

Matrizes em Python

Afonso Paiva

apneto@icmc.usp.br

Fundamentos da Programação de Computadores – SME0332

Matrizes

- Os valores são acessados através de diferentes **índices**
- Semelhante as matrizes em Álgebra Linear:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{12} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

- Em Python, **listas** podem ser usadas para armazenar matrizes, como uma lista de listas de forma recursiva.

Motivação

- Suponha que **um aluno** é avaliado com **5 provas**, logo precisaríamos de **uma lista com 5 posições** para guardar as notas desse aluno.

	0	1	2	3	4
notas =	7.3	5.5	3.6	4.3	9.0

```
notas = [7.3, 5.5, 3.6, 4.3, 9.0]
```

Se a turma tivesse **4 alunos**, como guardar as notas de todos os alunos?

Motivação

- Usando uma **matriz** bidimensional (isto é, uma lista com 2 índices)
 - cada **linha** representa um aluno
 - cada **coluna** representa as notas de uma prova

		notas				
		0	1	2	3	4
alunos	0	7.3	5.5	3.6	4.3	9.0
	1	1.6	6.2	7.1	6.5	6.1
	2	6.6	7.0	5.9	8.2	5.0
	3	5.1	4.2	1.4	5.0	3.3

Matrizes em Python

		notas				
		0	1	2	3	4
alunos	0	7.3	5.5	3.6	4.3	9.0
	1	1.6	6.2	7.1	6.5	6.1
	2	6.6	7.0	5.9	8.2	5.0
	3	5.1	4.2	1.4	5.0	3.3

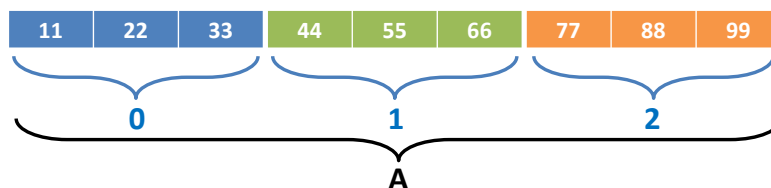
```
turma = [ [7.3,5.5,3.6,4.3,9.0], [1.6,6.2,7.1,6.5,6.1],  
          [6.6,7.0,5.9,8.2,5.0], [5.1,4.2,1.4,5.0,3.3] ]
```

Matrizes em Python

```
>>> A= [ [11,22,33], [44,55,66], [77,88,99] ]
```

$$A = \begin{pmatrix} 11 & 22 & 33 \\ 44 & 55 & 66 \\ 77 & 88 & 99 \end{pmatrix}$$

- Na memória uma matriz seria armazenada assim:



Matrizes: *acesso de elementos*

- Para acessar um elemento de uma matriz, basta informar os índices entre **colchetes**
 - **[i][j]**: acessa um elemento na linha *i* coluna *j*

```
>>> A[0][1]
```

```
22
```

```
>>> A[2][0]
```

```
77
```

	0	1	2
0	11	22	33
1	44	55	66
2	77	88	99

A

Matrizes: *acesso de elementos*

- Exemplo:** calcular a média da turma

```
turma = [ [7.3,5.5,3.6,4.3,9.0], [1.6,6.2,7.1,6.5,6.1],  
          [6.6,7.0,5.9,8.2,5.0], [5.1,4.2,1.4,5.0,3.3] ]  
  
media = 0  
# loop para percorrer as linhas  
for i in range(4):  
    # loop para percorrer as colunas  
    for j in range(5):  
        media += turma[i][j]  
  
media /= 20  
print(media)
```

Matrizes em Python

- **Exemplo:** criando uma matriz nula de ordem m X n

```
m = int(input('Quantidade de linhas: '))
n = int(input('Quantidade de colunas: '))
matriz = []

for i in range(m):
    linha = []
    for j in range(n):
        linha.append(0)
    matriz.append(linha)

print(matriz)
```

Matrizes em Python

- **Exemplo:** criando uma matriz nula de ordem m X n

```
m = int(input('Quantidade de linhas: '))
n = int(input('Quantidade de colunas: '))
matriz = []

for i in range(m):
    matriz.append([0]*n)

print(matriz)
```

Matrizes em Python

- **Exemplo:** criando uma matriz identidade de ordem n

```
n = int(input('Entre com a ordem da matriz: '))
matriz = []

for i in range(n):
    matriz.append([0]*n)
    matriz[i][i] = 1

# imprimir no formato de matriz
for i in range(n):
    print(matriz[i])
```

Matrizes em Python

- **Exemplo:** criando uma matriz identidade de ordem n

```
n = int(input('Entre com a ordem da matriz: '))
matriz = [[0]*n]*n

for i in range(n):
    matriz[i][i] = 1

# imprimir no formato de matriz
for i in range(n):
    print(matriz[i])
```

A resposta desse programa é uma matriz
cujos coeficientes são **todos iguais a 1!**
Por quê? (Python Tutor)



Matrizes em Python

- **Exemplo:** contando os números ímpares em uma matriz quadrada de ordem 3

```
matriz = []
for i in range(3):
    linha = []
    for j in range(3):
        linha.append(int(input('A['+str(i)+' '+str(j)+']:')))
    matriz.append(linha)

# contando os ímpares
impares = 0
for i in range(3):
    for j in range(3):
        if matriz[i][j] % 2:
            impares += 1

print('A matriz contém', impares, 'números ímpares')
```

Matrizes: *iterando sobre linhas*

- Lembra que o **for** pode iterar sobre uma lista?

```
lista = [1,2,3,4]
for i in lista:
    print(i)
```

- Com matrizes não é diferente, já que **cada linha é uma lista!**

Matrizes: *iterando sobre linhas*

- **Exemplo:** contando os números ímpares em uma matriz quadrada de ordem 3

```
matriz = []
for i in range(3):
    linha = []
    for j in range(3):
        linha.append(int(input('A['+str(i)+' '+str(j)+']:')))
    matriz.append(linha)

# contando os ímpares
impares = 0
for linha in matriz:
    for valor in linha:
        if valor % 2:
            impares += 1
print('A matriz contém', impares, 'números ímpares')
```

Matrizes: *misturando tipos*

- Python permite misturar tipos de dados em uma matriz
- **Exemplo:**
 - uma tabela de carros com suas respectivas quilometragens

Carro	Km
Gol	932
Uno	651
Palio	780
Fiesta	200
Civic	120

Matrizes: *misturando tipos*

- **Exemplo:** determinando o carro mais novo (menor Km)

```
carros = ['Gol', 'Uno', 'Palio', 'Fiesta', 'Civic']
Km = [932, 651, 780, 200, 120]
tabela = []
# monta a matriz
for i in range(5):
    L = []
    L.append(carros[i])
    L.append(Km[i])
    tabela.append(L)
# procura o carro com menor Km
menor, pos = tabela[0][1], 0
for i in range(5):
    if tabela[i][1] < menor:
        menor = tabela[i][1]
        pos = i
# imprime a matriz
for i in range(5):
    print(tabela[i])
print('O carro mais novo é o', tabela[pos][0])
```

Carro	Km
Gol	932
Uno	651
Palio	780
Fiesta	200
Civic	120

Copiando Matrizes

- Vamos precisar do módulo **copy** e da função **deepcopy**

```
>>> import copy
>>> A = [[1, 2], [3, 4]]
>>> B = copy.copy(A)      # cuidado, cópia rasa
>>> C = copy.deepcopy(A)  # ok, cópia profunda
>>> A[0][0] = 100
>>> A
[[100, 2], [3, 4]]
>>> B
[[100, 2], [3, 4]]
>>> C
[[1, 2], [3, 4]]
```

Matrizes n -dimensionais

- Uma matriz pode ter um número arbitrário de dimensões
 - Basta usar **um índice para cada dimensão**
- **Exemplo:**
 - Suponha que **cada turma tem 4 alunos** e **cada aluno é avaliado com 5 provas**. **E se tivéssemos 2 turmas?**

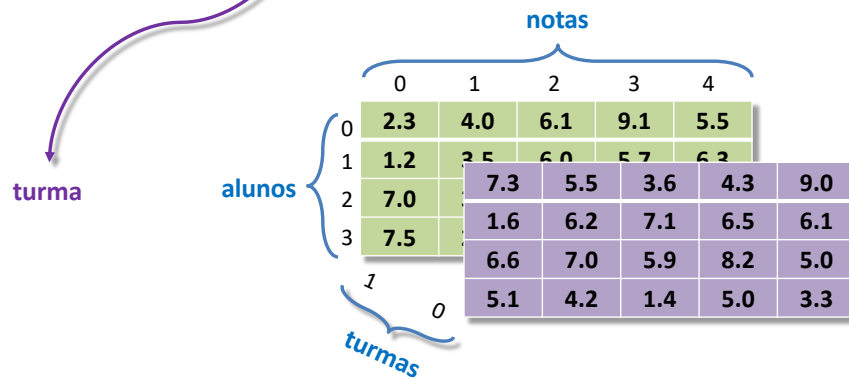
Matrizes n -dimensionais

```
>>> M = [[[7.3,5.5,3.6,4.3,9.0], [1.6,6.2,7.1,6.5,6.1],  
[6.6,7.0,5.9,8.2,5.0], [5.1,4.2,1.4,5.0,3.3]],  
[[2.3,4.0,6.1,9.1,5.5], [1.2,3.5,6.0,5.7,6.3],  
[7.0,3.1,2.9,6.6,3.3], [7.5,2.5,6.0,6.5,8.0]]]
```

		notas					
		0	1	2	3	4	
alunos	0	2.3	4.0	6.1	9.1	5.5	
	1	1.2	3.5	6.0	5.7	6.3	
	2	7.0	7.3	5.5	3.6	4.3	9.0
	3	7.5	1.6	6.2	7.1	6.5	6.1
	4		6.6	7.0	5.9	8.2	5.0
			5.1	4.2	1.4	5.0	3.3
		turmas					
		1	0				

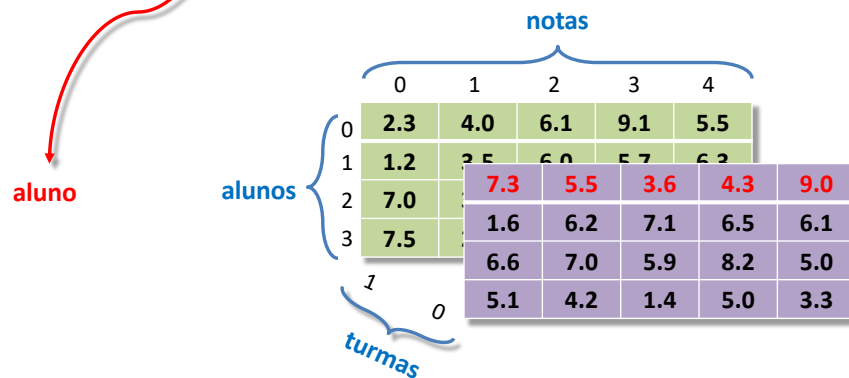
Matrizes n -dimensionais

```
>>> M = [[7.3, 5.5, 3.6, 4.3, 9.0], [1.6, 6.2, 7.1, 6.5, 6.1],  
[6.6, 7.0, 5.9, 8.2, 5.0], [5.1, 4.2, 1.4, 5.0, 3.3]],  
[[2.3, 4.0, 6.1, 9.1, 5.5], [1.2, 3.5, 6.0, 5.7, 6.3],  
[7.0, 3.1, 2.9, 6.6, 3.3], [7.5, 2.5, 6.0, 6.5, 8.0]]]
```



Matrizes n -dimensionais

```
>>> M = [[7.3, 5.5, 3.6, 4.3, 9.0], [1.6, 6.2, 7.1, 6.5, 6.1],  
[6.6, 7.0, 5.9, 8.2, 5.0], [5.1, 4.2, 1.4, 5.0, 3.3]],  
[[2.3, 4.0, 6.1, 9.1, 5.5], [1.2, 3.5, 6.0, 5.7, 6.3],  
[7.0, 3.1, 2.9, 6.6, 3.3], [7.5, 2.5, 6.0, 6.5, 8.0]]]
```



Matrizes n -dimensionais

```
>>> M = [[[7.3,5.5,3.6,4.3,9.0], [1.6,6.2,7.1,6.5,6.1],  
[6.6,7.0,5.9,8.2,5.0], [5.1,4.2,1.4,5.0,3.3]],  
[[2.3,4.0,6.1,9.1,5.5], [1.2,3.5,6.0,5.7,6.3],  
[7.0,3.1,2.9,6.6,3.3], [7.5,2.5,6.0,6.5,8.0]]]
```

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

