Python for physicists - exercise 5

Submission instructions - please read carefully:

• To be submitted by *** in the moodle (Lemida) system.

• *** files with py suffixes must be submitted - named exactly as detailed below for each exercise.

That is to say that:

• Do not submit complete projects, libraries, zip files, etc., and do not submit all exercises in one file, but in separate files with the names listed below.

• Make sure that the files run and do what is needed (on a recent version of Python, 3.5 or higher).

• Use only the commands we learned in the practice.

Exercise 1. Submit it as file name: ex05-01.py

Given a text file such that each line has a pair of coordinates, in the following structure:

```
10, 15

4, 30

8, 19
```

and so on.

You must write a program that reads the contents of this file and shows two graphs, in the same window, one above the other (i.e. use subplots): the first (top) graph shows all pairs of data, only as points (no connecting lines); the second graph (bottom) shows all the data except for the first 5 pairs (assume that the file has at least 6 lines), and in this graph lines must be shown connecting the points, not just the points themselves.

Exercise 2. Submit it as file name: ex05-02.py

Given a text file in the following structure:

```
# table 1
70, 30
80, 20
90, 16
14, 80


#table 2
1, 1
2, 4
3, 9
4, 16
5, 25


# table 3
8, 18
9, 20
```

You are requested to write a program that reads from the file only the data that is under table2, and displays to the screen a graph of the points in this table.

Please note that the software must work for every file in this structure, not necessarily with the same data numbers in the tables as in the example file above.

<u>**Instruction:**</u> First, open the file with the open command to find what line numbers table2 starts with and ends with.

Then, use the genfromtxt.np command with the header_skip and footer_skip parameters.

Exercise 3. Submit it as file name: ex05-03.py

Write a function that accepts two matrices (numpy arrays) and multiplies them. Assume that the function will receive normal parameters, meaning that the size of the matrices allows to multiply them. The function needs to return the product matrix.

You have to write the function yourself using loops, and without using numpy's matrix multiplication command.

Then, check the running speed of the function: run it on several inputs of different sizes and check using time.time() how long it took it to perform the calculations.

Compare the result with the time of the run obtained by multiplying numpy matrices (with the @ command).

Present a summary of the data (in the same .py file you submit, in a comment.)

Note: to get meaningful comparison results, make sure you start the timer before performing the operation and close it after it. Also, you should perform a large number of operations (in a loop) between opening the timer to close it, to allow a meaningful comparison.

(if you compare one multiplication operation, the time will be so fast and we won't get a meaningful result. That is why it is useful, for example, to measure the time of multiplying 1000 matrices, etc.)