Comandos Python: Matrizes

Agenda

- Matrizes
  - Matrizes e vetores multidimensionais
  - Criando matrizes
  - Acessando dados de uma matriz
  - Declarando vetores multidimensionais
- Exemplo com matrizes
- Exercícios
- Informações extras: NumPy
  - O tipo Array

Matrizes e Vetores Multidimensionais

- Suponha por exemplo que devemos armazenar as notas de cada aluno(a) (provas, laboratórios, atividades conceituais, exame), na matéria de PRAP.
- Assumindo que um(a) aluno(a) é avaliado(a) com 4 notas, seria necessário um vetor de 4 posições para guardar as notas de um(a) aluno(a).

Matrizes e Vetores Multidimensionais

- Contudo, assumindo que uma turma tem 130 aluno(a)s, seria necessário uma matriz bidimensional para guardar as notas de todo(a)s o(a)s alunos de uma turma.

notas

aluno(a)s	0	1	2	3
0	9.5	9	8	0
1	5	6	7	8.5
2	9	8	8	0

..


```
turma = [[9.5, 9, 8, 0], [5, 6, 7, 8.5], [9, 8, 8, 0],
         [3.6, 7.0, 9.1, 8.7], [5.0, 4.5, 7.0, 5.2],
         [2.1, 6.5, 8.0, 7.0], ... ]
```

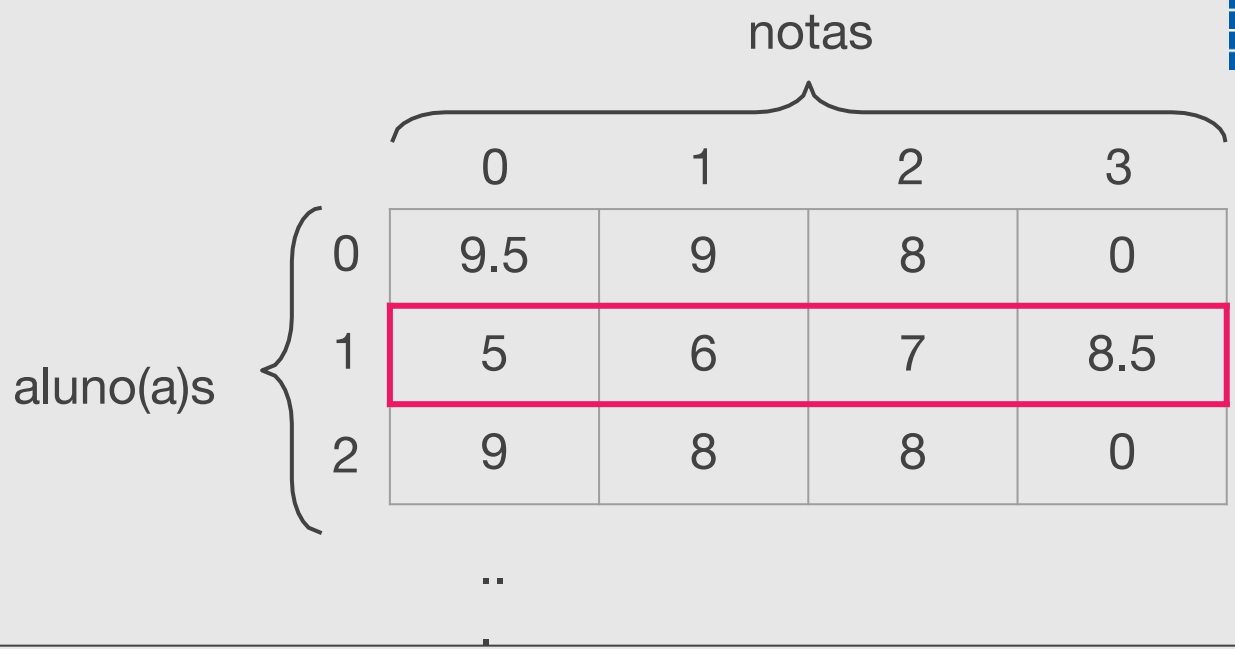
notas

aluno(a)s	0	1	2	3
0	9.5	9	8	0
1	5	6	7	8.5
2	9	8	8	0

..

.

```
turma = [[9.5, 9, 8, 0], [5, 6, 7, 8.5], [9, 8, 8, 0],
         [3.6, 7.0, 9.1, 8.7], [5.0, 4.5, 7.0, 5.2],
         [2.1, 6.5, 8.0, 7.0], ... ]
```




```
turma = [[9.5, 9, 8, 0], [5, 6, 7, 8.5], [9, 8, 8, 0],
         [3.6, 7.0, 9.1, 8.7], [5.0, 4.5, 7.0, 5.2],
         [2.1, 6.5, 8.0, 7.0], ... ]
```


```
turma = [[9.5, 9, 8, 0], [5, 6, 7, 8.5], [9, 8, 8, 0],
         [3.6, 7.0, 9.1, 8.7], [5.0, 4.5, 7.0, 5.2],
         [2.1, 6.5, 8.0, 7.0], ... ]
```

notas

aluno(a)s	0	1	2	3
0	9.5	9	8	0
1	5	6	7	8.5
2	9	8	8	0

..
.

```
turma = [[9.5, 9, 8, 0], [5, 6, 7, 8.5], [9, 8, 8, 0],
         [3.6, 7.0, 9.1, 8.7], [5.0, 4.5, 7.0, 5.2],
         [2.1,
type(turma)
```

```
list
```

Acesso aos Valores: [linha][coluna]

- Segunda nota do primeiro aluno(a): `turma[0][1]`
- Quarta nota do terceiro aluno(a): `turma[2][3]`

Declarando uma Matriz com Listas

- Para criar uma matriz de dimensões *l* x *c* inicialmente vazia podemos utilizar listas.
- Exemplo de uma matriz 3 x 4 inicialmente vazia:

```
mat = [[] for i in range(3)]
#dentro da lista externa cria-se vazia 3 listas []
mat
[[], [], []]
```

- Note que cada lista interna representa uma linha da matriz, e seu tamanho pode ser 4 ou qualquer outro valor.

Exemplo de Declaração de Matriz

- Criar matriz 3 x 4 onde cada posição (*i* , *j*) contém o valor de *i* * *j*.

	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	4	6

Exemplo de Declaração de Matriz

- Criar matriz 3 x 4 onde cada posição (*i* , *j*) contém o valor de *i* * *j*.

```
mat = []
for i in range(3): # para cada linha de 0 ate 2
    l = [] # linha começa vazia
    for j in range(4): # para cada coluna de 0 ate 3
        l.append(i*j) # preenche colunas da linha i
    mat.append(l) # adiciona linha na matriz
print(mat)
```

```
[[0, 0, 0, 0], [0, 1, 2, 3], [0, 2, 4, 6]]
```

Exemplo de Declaração de Matriz

- Obtendo o mesmo resultado utilizando compreensão de listas:

```
mat = [[i*j for j in range(4)] for i in
range(3)]  print(mat)
```

```
[[0, 0, 0, 0], [0, 1, 2, 3], [0, 2, 4, 6]]
```

Acessando os Dados da Matriz

```
nome_da_matriz[linha][coluna]
```

- Ex: `matriz[1][10]`: refere-se à variável na 2a linha e na 11a coluna da matriz.
- Lembre-se que, como a matriz está implementada com listas, a primeira posição em uma determinada dimensão começa no índice 0.
- O acesso a posições inválidas causa um erro de execução.

Declarando Vetores Multidimensionais

- Ainda para o exemplo da turma, assumindo que um curso tem duas turmas, seria necessário uma matriz tridimensional para guardar as notas de todo(a)s o(a)s aluno(a)s de todas as turmas do curso.

Declarando Vetores Multidimensionais


```
turmas = [[[9.5, 9, 8, 0], [5, 6, 7, 8.5], [9, 8, 8, 0]],
          [[2, 8, 10, 5], [8.6, 4.3, 7, 8.5], [8, 3, 5.7, 7.9]]]
```


```
turmas = [[[9.5, 9, 8, 0], [5, 6, 7, 8.5], [9, 8, 8, 0]],
          [[2, 8, 10, 5], [8.6, 4.3, 7, 8.5], [8, 3, 5.7, 7.9]]]
```

turma


```
turmas = [[[9.5, 9, 8, 0], [5, 6, 7, 8.5], [9, 8, 8, 0]],
          [[2, 8, 10, 5], [8.6, 4.3, 7, 8.5], [8, 3, 5.7, 7.9]]]
```

aluno(a)


```
turmas = [[[9.5, 9, 8, 0], [5, 6, 7, 8.5], [9, 8, 8, 0]],
          [[2, 8, 10, 5], [8.6, 4.3, 7, 8.5], [8, 3, 5.7, 7.9]]]
print(turmas[0][0][2]
)   8
```

Exemplos com Matrizes

- Criar programas com operações básicas sobre matrizes quadradas
  - Soma de 2 matrizes com dimensões *n* x *n*.
  - Subtração de 2 matrizes com dimensões *n* x *n*.
  - Cálculo da transposta de uma matriz de dimensão *n* x *n*.
  - Multiplicação de 2 matrizes com dimensões *n* x *n*.

Exemplos com Matrizes

- Primeiramente vamos implementar o código para fazer a leitura e a impressão de uma matriz.

```
def leMatriz(dimensao):
  mat = [[] for i in range(dimensao)]
  for i in range(dimensao):
    for j in range(dimensao):
      num = int(input("("+str(i+1)+","+str(j+1)+"): "))
      mat[i].append(num)
  return mat
```

Exemplos com Matrizes

- Primeiramente vamos implementar o código para fazer a leitura e a impressão de uma matriz.

```
def imprimeMatriz(mat):
  for linha in mat:
    for numero in linha:
      print(numero, end=" ")
      #imprime números na mesma linha separados por espaço
    print() #apos impressão de uma linha, pula uma linha
```

Exemplos com Matrizes

- Primeiramente vamos implementar o código para fazer a leitura e a impressão de uma matriz.

```
def imprimeMatriz(mat):
  for linha in mat:
    for numero in linha:
      print(numero, end=" ")
      #imprime números na mesma linha separados por espaço
    print() #apos impressão de uma linha, pula uma linha
```

```
dimensao = int(input("Digite a dimensão da matriz: "))
mat = leMatriz(dimensao)
imprimeMatriz(mat)
```

Exemplos com Matrizes

- Vamos implementar o código para fazer a **soma** de duas matrizes.
- Para cada posição `(i,j)` fazemos:

```
mat3[i][j] = mat1[i][j] + mat2[i][j]
```

tal que o resultado da soma das matrizes estará em `mat3`.

Exemplos com Matrizes

```
def somaMatriz(mat1,
  mat2):  tam = len(mat1)
  mat3 = [[0 for j in range(tam)] for i in range(tam)]
  for i in range(tam):
    for j in range(tam):
      mat3[i][j] = mat1[i][j] + mat2[i][j]
  return mat3
```

Exemplos com Matrizes

- Vamos implementar o código para fazer a **multiplicação** de duas matrizes (de dimensão *n* x *n*).
- Uma posição `(i,j)` de `mat3` terá o produto interno do vetor linha `i` de `mat1` com o vetor coluna `j` de `mat2`:

$$\texttt{mat3[i][j]} = \sum_{\texttt{k=0}}^{\texttt{n-1}} \texttt{mat1[i][k]} * \texttt{mat2[k][j]}$$

Exemplos com Matrizes

- Vamos implementar o código para fazer a **multiplicação** de duas matrizes (de dimensão *n* x *n*).

```
def multiplicaMatriz(mat1,
  mat2):  tam = len(mat1)
  mat3 = [[0 for j in range(tam)] for i in range(tam)]
  for i in range(tam):
    for j in range(tam):
        mat3[i][j] = (mat1[i][j] * mat2[i][j])
  return mat3
```

Exemplos com Matrizes: O Programa

```
n = int(input("Dimensão das matrizes: "))
mat1 = leMatriz(n)
mat2 = leMatriz(n)
print("mat1: ")
imprimeMatriz(mat1)
print("mat2: ")
imprimeMatriz(mat2)
mat3Soma = somaMatriz(mat1, mat2)
print("Soma: ")
imprimeMatriz(mat3Soma)
mat3Mult = multiplicaMatriz(mat1, mat2)
print("Multiplicação: ")
imprimeMatriz(mat3Mult)
```

Exercícios

Exercícios

- Escreva um programa que leia todas as posições de uma matriz de dimensão 10x10. O programa deve então exibir o número de posições não nulas na matriz.

Exercícios: Jogo da Velha

- Faça um programa para o Jogo da Velha. Seu programa deve ler uma matriz 3x3 que representa um tabuleiro de jogo da velha.

Exercícios

Como exemplo de utilização de Vetores, podemos imaginar um programa responsável por realizar a venda de passagens para uma excursão, em que será utilizado um ônibus com 50 lugares.

O nosso programa será responsável por controlar a quantidade de poltronas vendidas e também por não permitir a venda de mais de uma passagem para a mesma poltrona.

Exercícios:

- Como exemplo de utilização de Matrizes, podemos utilizar uma situação em que se quer medir os tempos de um piloto de corrida na etapa de classificação.

  Conforme as regras do campeonato, cada piloto deve dar quatro voltas válidas.
  Na categoria em questão, temos quatro pilotos participantes.

NumPy

NumPy

- NumPy é uma biblioteca para Python que contém tipos para representar vetores e matrizes juntamente com diversas operações, dentre elas operações comuns de álgebra linear.
- NumPy é implementado para trazer maior eficiência do código em Python para aplicações científicas.

NumPy

- Primeiramente deve-se instalar o NumPy baixando-se o pacote de http://www.numpy.org
- Para usar os itens deste pacote deve-se importá-lo inicialmente com o comando: `import numpy`

```
C:\Users\marci>pip freeze

C:\Users\marci>pip install numpy
Collecting numpy
  Downloading numpy-2.1.1-cp312-cp312-win_amd64.whl.metadata (59 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 59.7/59.7 kB 798.4 kB/s eta 0:00:00
Downloading numpy-2.1.1-cp312-cp312-win_amd64.whl (12.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.6/12.6 MB 6.3 MB/s eta 0:00:00
Installing collected packages: numpy
Successfully installed numpy-2.1.1

[notice] A new release of pip is available: 24.0 -> 24.2
[notice] To update, run: python.exe -m pip install --upgrade pip

C:\Users\marci>pip freeze
numpy==2.1.1
```

NumPy

- O objeto mais simples da biblioteca é o **array** que serve para criar vetores homogêneos multidimensionais.
- Um **array** pode ser criado a partir de uma lista:

```
import numpy as np
a =
np.array([1,2,3])
print(a.ndim)
print(a.size)
print(a)
```

```
1
3
array([1,2,3])
```

NumPy

- O objeto mais simples da biblioteca é o **array** que serve para criar vetores homogêneos multidimensionais.
- Um **array** pode ser criado a partir de uma lista:

```
import numpy as np
b = np.array([[1,2,3],[4,5,6]])
print(b.ndim)
print(b.size)
b
```

```
2
6
array([[1, 2, 3],
       [4, 5, 6]])
```

NumPy

- Um **array** pode ser criado com mais do que uma dimensão utilizando as funções **arange** e **reshape**.

```
a = np.arange(10)
a
a =
np.arange(10).reshape(2,5)  a
```

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
array([[0, 1, 2, 3, 4],
       [5, 6, 7, 8, 9]])
```

NumPy

- NumPy oferece a função **zeros** que cria um **array** contendo apenas zeros. Seu argumento de entrada é uma tupla.

```
a=np.zeros((3))
a
a=np.zeros((3,4))
a
```

```
array([ 0., 0., 0.])
array([[ 0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.]])
```

NumPy

- Os operadores `*`, `-` , `+` , `/`, `**`, quando utilizados sob arrays, são aplicados em cada posição do **array**.

```
m = np.ones((2,3))
m=m+1
print(m)
```

```
array([[ 2.,  2.,  2.],
       [ 2.,  2.,  2.]])
```

NumPy

- Os operadores `*`, `-` , `+` , `/`, `**`, quando utilizados sob arrays, são aplicados em cada posição do **array**.

```
m=m*4
```

```
array([[ 4.,  4.,  4.],
       [ 4.,  4.,  4.]])
```

NumPy

- Os operadores `*`, `-` , `+` , `/`, `**`, quando utilizados sob arrays, são aplicados em cada posição do **array**.

```
m = m + 1
m = m**3
```

```
array([[ 8.,  8.,  8.],
       [ 8.,  8.,  8.]])
```

NumPy

- Os operadores `*`, `-` , `+` , `/`, `**`, quando utilizados sob arrays, são aplicados em cada posição do **array**.

```
A = np.array([[1,1], [0,1]])
B = np.array([[2,0], [3,4]])
C= A*B
print(C)
```

```
array([[2, 0],
       [0, 4]])
```

- Multiplicação de elemento por elemento.

NumPy

- Os operadores `*`, `-` , `+` , `/`, `**`, quando utilizados sob arrays, são aplicados em cada posição do **array**.

```
A = np.array([[1,1], [0,1]])
B = np.array([[2,0], [3,4]])
C = np.dot(A,B) # multiplicação de matrizes
print(C)
```

```
array([[5, 4],
       [3, 4]])
```

NumPy

- Na biblioteca existe uma variedade de outras funções como funções para calcular autovalores e autovetores, resolução de um sistema de equações lineares, etc.

Referências & Exercícios

- Os *slides* deste curso foram baseados nos slides produzidos e cedidos gentilmente pela Professora Sandra Ávila, do Instituto de Computação da Unicamp. Parte dos slides foram baseados no material de MC102 do Prof. Eduardo Xavier (IC/Unicamp)

- https://panda.ime.usp.br/aulasPython/static/aulasPython/aula11.html
- http://www.galirows.com.br/meublog/programacao/exercicios-resolvidos-python/

  (Exercícios resolvidos, explicação com vídeo)

Referências & Exercícios

- Os *slides* deste curso foram baseados nos slides produzidos e cedidos gentilmente pela Professora Sandra Ávila, do Instituto de Computação da Unicamp. Parte dos slides foram baseados no material de MC102 do Prof. Eduardo Xavier (IC/Unicamp)

- https://wiki.python.org.br/ExerciciosFuncoes