # Forma alternativa/compacta de inicializar uma matriz
13 matriz = [[0 for j in range(c)] for i in range(l)]
```



# Declaração de Matrizes

- Inicializando uma matriz de dimensões  $l \times c$  e atribuindo valores de 1 até  $l \times c$  para os elementos da matriz:

```
1 l = int(input("Entre com o número de linhas: ")) # l = 3
2 c = int(input("Entre com o número de colunas: ")) # c = 4
3 matriz = []
4
5 for i in range(l):
6     linha = []
7     for j in range(c):
8         linha.append(i * c + j + 1)
9     matriz.append(linha)
10
11 print(matriz)
12 # [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]]
```

# Acessando Elementos de uma Matriz

- Note que uma matriz é uma lista de listas.
- Podemos acessar um elemento de uma matriz, localizado em uma determinada linha e coluna, da seguinte forma:

```
1 matriz[linha][coluna]
2 # Lembrete: linhas e colunas são numeradas
3 #           a partir da posição zero
```

- Exemplo:

```
1 matriz = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
2 print(matriz[0][2])
3 # 3
4 print(matriz[2][1])
5 # 8
```

# Acessando Elementos de uma Matriz

- Similar ao que vimos em listas e tuplas, caso ocorra uma tentativa de acessar uma posição inexistente da matriz, um erro será gerado.
- Exemplo:

```
1 matriz = [[1, 2], [3, 4]]
2 print(matriz[0][0])
3 # 1
4 print(matriz[1][1])
5 # 4
6 print(matriz[2][2])
7 # IndexError: list index out of range
```

# Alterando Elementos de uma Matriz

- Podemos alterar um elemento de uma matriz, localizado em uma determinada linha e coluna, da seguinte forma:

```
1 matriz[linha][coluna] = valor
```

- Exemplo:

```
1 matriz = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
2 matriz[0][0] = 0
3 matriz[2][2] = 10
4 print(matriz)
5 # [[0, 2, 3], [4, 5, 6], [7, 8, 10]]
```

# Criando uma Cópia de uma Matriz

- Para criar uma cópia de uma matriz, precisamos criar uma nova matriz com as cópias de cada uma das linhas da matriz original.
- Exemplo:

```
1 A = [[1, 2], [3, 4]]
2
3 B = A.copy()
4
5 B[0][0] = 0
6
7 print(A)
8 # [[0, 2], [3, 4]]
9 print(B)
10 # [[0, 2], [3, 4]]
```

# Criando uma Cópia de uma Matriz

- Para criar uma cópia de uma matriz, precisamos criar uma nova matriz com as cópias de cada uma das linhas da matriz original.
- Exemplo:

```
1 A = [[1, 2], [3, 4]]
2
3 B = [linha.copy() for linha in A]
4
5 B[0][0] = 0
6
7 print(A)
8 # [[1, 2], [3, 4]]
9 print(B)
10 # [[0, 2], [3, 4]]
```

# Criando uma Cópia de uma Matriz

- Para criar uma cópia de uma matriz, precisamos criar uma nova matriz com as cópias de cada uma das linhas da matriz original.
- Exemplo:

```
1 A = [[1, 2], [3, 4]]
2
3 B = [linha[:] for linha in A]
4
5 B[0][0] = 0
6
7 print(A)
8 # [[1, 2], [3, 4]]
9 print(B)
10 # [[0, 2], [3, 4]]
```

# Criando uma Cópia de uma Matriz

- Para criar uma cópia de uma matriz, precisamos criar uma nova matriz com as cópias de cada uma das linhas da matriz original.
- Exemplo:

```
1 A = [[1, 2], [3, 4]]
2
3 B = [list(linha) for linha in A]
4
5 B[0][0] = 0
6
7 print(A)
8 # [[1, 2], [3, 4]]
9 print(B)
10 # [[0, 2], [3, 4]]
```



- Até agora criamos matrizes bidimensionais, mas podemos criar objetos com mais dimensões.
- Podemos criar objetos com  $d$  dimensões utilizando a mesma ideia de listas de listas.
- Exemplo de um objeto com dimensões  $2 \times 2 \times 2$ :

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

# Objetos Multidimensionais

- Podemos acessar um elemento em um objeto com dimensões  $d_1 \times d_2 \times \dots \times d_n$  da seguinte forma:

```
1 objeto[index_1][index_2]...[index_n]
```

- Exemplo:

```
1 obj = [[[1, 2], [3, 4]], [[5, 6], [7, 8]]] # 2 x 2 x 2
2 print(obj[0][0][0])
3 # 1
4 print(obj[1][0][0])
5 # 5
6 print(obj[1][1][0])
7 # 7
8 print(obj[1][1][1])
9 # 8
```

- Podemos alterar um elemento em um objeto com dimensões  $d_1 \times d_2 \times \dots \times d_n$  da seguinte forma:

```
1 objeto[index_1][index_2]...[index_n] = valor
```

- Exemplo:

```
1 obj = [[[0, 0], [0, 0]], [[0, 0], [0, 0]]] # 2 x 2 x 2
2 obj[1][0][1] = 5
3 obj[0][1][0] = 3
4 print(obj)
5 # [[[0, 0], [3, 0]], [[0, 5], [0, 0]]]
```

## **Exercícios**

---

1. Escreva uma função que leia e retorne uma matriz de inteiros fornecida pelo usuário. Sua função deve ler os números linha a linha. Os números devem estar separados por espaços em branco. Sua função deve interromper a leitura ao receber uma linha em branco.
2. Escreva uma função que, dada uma lista bidimensional (lista de listas), verifique se ela é uma matriz. Em caso positivo, sua função deve retornar uma tupla com o número de linhas e de colunas da matriz. Em caso negativo, deve retornar uma tupla vazia.
3. Escreva uma função que imprime, linha a linha, os valores de uma matriz bidimensional dada como argumento.

## Exercício 1 - Lendo uma Matriz

```
1 def lê_matriz():
2     M = []
3     while True:
4         temp = input().split()
5         if temp == []:
6             return M
7         linha = []
8         for i in temp:
9             linha.append(int(i))
10        M.append(linha)
```

## Exercício 2 - Dimensões de uma Matriz

```
1 def dimensões(M):  
2     linhas = len(M)  
3     colunas = len(M[0])  
4     for i in range(1, linhas):  
5         if len(M[i]) != colunas:  
6             return ()  
7     return (linhas, colunas)
```

## Exercício 3 - Imprimindo uma Matriz

```
1 def imprime_matriz(M):  
2     (linhas, colunas) = dimensões(M)  
3     for i in range(linhas):  
4  
5         for j in range(colunas):  
6             print(M[i][j], end = " ")  
7         print()
```



## Exercício 3 - Imprimindo uma Matriz

```
1 def imprime_matriz(M):
2     (linhas, colunas) = dimensões(M)
3     for i in range(linhas):
4         print(M[i][0], end = "")
5         for j in range(1, colunas):
6             print(" ", M[i][j], end = "")
7         print()
```

## Exercício 3 - Imprimindo uma Matriz

```
1 def imprime_matriz(M):  
2     for linha in M:  
3         # converte os elementos da lista para string  
4         aux = [str(i) for i in linha]  
5         print(" ".join(aux))
```

4. Escreva uma função que, dada uma matriz ( $M$ ), calcule a sua transposta ( $M^t$ ). Exemplo:

$$\begin{array}{c} M \\ \left[ \begin{array}{cccc} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{array} \right] \end{array} \quad \begin{array}{c} M^t \\ \left[ \begin{array}{cc} 1 & 5 \\ 2 & 6 \\ 3 & 7 \\ 4 & 8 \end{array} \right] \end{array}$$

5. Escreva uma função que recebe duas matrizes ( $A$  e  $B$ ). Se as duas matrizes tiverem dimensões compatíveis, sua função deve retornar a soma das duas ( $C = A + B$ ). Caso contrário, sua função deve retornar uma lista vazia. Exemplo:

$$\begin{array}{c} A \\ \left[ \begin{array}{cc} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{array} \right] \end{array} + \begin{array}{c} B \\ \left[ \begin{array}{cc} 5 & 6 \\ 1 & 3 \\ 4 & 2 \end{array} \right] \end{array} = \begin{array}{c} C \\ \left[ \begin{array}{cc} 6 & 8 \\ 4 & 7 \\ 9 & 8 \end{array} \right] \end{array}$$

## Exercício 4 - Matriz Transposta

```
1 def transposta(M):  
2     T = []  
3     (linhas, colunas) = dimensões(M)  
4     for j in range(colunas):  
5         linha = []  
6         for i in range(linhas):  
7             linha.append(M[i][j])  
8         T.append(linha)  
9     return T
```

## Exercício 4 - Matriz Transposta

```
1 def transposta(M):  
2     T = []  
3     (linhas, colunas) = dimensões(M)  
4     for j in range(colunas):  
5         T.append([])  
6         for i in range(linhas):  
7             T[j].append(M[i][j])  
8     return T
```

## Exercício 5 - Soma de Matrizes

```
1 def soma(A, B):
2     C = []
3     dim_a = dimensões(A)
4     dim_b = dimensões(B)
5     if dim_a == dim_b:
6         (linhas, colunas) = dim_a
7         for i in range(linhas):
8             linha = []
9             for j in range(colunas):
10                linha.append(A[i][j] + B[i][j])
11            C.append(linha)
12    return C
```

6. Escreva uma função que recebe duas matrizes ( $A$  e  $B$ ). Se as duas matrizes tiverem dimensões compatíveis, sua função deve retornar o produto das duas ( $C = A \times B$ ). Caso contrário, sua função deve retornar uma lista vazia. Exemplo:

$$\begin{bmatrix} A \\ 1 & 2 \\ 3 & 4 \end{bmatrix} \times \begin{bmatrix} B \\ 5 \\ 6 \end{bmatrix} = \begin{bmatrix} C \\ 17 \\ 39 \end{bmatrix}$$

7. Escreva uma função que, dada uma matriz quadrada, verifique se ela é uma matriz diagonal. Exemplo:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$

8. Escreva uma função que, dada uma matriz quadrada, verifique se ela é uma matriz triangular inferior. Exemplo:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 \\ 0 & 4 & 4 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 9 & 0 & 2 & 3 \end{bmatrix}$$

9. Escreva uma função que, dada uma matriz quadrada, verifique se ela é uma matriz triangular superior. Exemplo:

$$\begin{bmatrix} 1 & 0 & 8 & 9 & 8 \\ 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 7 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$



10. Uma matriz quadrada de números inteiros é um *quadrado mágico* se o valor da soma dos elementos de cada linha, de cada coluna e da diagonal principal e da diagonal secundária é o mesmo. Além disso, a matriz deve conter todos os números inteiros do intervalo  $[1..n \times n]$ . Exemplo:

$$\begin{bmatrix} 15 & 8 & 1 & 24 & 17 \\ 16 & 14 & 7 & 5 & 23 \\ 22 & 20 & 13 & 6 & 4 \\ 3 & 21 & 19 & 12 & 10 \\ 9 & 2 & 25 & 18 & 11 \end{bmatrix}$$

A matriz acima é um quadrado mágico, cujas somas valem 65. Escreva um programa que, dada uma matriz quadrada, verifique se ela é um *quadrado mágico*.

11. Uma matriz de permutações é uma matriz quadrada cujos elementos são zeros ou uns, tal que em cada linha e em cada coluna exista exatamente um elemento igual a 1. Exemplo:

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Escreva um programa que, dada uma matriz quadrada, verifique se ela é uma matriz de permutações.