



ICC901 – Introdução à Programação de Computadores
IECo81 – Introdução à Ciência dos Computadores
IECo37 – Introdução à Programação de Computadores

Aula 07 – Matrizes

Atualização: 8/mar/19



Você tem a liberdade de:



Compartilhar: copiar, distribuir e transmitir esta obra.

Remixar: criar obras derivadas.

Sob as seguintes condições:



Atribuição: você deve creditar a obra da forma especificada pelo autor ou licenciante (mas não de maneira que sugira que estes concedem qualquer aval a você ou ao seu uso da obra).



Uso não comercial: você não pode usar esta obra para fins comerciais.

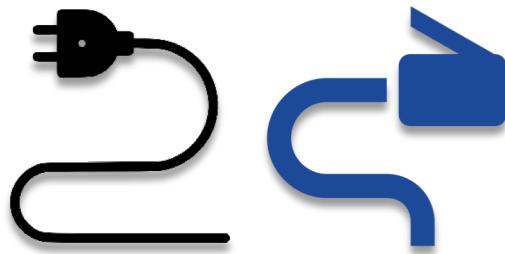


Compartilhamento pela mesma licença: se você alterar, transformar ou criar em cima desta obra, poderá distribuir a obra resultante apenas sob a mesma licença, ou sob uma licença similar à presente.

Conserve o laboratório



Os equipamentos são frágeis:
use-os com cuidado



Não mexa nos cabos



Não consuma alimentos ou bebidas.
Mantenha sua garrafa de água
tampada.

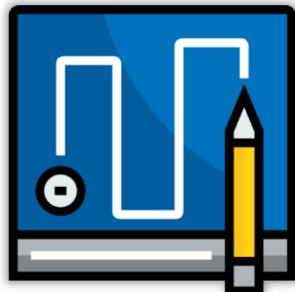
Antes de começar...



Está atento ao **calendário**?



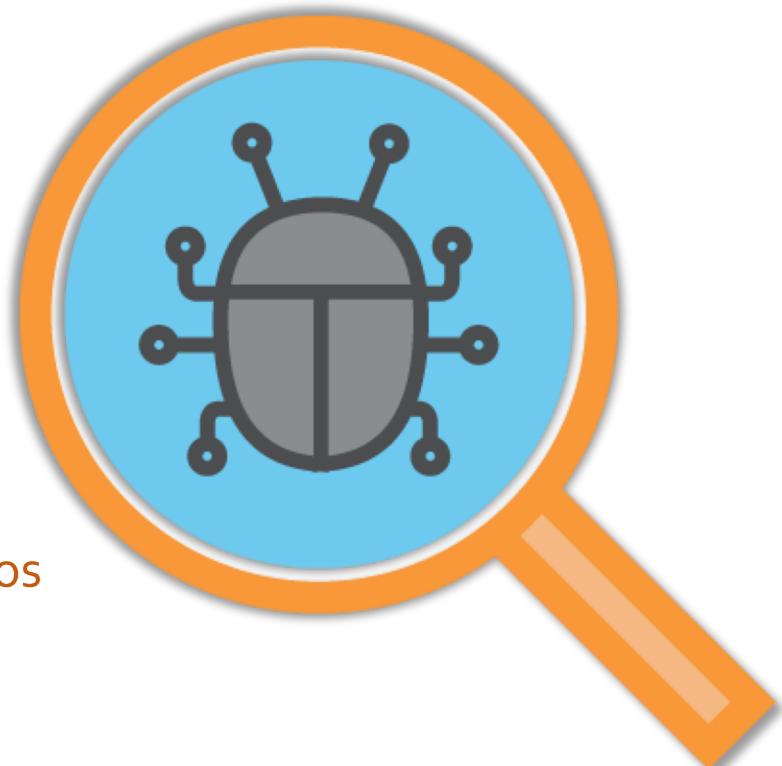
Este módulo tem **Peso 3** na avaliação



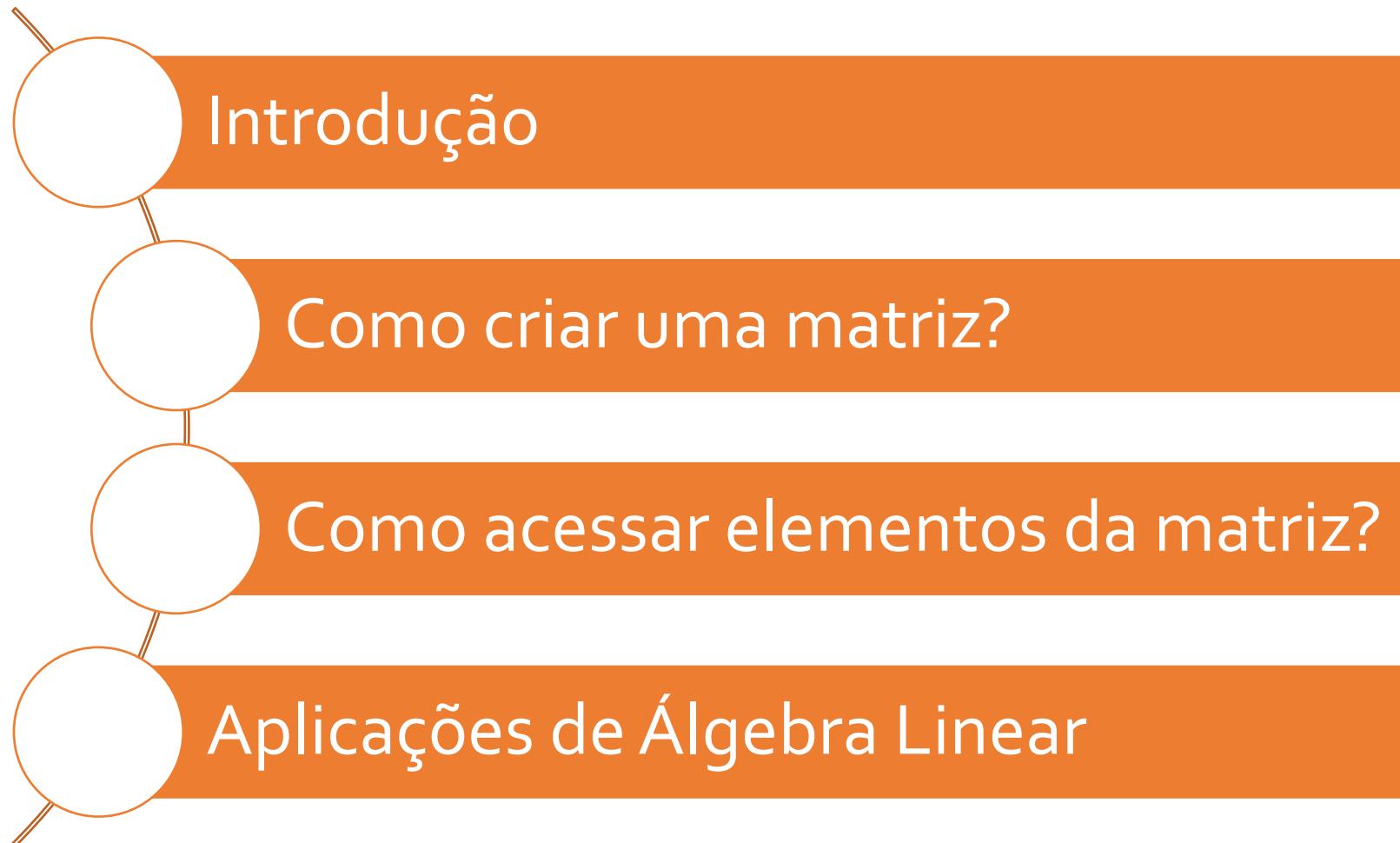
Como está sua **tática** de estudo?

Como lidar com erros de programação?

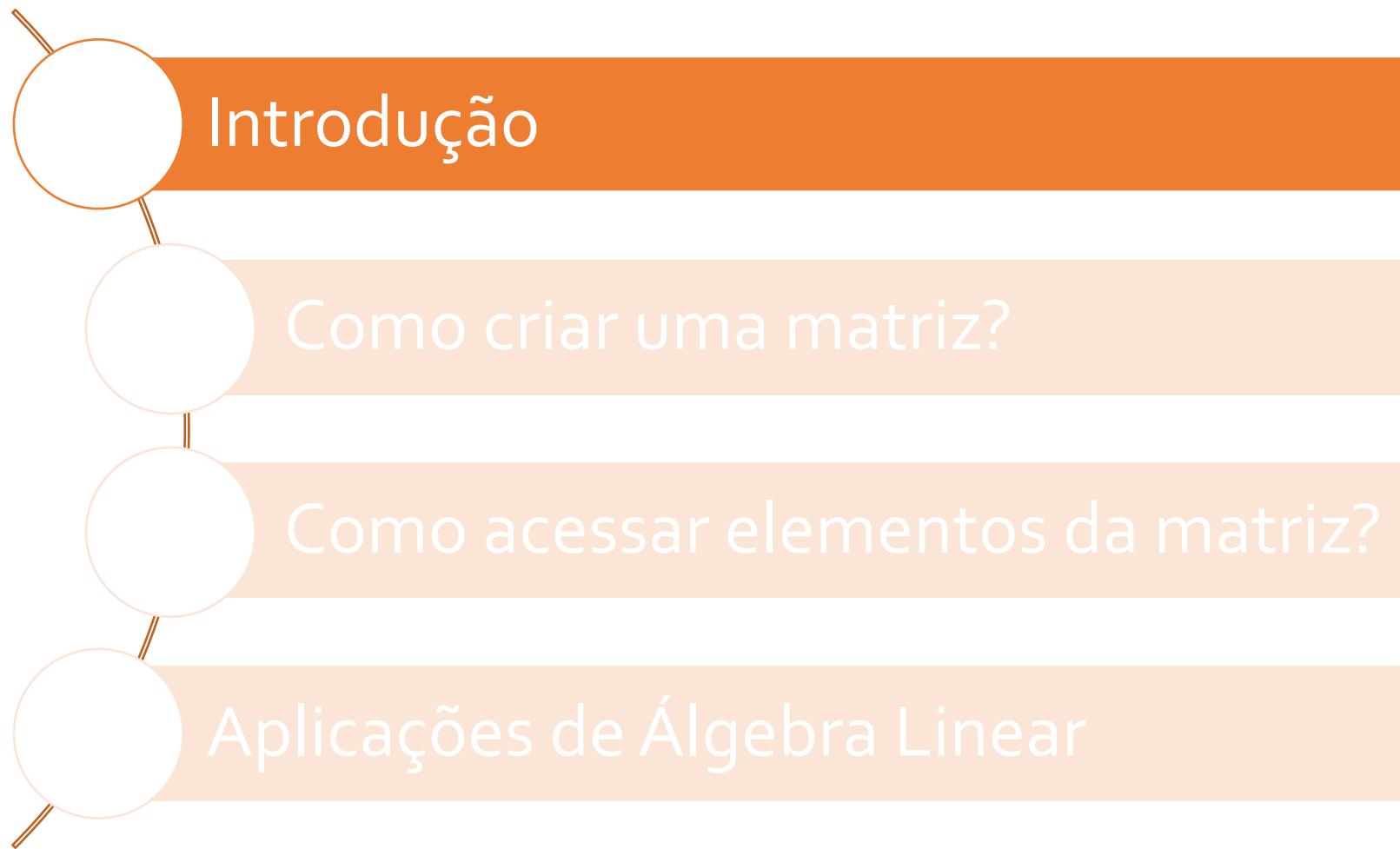
- 🛡️ Não se pergunte: “por que o programa não funciona?”
- 🛡️ Em vez disso, pergunte-se: “**por que o programa está funcionando deste jeito?**”
- 🛡️ Estratégias:
 - 🛡️ Não se limite ao exemplo: teste **vários** casos
 - 🛡️ Imprima resultados **intermediários**
 - 🛡️ Tente **explicar** o problema para outra pessoa
 - 🛡️ Dê um **tempo** e tente de novo mais tarde



Conteúdo

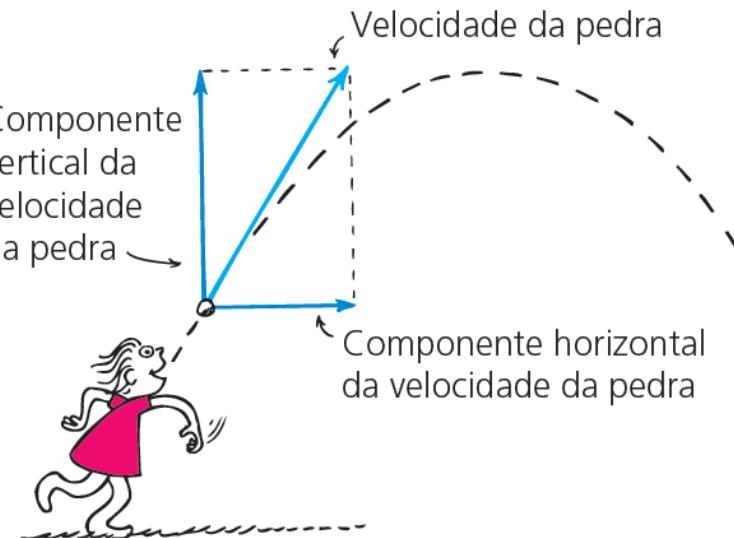


Conteúdo



Matrizes guardam dados bidimensionais

- 💡 Certos problemas exigem que os dados sejam organizados na forma de **matriz bidimensional**:
 - 💡 **Tempo de lançamento de um projétil e os respectivos componentes x e y da velocidade.**



Tempo	v_x	v_y
1	1.3	0.8
2	2.7	1.2
3	4.6	3.1
4	5.8	4.2
5	7.9	5.5

Matrizes

:: Notação Matemática

- Em notação matemática, em uma matriz A qualquer, cada elemento é indicado por a_{ij} .
 - O índice i indica a **linha**.
 - O índice j indica a **coluna**.

$$A_{m \times n} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

Nº de linhas Nº de colunas

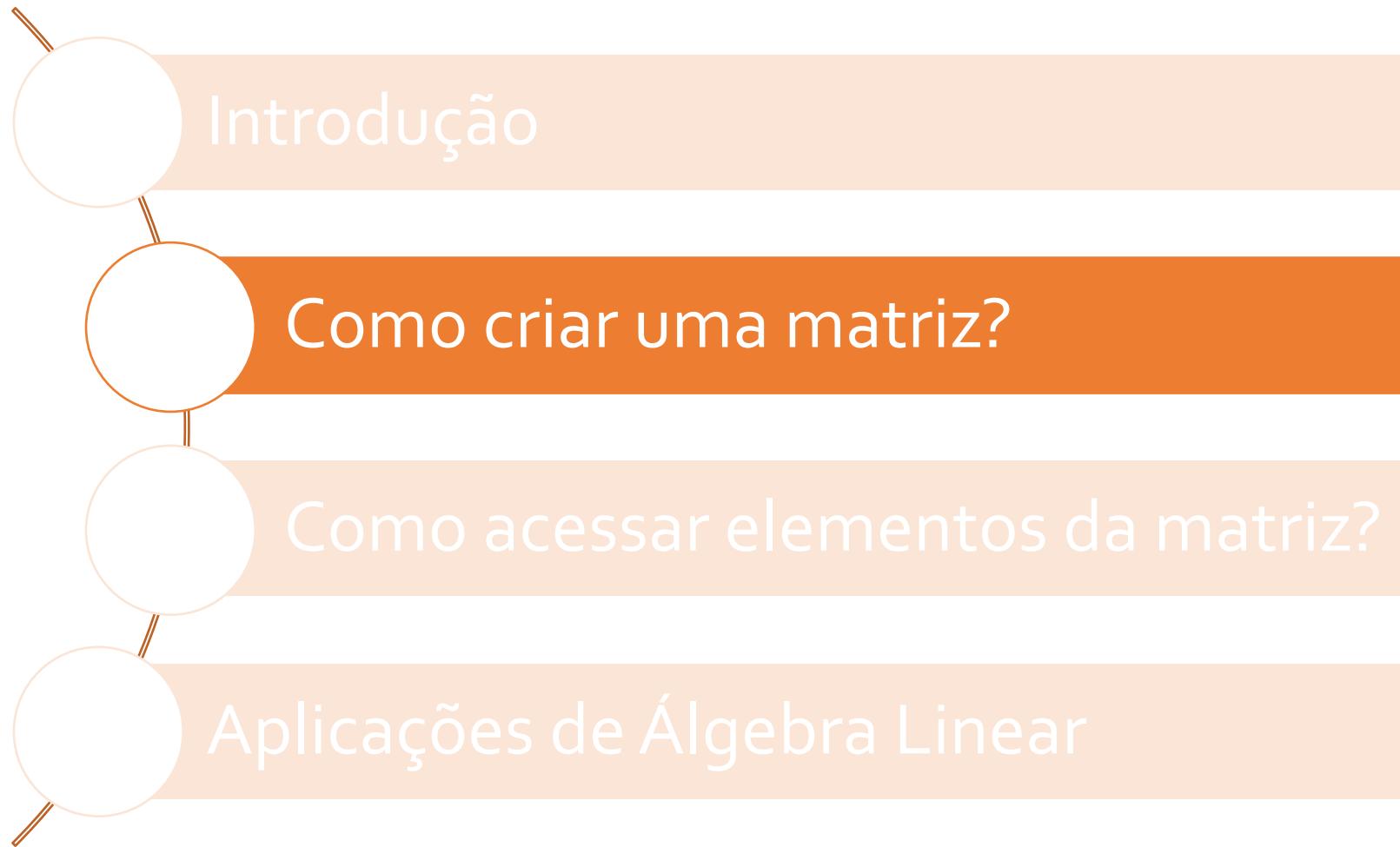
Matrizes em Python

- 🛡️ Para operar com matrizes, utilize os módulos:
 - 🛡️ **numpy** – manipulação de vetores e matrizes,
 - 🛡️ **numpy.linalg** – operações de Álgebra Linear envolvendo matrizes.

```
from numpy import *
from numpy.linalg import *
```

- 🛡️ Mais detalhes:
 - 🛡️ docs.scipy.org/doc/numpy/reference/

Conteúdo



Matrizes em Python

:: Declaração de matrizes

- 🛡 Cada linha da matriz é um **vetor**:
 - 🛡 Os **elementos** de uma mesma linha devem ser **separados por vírgula** e estar **dentro de colchetes**.
- 🛡 A matriz é um **vetor de vetores**:
 - 🛡 Separe cada linha por **vírgula**.

```
A = array([[3, -4, 0], [5, 2, -1]])
```

Lembre-se de
importar o **numpy**

$$A = \begin{bmatrix} 3 & -4 & 0 \\ 5 & 2 & -1 \end{bmatrix}$$

Como criar uma matriz?

:: Direto no script

- 🛡 Quadro de medalhas das Olimpíadas de 2016:

	Ouro	Prata	Bronze
EUA	46	37	38
Grã-Bretanha	27	23	17
China	26	18	26
Rússia	19	18	19
Alemanha	17	10	15
Japão	12	08	21
França	10	18	14

The screenshot shows a Python code editor window with the following code:

```
Arquivo Editar Buscar Executar Ferramentas
Python 3 main.py
from numpy import *
quadro = array([
[46, 37, 38],
[27, 23, 17],
[26, 18, 26],
[19, 18, 19],
[17, 10, 15],
[12, 8, 21],
[10, 18, 14]
])
```

The code imports the numpy module and creates a 2D array named 'quadro' containing the medal counts for seven countries. The array has 7 rows and 3 columns, corresponding to the data in the table above.

Como criar uma matriz? :: Via teclado

- 🛡 Os **elementos** de uma mesma linha devem ser **separados** por **vírgula**.
- 🛡 Cada **linha** da matriz deve estar **entre colchetes**.
- 🛡 As **linhas** devem ser **separadas** por **vírgula**.

```
Arquivo Editar Buscar Executar Ferramentas
Python 3 main.py
from numpy import *

# Leitura da matriz 'quadro'
quadro =
array(eval(input("M: ")))

print(quadro)

Console Shell
M: [[46, 37, 38], [27, 23, 17]]
[[ 46   37   38]
 [ 25   23   17]]
```

Como criar uma matriz? :: Zeros ou uns

A screenshot of a Python code editor window titled "main.py". The code imports numpy and creates a 3x2 matrix of zeros using the zeros() function. A green callout bubble points to the inner parentheses of the zeros() function, containing the text "Use os dois conjuntos de parênteses!". The output console shows the resulting 3x2 matrix of zeros.

```
from numpy import *

# Matriz 3x2 de ZEROS
m0 = zeros( (3,2) )

print(m0)
```

Console output:

```
[[ 0.  0.]
 [ 0.  0.]
 [ 0.  0.]]
```

A screenshot of a Python code editor window titled "main.py". The code imports numpy and creates a 2x5 matrix of ones using the ones() function. The output console shows the resulting 2x5 matrix of ones.

```
from numpy import *

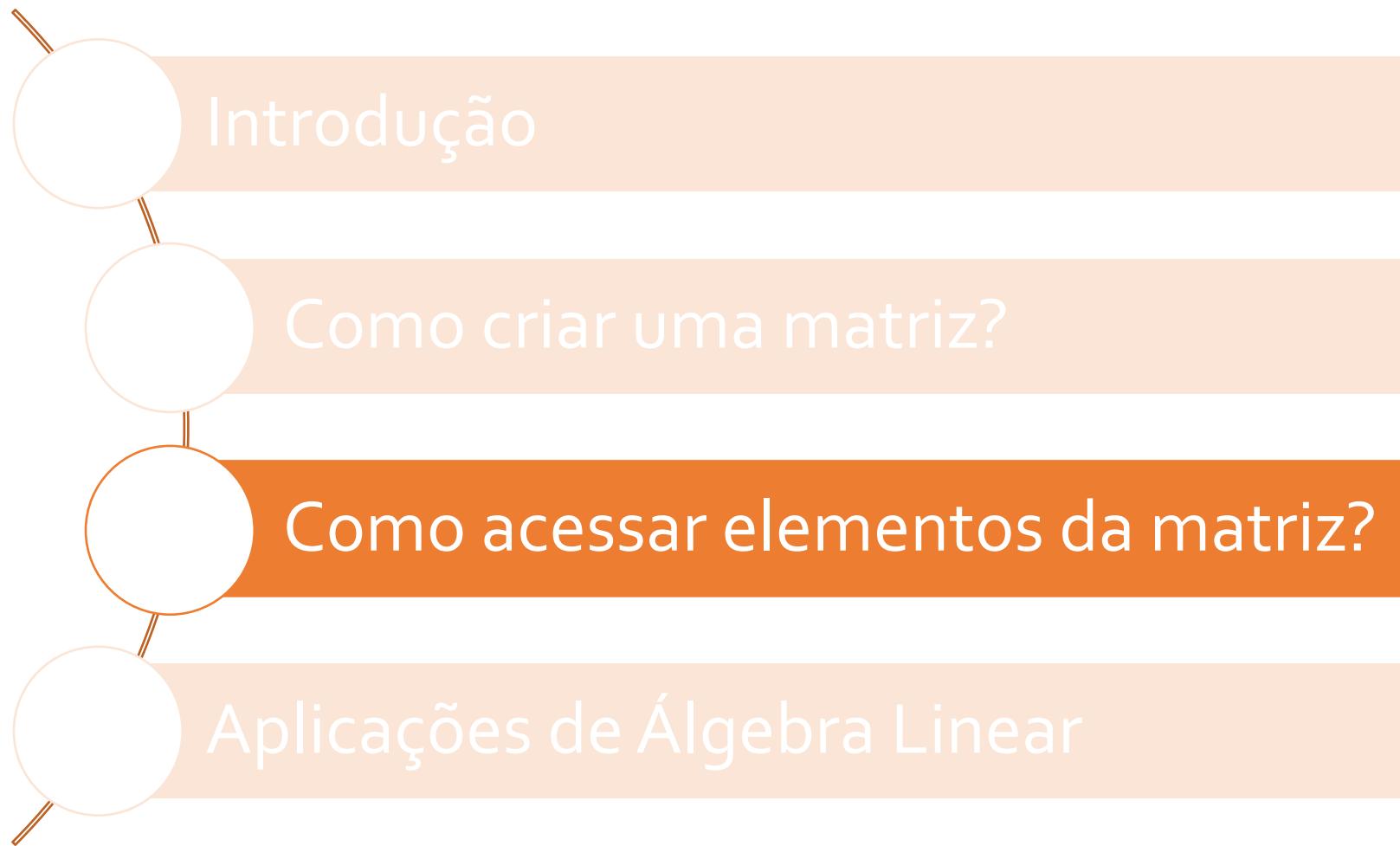
# Matriz 2x5 de UNS
m1 = ones( (2,5) )

print(m1)
```

Console output:

```
[[ 1.  1.  1.  1.  1.]
 [ 1.  1.  1.  1.  1.]]
```

Conteúdo

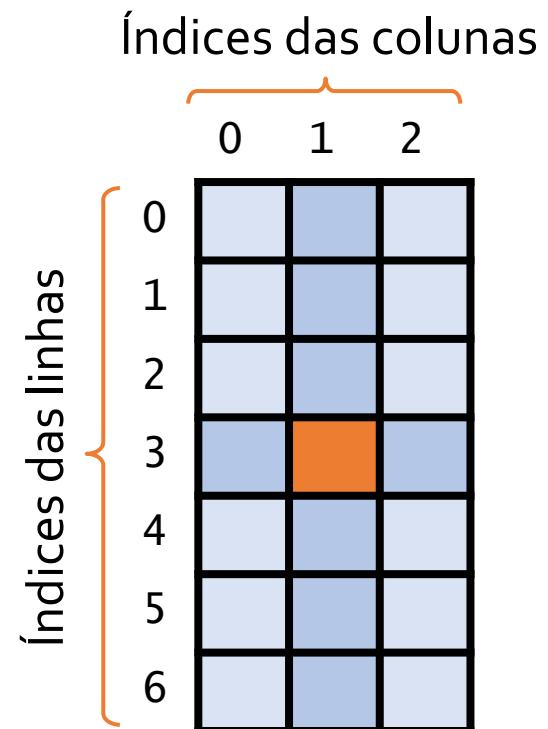


Como acessar elementos da matriz?

`x = mat[3, 1]`

Índice da
linha

Índice da
coluna



Quais as dimensões de uma matriz?

- ◆ Número de **elementos**:

```
TAM = size(quadro)
```

x x x x x x
x x x x x x
x x x x x x
x x x x x x
x x x x x x
x x x x x x
x x x x x x

- ◆ Vetor com número de **linhas** e de **colunas**:

```
FORMA = shape(quadro)
```

x x x x x x
x
x
x
x
x

- ◆ Número de **linhas**:

```
LIN = shape(quadro) [0]
```

x
x
x
x
x
x
x

- ◆ Número de **colunas**:

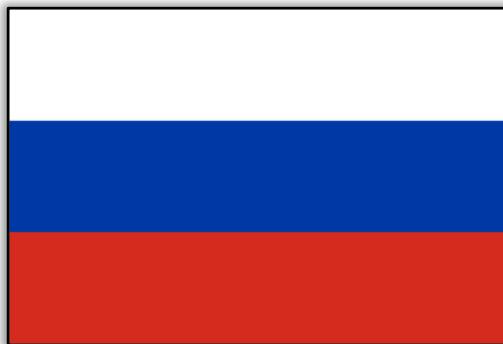
```
COL = shape(quadro) [1]
```

x x x x x x

Problema 01

:: Soma de uma linha

- 🛡 Qual o total de medalhas da **Rússia** nas Olimpíadas de 2016?



	Ouro	Prata	Bronze
EUA	46	37	38
Grã-Bretanha	27	23	17
China	26	18	26
Rússia	19	18	19
Alemanha	17	10	15
Japão	12	08	21
França	10	18	14

Problema 01

2 – Identifique as entradas e saídas

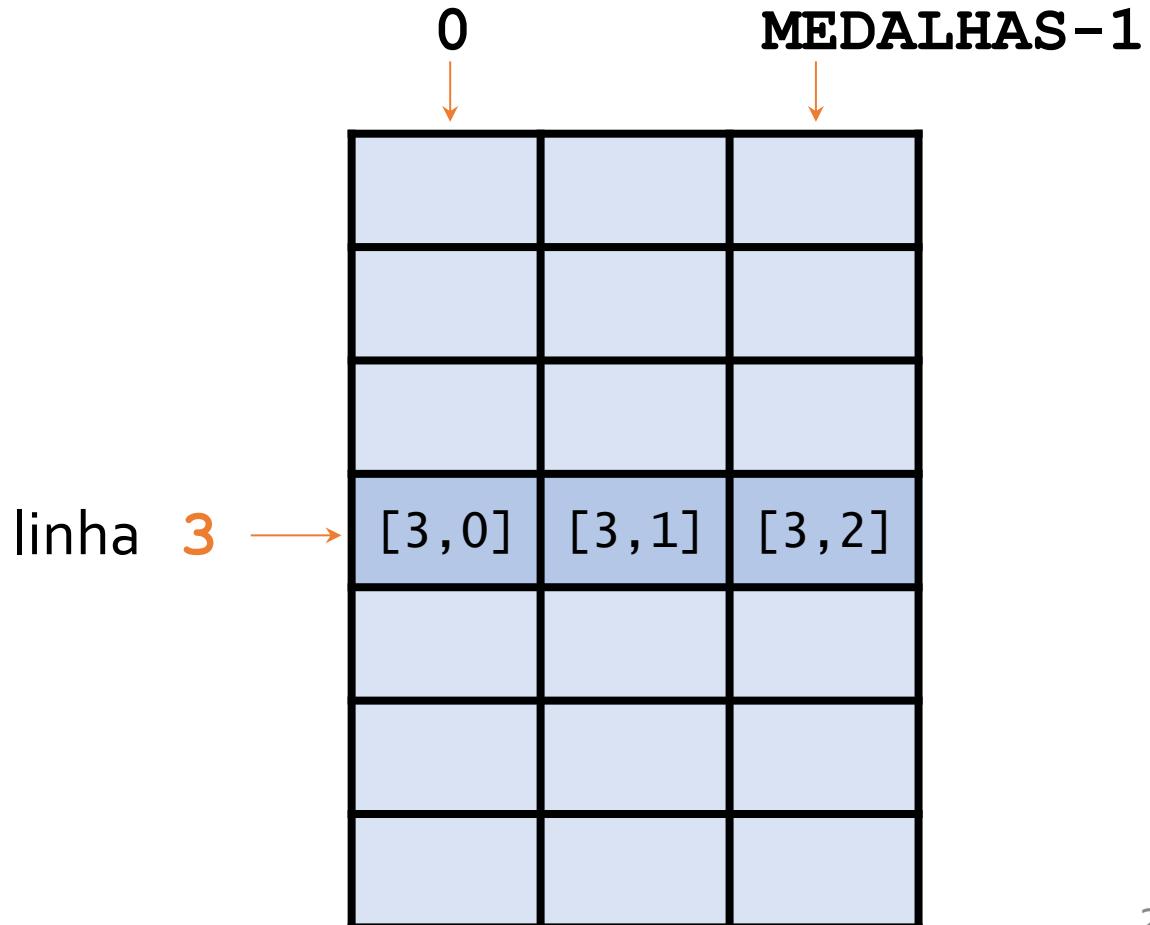
	Grandeza	Unidade de medida	Faixa de valores
Entradas	Quadro de medalhas (matriz)	---	$[\geq 0]$
Saídas	Total de medalhas da Rússia (escalar)	---	≥ 0

Problema 01

3 – Projete o script

- Qual a linha que corresponde às medalhas da Rússia?

Linha 3



Problema 01

4 – Codifique em Python 1/2

```
from numpy import *

# Definicao do quadro de medalhas
quadro = array([
[46, 37, 38],
[27, 23, 17],
[26, 18, 26],
[19, 18, 19],
[17, 10, 15],
[12, 8, 21],
[10, 18, 14]
])
```

Problema 01

4 – Codifique em Python 2/2

```
# Tipos de medalhas (no. de colunas)
MEDALHAS = shape(quadro) [1]

# Variavel acumuladora
total = 0

# Processa a j-esima coluna na linha 4
for j in range(MEDALHAS):
    total = total + quadro[3, j]
print(total)
```

Gera índices de cada
coluna da matriz

Problema 01

5 – Teste o script

🛡️ Teste 1:

- 🛡️ O resultado do script corresponde à soma manual?

Resultado: **56**

Soma manual: $19 + 18 + 19 = \textcolor{blue}{56}$ ✓

🛡️ Teste 2:

- 🛡️ O script funciona para outro país (outra linha)?

Linha: 6 (França)

Resultado: **42**

Soma manual: $10 + 18 + 14 = \textcolor{blue}{42}$ ✓

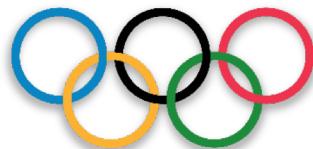
Problema 02

:: Soma de uma coluna

- 💡 Qual o total de medalhas de **ouro** dos sete primeiros colocados nas Olimpíadas de 2016?



Rio2016



	Ouro	Prata	Bronze
EUA	46	37	38
Grã-Bretanha	27	23	17
China	26	18	26
Rússia	19	18	19
Alemanha	17	10	15
Japão	12	08	21
França	10	18	14

Problema 02

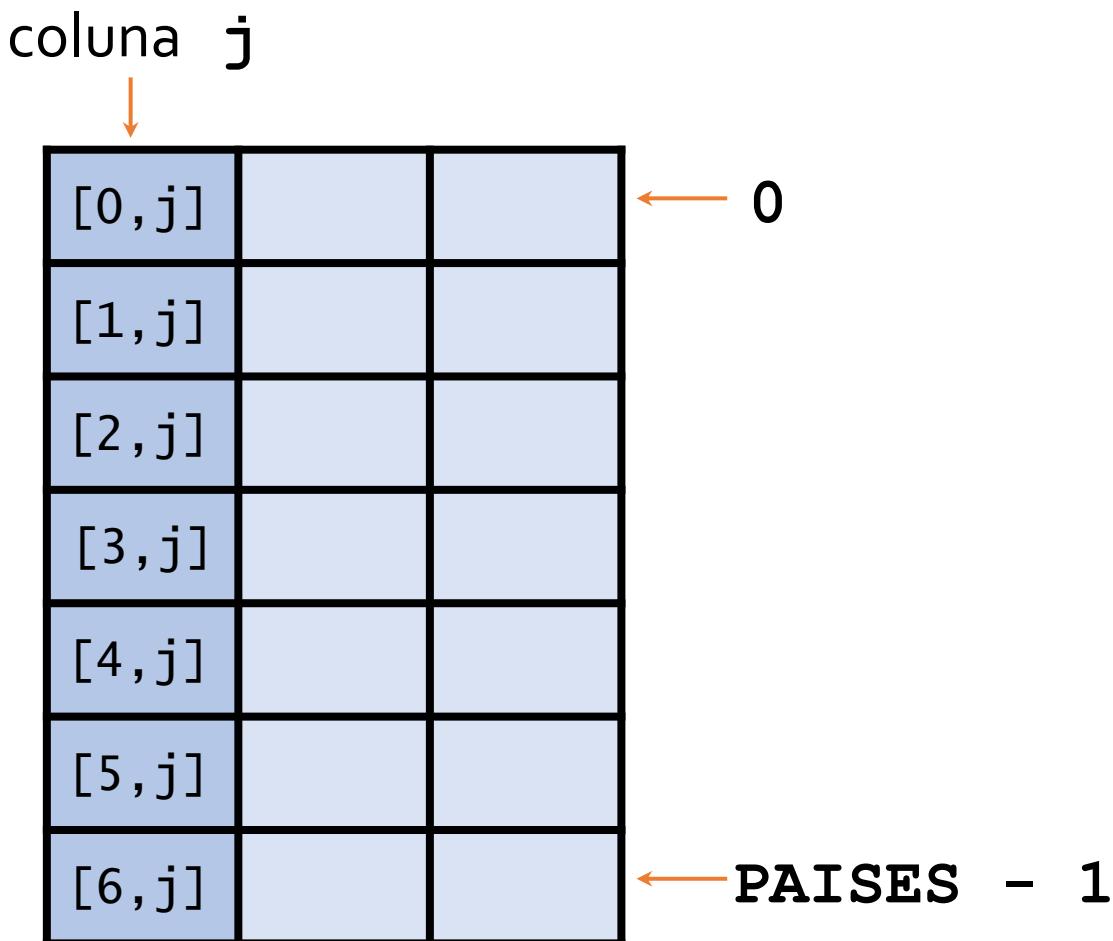
2 – Identifique as entradas e saídas

	Grandeza	Unidade de medida	Faixa de valores
Entradas	Quadro de medalhas (matriz)	---	$[\geq 0]$
Saídas	Total de medalhas da ouro (escalar)	---	≥ 0

Problema 02

3 – Projete o script

- Qual o total de medalhas do tipo j entre os sete países?



Problema 02

4 – Codifique em Python 1/2

```
from numpy import *

# Definicao do quadro de medalhas
quadro = array([
[46, 37, 38],
[27, 23, 17],
[26, 18, 26],
[19, 18, 19],
[17, 10, 15],
[12, 8, 21],
[10, 18, 14]
])
```

Problema 02

4 – Codifique em Python 2/2

```
# No. de países (no. de linhas)
PAISES      = shape(quadro)[0]

# Variável acumuladora
ouro = 0

# Percorre a i-esima linha na coluna 0
for i in range(PAISES) : Gera índices de cada linha da matriz
    ouro = ouro + quadro[i,0]

print(ouro)
```

Problema 02

5 – Teste o script

🛡️ Teste 1:

- 🛡️ O resultado do script corresponde à soma manual?

Resultado: **157**

Soma manual: $46 + 27 + 26 + 19 + 17 + 12 + 10 = \textcolor{blue}{157}$



🛡️ Teste 2:

- 🛡️ O script funciona para outra medalha (outra coluna)?

Coluna: 2 (bronze)

Resultado: **150**

Soma manual: $38 + 17 + 26 + 19 + 15 + 21 + 14 = \textcolor{blue}{150}$



Seleção de elementos

- Seleção de todos os elementos da linha i :

```
x = quadro[i, :]
```

x	x	x	x	x	x	x	x
x	x	x	x	x	x	x	x
x	x	x	x	x	x	x	x
x	x	x	x	x	x	x	x
x	x	x	x	x	x	x	x
x	x	x	x	x	x	x	x
x	x	x	x	x	x	x	x
x	x	x	x	x	x	x	x

- Seleção de todos os elementos da coluna j :

```
x = quadro[:, j]
```

x	x	x	x	x	x	x	x
x	x	x	x	x	x	x	x
x	x	x	x	x	x	x	x
x	x	x	x	x	x	x	x
x	x	x	x	x	x	x	x
x	x	x	x	x	x	x	x
x	x	x	x	x	x	x	x
x	x	x	x	x	x	x	x

Problema 03

:: Soma de cada linha

- 🛡 Qual o total de medalhas de cada país?

	Ouro	Prata	Bronze
EUA	46	37	38
Grã-Bretanha	27	23	17
China	26	18	26
Rússia	19	18	19
Alemanha	17	10	15
Japão	12	08	21
França	10	18	14



Problema 03

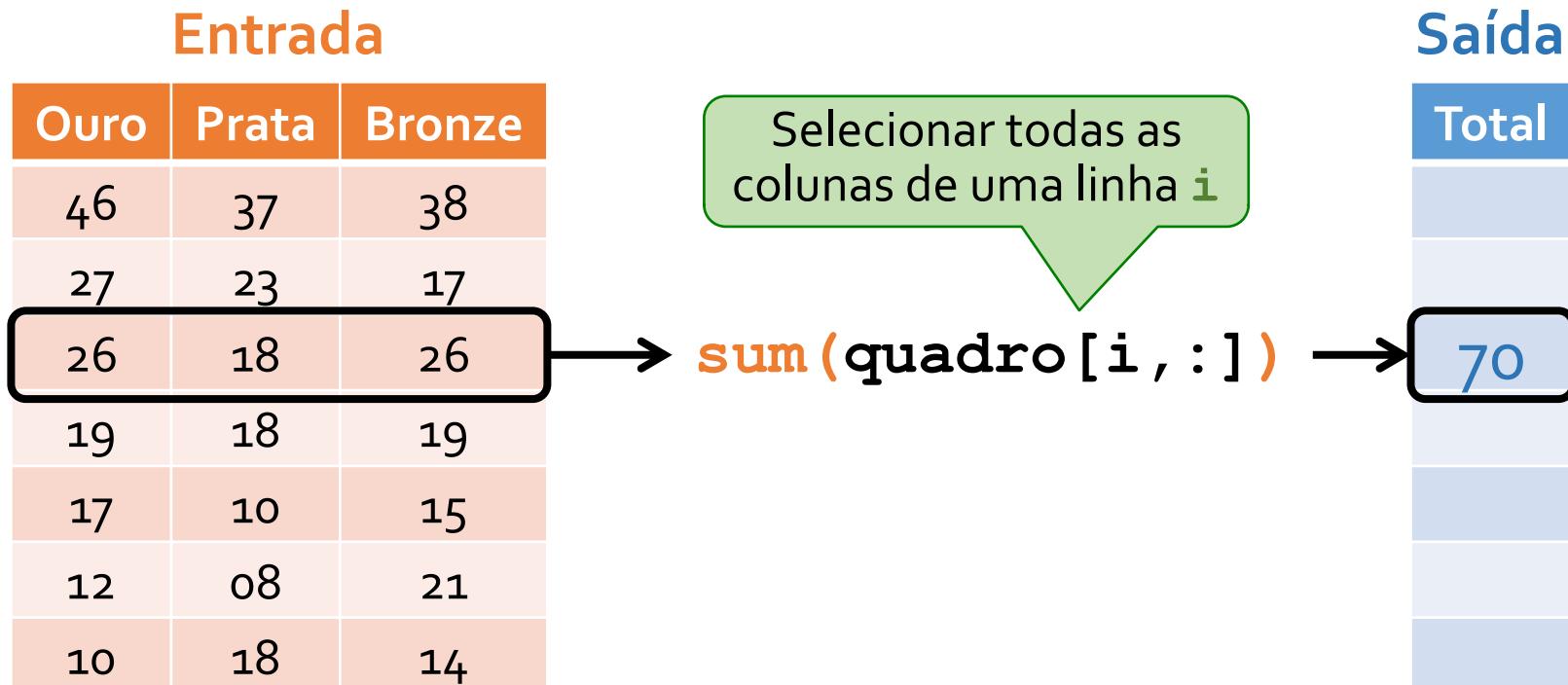
2 – Identifique as entradas e saídas

	Grandeza	Unidade de medida	Faixa de valores
Entradas	Quadro de medalhas (matriz)	---	$[\geq 0]$
Saídas	total de medalhas de cada país (vetor)	---	$[\geq 0]$

Problema 03

3 – Projete o script

- Devemos percorrer cada linha da matriz.
- Para uma linha (=país) selecionada, somamos todas as suas colunas (=medalhas):



Problema 03

4 – Codifique em Python

```
# Incluir código inicial dos outros problemas

# No. de países (no. de linhas)
PAISES = shape(quadro)[0]

# Vetor de contagem de medalhas de cada país
total = zeros(PAISES, dtype=int)

# Soma as medalhas ganhas para cada país
for i in range(PAISES):
    total[i] = sum(quadro[i,:])

print(total)
```

Problema 03

5 – Teste o script

🛡 Faça um checklist:

- 🛡 São impressos 07 valores, uma para cada país (linha)?
- 🛡 Confira manualmente as somas da primeira e da última linha:

121

$$\rightarrow 46 + 37 + 38$$

67

70

56

42

41

42

$$\rightarrow 10 + 18 + 14$$

Localizando elementos adjacentes

- Alguns programas que trabalham com matrizes precisam localizar os elementos adjacentes a outro elemento

$[i, j]$.



Cuidado para não obter índices negativos

$[i-1, j-1]$	$[i-1, j]$	$[i-1, j+1]$
$[i, j-1]$	$\boxed{i, j}$	$[i, j+1]$
$[i+1, j-1]$	$[i+1, j]$	$[i+1, j+1]$

Problema 04

- 🛡️ Escreva um script que imprima uma matriz quadrada de dimensão **N** contendo:
 - 🛡️ **1** nos elementos abaixo da diagonal principal
 - 🛡️ **0** na diagonal principal
 - 🛡️ **-1** nos elementos acima da diagonal principal

	a	b	c	d	e
a	1	0	0	0	0
b	0	1	0	0	0
c	0	0	1	0	0
d	0	0	0	1	0
e	0	0	0	0	1

Problema 04

2 – Identifique as entradas e saídas

	Grandeza	Unidade de medida	Faixa de valores
Entradas	Dimensão N da matriz (escalar)	---	≥ 1
Saídas	Matriz quadrada $N \times N$ (matriz)	---	{-1, 0, +1}

Problema 04

3 – Projete o script

		coluna [j]			
		[0,0]	[0,1]	[0,2]	[0,3]
linha [i]		[1,0]	[1,1]	[1,2]	[1,3]
		[2,0]	[2,1]	[2,2]	[2,3]
		[3,0]	[3,1]	[3,2]	[3,3]

Diagonal principal: $i = j$

Elementos acima: $i < j$

Elementos abaixo: $i > j$

Problema 04

4 – Codifique em Python

```
from numpy import *
N = int(input("Dimensao da matriz: "))
# Cria matriz de zeros, no formato inteiro
mat = zeros((N,N), dtype=int)

# Preenchimento da matriz
for i in range(N):
    for j in range(N):
        # Verifica se termo estah ABAIXO
        if (i > j):
            mat[i,j] = 1
        # Verifica se termo estah ACIMA
        elif (i < j):
            mat[i,j] = -1
        # Elementos da diagonal principal
        else:
            mat[i,j] = 0
print(mat)
```

Cada laço `for` percorre uma das dimensões da matriz

Problema 04

5 – Teste o script

- 🛡 **Teste 1:** $N = 1$ (menor valor possível)

```
[[0]]
```

- 🛡 **Teste 2:** $N = 2$ (segundo menor valor possível)

```
[[ 0 -1]
 [ 1  0]]
```

- 🛡 **Teste 3:** $N = 5$ (outro qualquer)

```
[[ 0 -1 -1 -1 -1]
 [ 1  0 -1 -1 -1]
 [ 1  1  0 -1 -1]
 [ 1  1  1  0 -1]
 [ 1  1  1  1  0]]
```

Laço interno é executado mais “rápido” que o laço externo

```
for i in range(N):  
    for j in range(N):  
        mat[i,j] = 1
```

1	1	1	1	...

Todas as colunas j de uma linha i serão percorridas antes de passar para a linha $i + 1$

```
for j in range(N):  
    for i in range(N):  
        mat[i,j] = 1
```

1			
1			
1			
...			

Todas as linhas i de uma coluna j serão percorridas antes de passar para a coluna $j + 1$

```
for i in range(N):  
    for j in range(N):  
        mat[j,i] = 1
```

1			
1			
1			
...			

Todas as linhas j de uma coluna i serão percorridas antes de passar para a coluna $i + 1$

Problema 05

:: Qual a nota de cada aluno?

- Oito estudantes responderam um teste de dez questões.
- As **respostas** são armazenadas em uma **matriz**. Cada linha registra as respostas de um aluno para as questões.

	0	1	2	3	4	5	6	7	8	9
Student 0	A	B	A	C	C	D	E	E	A	D
Student 1	D	B	A	B	C	A	E	E	A	D
Student 2	E	D	D	A	C	B	E	E	A	D
Student 3	C	B	A	E	D	C	E	E	A	D
Student 4	A	B	D	C	C	D	E	E	A	D
Student 5	B	B	E	C	C	D	E	E	A	D
Student 6	B	B	A	C	C	D	E	E	A	D
Student 7	E	B	E	C	C	D	E	E	A	D

Problema 05 (cont.)

- O **gabarito** é armazenado em um **vetor** de dez elementos.
- Qual a **nota** de cada aluno, considerando que cada questão vale um ponto?

0 1 2 3 4 5 6 7 8 9

D B D C C D A E A D

Problema 05

2 – Identifique as entradas e saídas

	Grandeza	Unidade de medida	Faixa de valores
Entradas	Matriz de respostas da turma (matriz)	---	{A, B, C, D, E}
	Vetor de gabarito (vetor)	---	{A, B, C, D, E}
Saídas	Nota de cada aluno (vetor)	---	[[0, 10]]

Problema 05

3 – Projete o script

1. Definir matriz de respostas (**resp**)
2. Definir vetor de gabarito (**gab**)
3. Criar um vetor de notas (**notas**), com número de elementos igual ao número de linhas da matriz **resp**
4. Para cada aluno (linha) na matriz **resp**:
 - A. Para cada resposta (coluna) na matriz **resp**:
 - i. Comparar resposta do aluno com gabarito
 - ii. Se forem iguais, incrementar posição correspondente no vetor **notas**
 - iii. Caso contrário, não faz nada
5. Imprimir nota do aluno

Problema 05

4 – Codifique em Python – definições

```
from numpy import *

# Respostas dos alunos às questões
resp = array([
    ['A', 'B', 'A', 'C', 'C', 'D', 'E', 'E', 'A', 'D'],
    ['D', 'B', 'A', 'B', 'C', 'A', 'E', 'E', 'A', 'D'],
    ['E', 'D', 'D', 'A', 'C', 'B', 'E', 'E', 'A', 'D'],
    ['C', 'B', 'A', 'E', 'D', 'C', 'E', 'E', 'A', 'D'],
    ['A', 'B', 'D', 'C', 'C', 'D', 'E', 'E', 'A', 'D'],
    ['B', 'B', 'E', 'C', 'C', 'D', 'E', 'E', 'A', 'D'],
    ['B', 'B', 'A', 'C', 'C', 'D', 'E', 'E', 'A', 'D'],
    ['E', 'B', 'E', 'C', 'C', 'D', 'E', 'E', 'A', 'D'])

# Gabarito das questões
gab = array([
    ['D', 'B', 'D', 'C', 'C', 'D', 'A', 'E', 'A', 'D'])
```

Problema 05

4 – Codifique em Python – nota

```
# Constantes
NALUNOS = shape(resp)[0]           # No. de alunos
NQUEST = shape(resp)[1]             # No. de questoes

# Vetor das notas de cada aluno
notas = zeros((NALUNOS), dtype=int)

# Percorre cada linha (aluno)
for i in range(NALUNOS):
    # Pontua um aluno i, verificando cada questao
    for j in range(NQUEST):
        if (resp[i,j] == gab[j]):
            notas[i] = notas[i] + 1

print(notas)
```

Contador *i* controla as linhas e
contador *j* controla as colunas

Há um gabarito para
cada questão (coluna
da matriz)

Há uma nota para cada
aluno (linha da matriz)

Problema 05

5 – Teste o script

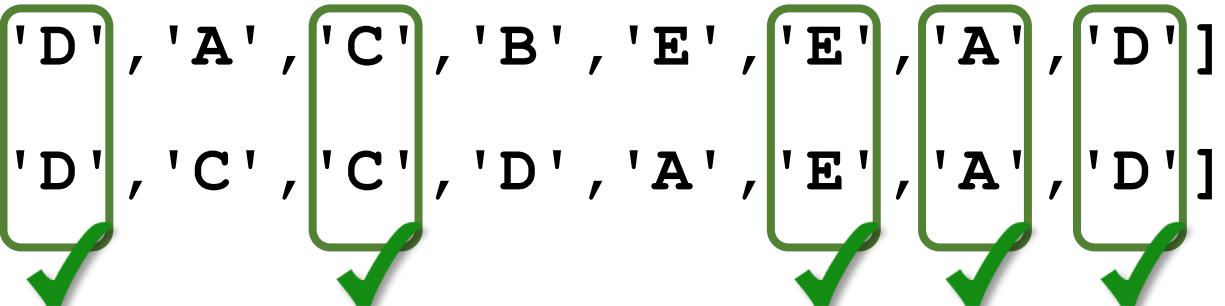
- Escolha algumas linhas da matriz de resposta, compare com o vetor de gabarito e confira o resultado com o vetor de saída do script:

Saída: [7 6 5 4 8 7 7 7]

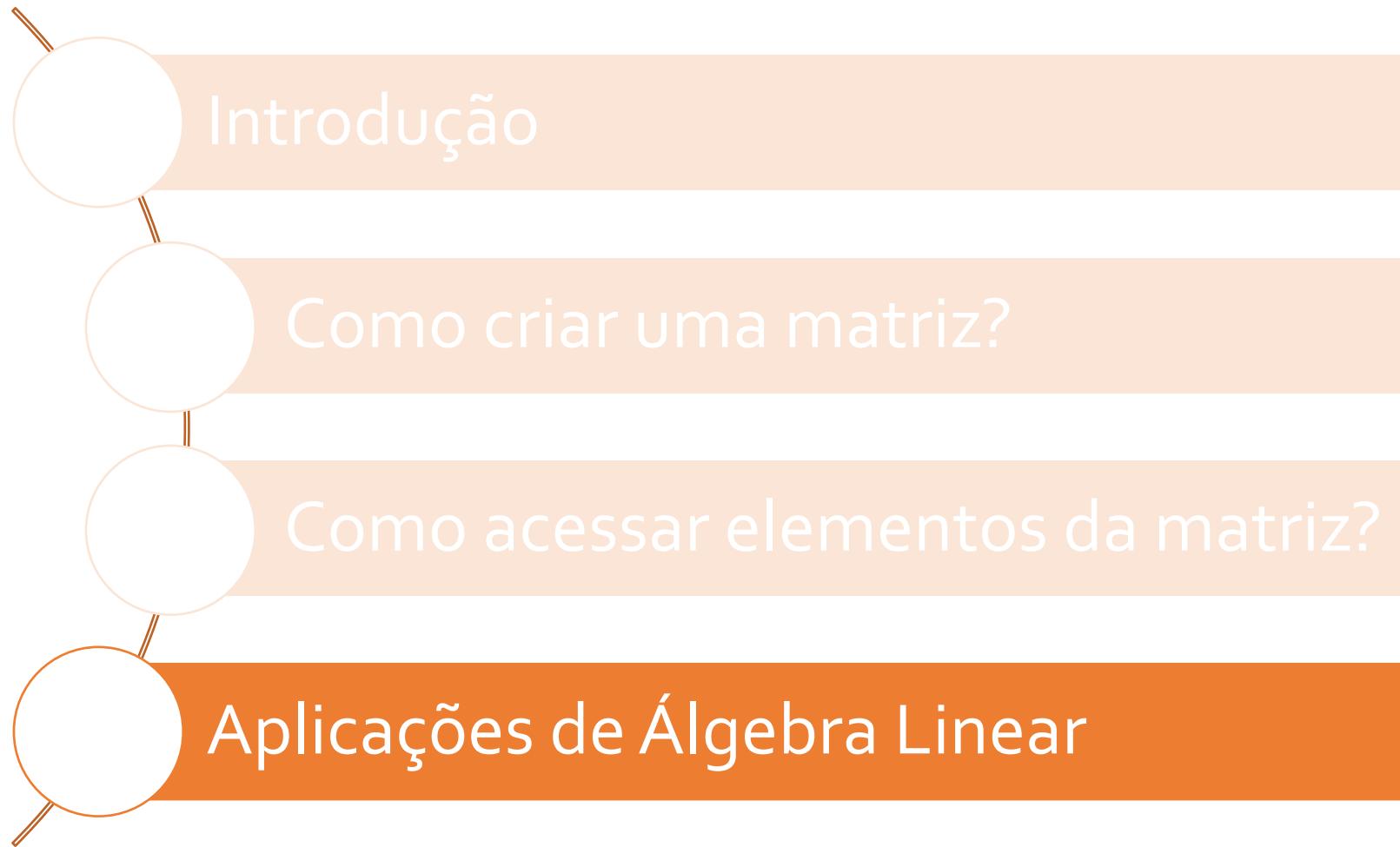
- Não adianta escolher as linhas cujos resultados são 7, pois será difícil distinguir se houve coincidência ou não.

Linha 2:

```
[ 'E' , 'D' , 'D' , 'A' , 'C' , 'B' , 'E' , 'E' , 'A' , 'D' ]  
[ 'D' , 'B' , 'D' , 'C' , 'C' , 'D' , 'A' , 'E' , 'A' , 'D' ]
```



Conteúdo



Matrizes em Python

:: Operações

Estruturais

- Operações entre matrizes executadas elemento a elemento de mesma posição

Matriciais

- Seguem as regras da álgebra linear

Adição e Subtração

- 🛡 Soma/subtração estrutural é **idêntica** à soma/subtração matricial.
- 🛡 Matrizes devem ter **mesmas dimensões**.
- 🛡 Exemplo:

$$\begin{bmatrix} 3 & -4 & 0 \\ 5 & 2 & -1 \end{bmatrix} + \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} = \begin{bmatrix} 4 & -2 & 3 \\ 9 & 7 & 5 \end{bmatrix}$$

- 🛡 Em Python:

```
total = a + b
```

Multiplicação

Multiplicação estrutural

- Multiplica elementos de mesma posição em **a** e **b**.
- As duas matrizes precisam ter as mesmas dimensões.

```
prod = a * b
```

Multiplicação matricial

- Multiplica as matrizes **a** e **b**.
- O número de colunas em **a** precisa ser igual ao de linhas em **b**.

```
prod = dot(a, b)
```

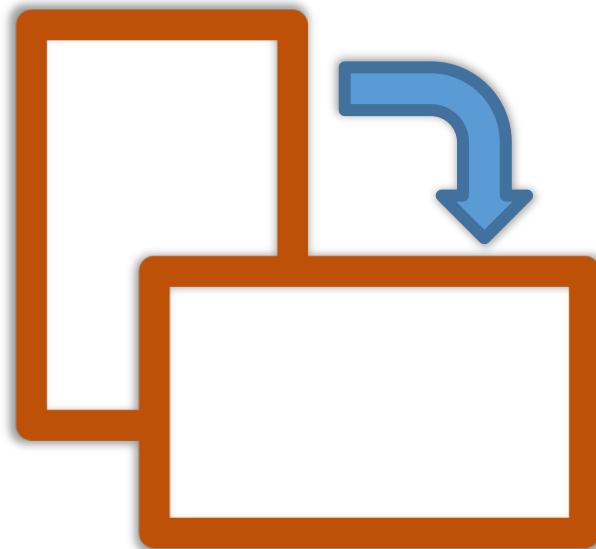
Outras operações com matrizes

Operação	Em Python	Módulo
Transposta	<code>m.T</code>	<code>numpy</code>
Determinante	<code>det(m)</code>	<code>numpy.linalg</code>
Inversa	<code>inv(m)</code>	<code>numpy.linalg</code>

Problema 06

:: Matriz transposta

- 🛡 Leia uma matriz $M \times N$ a partir do teclado.
- 🛡 Imprima a transposta da matriz fornecida.



Problema 06

2 – Identifique as entradas e saídas

	Grandeza	Unidade de medida	Faixa de valores
Entradas	Matriz $M \times N$ (matriz)	---	[]
Saídas	Matriz $N \times M$ transposta (matriz)	---	[]

Problema 06

3 – Projete o script

Matriz **mat**

[0,0]	[0,1]	[0,2]
[1,0]	[1,1]	[1,2]
[2,0]	[2,1]	[2,2]
[3,0]	[3,1]	[3,2]

Matriz **tp**



tp[i, j] = mat[j, i]

mat[0,0]	mat[1,0]	mat[2,0]	mat[3,0]
mat[0,1]	mat[1,1]	mat[2,1]	mat[3,1]
mat[0,2]	mat[1,2]	mat[2,2]	mat[3,2]

tp = mat.T

Problema 06

4 – Codifique em Python

```
from numpy import *
# Leitura da matriz
mat = array(eval(input("Matriz: ")))
# Imprime matriz transposta
print(mat.T)
```

Problema 06

5 – Teste o script

🛡️ Teste 1:

- 🛡️ A entrada é uma matriz **quadrada**

$$[[1, 2], [3, 4]] \rightarrow [[1 \ 3] \\ [2 \ 4]]$$

🛡️ Teste 2:

- 🛡️ A entrada é uma matriz **retangular**

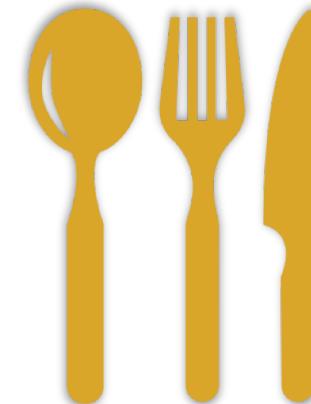
$$[[1, 2, 3], [4, 5, 6]] \rightarrow [[1 \ 4] \\ [2 \ 5] \\ [3 \ 6]]$$

Problema 07

:: Sistemas de Equações Lineares

- Uma fábrica produz três talheres: faca, colher e garfo, a partir de três tipos de materiais: metal, plástico e borracha.
- As quantidades (em grama) de material exigido para produzir cada componente são:

	Faca (x)	Colher (y)	Garfo (z)
Metal	30	20	10
Plástico	60	50	20
Borracha	40	30	20



- Se houver disponíveis 2350g, 5000g e 3600g de metal, plástico e borracha, respectivamente, quantos talheres podem ser produzidos de cada tipo?

Problema 07

2 – Identifique as entradas e saídas

	Grandeza	Unidade de medida	Faixa de valores
Entradas	Matriz de coeficientes (matriz)	---	[]
Saídas	Quantidade de cada talher (vetor)	---	[]

Embora a quantidade de itens seja um número inteiro positivo, a resposta de um sistema de equações lineares é um valor real.

Problema 07

3 – Projete o script 1/2

Cada linha do sistema é a soma do gasto de um **mesmo material** em cada talher

$$\begin{aligned}30 \cdot x + 20 \cdot y + 10 \cdot z &= 2350 \\60 \cdot x + 50 \cdot y + 20 \cdot z &= 5000 \\40 \cdot x + 30 \cdot y + 20 \cdot z &= 3600\end{aligned}$$

$$\begin{bmatrix} 30 & 20 & 10 \\ 60 & 50 & 20 \\ 40 & 30 & 20 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 2350 \\ 5000 \\ 3600 \end{bmatrix}$$

$$A \cdot r = B$$

$$\downarrow$$

$$r = A^{-1}B$$

Problema 07

3 – Projete o script 2/2

- Use o módulo `numpy.linalg` para resolver a equação:

$$r = A^{-1}B$$

```
r = dot(inv(A), B)
```

Problema 07

4 – Codifique em Python

```
from numpy import *
from numpy.linalg import *

# Insercao dos valores do enunciado
mat = array([[30,20,10],
             [60,50,20],
             [40,30,20]])
qtde = array([2350, 5000, 3600])  
Matriz linha

# Resolucao do sistema de equacoes lineares
r = dot(inv(mat), qtde.T)
Transpor para virar matriz coluna

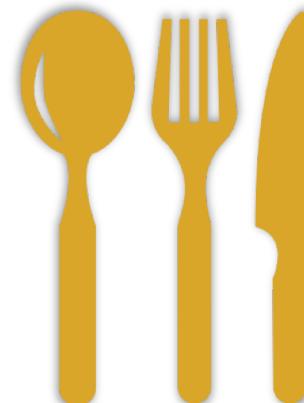
# Impressao do resultado
print(r)
```

Problema 07

5 – Teste o script

[40 . 30 . 55 .]

- 🛡 Com 2350g de metal, 5000g de plástico e 3600g de borracha disponíveis, é possível produzir:
 - 🛡 40 facas
 - 🛡 30 colheres
 - 🛡 55 garfos



Referências bibliográficas

-  Menezes, Nilo Ney Coutinho (2014). *Introdução à Programação com Python*, 2 ed. Editora Novatec.
-  HETLAND, Magnus Lie (2008). *Beginning Python: From Novice to Professional*. Springer eBooks, 2^a edição.
Disponível em: <http://dx.doi.org/10.1007/978-1-4302-0634-7>.
-  Horstmann, Cay & Necaise, Rance D. (2013). *Python for Everyone*. John Wiley & Sons.
-  Liang, Y. D. (2013). *Introduction to Programming Using Python*. Pearson

Dúvidas?

