ITI1120
Lab 8

Exercises with Arrays

Objectives

- Arrays and et exercises using them
  - Examples:
  - 2D Lists
  - Display an array
  - Read an array from the keyboard
  - Sum of the values in the upper triangle

  - Exercise 1: Matrix transposed
  - Exercise 2: Sum of an array
  - Exercise 3: Multiplication with arrays

Arrays

- An array is a 2 dimensional rectangular table:

$$M = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

- The dimensions are the number of row and columns (3x3 for the above example).
- We can refer to an element of M by specifying its row and column in that order.
  - In mathematics: rows and columns start at 1, on the upper left corner:
    - With a mathematical notation , $M_{1,2} = 2$
  - In Python, we usedes indexes starting at 0, similar to the lists.
    - With an algorithmic notation , M[0][1] ← 2

An array in Python is a 2D list

- To creater and initialize a 2D list (array of 2x3)

```
>>> m = [[1, 2, 3], [4, 5, 6]]
>>> print(m)
>>> [[1, 2, 3], [4, 5, 6]]
```

- The function **len** returns the size (number of rows):

```
>>> len(m)
>>> 2
>>> len(m[0])    # number of columns?
>>> 3
```

- Recall that a matrix in an array where the number of rows is equal to those of columns
- >>> liste1 = [[1,2], [3,4,5]]

- 3D List (2x2x2)

```
>>> m3 = [[[1,2],[3,4], [5,6]]]
>>> m3[0][0][0]
>>> 1
```

Display an array

```
matrix = [[1,2,3],[4,5,6],[7,8,9]]

for i in matrix:    # visi each row
  for j in i:        # visit each element  of the row
    print(j, end=" ")
  print()

# alternative
i = 0
while i < len(matrix):
  j = 0
  while j < len(matrix[i]):
    print(matrix[i][j], end=" ")
    j = j + 1
  i = i + 1
  print()
```

Lecture of an array from a keyboard

```
m = int(input("Enter the number of rows: "))
n = int(input("Enter the number of columns: "))
matrix = []
i = 0
while (i < m):
  j = 0
  matrix.append([])
  while j < n:
     v = int(input("matrix["+str(i)+","+str(j) +"]="))
     matrice[i].append(v)
     j = j + 1
  i = i + 1

# values sare converted in int (or other type as needed)
```

Lecture of an array from a keyboard (version 2)

```
m = int(input("Enter the number of rows: "))
matrix = []
i = 0
while (i < m):
     print("Enter the row", i,
           "(integers separated by spaces)")
     row = [int(val) for val in input().split()]
     matrix.append(row)
     i = i + 1
```

Lecture of an array from a keyboard (version 3)

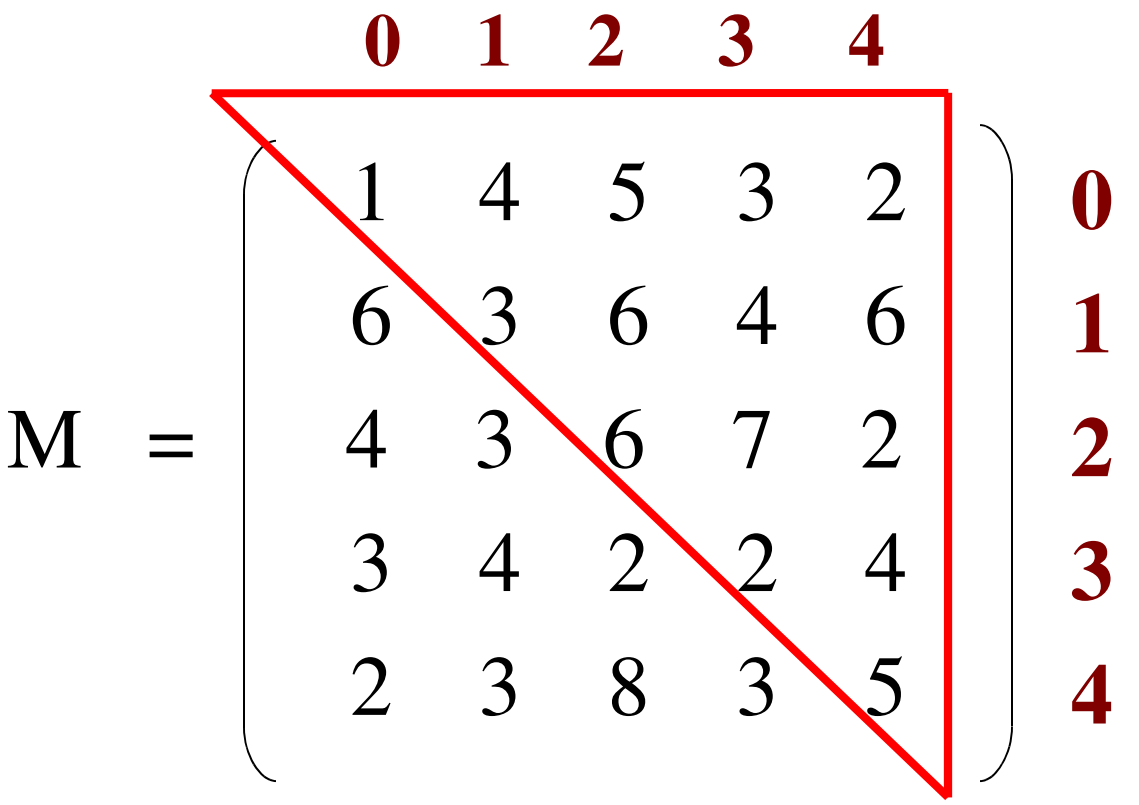
```
print("Enter the numbes with spaces between columns.")
print(« One row per line, and an empty line une ligne vide a la fin.")
matrix = []
while True:
      row = input()
      if not row: break
      values = row.split()
      column = [int(val) for val in values]
      matrix.append(row)

#Rows do not need to be of the same size unless if we
# need a matrix
```

Traitement des éléments d'une matrice

- Pour visiter tous les éléments d'une liste, nous avions besoin d'une boucle.
- Pour visiter tous les éléments d'une matrice, nous aurons besoin de *deux* boucles imbriquées:
  - Boucle extérieure: parcourt les rangées
  - Boucle intérieure: parcourt les colonnes pour une rangée donnée.

Example of a matrix

Derive a Python program that sums up elements of the upper right triangle.



How to determine if an element is on the diagonal or above?

row_index <= col_index

Example - suite

DATA:

M *(matrix numbers)*

N *(size of M)*

RESULT:

Sum *(sum of the upper right triangle)*

INTERMEDIARIES:

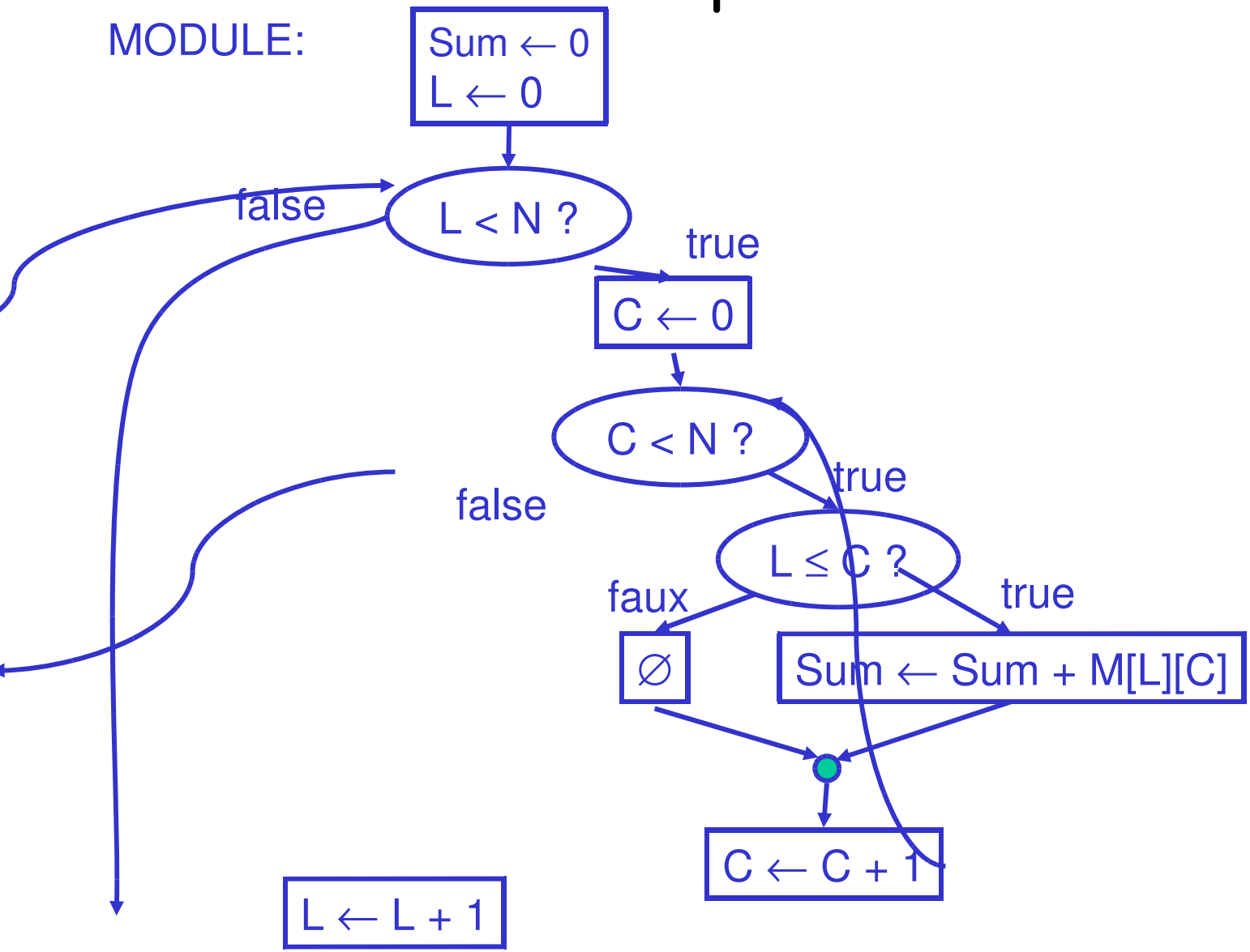
L *(row index)*

C *(column index)*

EN-TÊTE:

Sum ← CalculateUpperTriangle(M, N)

Example - suite



Python Implementation

```
def calculateUpperTriangle(m):
  ''' (list) -> list
  Returns the sum of the upper right triangle
  Precondition: m is an integer matrix
  '''
  sum = 0
  L = 0
  while L < len(m):
     C = 0
     while C < len(m[L]):
        if L <= C:
          sum = sum + m[L][C]
        C = C + 1
     L = L + 1
  return sum

print(calculeUpperTriagle([[1,2],[3,4]]))
```

Exercise 1: Transposed Matrix

- Derive an algorithm that takes as input an integer matrix A and transposes this matrix to produce a new matrix A^T. Transposing a matrix requires each element a_{lc} of the original matrix to become the element $a^T{}_{cl}$ in the transposed matrix. The number of rows in A becomes the number of columns in A^T, and the number of columns in A the number of rows in A^T.

- Par example:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \qquad A^T = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

Transposed matrix in Python

- Create a function which takes a matrix and returns a new one that is the transposed of the initial one.
- The main program must read the matrix from the keyboard,derive the transposed matrix and display it.

Example:

```
>>> L = [[1,2,3],[4,5,6]]
>>> L1 = transpose(L)
>>> L1
[[1, 4], [2, 5], [3, 6]]
```

Exercise 2: Sum of a matrix

- Suppose a matrix (*mxn*) A and B being from the same size. Element in row i and column j f A or is denoted by a_{ij}.

- Let $C = A + B$. Thus, C is a matrix $m \times n$, and for $0 \leq i < m$, et $0 \leq j < n$:

$$c_{ij} = \sum_{k=0}^{n-1} a_{ij} + b_{ij}$$

- Derive a function Python that sums up matrixes A and B of same dimensions.

Sum of matrixes in Python

- Create a function that takes 2 matrices and returns a new matrix that is their sum.
- The main program must read two matrixes and display their sum, the result.

Example:

```
>>> m = sum_matrixes([[1,2],[3,4]], [[1,1],[1,1]])
>>> m
[[2, 3], [4, 5]]
```

Exercise 3: Multiplication of matrixes

- Suppose A is a matrix $m \times n$ and B a matrix $n \times p$. The element at the row i and the column j of A is denoted by a_{ij}.

- Let $C = A \times B$. Thus, C is a matrix $m \times p$, and for $0 \leq i < m$, et $0 \leq j < p$

$$c_{ij} = \sum_{k=0}^{n-1} a_{ik} b_{kj}$$

Derive a function Python that multiplies two matrixes A and B of compatibles dimensions .

Multiplication of matrixes in Python

- Create a function that takes two matrixes and returns a new matrix that is the product of the two.
- The main program must take two matrixes and display the result of their multiplication.

Example:

```
>>> m = product_matrixes([[1,2,3],[4,5,6]], [[1,2],
  [3,4],[5,6]])
>>> m
[[22, 28], [49, 64]]
```