Objetos Multidimensionais

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:

```
obj = [
  7, 42, True , "Algoritmos", 3.14,
  [0.1, 0.2, 0.3]
]
```

- Exemplo de declaração de um objeto multimensional:

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

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 × 2:

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

- Exemplo de declaração de uma matriz 3 × 4:

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

Declaração de Matrizes

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

```
l = int(input("Entre com o número de linhas: "))
c = int(input("Entre com o número de colunas: "))
matriz = []


for i in
r
  linha = []
  for j in range(c):
    linha.append(int(input())) # recebendo os dados
   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:

```
l = int(input("Entre com o número de linhas: "))  # l = 3
c = int(input("Entre com o número de colunas: ")) # c = 4
matriz = []
for i in range(l):
  linha = []
  for j in range(c):
     linha.append(0)
  matriz.append(linha)
print(matriz)
# [[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0,
0]]

# Forma alternativa/compacta de inicializar uma
matriz
matriz = [[0 for i in range(c)] for j in range(l)]
```

Declaração de Matrizes

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

```
l = int(input("Entre com o número de linhas: "))  # l = 3
c = int(input("Entre com o número de colunas: ")) # c = 4
matriz = []

for i in range(l):
  linha = []
  for j in range(c):
    linha.append(i * c + j + 1)
  matriz.append(linha)

print(matriz)
# [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11,
12]]
```

Acessando Elementos de uma Matriz

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

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

- Exemplo:

```
matriz = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
print(matriz[0][2])    # 3
print(matriz[2][1])    # 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])   # 1
3 print(matriz[1][1])   # 4
4 print(matriz[2][2])
5 # IndexError: list index out of range
6
7
```

Alterando Elementos de uma Matriz

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

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

- Exemplo:

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

Objetos Multidimensionais

- 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 $2 \times 2 \times 2$:

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

Objetos Multidimensionais

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

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

- Exemplo:

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

Objetos Multidimensionais

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

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

- Exemplo:

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

Exercícios

Exercícios

1. Escreva uma função que leia e retorne uma matriz de inteiros fornecida pelo usuário. Sua matriz 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 um 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()  if temp == []:
5        return M  linha = []
6      for i in temp:  linha.append(int(i))
7    M.append(linha)
8
9
1
0
```

Exercício 2 - Dimensões de uma Matriz

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

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):  print(M[i][j], end = " ")
6   print()
7
```

Exercício 3 - Imprimindo uma Matriz

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

Exercício 3 - Imprimindo uma Matriz

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

Exercícios

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

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:

Exercício 4 - Matriz Transposta

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

Exercício 4 - Matriz Transposta

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

Exercício 5 - Soma de Matrizes

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

Exercícios

Faça um programa que peça:

1) Tamanho da Matriz

2) Peça para preencher a matriz campo a campo

Randomicamente deve ele inserir um valor em um ponto da matriz

Deve imprimir a matriz indicando onde está o ponto.