



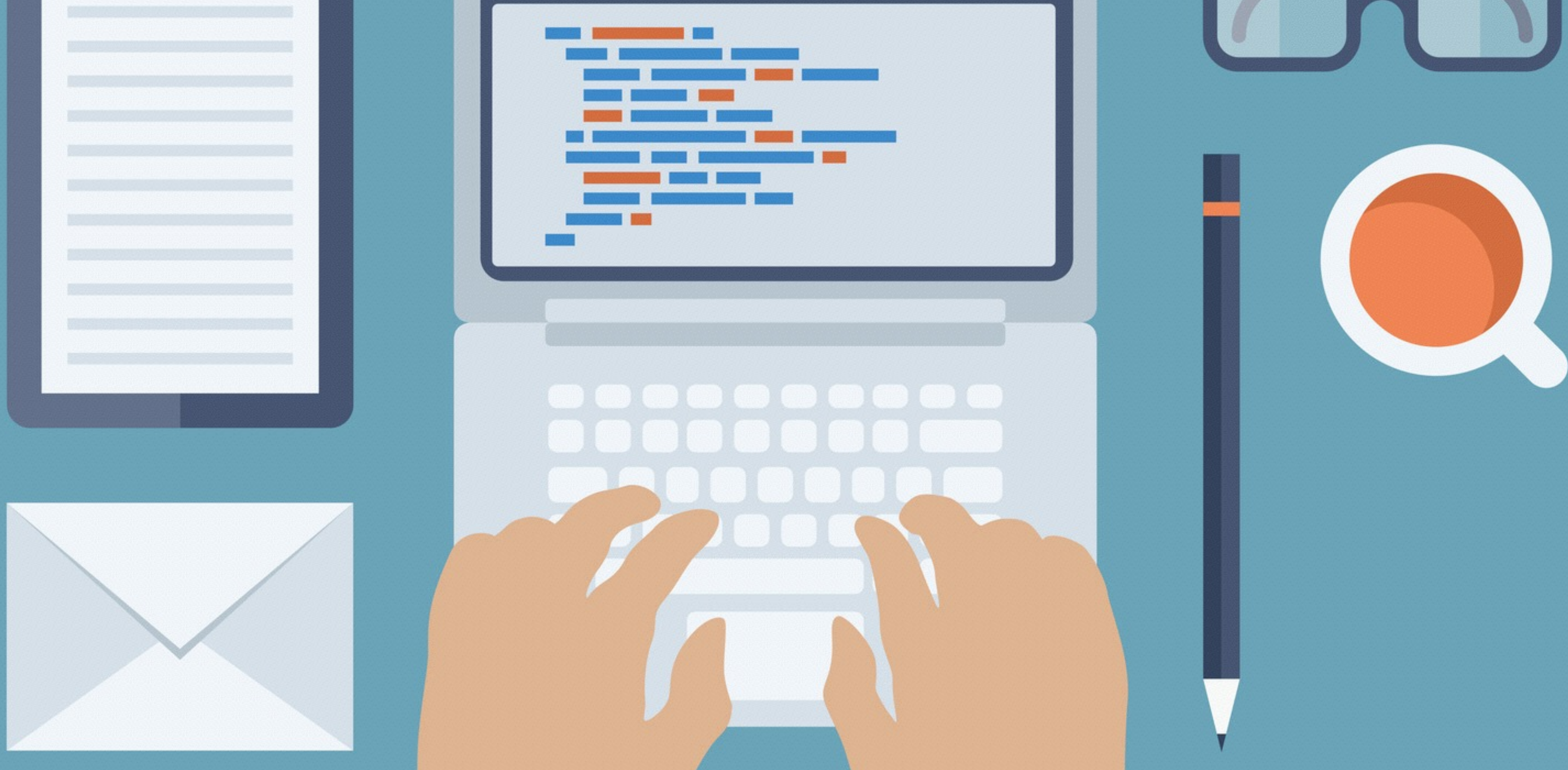
# CK0211 - Fundamentos de Programação: Matrizes

Emanuele Santos

Bibliografia: Ascencio, Cap. 7

# Objetivos

- Aprender a usar matrizes em algoritmos e em Python



# MATRIZ EM ALGORITMOS

# Definição de Matriz

- Variável **composta homogênea multidimensional**
- Conjunto de variáveis de um **mesmo tipo**
- O conjunto possui um **único identificador**
- As variáveis que compõem o conjunto são alocadas sequencialmente na memória
- O que distingue as variáveis são **índices** que referenciam sua posição/localização na estrutura (conjunto)
- Utiliza-se um **índice** para cada uma de suas **dimensões**

# Declaração de matriz

**DECLARE** nome [dimensão1, dimensão2, ..., dimensãoN] **TIPO**

- onde:
  - nome é o nome da variável do tipo matriz;
  - dimensão1 é a quantidade de elementos da 1ª dimensão (muitas vezes, chamada linha);
  - dimensão2 é a quantidade de elementos da 2ª dimensão (muitas vezes, chamada coluna);
  - dimensãoN é a quantidade de elementos da enésima dimensão;
  - TIPO é o tipo básico de cada elemento da matriz

# Exemplos de matriz

**DECLARE  $m[3, 5]$  NUMÉRICO**

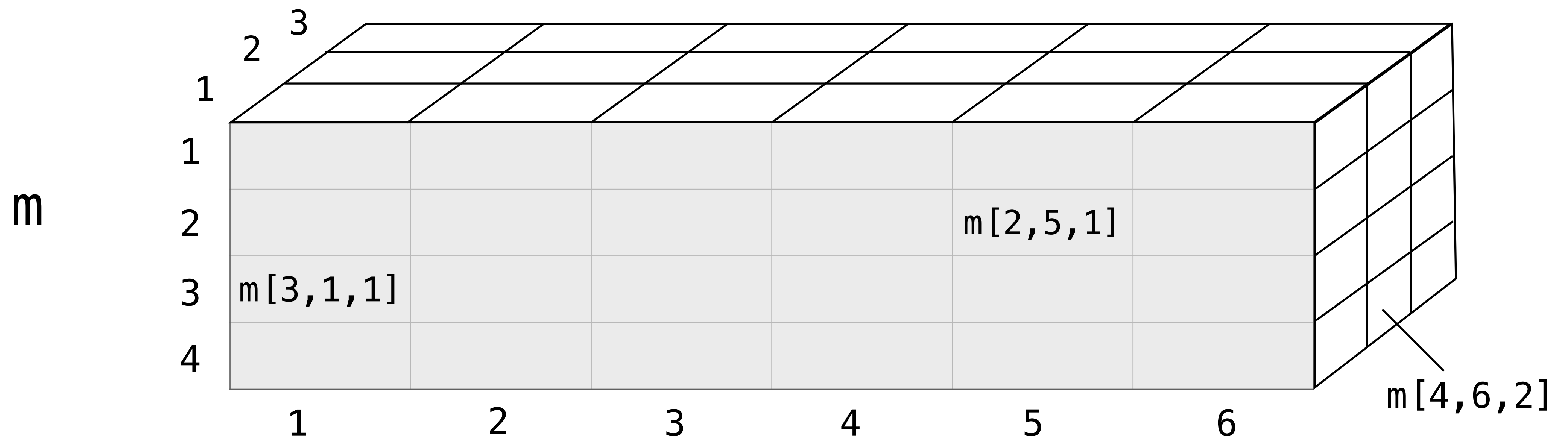
- $m$  é uma matriz bidimensional, onde o tamanho da 1ª dimensão (linha) é 3 e o tamanho da 2ª dimensão (coluna) é 5

		1	2	3	4	5
m	1	$m[1, 1]$				
	2				$m[2, 4]$	
	3	$m[3, 1]$				$m[3, 5]$

# Exemplos de matriz

**DECLARE  $m[4, 6, 3]$  NUMÉRICO**

- $m$  é uma matriz tridimensional, onde o tamanho da 1ª dimensão (linha) é 4, o tamanho da 2ª dimensão (coluna) é 6 e o tamanho da 3ª dimensão (profundidade) é 3



# Atribuindo valores a uma matriz: Exemplo 1

- Cada elemento de uma matriz pode armazenar um valor
- A atribuição deve conter o índice da posição desejada em cada dimensão
- mat possui 2 dimensões: 5 linhas e 4 colunas
- mat poderá armazenar 20 elementos numéricos

```
DECLARE mat [5, 4] NUMÉRICO
mat [2, 4] ← 45
mat [3, 1] ← 8
mat [1, 3] ← 10
```

	1	2	10	
	2			45
mat	8			
	4			
	5			
	1	2	3	4

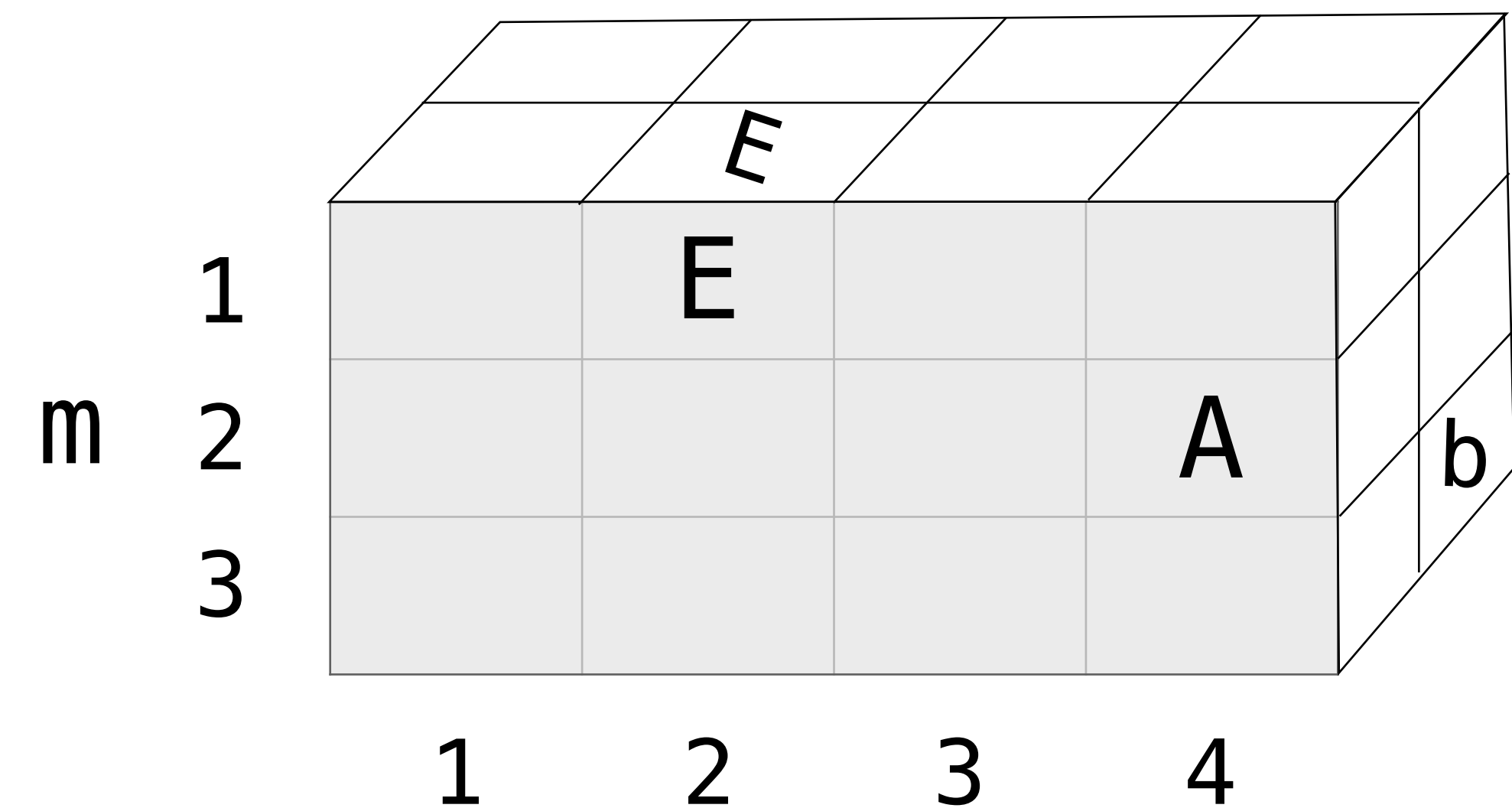


# Atribuindo valores a uma matriz: Exemplo 2

```

DECLARE m[3,4,2] LITERAL
mat[2,4,1] ← "A"
mat[1,2,1] ← "E"
mat[3,4,2] ← "b"
    
```

- m possui 3 dimensões: 3 linhas, 4 colunas e profundidade 2
- m pode armazenar 24 elementos literais



# Preenchendo uma matriz

- Para preencher uma matriz precisamos identificar todas as suas posições
- Para isso, precisaremos de um índice para cada dimensão da matriz

mat

1					
2					
3					
	1	2	3	4	5

# Preenchendo uma matriz: Exemplo 1

```
DECLARE mat[3,5] NUMÉRICO
PARA i ← 1 ATÉ 3 FAÇA
INÍCIO
  PARA j ← 1 ATÉ 5 FAÇA
  INÍCIO
    ESCREVA “Digite o número da linha”, i, “e coluna ”, j
    LEIA mat[i,j]
  FIM
FIM
FIM
```

# Preenchendo uma matriz: Exemplo 2

```
DECLARE mat[4,3,2] NUMÉRICO
PARA i ← 1 ATÉ 4 FAÇA
INÍCIO
  PARA j ← 1 ATÉ 3 FAÇA
  INÍCIO
    PARA k ← 1 ATÉ 2 FAÇA
    INÍCIO
      ESCREVA "Digite o número da linha", i, "e coluna ", j, " e
profundidade ", k
      LEIA mat[i,j,k]
    FIM
  FIM
FIM
FIM
FIM
```

# Mostrando os elementos de uma matriz

- Para mostrar os elementos de uma matriz também precisamos identificar todas as suas posições
- Vamos usar novamente um índice para cada dimensão da matriz

# Mostrando os elementos de uma matriz: Exemplo 1

```
PARA i ← 1 ATÉ 3 FAÇA
INÍCIO
  PARA j ← 1 ATÉ 5 FAÇA
  INÍCIO
    ESCREVA mat[i,j]
  FIM
FIM
FIM
```

# Percorrendo uma matriz: Forma 1

- Usamos as estruturas de repetição para controlar a forma de percorrer a matriz
- Nos exemplos anteriores e no exemplo a seguir, queremos percorrer a matriz `mat` de modo a percorrer uma linha inteira antes de ir para a linha de baixo.

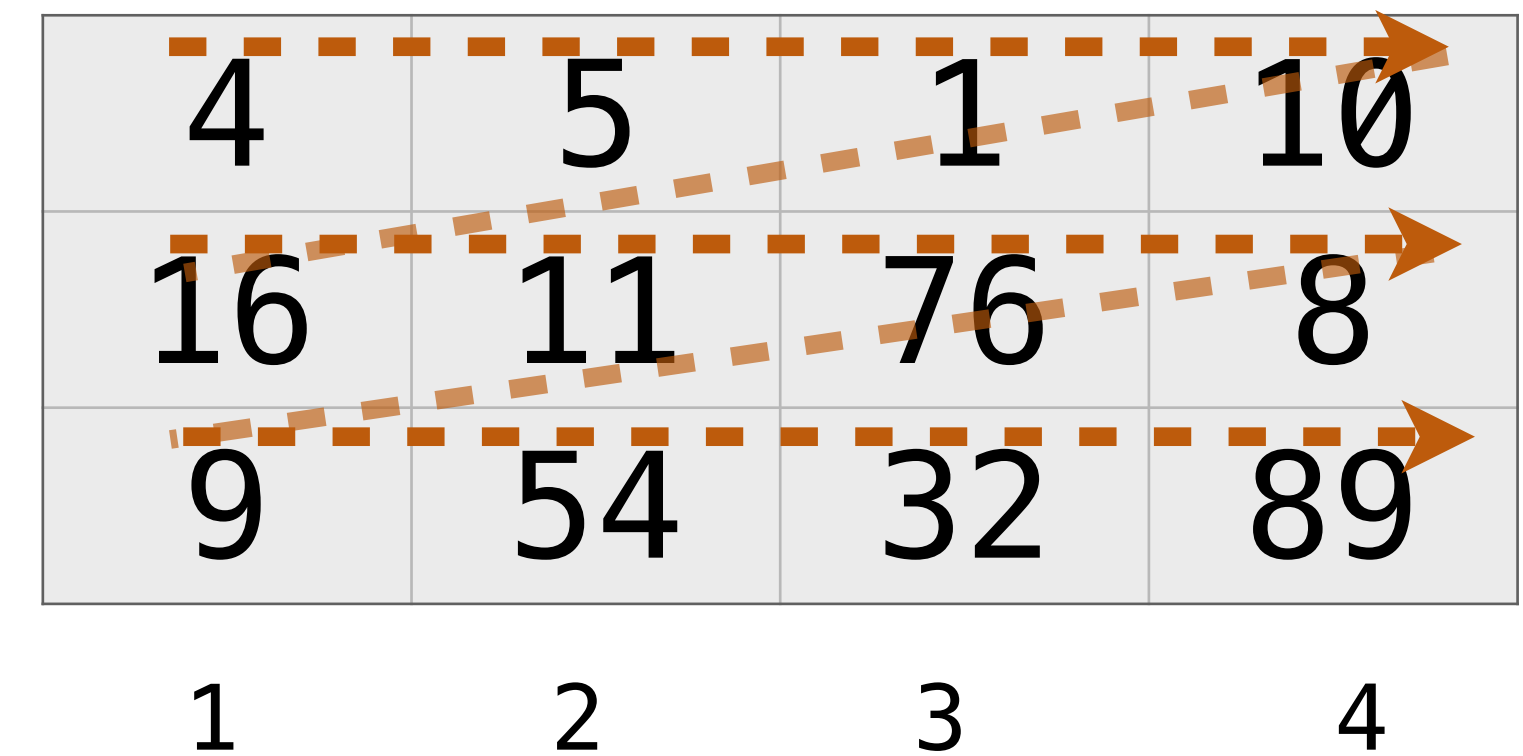
1	4	5	1	10
2	16	11	76	8
3	9	54	32	89
	1	2	3	4

mat

# Preenchendo uma matriz: Forma 1

mat

1	4	5	1	10
2	16	11	76	8
3	9	54	32	89
	1	2	3	4



```

PARA i ← 1 ATÉ 3 FAÇA
INÍCIO
  ESCREVA “Elementos da linha ”, i
  PARA j ← 1 ATÉ 4 FAÇA
  INÍCIO
    ESCREVA mat[i,j]
  FIM
FIM
FIM

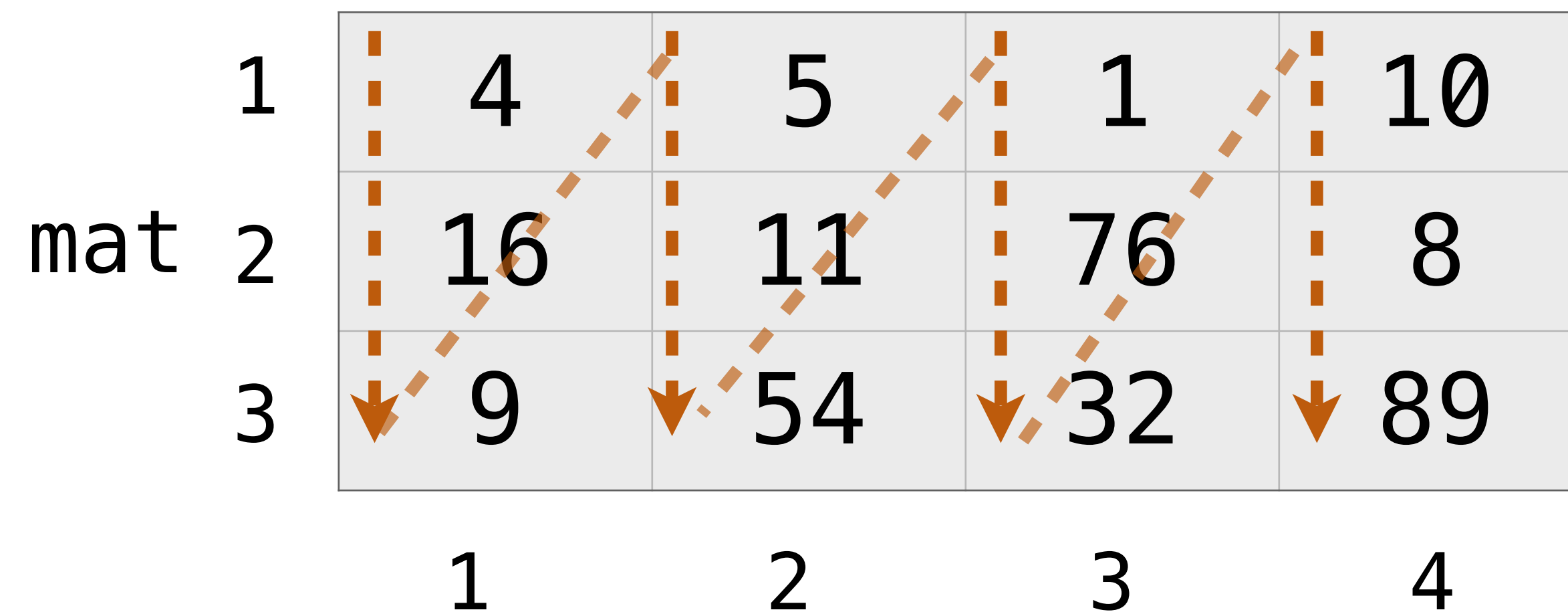
```



# Percorrendo uma matriz: Forma 2

- Nos exemplos anteriores e no exemplo a seguir, queremos percorrer a matriz `mat` de modo a percorrer uma coluna inteira antes de ir para a próxima coluna.

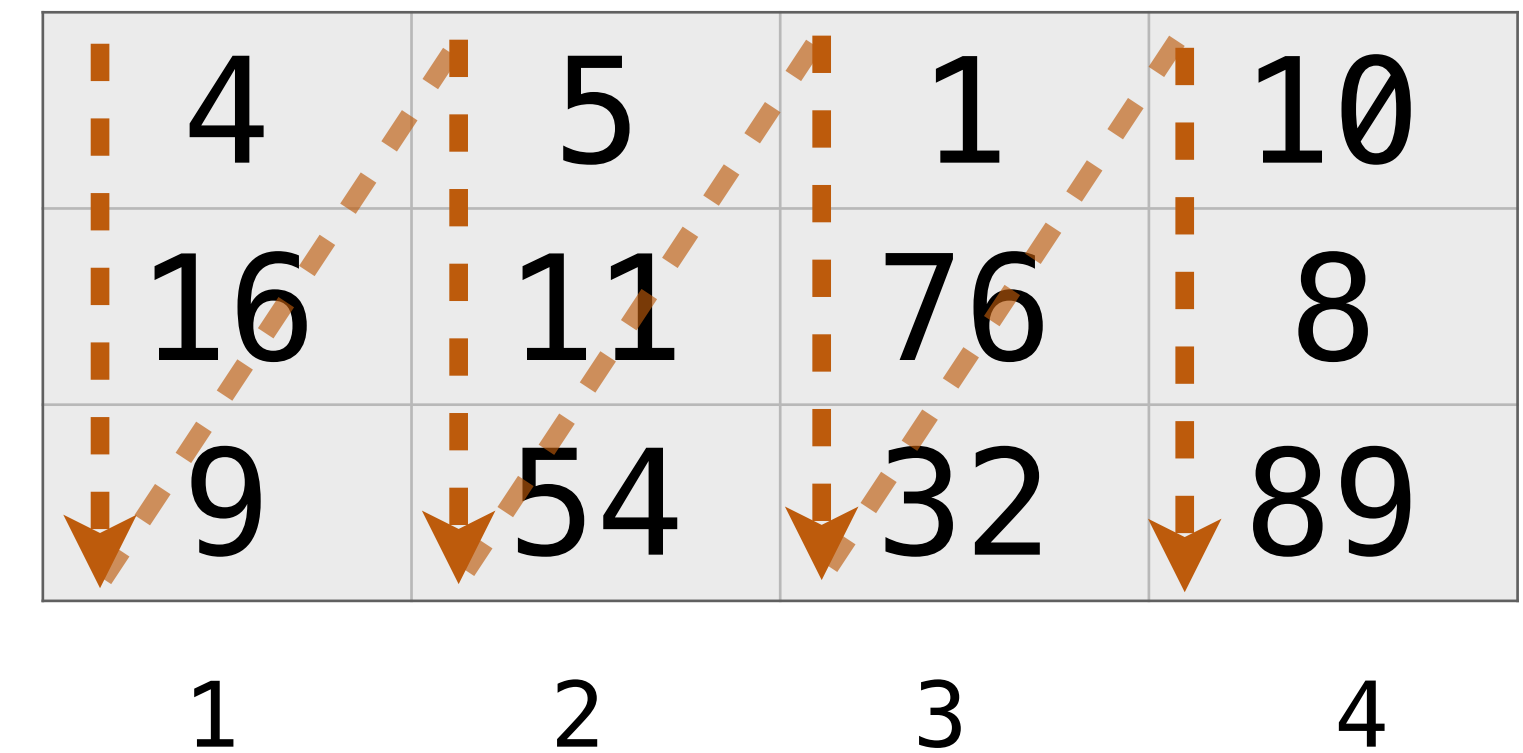
	1	4	5	1	10
mat	2	16	11	76	8
	3	9	54	32	89
		1	2	3	4



# Preenchendo uma matriz: Forma 2

mat

1	4	5	1	10
2	16	11	76	8
3	9	54	32	89
	1	2	3	4



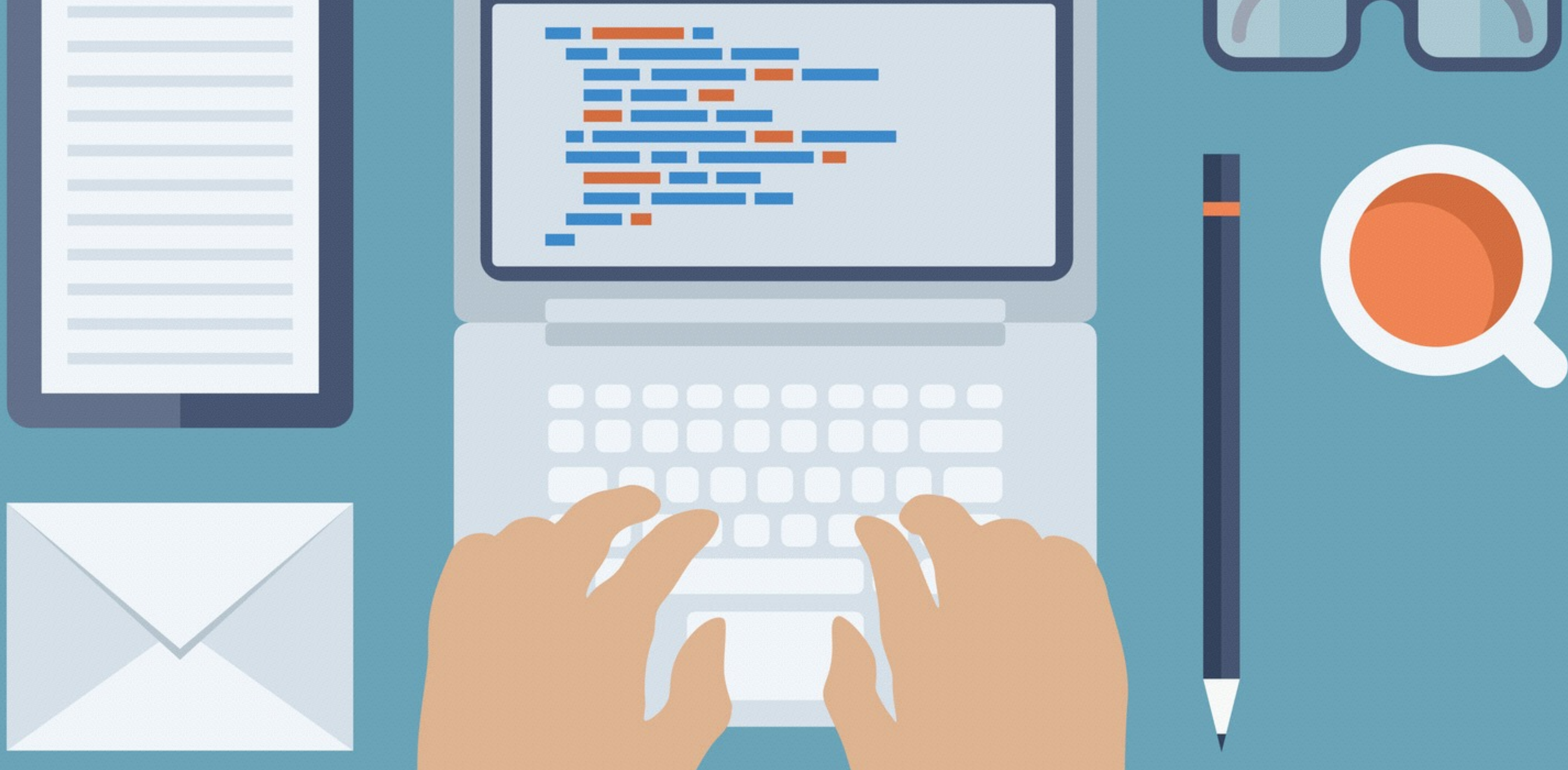
```

PARA j ← 1 ATÉ 4 FAÇA
INÍCIO
  ESCREVA “Elementos da coluna ”, j
  PARA i ← 1 ATÉ 3 FAÇA
  INÍCIO
    ESCREVA mat[i,j]
  FIM
FIM

```

# Preenchendo uma matriz: Observações

- Para evitar confusão, é melhor sempre usar a mesma variável para percorrer uma dada dimensão, por exemplo:
  - $i$  para linhas
  - $j$  para colunas
  - $k$  para profundidade, etc
- O que vai mudar a forma de percorrer a matriz é apenas a ordem na estrutura de repetição: a dimensão a ser percorrida primeiro aparece primeiro, mas o elemento será sempre acessado por  $m[i, j, k]$



# MATRIZES EM PYTHON

# Implementação de matrizes

- Assim como em vetores, a maneira da qual uma matriz é implementada em um programa varia de linguagem para linguagem
- Veremos a seguir o caso específico para a linguagem Python. Para saber a implementação em outras linguagens de programação, veja as seções 7.2, 7.3 e 7.4 do Ascencio

# Matrizes em Python

- Em Python, matrizes são implementadas como listas de listas
- Os índices das dimensões de uma matriz iniciam em zero

# Inicializando matrizes

```
# criando uma matriz de 3 linhas e 4 colunas
mat = [[0, 0, 0, 0],
        [0, 0, 0, 0],
        [0, 0, 0, 0]]
# que é equivalente a:
mat = [[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]
# e que também é equivalente a:
mat = []
for i in range(3):
    linha = [0]*4
    mat.append(linha)
```

# Inicializando matrizes

```
# criando uma matriz mat de 4 linhas, 3 colunas e profundidade 3
mat = [[0, 0, 0], [0, 0, 0], [0, 0, 0]],
      [[0, 0, 0], [0, 0, 0], [0, 0, 0]],
      [[0, 0, 0], [0, 0, 0], [0, 0, 0]],
      [[0, 0, 0], [0, 0, 0], [0, 0, 0]]]

# e que é a mesma coisa de:
mat = []
for i in range(4):
    linha = []
    for j in range(3): #colunas
        col = [0] * 3
        linha.append(col)
    mat.append(linha)
```



# Atribuindo valores a uma matriz

```
# mat é uma matriz com 3 linhas e 4 colunas
mat[0][2] = 5
mat[2][0] = 7
mat[1][1] = 11
mat[2][3] = 8
```

	0			5	
mat	1		11		
	2	7			8
		0	1	2	3

# Atribuindo valores a uma matriz

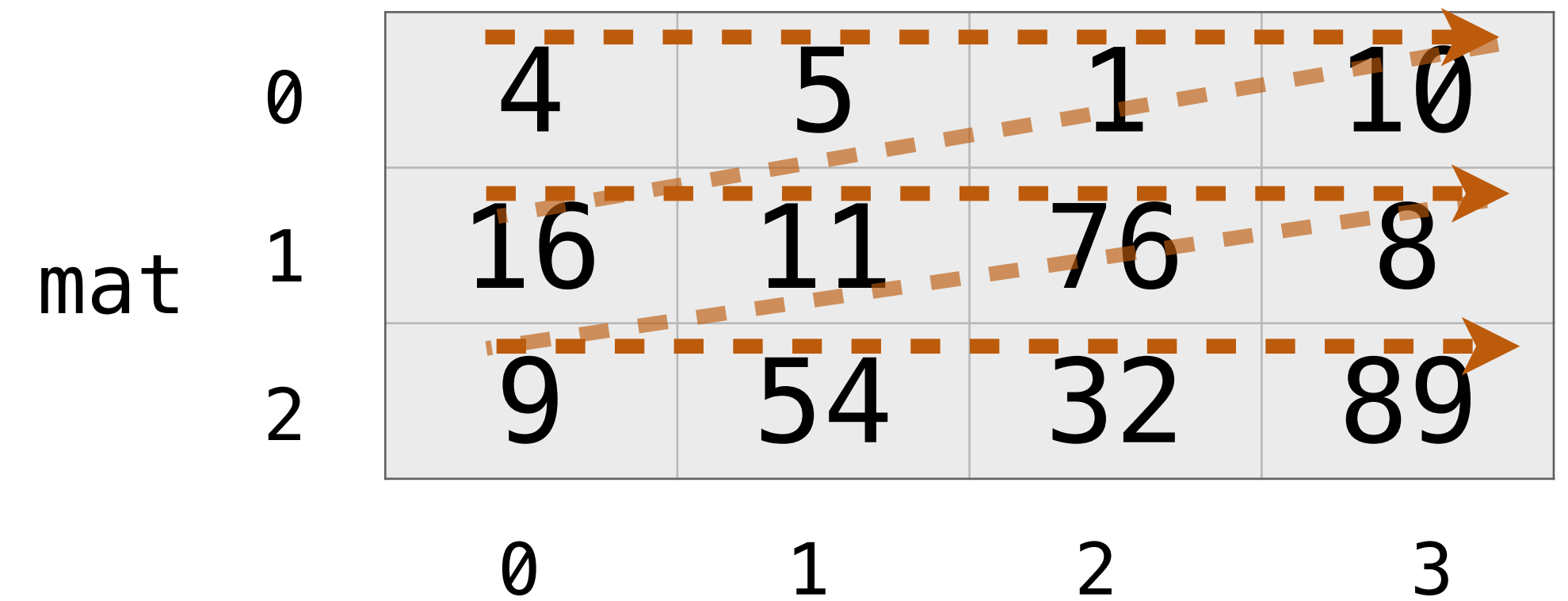
```
# mat é uma matriz com 4 linhas e 3 colunas  
e profundidade 2  
mat[0][2][0] = 5  
mat[2][1][1] = 7  
mat[3][2][1] = 11
```

# Preenchendo uma matriz

```
for i in range(4):  
    for j in range(3):  
        mat[i][j] = int(input("digite: "))
```

```
for i in range(5):  
    for j in range(4):  
        for k in range(2):  
            mat[i][j][k] = int(input("digite: "))
```

# Percorrendo uma matriz: forma 1



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

# Percorrendo uma matriz: forma 2

mat

0	4	5	1	10
1	16	11	76	8
2	9	54	32	89
	0	1	2	3

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

# Algoritmo 1

- Faça um programa que preencha uma matriz  $2 \times 2$  e calcule e mostre o seu maior elemento

# Algoritmo 1

## ALGORITMO

DECLARE  $M[2,2]$ ,  $i$ ,  $j$ , maior NUMÉRICO

PARA  $i \leftarrow 1$  ATÉ 2 FAÇA

INÍCIO

PARA  $j \leftarrow 1$  ATÉ 2 FAÇA

INÍCIO

LEIA  $M[i,j]$

FIM

FIM

maior  $\leftarrow M[1,1]$

PARA  $i \leftarrow 1$  ATÉ 2 FAÇA

INÍCIO

PARA  $j \leftarrow 1$  ATÉ 2 FAÇA

INÍCIO

SE  $M[i,j] >$  maior ENTÃO maior  $\leftarrow M[i,j]$

FIM

FIM

ESCREVA maior

FIM\_ALGORITMO.

```
#coding: utf-8
M = [[0,0],[0,0]]
for i in range(2):
    for j in range(2):
        M[i][j] = float(input("Digite um número: "))
maior = M[0][0]
for i in range(2):
    for j in range(2):
        if M[i][j] > maior:
            maior = M[i][j]
print(maior)
```



# Algoritmo 2

- Faça um programa que preencha uma matriz  $2 \times 2$ , calcule e mostre a matriz  $R$ , resultante da multiplicação dos elementos de  $M$  pelo seu maior elemento

## ALGORITMO

DECLARE  $M[2,2]$ ,  $R[2,2]$ ,  $i$ ,  $j$ , maior NUMÉRICO

PARA  $i \leftarrow 1$  ATÉ 2 FAÇA

INÍCIO

PARA  $j \leftarrow 1$  ATÉ 2 FAÇA

INÍCIO

LEIA  $M[i,j]$

FIM

FIM

maior  $\leftarrow M[1,1]$

PARA  $i \leftarrow 1$  ATÉ 2 FAÇA

INÍCIO

PARA  $j \leftarrow 1$  ATÉ 2 FAÇA

INÍCIO

SE  $M[i,j] >$  maior ENTÃO maior  $\leftarrow M[i,j]$

FIM

FIM

...

```
...  
PARA i ← 1 ATÉ 2 FAÇA  
INÍCIO  
  PARA j ← 1 ATÉ 2 FAÇA  
  INÍCIO  
    R[i,j] ← maior * M[i,j]  
  FIM  
FIM  
  
PARA i ← 1 ATÉ 2 FAÇA  
INÍCIO  
  PARA j ← 1 ATÉ 2 FAÇA  
  INÍCIO  
    ESCREVA R[i,j]  
  FIM  
FIM  
FIM_ALGORITMO.
```

```
#coding: utf-8
M = [[0,0],[0,0]]
for i in range(2):
    for j in range(2):
        M[i][j] = float(input("Digite um número: "))
maior = M[0][0]
for i in range(2):
    for j in range(2):
        if M[i][j] > maior:
            maior = M[i][j]
R = [[0,0],[0,0]]
for i in range(2):
    for j in range(2):
        R[i][j] = maior * M[i][j]
for i in range(2):
    for j in range(2):
        print(R[i][j])
```

# Algoritmo 3

- Faça um programa que preencha uma matriz  $5 \times 3$  com as notas de 5 alunos em três provas. O programa deverá mostrar um relatório com o número dos 5 alunos e a prova em que cada aluno obteve menor nota. Ao final do relatório, deverá mostrar quantos alunos tiveram menor nota em cada uma das provas: na prova 1, na prova 2, na prova 3:

## ALGORITMO

**DECLARE** notas[5,3], i, j, menor, menor\_p, q1, q2, q3 **NUMÉRICO**

**PARA** i ← 1 **ATÉ** 5 **FAÇA**

**INÍCIO**

**PARA** j ← 1 **ATÉ** 3 **FAÇA**

**INÍCIO**

**LEIA** notas[i,j]

**FIM**

**FIM**

q1 ← 0

q2 ← 0

q3 ← 0

...

```
...
PARA i ← 1 ATÉ 5 FAÇA
INÍCIO
  ESCREVA i
  menor ← notas[i,1]
  menor_p ← 1
  PARA j ← 1 ATÉ 3 FAÇA
  INÍCIO
    SE notas[i,j] < menor ENTÃO
      INÍCIO
        menor ← notas[i,1]
        menor_p ← j
      FIM
    FIM
  FIM
  ESCREVA p_menor
  SE menor_p = 1 ENTÃO q1 ← q1 + 1
  SE menor_p = 2 ENTÃO q2 ← q2 + 1
  SE menor_p = 3 ENTÃO q3 ← q3 + 1
FIM
ESCREVA q1, q2, q3
FIM_ALGORITMO.
```

```
#coding: utf-8
notas = []
for i in range(5):
    linha = [0] * 3
    notas.append(linha)

for i in range(5):
    for j in range(3):
        notas[i][j] = float(input("Digite a nota %d do aluno %d: "% (j
+1, i+1)))
q1 = 0
q2 = 0
q3 = 0
# continua no próximo slide
```



```
for i in range(5):
    menor = notas[i][0]
    menor_p = 0
    for j in range(3):
        if notas[i][j] < menor:
            menor = notas[i][j]
            menor_p = j
    print(i, menor_p)
    if menor_p == 0:
        q1 += 1
    if menor_p == 1:
        q2 += 1
    if menor_p == 2:
        q3 += 1
print(q1, q2, q3)
```