

Exercise 3

Submission date: 30.4.23 23:00

In the following exercise you are allowed to use only basic python functions and the *deepcopy* package. Other packages are not allowed. Do not 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
```

(If you are working alone, your first row should be *# id1*)

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.

Part A - Classes:

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.
$$R_{row_index1} + (scalar) \cdot R_{row_index2} \rightarrow R_{row_index1}$$
- d. Matrix multiplication – multiple the current matrix with another matrix (another *Matrix* instance and not list of numbers).

* All these methods change the value of the *mat* attribute (see explanation below) and do not return any value.

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 (int or float).

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 $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$.

In this question you will need to check whether the parameters used to create an instance are valid (i.e. the parameters of the constructor.). 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 (*raise* command. Do not print the 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) For each method, **write in the answer file** the possible errors that may occur. Do not implement it in the code file for the methods that are not the constructor.

Part B – Inheritance and Magic Operators:

In this question you will create another matrix class, this time called *SquaredMatrix*.

- 1) Write the constructor of the class *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 part A. In this question, you are required to check the validity of the parameters used to create an instance of *SquaredMatrix*.

- 2) Implement the method `__pow__` in the *SquaredMatrix*.

`__pow__(self, n)` – multiple the matrix with itself n times. The method returns an instance of a squared matrix (differently from the methods in part A). Make sure you are not changing the values of the *mat* attribute of the original matrix instance.

For example:

```
m_res = m ** 3
```

```
m_res.mat == m.matrix_mult(m.matrix_mult(m))
```

You are also required to check the validity of the `__pow__` parameter – **positive** integer.

- 3) **In the answer file** - explain how you plan to implement this method. Try to justify why you decided to implement it like this.

Good Luck!