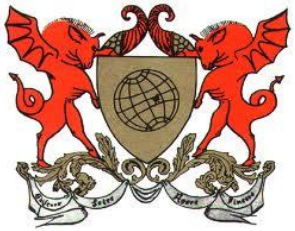
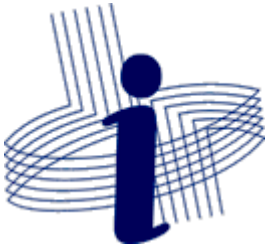


Universidade Federal de Viçosa
Departamento de Informática
Centro de Ciências Exatas e Tecnológicas



INF 100 – Introduction to Programming

Arrays

(cont.)

Multidimensional arrays

matrix

Indexes	0	1	2	3
0	1	2.5	0	-4
1	0	1	-2	0.5
2	0	4	1	0.7

Multidimensional arrays

matrix

Indexes	0	1	2	3
0	1	2.5	0	-4
1	0	1	-2	0.5
2	0	4	1	0.7

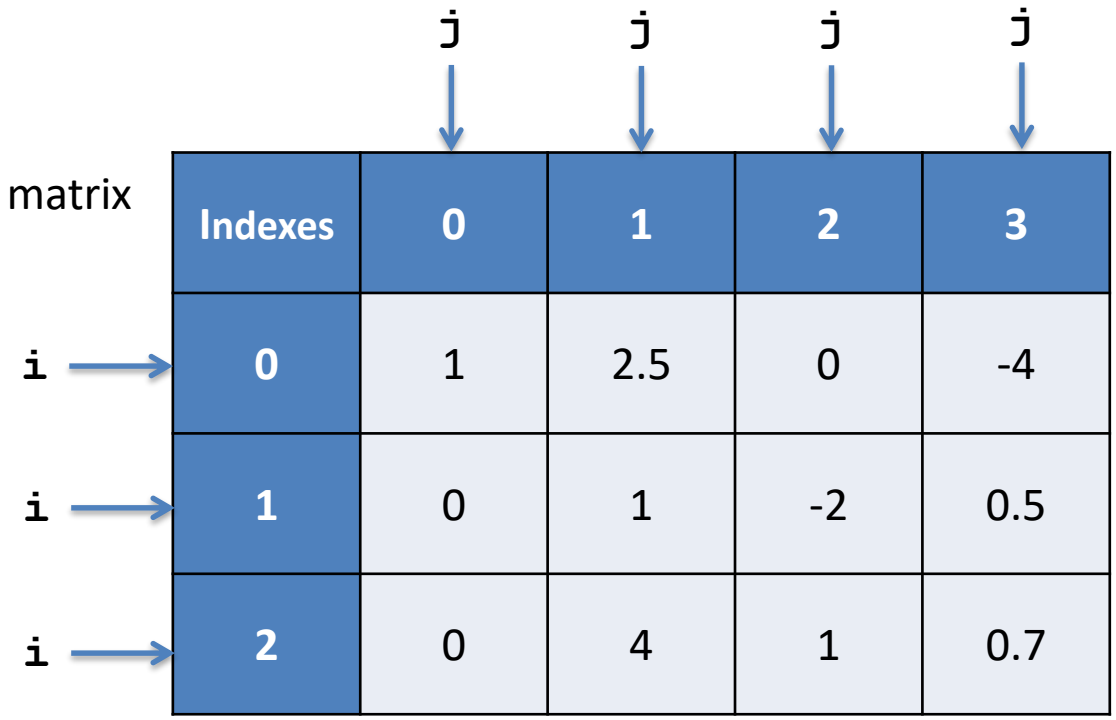
Space allocated in RAM memory

Representation in Python:

```
[
[1,   2.5,   0,    -4],
[0,   1,     -2,   0.5],
[0,   4,     1,    0.7]
]
```

Multidimensional arrays

```
matrix[i][j]   # ref. to line i, column j of matrix
```



Processing elements in a matrix

```
for each line i of the matrix M
    for each column j of the matrix M
        do something with the element M[i][j]
```

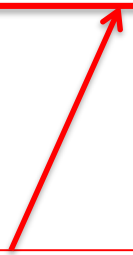
Processing elements in a matrix

```
for each line i of the matrix M
    for each column j of the matrix M
        do something with the element M[i][j]
```

This loop is executed FOR EACH value of i

Processing elements in a matrix

```
for each line i of the matrix M
    for each column j of the matrix M
        do something with the element M[i][j]
```



This command is executed FOR EACH value of i and j

Reading elements in a matrix

```
import numpy as np

A = np.empty( (3, 4) )  # matriz A: 3 x 4

for i in range(0, 3): # lines 0 to 2
    for j in range(0, 4): # columns 0 to 3
        s = 'Type element A[%d][%d]: ' % (i, j)
        A[i][j] = float (input( s ))
```


Reading elements in a matrix

```
import numpy as np

# dimensions defined at runtime by the user
m = int( input('Number of lines : '))
n = int( input('Number of columns: '))
A = np.empty( (m, n) )

for i in range( 0, m ): # lines 0 to m-1
    for j in range( 0, n ): # columns 0 to n-1
        s = 'Type element A[%d][%d]: ' % (i, j)
        A[i][j] = float (input( s ))
```

“Pretty-printing” a matrix

```
# printing matrix with integer elements

mi = np.asarray([[1, 2, 3],
                 [4, 5, 6],
                 [7, 8, 9],
                 [10, 11, 12]])

for i in range(0, 4):
    for j in range(0, 3):
        print('%3d' % (mi[i][j]), end='')
    print()
```

Output:

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

“Pretty-printing” a float matrix

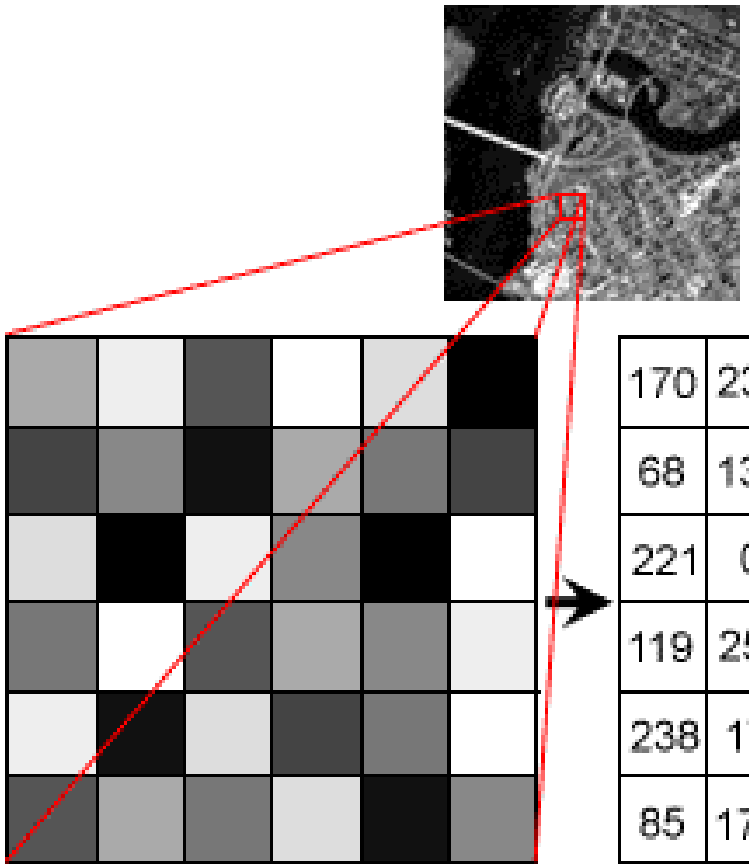
```
mf = np.asarray(
     [[1.1, 2.5, 3],
     [4, 5.2, 6],
     [7.8, 8, 9.9],
     [10, 11, 12.3]])

for i in range(0, 4):
    for j in range(0, 3):
        print('%5.1f' % (mf[i][j]), end='')
    print()
```

Output:

```
  1.1  2.5  3.0
  4.0  5.2  6.0
  7.8  8.0  9.9
 10.0 11.0 12.3
```

Application - digital images



170	238	85	255	221	0
68	136	17	170	119	68
221	0	238	136	0	255
119	255	85	170	136	238
238	17	221	68	119	255
85	170	119	221	17	136

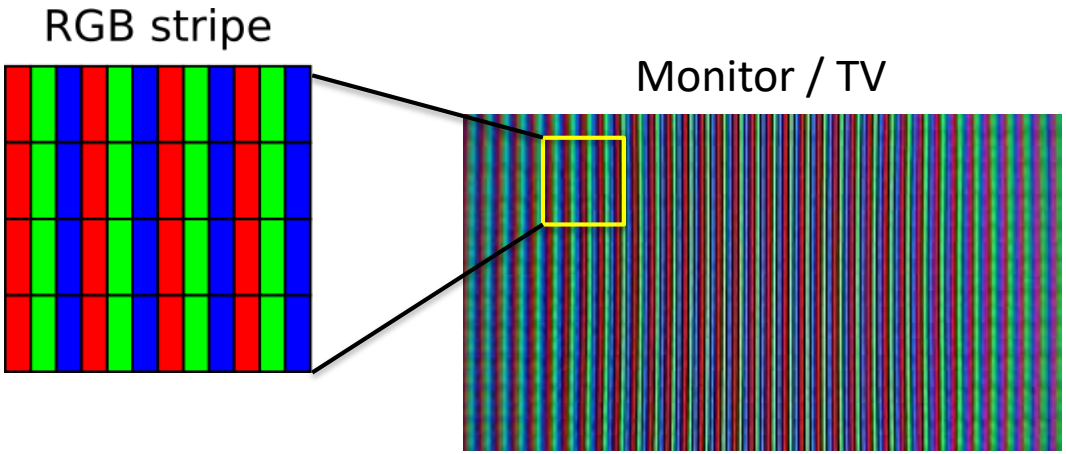
1 pixel

Grayscale images:
256 levels of gray
0 = black
255 = white

Application - digital images

Color images:
Each RGB componente of a pixel is a value between 0 and 255

$256^3 \approx 16$ million colors



Digital image processing
(generic algorithm)

```
For each pixel of the image:
    c = get the value (color) of the pixel
    modify c (according to a specific algorithm)
    write the new value c in the image
```


Digital image processing
(generic algorithm)

```
width = get the width of the image
height = get the height of the image

for y in range(0, height):
    for x in range(0, width):
        r, g, b = image.getpixel( (x,y) )
        Modify the values of r, g, b
        imagem.putpixel( (x,y), (r,g,b) )
```


Exercise 1

- Write a program that reads a matrix A(*m* x *n*) with float elements and then:
  a) Create an array B with *m* elements, where each cell stores the sum of the elements of the corresponding line of the matrix A;
  b) Create an array C with *n* elements, where each cell stores the sum of the elements of the corresponding column of the matrix A.


Solution

```
import numpy as np

m = int( input('Número de linhas : '))
n = int( input('Número de colunas: '))
A = np.zeros( (m, n) )
for i in range(0, m): # linhas 0 a m-1
    for j in range(0, n): # colunas 0 a n-1
        s = 'Informe o elemento A[%d][%d]: ' % ( i, j )
        A[i][j] = float (input( s ))

print('\nMatriz lida:\n')
for i in range(0, m):
    for j in range(0, n):
        print('%6.1f' % ( A[i][j] ), end='')
    print()

B = np.zeros( m )
for i in range(0, m):
    for j in range(0, n):
        B[i] = B[i] + A[i][j]

C = np.zeros( n )
for j in range(0, n):
    for i in range(0, m):
        C[j] = C[j] + A[i][j]

# Escreve B
print('\nB = ', end='')
for i in range(0, m):
    print('%6.1f' % ( B[i] ), end='')
print()

# Escreve C
print('\nC = ', end='')
for i in range(0, n):
    print('%6.1f' % ( C[i] ), end='')
print()
```

Exercise 2

- Write a program that reads a matrix A(*n* x *n*) and then:
  a) Prints the product of the values on the main diagonal;
  b) Prints the product of the values on the antidiagonal;
  c) Calculates the average of all values of the matrix and prints the number of values that are below this average.


Solution

```
import numpy as np

m = int( input('m = '))
A = np.zeros( (m, m) )
for i in range(0, m): # linhas 0 a m-1
    for j in range(0, m): # colunas 0 a m-1
        s = 'Informe o elemento A[%d][%d]: ' % ( i, j )
        A[i][j] = float (input( s ))

print('\nMatriz lida:\n')
for i in range(0, m):
    for j in range(0, m):
        print('%6.1f' % ( A[i][j] ), end='')
    print()

p = 1
for i in range(0, m):
    p = p * A[i][i]
print('\nProduto dos elementos da diagonal principal =', p
)

p = 1
for i in range(0, m):
    p = p * A[i][m-1-i]
print('Produto dos elementos da diagonal secundária =', p )

# Calculando a média...
soma = 0
for i in range(0, m):
    for j in range(0, m):
        soma = soma + A[i][j]
media = soma/(m*m)
print('Média dos elementos da matriz:', media )

# Nº elementos abaixo da média...
nAbaixo = 0
for i in range(0, m):
    for j in range(0, m):
        if A[i][j] < media: nAbaixo = nAbaixo + 1
print('Nº de elementos abaixo da média:', nAbaixo )
```

Exercise 3 – Determinant of a matrix

- Write a program that reads a matrix A(*n* x *n*) with *n* assuming only 2 or 3, and then calculates the determinant of the matrix.


Exercise 3 – Determinant of a matrix

The determinant of a 2 × 2 matrix is defined by

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc.$$

The determinant of a 3 × 3 matrix is defined by

$$\begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = a\begin{vmatrix} e & f \\ h & i \end{vmatrix} - b\begin{vmatrix} d & f \\ g & i \end{vmatrix} + c\begin{vmatrix} d & e \\ g & h \end{vmatrix}$$

$$= a(ei - fh) - b(di - fg) + c(dh - eg)$$

$$= aei + bfg + cdh - ceg - bdi - afh.$$

Solution (1/2)

```
import numpy as np

while True:
    k = int(input('Type size of the matrix (2 or 3): '))
    if k == 2 or k == 3:
        break
    print("Must be 2 or 3")

print()
M = np.zeros( (k, k) )
for i in range(0, k):
    for j in range(0, k):
        s = 'Type element M[%d][%d]: ' % (i, j)
        M[i][j] = float(input(s))

print('\nMatrix typed:')
for i in range(0, k):
    for j in range(0, k):
        print('%5.1f' % (M[i][j]), end='')
    print()
```

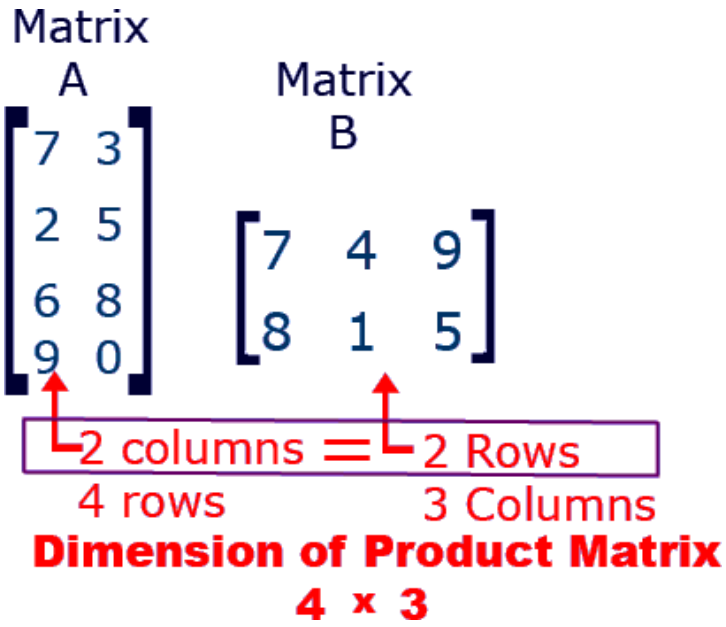

Solution (2/2)

```
if k == 2:
    det = M[0][0]*M[1][1]-M[0][1]*M[1][0]
else:
    det = M[0][0]*M[1][1]*M[2][2] \
    + M[0][1]*M[1][2]*M[2][0] \
    + M[0][2]*M[1][0]*M[2][1] \
    - M[0][2]*M[1][1]*M[2][0] \
    - M[0][0]*M[1][2]*M[2][1] \
    - M[0][1]*M[1][0]*M[2][2]

print("Determinant= ", det)
```


Exercise 4 – matrix multiplication

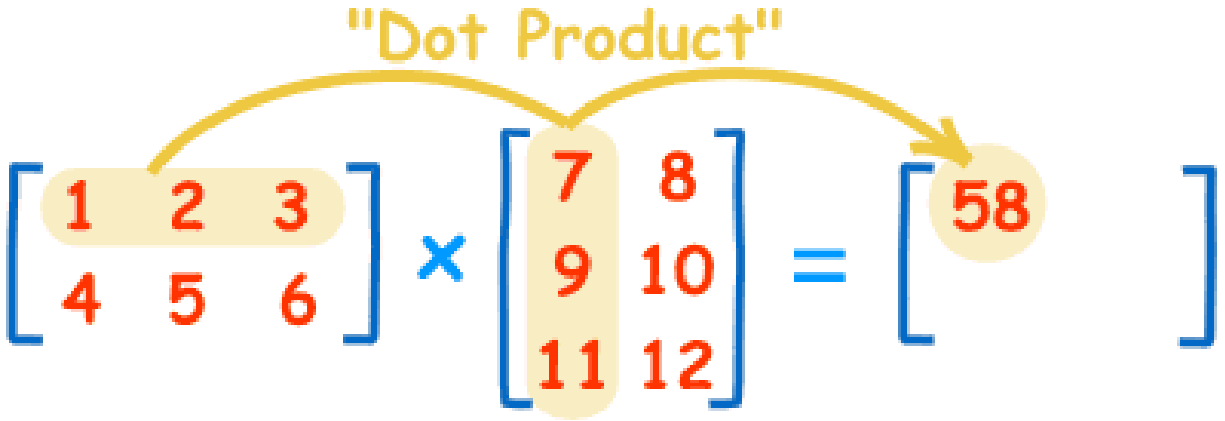
- Write a program that reads 2 integer matrices A_{mn} and B_{np} and then calculates and prints a matrix C_{mp}, resulting from the product A B.



Exercise 4 – matrix multiplication

$$P \in \mathbb{M}^{mxn},\ Q \in \mathbb{M}^{nxp}$$

$$(PxQ)(i,j) = \sum_{k=0}^{n} \left(P(i,k) \times Q(k,j) \right)$$



Solution (1/2)

```
import numpy as np

m = int( input('Number of lines in matrix A: '))
n = int( input('Number of columns in matrix A: '))
p = int( input('Number of columns in matrix B: '))

print()
A = np.zeros( (m, n) )
for i in range(0, m):
    for j in range(0, n):
        s = 'Type element A[%d][%d]: ' % (i, j)
        A[i][j] = float(input(s))

print('\nMatrix A:')
for i in range(0, m):
    for j in range(0, n):
        print('%5.1f' % (A[i][j]), end='')
    print()

print()
B = np.zeros( (n, p) )
for i in range(0, n):
    for j in range(0, p):
        s = 'Type element B[%d][%d]: ' % (i, j)
        B[i][j] = float(input(s))

print('\nMatrix B:')
for i in range(0, n):
    for j in range(0, p):
        print('%5.1f' % (B[i][j]), end='')
    print()
```

Solution (2/2)

```
C = np.zeros( (m, p) )
for i in range(0, m):
    for j in range(0, p):
        for k in range(0, n):
            C[i][j] += A[i,k] * B[k,j]

print('\nProduct AB :')
for i in range(0, m):
    for j in range(0, p):
        print('%5.1f' % (C[i][j]), end='')
    print()
```


Exercise 5

- In the game **Minesweeper**, a certain number of bombs is distributed in a map, represented by a matrix. A cell that does not contain a bomb must indicate the number of bombs it has around it.
- Write a program that reads the size of the map and the number of bombs. Then it must distribute the bombs randomly and afterwards it must calculate the number of bombs around each non-bomb cell.


Sample execution
(suppose that a bomb is represented by “9”)

```
Size of the square field: 4
Number of bombs: 5

Distributing bombs:
 0 0 0 0
 0 9 9 0
 9 9 0 0
 0 0 9 0

Calculating bombs around each cell:
 1 2 2 1
 3 9 9 1
 9 9 4 2
 2 3 9 1
```


Solution (1/2)

```
import numpy as np
import random

bomb = 9

n = int(input("Size of the square map: "))
b = int(input("Number of bombs: "))

map = np.zeros((n, n), dtype=int)

k = b
while k > 0:
    i = random.randint(0, n-1)
    j = random.randint(0, n-1)
    if map[i][j] == 0:
        map[i][j] = bomb
        k -= 1

print()
for i in range(0, n):
    for j in range(0, n):
        print('%2d' % (map[i][j]), end='')
    print()
```

Solution (2/2)

```
for i in range(0, n):
    for j in range(0, n):
        if map[i][j] == bomb:
            if i-1 >= 0 and j-1 >= 0 and i-1 < n and j-1 < n and map[i-1][j-1] != bomb:
                map[i-1][j-1] += 1
            if i-1 >= 0 and j >= 0 and i-1 < n and j < n and map[i-1][j] != bomb:
                map[i-1][j] += 1
            if i-1 >= 0 and j+1 >= 0 and i-1 < n and j+1 < n and map[i-1][j+1] != bomb:
                map[i-1][j+1] += 1
            if i >= 0 and j-1 >= 0 and i < n and j-1 < n and map[i][j-1] != bomb:
                map[i][j-1] += 1
            if i >= 0 and j+1 >= 0 and i < n and j+1 < n and map[i][j+1] != bomb:
                map[i][j+1] += 1
            if i+1 >= 0 and j-1 >= 0 and i+1 < n and j-1 < n and map[i+1][j-1] != bomb:
                map[i+1][j-1] += 1
            if i+1 >= 0 and j >= 0 and i+1 < n and j < n and map[i+1][j] != bomb:
                map[i+1][j] += 1
            if i+1 >= 0 and j+1 >= 0 and i+1 < n and j+1 < n and map[i+1][j+1] != bomb:
                map[i+1][j+1] += 1

print()
for i in range(0, n):
    for j in range(0, n):
        print('%2d' % (map[i][j]), end='')
    print()
```