Exercise 2

In the following exercise you are not allowed to use any packages at all, only basic python functions. Don't forget to add documentation to each class and each method.

Remember to write your ID numbers in the answers file and to start each code file with:

id1, id2
6

For your comfort, we uploaded a python file with the classes as asked in the exercise. The classes are waiting for your magnificent coding skills. Please do not change the methods and classes names.

During this exercise you will check the validity of certain parameters. As explain in the lecture, you are allowed to use only *ValueError* or *TypeError*. Make sure you use them correctly. You can find information about each error type in the lecture slides or online.

<u>Part A - Classes</u>:

1) Create a class of a matrix, called Matrix, with the following methods:
   a. Switch between rows.
   b. Multiply row with a number.
   c. Add a row multiplied by a scalar to another row.
      I. First argument (*row_index1*) is the row that is being changed.

$$R_{row\_index1} + (scalar)R_{row\_index2} \rightarrow R_{row\_index1}$$

   d. Gauss algorithm – rank the matrix.
      I. Separate this step to additional methods\functions.

The Matrix also has the following attributes.

a. *n_rows* – number of rows.
b. *n_cols* – number of columns.
c. *mat* – list of lists that hold the matrix items. First list is the first row, the second list is the second row.
   For example: $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \rightarrow [[1,2,3],[4,5,6]]$

The constructor of the Matrix class receives 3 arguments:

a. *n_rows* – number of rows.
b. *n_cols* – number of columns.
c. *items* – list of numbers.

Part of the constructor is already implemented in the py file uploaded to the Moodle. The constructor changes a list of numbers (*items)* to a list containing lists of numbers according to the size of *n_rows* and *n_cols* and saves the new object in *mat* attribute. For example: after creating the following matrix instance, *Matrix(2,3,[1,2,3,4,5,6]),* the *mat* attribute will be equal to $[[1,2,3],[4,5,6]]$.

In this question <u>you will need to check whether the parameters used to create an instance are valid (i.e. the parameters of the constructor.)</u>. For example, if the list contains a string or the *n_rows* is negative, the parameters are invalid. In this case of invalid parameters, you will raise an informative error. You do not need to check the parameters for the other methods.

Testing example:

m = Matrix(3, 2, [1, 2, 3, 4, 5, 6]) # 3 - number of lines, 2 - number of columns

m.switch_rows(1, 2) # indices start from 0

m.mat == [[1, 2], [5, 6], [3, 4]] # should return True

2) Explain the following points in the answers file:
   a. Explain how you implemented each method. 2-3 sentences for each method.
   b. Explain your constructor and how you checked the parameters.
   c. Explain which methods you added and what each method does.

<u>Question 2 – Inheritance and Magic Operators</u>:

Create another matrix class, this time called SquaredMatrix.

The constructor of this class should only receive two arguments (rather than 3 in the previous question). The new class inherits all the methods from your matrix class implemented in question 2. In addition, the SquaredMatrix should include the following method:

a. __pow__(self, n) – multiple the matrix with itself n times.
   The method returns a list of lists of the items (similar as it appears in the *mat* attribute).
   (m ** 3 = m.mat * m.mat * m.mat)

In this question, you are required to check the validity of the parameters used to create an instance. Moreover, in this question <u>you are also required to check the validity of</u> the __*pow*__ parameter as well.

Explain how you implemented this method. Try to justify why you decided to implement it like this. You are allowed to add method to the Matrix class as well. Write your answer in the answers file.

Good Luck!