Programação Script

Matrizes

Aula 09

Prof. Felipe A. Louza

Roteiro

1. Matrizes
2. Inicialização de Matrizes
3. Entrada e Saída
4. Exemplos com Matrizes
5. Matrizes Multidimensionais
6. Referências

Roteiro

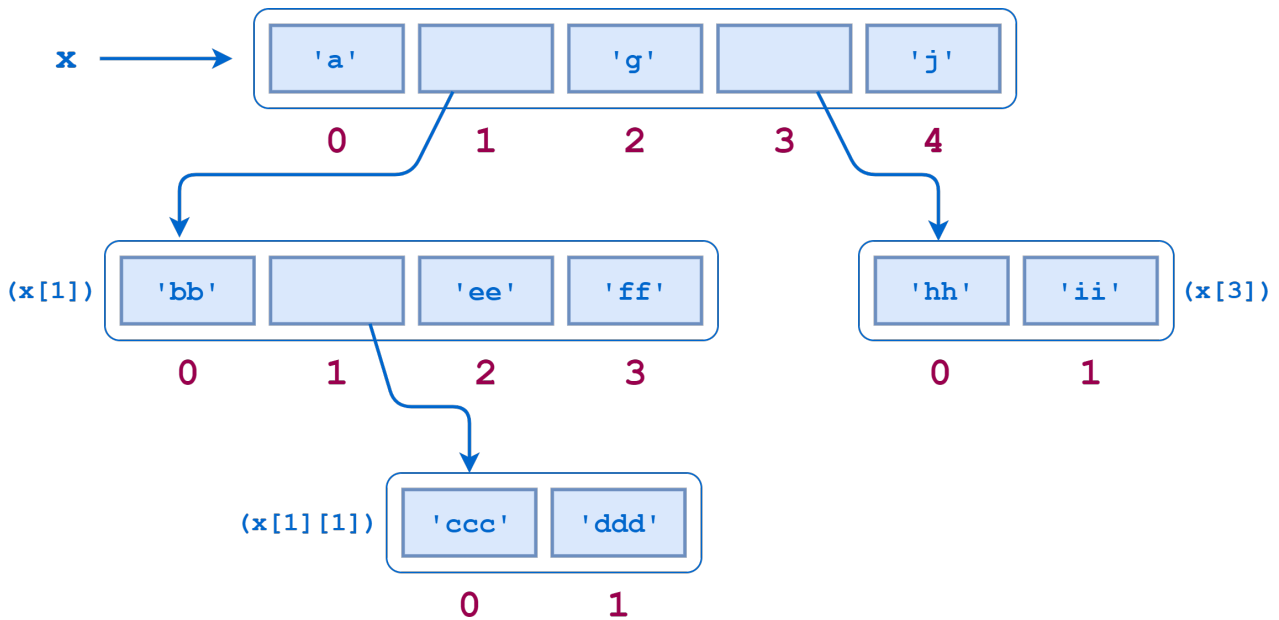
1. Matrizes
2. Inicialização de Matrizes
3. Entrada e Saída
4. Exemplos com Matrizes
5. Matrizes Multidimensionais
6. Referências

Listas Multidimensionais

Em **Python**, **listas** podem conter qualquer tipo de objeto, inclusive outras listas:

- Exemplo:

```
1 x = ['a', ['bb', ['ccc', 'ddd'], 'ee', 'ff'], 'g', ['hh', 'ii'], 'j']
```



Matrizes

Podemos usar uma lista de listas para representar uma **matriz 2D**:

- Cada elemento da lista contém uma linha da matriz;

```
>>> M = [[ 1,  2,  3], [ 4,  5,  6], [ 7,  8,  9], [10, 11, 12]]
```

- Cada linha corresponde a uma lista com os elementos das colunas da matriz.

```
>>> M = [[ 1,  2,  3],
         [ 4,  5,  6],
         [ 7,  8,  9],
         [10, 11, 12]]
```

```
>>> type(M)
<class 'list'>
>>> type(M[0])
<class 'list'>
```

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix}$$

$M_{4\times 3}$

Matrizes

O acesso aos valores M_{ij} é feito da seguinte forma: `[linha][coluna]`

- `M[i][j]` acessa a linha i e coluna j.

```
>>> M = [[ 1,  2,  3],
         [ 4,  5,  6],
         [ 7,  8,  9],
         [10, 11, 12]]
```

```
>>> M[0][0]
1
>>> M[3][2] = 0
```

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 0 \end{bmatrix}$$

$M_{4\times 3}$

Matrizes

Cuidado para não acessar uma posição inválida da matriz:

- `IndexError:`

```
>>> M = [[ 1,  2,  3],
         [ 4,  5,  6],
         [ 7,  8,  9],
         [10, 11, 12]]
```

```
>>> M[3][3]
Traceback (most ... last):
  ...
IndexError: list index out of range
```

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix}$$

$M_{4\times 3}$

Matrizes

Em **Python** nem todas as linhas de uma matriz precisam ter o mesmo número de colunas.

- `IndexError:`

```
>>> M = [[ 1],
         [ 4,  5],
         [ 7,  8, 9]]
```

```
>>> M[0][0]
1
>>> M[0][1]
Traceback (most ... last):
  ...
IndexError: list index out of range
```

$$M = \begin{bmatrix} 1 & & \\ 4 & 5 & \\ 7 & 8 & 9 \end{bmatrix}$$

- Vamos ver exemplos com matrizes completas.

Roteiro

1. Matrizes
2. Inicialização de Matrizes
3. Entrada e Saída
4. Exemplos com Matrizes
5. Matrizes Multidimensionais
6. Referências

Inicialização de matrizes

Vamos ver como criar uma matriz de dimensões $l \times c$ inicialmente vazia com compreensão de listas.

```
>>> M = [[] for i in range(3)] # matriz com 3 listas vazias
>>> M
[[],[],[]]
```

- Cada lista interna representa uma linha da matriz.

```
>>> M[0] = [1, 2, 'a', 'b']
>>> M[1] = [3, 4, 'c', 'd']
>>> M[2] = [5, 6, 'e', 'f']
>>>
```

$$\begin{bmatrix} 1 & 2 & \text{`a'} & \text{`b'} \\ 3 & 4 & \text{`c'} & \text{`d'} \\ 5 & 6 & \text{`e'} & \text{`f'} \end{bmatrix}$$

$M_{3\times 4}$

Inicialização de matrizes

Podemos criar uma matriz de dimensões $l \times c$ inicialmente com valores zero:

```
1 >>> M = [[0 for j in range(4)] for i in range(3)] # matriz 3 x 4
2 >>> M
3 [[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]
```

- Depois podemos modificar um elemento por vez na matriz.

```
1 >>> M[0][0] = 1
2 >>> M[1][1] = 2
3 >>> M[2][2] = 3
4 >>> M[2][3] = 4
```

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

$M_{3\times4}$

Roteiro

1. Matrizes
2. Inicialização de Matrizes
3. Entrada e Saída
4. Exemplos com Matrizes
5. Matrizes Multidimensionais
6. Referências

Leitura do teclado

Após inicializar uma matriz com valores zero, podemos ler os valores do teclado:

```
>>> M = [[0 for j in range(4)] for i in range(3)] # matriz 3 x 4
>>> M
[[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]
```

- `input()` os valores do teclado:

```
>>> M[0][0] = input() # 1
>>> M[1][1] = input() # 2
>>> M[2][2] = input() # 3
>>> M[2][3] = input() # 4
```

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

$M_{3\times4}$

Leitura do teclado

Para ler todos os valores do teclado, vamos utilizar um laço aninhado:

```
>>> M = [[0 for j in range(4)] for i in range(3)] # matriz 3 x 4
>>> M
[[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]
```

- `input()` os valores do teclado:

```
#para cada linha
for i in range(3):
  #para cada coluna
  for j in range(4):
    M[i][j] = int(input())
```

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

$M_{3\times4}$

Leitura do teclado

Outra opção é criar uma lista vazia, ler os valores do teclado, e ir adicionando uma linha por vez na Matriz:

```
>>> M = [] # matriz 3 x 4
```

- append() as novas linhas:

```
#para cada linha
for i in range(3):
  linha = []
  #para cada coluna
  for j in range(4):
    valor = int(input())
    linha.append(valor)
  #adiciona linha i
  M.append(linha)
```

$$\begin{bmatrix} - & - & - & - \\ - & - & - & - \\ - & - & - & - \end{bmatrix}$$

$M_{3\times4}$

Leitura do teclado

Caso o usuário digite todos os valores de uma linha, ainda assim podemos criar uma Matriz $M_{3\times4}$:

```
>>> M = [] # matriz 3 x 4
```

- `split()` divide a entrada em uma lista:

```
1 2 3 4
5 6 7 8
9 10 11 12
```

```
#para cada linha
for i in range(4):
  linha = [int(x) for x in input.split()]
  #adiciona linha i
  M.append(linha)
```

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 5 & 7 & 8 \\ 7 & 10 & 11 & 12 \end{bmatrix}$$

$M_{3\times4}$

Imprimindo na tela

É fácil imprimir uma matriz na tela como uma lista de listas:

- `print()`:

```
>>> M = [[ 1,  2,  3],
         [ 4,  5,  6],
         [ 7,  8,  9],
         [10, 11, 12]]
```

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

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix}$$

$M_{4\times 3}$

Imprimindo na tela

Para imprimir em um formato matricial, primeiro vamos descobrir as dimensões da matriz:

- `len()`:

```
>>> M = [[ 1,  2,  3],
         [ 4,  5,  6],
         [ 7,  8,  9],
         [10, 11, 12]]
```

```
>>> len(M)     # número de linhas
4
>>> len(M[0]) # número de colunas
3
```

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix}$$

$M_{4\times 3}$

- Se a matriz estiver vazia, a linha 0 não existe, portanto, `len(M[0])` resulta em um erro.

Imprimindo na tela

Em seguida, criamos um laço aninhado para percorrer todas as linhas e colunas:

- `for:`

```
>>> M = [[ 1,  2,  3],
         [ 4,  5,  6],
         [ 7,  8,  9],
         [10, 11, 12]]
```

```
def imprimeMatriz(M):
  linhas = len(M)
  colunas = len(M[0])
  #para cada linha
  for i in range(linhas):
    #para cada coluna
    for j in range(colunas):
      print(M[i][j], end="\t")
    #pula linha
    print()
```

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix}$$

$M_{4\times 3}$

Roteiro

1. Matrizes
2. Inicialização de Matrizes
3. Entrada e Saída
4. Exemplos com Matrizes
5. Matrizes Multidimensionais
6. Referências

Exemplo 1: Soma de Matrizes

Escreva uma função em **Python** que receba 2 matrizes e calcule a soma entre elas:

$$\underbrace{\begin{bmatrix} 0 & 1 & 0 & 2 & 3 \\ 0 & 1 & 0 & 2 & 3 \\ 0 & 1 & 0 & 2 & 3 \\ 0 & 1 & 0 & 2 & 3 \\ 0 & 1 & 0 & 2 & 3 \end{bmatrix}}_{A_{5\times 5}} + \underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 \end{bmatrix}}_{B_{5\times 5}} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 2 & 3 \\ 1 & 2 & 1 & 3 & 4 \\ 0 & 1 & 0 & 2 & 3 \\ 2 & 3 & 2 & 4 & 5 \\ 3 & 4 & 3 & 5 & 6 \end{bmatrix}}_{C_{5\times 5}}$$

Exemplo 1: Soma de Matrizes

- Primeiro, temos que verificar se as matrizes possuem dimensões iguais:

```
1 def dimensoes(M):
2   linhas = len(M)
3   colunas = 0
4   if linhas > 0:
5     colunas = len(M[0])
6   #print("%dX%d" %(linhas, colunas))
7   return linhas, colunas
```

Exemplo 1: Soma de Matrizes

- Agora utilizando a função anterior:

```
1  def soma_matrizes(A, B):
2    if(dimensoes(A)!=dimensoes(B)):
3      return False
4    else:
5      C = []
6      linhas = len(A)
7      colunas = len(A[0])
8      for i in range(linhas):        # para cada linha
9        linha = []
10       for j in range(colunas):  # para cada coluna
11         linha.append(A[i][j] + B[i][j])
12       #adiciona linha
13       C.append(linha)
14     #resultado
15     return C
```

```
1  A = [[1, 2, 3], [4, 5, 6]]
2  B = [[2, 3, 4], [5, 6, 7]]
3  soma_matrizes(A, B)
4  >>> [[3, 5, 7], [9, 11, 13]]
```

Exemplo 2: Transposta de Matrizes

Escreva uma função em **Python** que receba todos os elementos de uma matriz `A` com dimensões $l \times c$ e retorne a sua transposta A^T.

$$\underbrace{\begin{bmatrix} 0 & 1 & 2 \\ 0 & 1 & 2 \\ 0 & 1 & 2 \\ 0 & 1 & 2 \end{bmatrix}}_{A_{4\times 3}} \qquad \underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \end{bmatrix}}_{A^T_{3\times 4}}$$

Exemplo 2: Transposta de Matrizes

- Vamos criar uma matriz com valores zero com as dimensões (número de linhas e colunas) invertidas:

```
1  def transposta(A):
2    l, c = dimensoes(A)
3    #inicializa a matriz com zeros
4    #No caso, l colunas [0, 0, 0, ..., 0]
5    #c vezes == c linhas
6    transposta = [[0]*l for i in range(c)]
7    for i in range(l):
8      for j in range(c):
9        transposta[j][i] = A[i][j]
10   return transposta
```

```
1  >>> A = [[0,  1,  2],
2           [0,  1,  2],
3           [0,  1,  2],
4           [0,  1,  2]]
5
6  >>> imprimeMatriz(transposta(A))
```

Exemplo 3: Matrizes Simétricas

Escreva uma função em **Python** que receba todos os elementos de uma matriz `A` com dimensões $l \times c$ e verifica se `A` é simétrica.

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 5 & 4 \\ 3 & 4 & 7 \end{bmatrix}$$

$A_{3\times 3}$

Exemplo 3: Matrizes Simétricas

- Se a matriz A não for quadrada, ela não pode ser simétrica.

```
1 def matrizQuadrada(A):
2   l, c = dimensoes(A)
3   if (l==c):
4     return True
5   else:
6     return False
```

Exemplo 3: Matrizes Simétricas

- Se a matriz $A = A^T$, então ela é simétrica:

$$\underbrace{\begin{bmatrix} 1 & 2 & 3 \\ 2 & 5 & 4 \\ 3 & 4 & 7 \end{bmatrix}}_{A_{3\times 3}} \quad \underbrace{\begin{bmatrix} 1 & 2 & 3 \\ 2 & 5 & 4 \\ 3 & 4 & 7 \end{bmatrix}}_{A^T_{3\times 3}}$$

Exemplo 3: Matrizes Simétricas

- Utilizando as funções anteriores:

```
1 def simetrica(A):
2   if(matrizQuadrada(A)==False):
3     return False
4
5   T = transposta(A)
6   if (A==T):
7     return True
8   else:
9     return False
```

```
1 A = [[1, 2, 3],
2      [2, 3, 4],
3      [1, 4, 7]]
4 print("A = ")
5 imprimeMatriz(A)
6 print("A^T = ")
7 imprimeMatriz(transposta(A))
8 print(simetrica(A))
```

Exemplo 4: Linhas e Colunas nulas

Escreva uma função que receba uma matriz $A_{l \times c}$ de números inteiros e retorna o número de linhas e colunas que tem **apenas zeros**.

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

$A_{4 \times 5}$

```
contagemZeros(M)

>> Linhas  nulas = 2
>> Colunas nulas = 3
```

Exemplo 4: Linhas e Colunas nulas

- Vamos utilizar uma variável acumuladora `soma` para guardar o **número de zeros** em cada linha e coluna.

```
1  def contagemZeros(M):
2    l, c = dimensoes(M)
3    linhas_nulas = 0
4    colunas_nulas = 0
5    ##
6    #para cada linha
7    for i in range(l):
8      sum=0
9      for item in M[i]:
10       sum+=item
11     if(sum==0): linhas_nulas+=1
12   ##
13   #para cada coluna
14   for  j in range(c):
15     sum=0
16     for i in range(l):
17       sum += M[i][j]
18     if(sum==0): colunas_nulas+=1
19
20   return linhas_nulas, colunas_nulas
```

Exemplo 4: Linhas e Colunas nulas

- Podemos testar com `A`:

```
A = [[0, 0, 0, 0, 1],
     [0, 0, 0, 0, 0],
     [0, 1, 0, 0, 0],
     [0, 0, 0, 0, 0]]
```

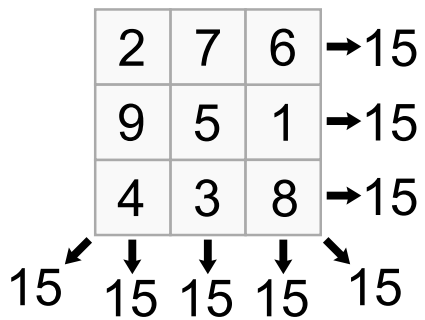
```
contagemZeros(A)

>> Linhas  nulas = 2
>> Colunas nulas = 3
```

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

$A_{4\times 5}$

Exemplo 5: Quadrado Mágico

Dizemos que uma matriz quadrada é um **quadrado mágico** se a somas dos elementos de cada linha, de cada coluna e das diagonais são todas iguais.



- Dada uma matriz $A_{l \times c}$, escreva uma função em **Python** que verifique se `A` é um quadrado mágico.

Exemplo 5: Quadrado Mágico

- Primeiro verificamos se a matriz é quadrada:

```
1 def quadradoMagico(A):
2   if(matrizQuadrada(A)==False):
3     return False
4
5   l, c = dimensoes(A)
```

- Depois, guardamos a soma das linhas em uma lista:

```
1   #soma das linhas
2   somaLinhas = []
3   for i in range(l):
4     soma = 0
5     for j in range(c):
6       soma+=A[i][j]
7     somaLinhas.append(soma)
8   print(somaLinhas)
```

$$\begin{bmatrix} 2 & 7 & 6 \\ 9 & 5 & 1 \\ 4 & 3 & 8 \end{bmatrix}$$

$A_{3\times 3}$

Exemplo 5: Quadrado Mágico

- O mesmo para a soma das colunas:

```
1  #soma das colunas
2  somaColunas = []
3  for j in range(c):
4    soma = 0
5    for i in range(l):
6      soma+=A[i][j]
7    somaColunas.append(soma)
8  print(somaColunas)
```

$$\begin{bmatrix} 2 & 7 & 6 \\ 9 & 5 & 1 \\ 4 & 3 & 8 \end{bmatrix}$$

$A_{3\times 3}$

Exemplo 5: Quadrado Mágico

- E para as diagonais

```
1   #soma das diagonal 1
2   somaDiagonal1 = 0
3   for i in range(l):
4     somaDiagonal1 += A[i][i]
5   print(somaDiagonal1)
6
7   #soma das diagonal 2
8   somaDiagonal2 = 0
9   for i in range(l):
10    somaDiagonal2 += A[i][c-i-1]
11  print(somaDiagonal2)
```

$$\begin{bmatrix} 2 & 7 & 6 \\ 9 & 5 & 1 \\ 4 & 3 & 8 \end{bmatrix}$$

$A_{3\times 3}$

Exemplo 5: Quadrado Mágico

- Finalmente, verificamos se todas as somas são iguais:

```
    #verifica todas as somas
    soma = somaDiagonal2
    correto = True
    for item in somaLinhas:
        if(item != soma): return False
    for item in somaColunas:
        if(item != soma): return False
    if(somaDiagonal1 != soma): return False

    #se não falhou em nenhuma comparação
    return True
```

```
A = [[2, 7, 6],
     [9, 5, 1],
     [4, 3, 8]]
print(quadradoMagico(A))
```

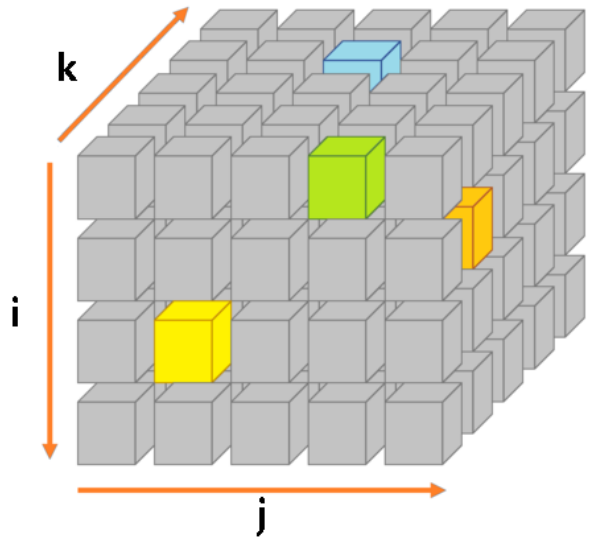
Roteiro

1. Matrizes
2. Inicialização de Matrizes
3. Entrada e Saída
4. Exemplos com Matrizes
5. Matrizes Multidimensionais
6. Referências

Matrizes Multidimensionais

Em **Python**, podemos implementar Matrizes Multidimensionais como generalizações de listas de listas.

```
1 M=[[[1, 0, 0, 0, -1], [2, 0, 0, 0, -2], [3, 0, 0, 0, -3], [4, 0, 0, 0, -4]],
2    [[5, 0, 0, 0, -5], [6, 0, 0, 0, -6], [7, 0, 0, 0, -7], [8, 0, 0, 0, -8]]]
```



`M[k][i][j]`

Matrizes Multidimensionais

Descobrindo as dimensões de uma matriz 3D:

```
M=[[[1, 0, 0, 0, -1], [2, 0, 0, 0, -2], [3, 0, 0, 0, -3], [4, 0, 0, 0, -4]],
   [[5, 0, 0, 0, -5], [6, 0, 0, 0, -6], [7, 0, 0, 0, -7], [8, 0, 0, 0, -8]]]
```

```
>>> len(M)      #profundidade
2
>>> len(M[0])   #linhas
4
>>> len(M[0][0]) #colunas
5
```

$$\begin{bmatrix} 5 & 0 & 0 & 0 & -5 \\ 6 & 0 & 0 & 0 & -6 \\ 7 & 0 & 0 & 0 & -7 \\ 8 & 0 & 0 & 0 & -8 \end{bmatrix}$$

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

$M_{2\times4\times5}$

Matrizes Multidimensionais

Vamos ver como imprimir os valores da matriz 3D:

```
M=[[[1, 0, 0, 0, -1], [2, 0, 0, 0, -2], [3, 0, 0, 0, -3], [4, 0, 0, 0, -4]],
   [[5, 0, 0, 0, -5], [6, 0, 0, 0, -6], [7, 0, 0, 0, -7], [8, 0, 0, 0, -8]]]
```

```
def imprimeMatriz3D(M):
  p = len(M)
  l = len(M[0])
  c = len(M[0][0])

  for k in range(p):
    for i in range(l):
      for j in range(c):
        print(M[k][i][j], end=" ")
      print()
    print()
```

$$\begin{bmatrix} 5 & 0 & 0 & 0 & -5 \\ 6 & 0 & 0 & 0 & -6 \\ 7 & 0 & 0 & 0 & -7 \\ 8 & 0 & 0 & 0 & -8 \end{bmatrix}$$

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

$M_{2\times4\times5}$

Matrizes Multidimensionais

Um exemplo em que utilizamos uma **matriz 3D**:

- Armazenar a quantidade de chuva em um dado **ano**, **mês** e **dia**, para cada um dos últimos anos:

```
1 M[2021][4][31] = 6.0
```

- Antes, precisamos inicializar a matriz:

```
1 #colunas, linhas e profundidade
2 M = [[[0 for i in range(32)] for j in range(13)] for k in range(9999)]
```

Fim

Dúvidas?

Leitura complementar

Leitura complementar:

1. panda.ime.usp.br/cc110/static/cc110/12-matrizes.html
2. panda.ime.usp.br/pensepy/static/pensepy/09-Listas/listas.html#listas-aninhadas
3. Tutorial do numpy: https://docs.scipy.org/doc/numpy-dev/user/quickstart.html

Roteiro

1. Matrizes
2. Inicialização de Matrizes
3. Entrada e Saída
4. Exemplos com Matrizes
5. Matrizes Multidimensionais
6. Referências

Referências

1. Materiais adaptados dos slides do Prof. Eduardo C. Xavier, da Universidade Estadual de Campinas.