ITI 1120

Lab # 8

(2D) lists, matrices

Starting Lab 8

- Open a browser and log into Blackboard Learn
- On the left hand side under Labs tab, find lab6 material contained in lab8-students.zip file
- Download that file to the Desktop and unzip it.

Before starting, always make sure you are running Python 3

This slide is applicable to all labs, exercises, assignments … etc

ALWAYS MAKE SURE FIRST that you are running Python 3.4 (3.5 is fine too)

That is, when you click on IDLE (or start python any other way) look at the first line that the Python shell displays. It should say Python 3.4 or 3.5 (and then some extra digits)

If you do not know how to do this, read the material provided with Lab 1. It explains it step by step

Introduction to matrices

- A matrix is a two dimensional rectangular grid of numbers:

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

- The dimensions of the matrix are the numbers of rows and columns (in the above case: row dimension 3, column dimension 3).
- A value within a matrix is referred to by its row and column indices, in that order.
  - Math: number rows and columns from 1, from upper left corner
    - In math notation, $M_{1,2} = 2$
  - In Python, matrices are implemented via 2D lists and indices start from 0, as they do in (1D) lists.
    - Thus, M[0][1] is 2

Matrix element processing

- To visit every element of an array, we had to use a loop.
- To visit every element of a matrix, we need to use a loop inside a loop:
  - Typically the outer loop goes through each row of a matrix
  - And the inner loop goes through each column within one row of a matrix.

Intro to matrices in python and programming exercises 1 to 5

Open a file called basics-2Dlists.py and spend time studying all the matrix functions there.

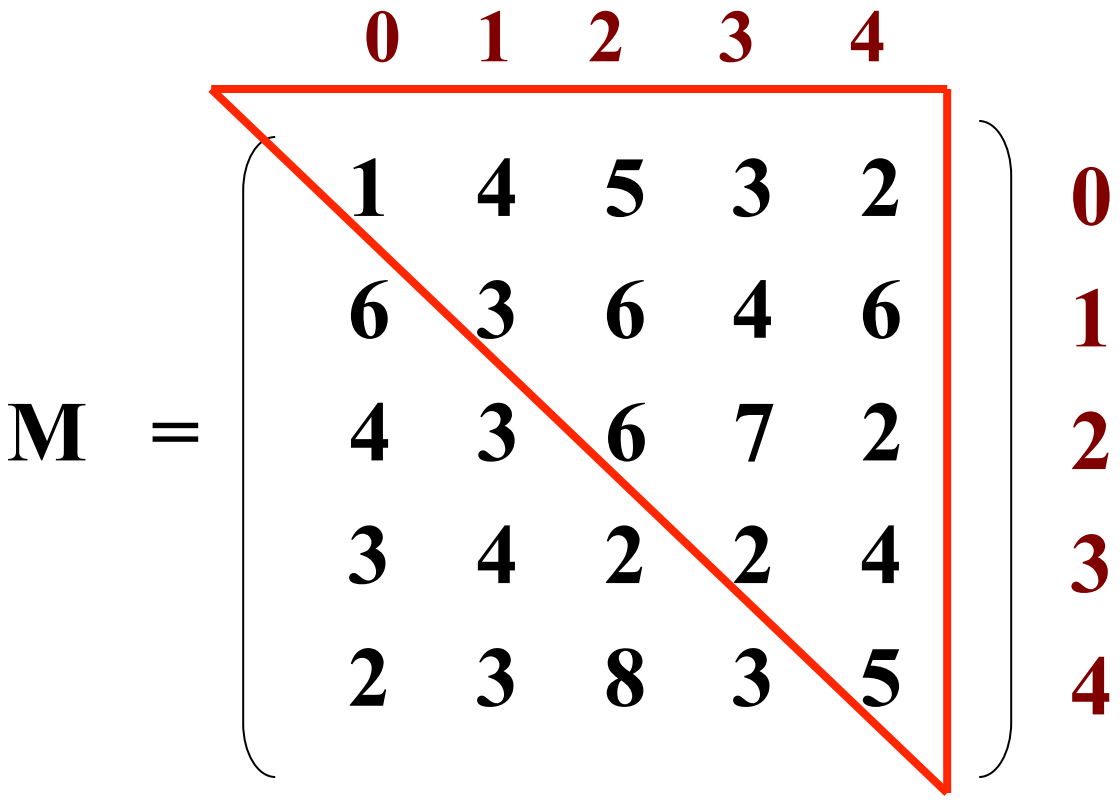
The open the file called basics-2Dlists-todo.py and implement (and test) the 5 functions labeled as programming exercise 1 to 5 in that file.

Two notes about exercises 2 and 5:

- Q2: For clarification of programming exercise 2 see the next page
- Q5: In programming exercise 5, try to find a solution that does not create any extra list. Or even better, find two solutions, one that creates an extra list and one that does not

Details about Prog Ex 2

Find the sum of the upper triangle of a square matrix (i.e. the diagonal and up).



How do we know if an element of a square matrix is on or above the main diagonal?

row_index <= column_index

Programming exercise 6 : Magic square

An *n* x *n* matrix forms am *magic square* if the following conditions are met

1. The elements of the matrix are numbers *1, 2, 3, …, n^2*
2. The sum of the elements in each row, in each column and in the two diagonals is the same value

https://en.wikipedia.org/wiki/Magic_square

Open magic.py and complete the function that tests if the given matrix m forms a magic square.

BONUS programming exercise (Back to 1D lists)

Write a function called move_zeros that takes as input list of integers and moves all the zeros in that list to the end of the list (without affecting the relative order of other numbers). For example, if the list was [1, 0, 3, 0, 0, 5, 7] the new changed list should be [1, 3, 5, 7, 0, 0, 0]

- Write THREE different solutions (Version 3 is challenging)
  - move_zeros_v1 should create a second, tmp, list to build the result (easier problem) and return that 2nd list. It should not change the given list.
  - move_zeros_v2 modify the previous question so that the given list IS modified Inside of the function (i.e. its zeros are at the end) . This function returns nothing.
  - move_zeros_v3 You may change the relative order of other elements here. This version should be moving elements in the same list without creating any additional lists (so called, in-place, solutions). (harder problem) This function returns nothing. You can use one extra variable to store one integer but even that is not necessary since in Python you can swap what variables `a` and `b` refer to by doing `a,b=b,a` For both problems the TAs will first discuss algorithmic approaches to solve the problems

```
>>> x = [1, 0, 3, 0, 0, 5, 7]
>>> y=move_zeros_v1(x)
>>> print(x, y)
[1, 0, 3, 0, 0, 5, 7] [1, 3, 5, 7, 0, 0, 0]
>>> x = [1, 0, 3, 0, 0, 5, 7]
>>> z=move_zeros_v2(x)
>>> print(x, z)
[1, 3, 5, 7, 0, 0, 0] None
>>> x = [1, 0, 3, 0, 0, 5, 7]
>>> t=move_zeros_v3(x)
>>> print(x, t)
[1, 7, 3, 5, 0, 0, 0] None
```