ITI1120
Lab 7

Arrays and their Illustrations

Objectives

- Arrays and and their applicatins
  - Examples:
    - 2D Lists
    - Display an array
    - Read an array from the keyboard
    - Sum of the values in the upper triangle
  - Exercise 1: Transposed matrix
  - 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 use 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 create 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 is 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 of an array

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

for i in matrix:   # visit 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 are converted in int (or other types 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 number with spaces between
columns.")
print(« One row per line, and an empty line at the
end.")
matrice = []
while True:
     line = input()
     if not line: break
     valeurs = line.split()
     rangee = [int(val) for val in valeurs]
     matrice.append(rangee)

# Rows do not have to be of the same size,
# unless it is a matrix.
```

Processing data in an array

- To go through every elements of a list, we need a loop.
- Similarly for an array we will need a *two* nested loops:
  - The outside loop: visits the rows
  - The inside loop: visits the columns for a given row.

Example of an array

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

$$
M = \begin{array}{c c}
\begin{array}{ccccc} \mathbf{0} & \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} \end{array} & \\
\begin{pmatrix} 1 & 4 & 5 & 3 & 2 \\ 6 & 3 & 6 & 4 & 6 \\ 4 & 3 & 6 & 7 & 2 \\ 3 & 4 & 2 & 2 & 4 \\ 2 & 3 & 8 & 3 & 5 \end{pmatrix} & \begin{array}{c} \mathbf{0} \\ \mathbf{1} \\ \mathbf{2} \\ \mathbf{3} \\ \mathbf{4} \end{array}
\end{array}
$$

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)

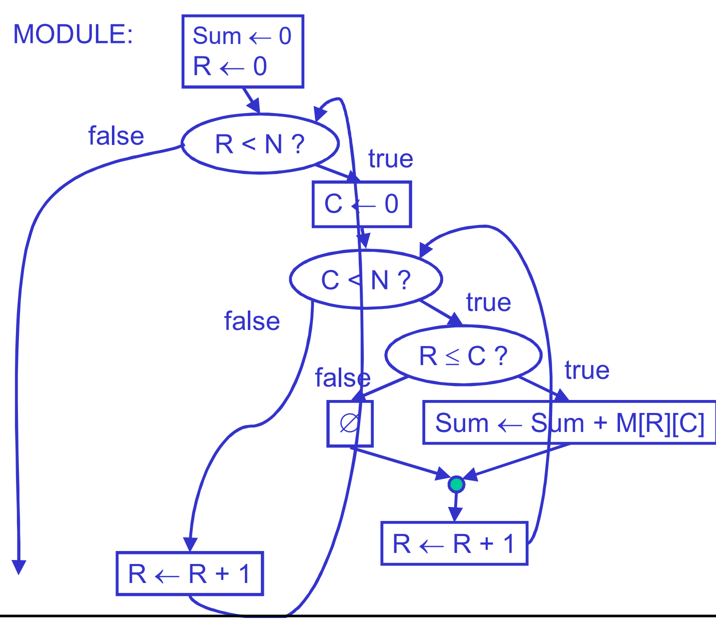
RESULT:
Sum (sum of the upper right triangle)

INTERMEDIAIRIES:
R (row index)
C (column index)

HEADER:
Sum ← ComputeUpperTriangle(M, N)

Example - suite

MODULE:



Implementation in Python

```
def computeUpperTriangle(m):
  ''' (list) -> list
  returns the sum of the upper triangle
  Precondition: m has only integers
  '''
  sum = 0
  R = 0
  while R < len(m):
     C = 0
     while C < len(m[R]):
        if R <= C:
          sum = sum + m[R][C]
        C = C + 1
     R = R + 1
  return sum

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

Exercise 1: Transposed Matrix

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

- 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

- Creates a function which takes a matrix and returns a new matrix that is the transposed of the original 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 that A is a matrix ($m \times n$) and that B is a matrix of the same size $m \times n$. An element in the row i and columne j of A is denoted by a_{ij}.
- Let $C = A + B$. Thus C is a matrix $m \times n$, so that for $0 \le i < m$, and $0 \le j < n$:

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

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

Sum of matrixes in Python

- Derive a function that takes 2 matrixes 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

- Assume that A is a matrixe $m \times n$ and that B is a matrix $n \times p$. The element in row i and column j of A is denoted by par a_{ij}.
- Let $C = A \times B$. Thus, C is a matrix $m \times p$, so that for $0 \leq i < m$, and $0 \leq j < p$:

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

- Derive a Python function that multiplies two matrixes A and B ofcompatible sizes.

Multiplication of matrixes in Python

- Derive a function that takes two matrixes and returns a new matrix that is their product.
- The main program must take two matrixes and display their product (the result).

Example:

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