ITI 1120

Lab # 8

(2D) lists, matrices, files

Starting Lab 8

- Open a browser and log into Brightspace 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

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. (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.

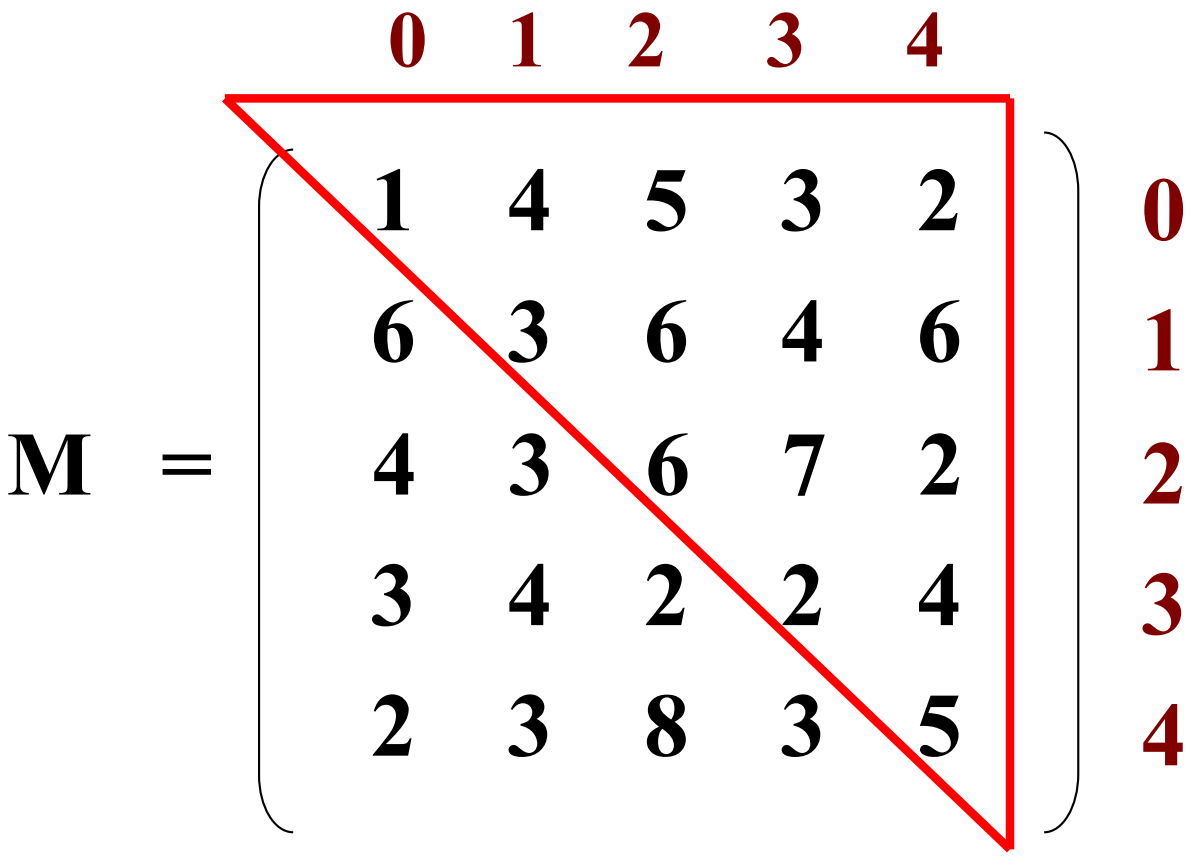
The open the file called basics-2Dlists-todo.py and implement (and test) the 5 functions labeled as programming exercise 1 to 5 in that file.

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.

Learn how to populating a 2D list from a file

View the content of the file called `alkaline_metals.txt`
This file contains the name, atomic number, and atomic weight of the alkaline earth metals separated by space. Records about different metals are separated by new line.

The goal of this exercise is to learn how to write a Python program that opens and reads that file and creates and populates a 2D lists with the relevant info about the 6 alkaline metals. Specifically the program needs to have a function, called `create_alctable(file_name)`, that given a string representing the name of the file as input, opens that file, reads it and returns the following 2D list:

```
[['beryllium', 4, 9.012], ['magnesium', 12, 24.305],
['calcium', 20, 20.078], ['strontium', 38, 87.62], ['barium',
56, 137.327], ['radium', 88, 226.0]]
```

Open working-with-files.py, that solves this problem. Press Run Module. When prompted for the file name enter: `alkaline_metals.txt`
Study what the program prints and the solution code. The printouts should elp you understand the code.
You will need to do something similar in the next programming exercise of this lab and in your Assignment 5.

Programming exercise 7: NY times bestsellers

For this you are provided with 2 files NY_short.txt and NY_long.txt each containing list of New York Times for some years. Each line in a file contains the information for a separate book, which includes: title, author, publisher, date it first reached #1 on one of the best seller lists, and category (fiction or nonfiction). There is a tab character between each of these pieces of information.

For this exercise you will first write a function `create_books_2Dlist(file_name)` that opens the file, reads it and returns a 2D lists containing a sublist with the info about each book. For example here is the beginning of required 2D list. See the next page for what the whole list should look like for

```
[['1976-04-11', '1876', 'Gore Vidal', 'Random House', 'Fiction'], … ]
```

Note that within each (sub)list about a book, the first element should be a string contaning the date in iso format

`YYYY-MM-DD`

Finally you should write a function called `search_by_year(books,year1,year2)` that given a 2D list of books in the above format prints all the bestsellers from year 1 to year2. See example runs on the next 2 pages for NYT_short.txt

While working use NYT_short.txt. Once you are done you can test your program with NYT_long.txt.

Programming exercise 7: example runs

```
>>> books=create_books_2Dlist("NYT_short.txt")
>>> books
[['1976-04-11', '1876', 'Gore Vidal', 'Random House', 'Fiction'], ['2011-11-27', '23337',
'Stephen King', 'Scribner', 'Fiction'], ['1984-07-08', '...and Ladies of the Club', 'Helen
Hooven Santmeyer', 'Putnam', 'Fiction'], ['2001-03-25', '1st to Die', 'James Patterson', '
Little, Brown', 'Fiction'], ['2002-03-24', '2nd Chance', 'James Patterson', 'Little, Brown
', 'Fiction'], ['2004-03-21', '3rd Degree', 'James Patterson', 'Little, Brown', 'Fiction']
, ['2005-05-22', '4th of July', 'James Patterson', 'Little, Brown', 'Fiction'], ['2010-06-
06', '61 Hours', 'Lee Child', 'Delacorte', 'Fiction'], ['2005-10-16', 'A Breath of Snow an
d Ashes', 'Diana Gabaldon', 'Delacorte', 'Fiction'], ['2001-04-29', 'A Common Life', 'Jan
Karon', 'Viking', 'Fiction'], ['2011-07-31', 'A Dance With Dragons', 'George R. R. Martin'
, 'Bantam', 'Fiction'], ['2001-02-04', 'A Day Late and a Dollar Short', 'Terry McMillan',
'Viking', 'Fiction'], ['2005-11-27', 'A Feast For Crows', 'George R. R. Martin', 'Bantam',
'Fiction'], ['1945-07-01', 'A Lion Is In the Streets', 'Adria Locke Langley', 'McGraw', 'F
iction'], ['1998-11-22', 'A Man In Full', 'Tom Wolfe', 'Farrar, Straus & Giroux', 'Fiction
'], ['2001-02-25', 'A Painted House', 'John Grisham', 'Doubleday', 'Fiction']]
```

Programming exercise 7: example runs

Below the books argument is the 2D lists from the previous slide

```
>>> search_by_year(books,2005,2005)

All Titles between 2005 and 2005:

4th of July, by James Patterson (2005-05-22)
A Breath of Snow and Ashes, by Diana Gabaldon (2005-10-16)
A Feast For Crows, by George R. R. Martin (2005-11-27)
>>> search_by_year(books,1945,1999)

All Titles between 1945 and 1999:

1876, by Gore Vidal (1976-04-11)
...and Ladies of the Club, by Helen Hooven Santmeyer (1984-07-08)
A Lion Is In the Streets, by Adria Locke Langley (1945-07-01)
A Man In Full, by Tom Wolfe (1998-11-22)
>>> search_by_year(books,2016,2017)

All Titles between 2016 and 2017:

No books found in that range of years.
>>>
```

EXTRA 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
```