Ex2 – Questions section – 316493758

Question 1:

a. Explain how you implemented each method. 2-3 sentences for each method.
   - switch_rows: I implemented it with simple switch between two lines on the matrix (by the given row indexes).
   - multiple_row: I implemented it with mapping each element in the row to its element multiply the given scalar.
   - add_row_to_row: I implemented it with two steps:
     - Save the second row multiple the scalar
     - Add it to the first row.
   - gauss_ranking: I implemented it with 4 steps:
     - Order it by the position of their first number (besides zero) for example, (1, 0 ,0) will come before (0,1,1).
     - Iterated every row(i) and for each row I iterate every row(j) the under her and subtract the rows that none of them will have the number (besides zero) in the first number in row i index.
     - Ordered again like in the first step (maybe some rows deleted)
     - D some cosmetic changes (turn -0.0 to 0 and stuff)

b. Explain your constructor and how you checked the parameters.

   My constructor does validations and then insert to his parameters the input.

   After it create mat by iterate row count and add columns count elements to each row.

   The parameters validations are first check if the row and column length is valid int and then if it bigger then 0. After that there is a validation if the items length fit the multiplication of the rows and columns count the if it and his properties are with the proper type.

c. Explain which methods you added and what each method does.
   - _validate_row_index(self, row_index): Validate row index input.
   - _cosmetic_changes(self): Do cosmetic changes to self.mat.
   - _get_first_index(self, row_index): Get first number (besides zero) in row at given index.
   - _order_mat_by_rank(self): Order mat by rank of row (index of the first number).
   - _do_gaussian_elimination(self): Do the actual gaussian elimination.
   - _get_multiple_row(self, row_index, number): Get row in the given index multiply by the given scalar.
   - _row_validations(row_count): Validate given rows count. (static function)
   - _col_validations(col_count): Validate given columns count. (static function)
   - _validate_scalar(n): Validate given scalar. (static function)
   - _is_digit(obj): Check if the given object is number (int or float). (static function)
   - _all_is_digits(lst): Check if the given list contains numbers (int or float) only. (static function)

- _init_validations(row_count, col_count, lst): Do all __init__ validations. (static function)
- _multiply_num_in_scalar(num, scalar): Multiply two numbers (for map). (static function)
- _add_numbers(num1, num2): Add two numbers (for map). (static function)
- _get_two_rows_adding(row1, row2): Get adding of two given rows. (static function)

Question 2:

Explain how you implemented this method (__pow__(self, n)).

* I validate n, checked that he is an int and bigger than 0.
* Then I saved the original mat.
* I iterate n time and every time I multiplied the current mat in the original mat.
* How one multiplication goes:

```
Create new matrix
- Transpose the original mat
- Iterate each row in current matrix:
        Create new row
        -Iterate every cell in row
                Create new cell
                sum the multiplication of cell and the cell in the same index in
                row at the transpose matrix.
                Add it to new cell

                Add new cell to new row

        Add new row to new mat
Current mat = new matrix
```