HOME ASSIGNMENT 2

DUE DATE: 14/5/20

In memory of John Horton Conway.

INTRODUCTION:

In this task, you need to build a class that can operate a session of "Conway's Game of Life". Before starting you should be familiar with Conway's Game of Life, we strongly recommend reading the Wikipedia page https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life.

For this work, you need to build a class that will inherit the properties of an interface named game_of_life_interface.py. You need to implement the different methods of the interface according to the restrictions.

CONSTRUCTOR INITIAL VARIABLES:

Your class could get up to four values during initialization: the size of the board, starting position, rle and rules.

- ***size_of_board*** is an integer bigger than 9 and smaller than 1000, i.e., an int in $\{10, \dots, 999\}$.
- ***board_start_mode*** is an integer. Only the integers defined in the section "Board start mode" should work, if a different integer is provided, ***board_start_mode***=1 should be chosen.
- ***rle (Run length encoding)*** *is a string that depicts the initial pattern to be placed on the board. if* ***rle*** *is not an empty string, the* ***board_start_mode*** *should be ignored. More information in the following sections.*
- ***pattern_position*** *is a pair (x, y). Represents the upper left corner position of the initial pattern on the board. This argument is only used for* ***rle*** *input, e.g., when* ***rle*** *is not an empty string.*
- ***rules*** is a string that holds the rules of the game. for more information/rules encodings read: https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life#Variations

PUBLIC METHODS TO IMPLEMENT

All the methods are also described in the interface file.

- ***update()***: This method updates the board game by the rules of the game. for **one** step.

- ***return_board()***: This method returns a list that encodes the board: The board is a two-dimensional list that every cell denotes if the cell is “dead” or “alive”. Dead will be denoted with a 0 while alive will be denoted with a 255.

- ***save_board_to_file****(file_name)*: This method saves the current state of the game to a file. file_name is a string that denotes the file name. The file should be a .png file, for example, file_name=’1000.png’. You should use Matplotlib for this.

- ***display_board()***: This method displays the current state of the game to the screen. You can use Matplotlib for this.

- **transform_rle_to_matrix**(rle): The method receive the rle as string and returns a two-dimensional list (list of lists) that is in the bounding dimensions of the pattern and has the number 255 in every cell that is a “live” cell of the pattern. The bounding dimensions of the pattern should be determined by the number and length of the rows.

BOARD START MODE:

see examples at the end of the document for more details.

- For **board_start_mode** = 1, the board should be arranged randomly as follows. Each cell is `alive’ with probability ½ (and `dead’ with the complement probability, that is, ½ as well). you can use the np.random module.

- For **board_start_mode** = 2, the board should be arranged randomly as follows. Each cell is `alive’ with probability 0.8 (and `dead’ with the complement probability, that is, 0.2). you can use the np.random module.

- For **board_start_mode** = 3, the board should be arranged randomly as follows. Each cell is `alive’ with probability 0.2 (and `dead’ with the complement probability, that is, 0.8). you can use the np.random module.

- For **board_start_mode** = 4, the board should start empty with a Gosper Glider Gun in top left cell at (10, 10).

RLE:

A text encoding of a pattern. *Most of the patterns can be found here* *https://www.conwaylife.com/wiki/Category:Patterns*

The text is encoded as follows:

‘b’	Dead cell
‘o’	Live cell
‘$’	End of line
‘2’ – ‘99’	Any number between 2-99
‘!’	End of pattern

The number **x** can appear before ‘b’ / ‘o’ / ‘$’ indicating **x** dead cells, **x** live cells or **x** new lines, respectively.

You can assume that all of the lines are with equal length, notice that the last line should be filled with dead cells if its encoding is shorter than the other lines.

Examples of RLE code:

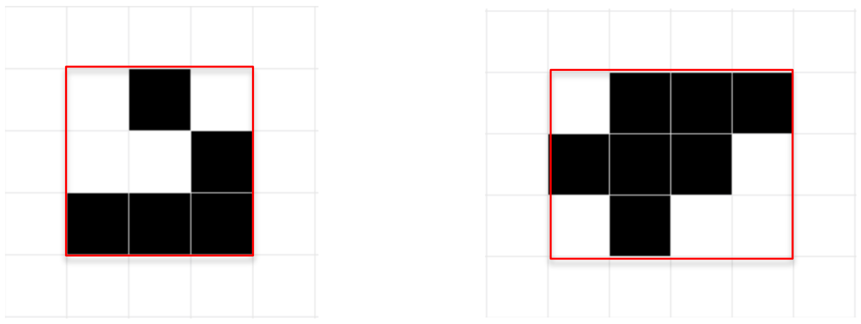


FIG1 IN THESE FIGURES WE HAVE TWO EXAMPLES: OF A GLIDER (LEFT) WITH THE RLE: “bob$2bo$3o!”, AND A C-HEPTOMINO (RIGHT) WITH RLE: “b3o$3ob$bo!”

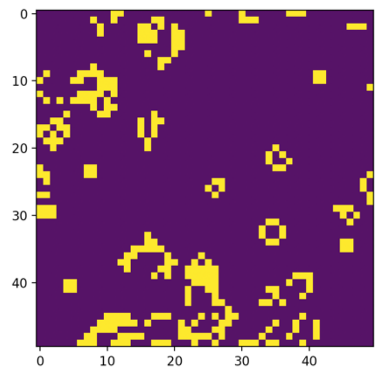
REMARKS:

- **An interface was uploaded to Moodle**. You should be familiar with it and use it while implementing your solution. You can start coding with our template file that was uploaded to Moodle.
- Your code will be uploaded using an import statement into our auto-grading code, therefore you should not run any code outside of the class member that you will create. For debugging purposes **you should use the statement if __name__ == ‘__main__’**. You can start coding with our template file that was uploaded to Moodle.
- **The board game is always a square and is a periodical space**. That is, a cell on the borders has also eight neighbors.
- If needed you can use only the following libraries:
    - **matplotlib**: https://matplotlib.org/3.2.1/api/_as_gen/matplotlib.pyplot.html
    - **numpy**: https://docs.scipy.org/doc/numpy-1.14.1/reference/routines.random.html

      https://docs.scipy.org/doc/numpy-1.14.1/index.html
- Document your code - for yourself, and for good practice. Your comments should be written in **English only.**
- You should submit a **single .py** file through the Moodle website with the name of YOUR_ID_NUMBER.py were YOUR_ID_NUMBER should be your id number.
- The run time is limited to 20 minutes.
- Avoid copying!

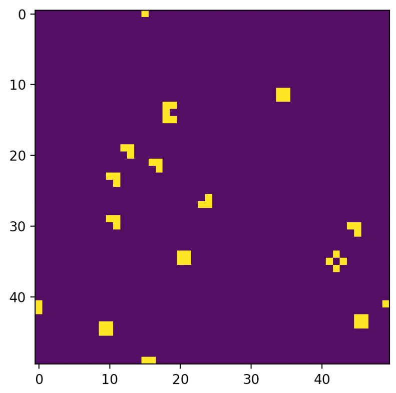
SOME EXAMPLES:

1. running the code for 100 iterations with a ***size_of_board***=*50,* **board_start_mode**=*1 and rules=B36/S23 will give us:*



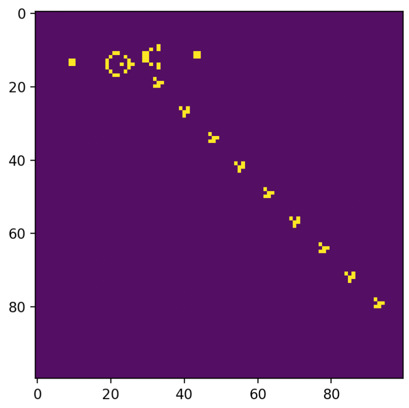
(run time was around 0.8 sec) (this example is based on a random seed, therefore don't expect it to be equal to your result)

2. running the code for 20 iterations with a ***size_of_board***=*50,* **board_start_mode**=*2 and rules=B45/S23 will give us:*



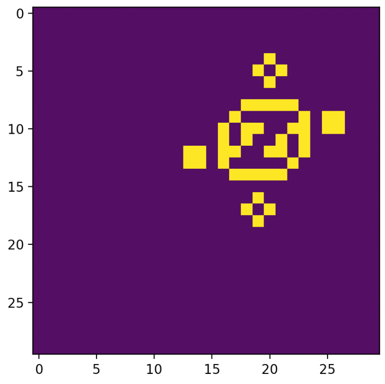
(run time was around 0.2 sec) (this example is based on a random seed, therefore don't expect it to be equal to your result)

3. running the code for 270 iterations with a ***size_of_board***=*100,* **board_start_mode**=*4 and* ***rules***=*B3/S23 will give us:*



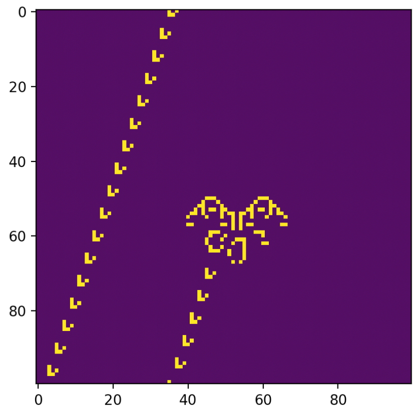
(run time was around 12 sec)

4. running the code for 20 iterations with a ***size_of_board***=*30,* ***rules***=*B3/S23,* ***rle***=*"7bo6b$6bobo5b$7bo6b2$5b5o4b$4bo5bob2o$3bob2o3bob2o$3bobo2bobo3b$2obo3b2obo3b$2obo5bo4b$4b5o5b2$6bo7b$5bobo6b$6bo!",* ***pattern_position***=*(4,13). will give us:*



(run time was around 0.4 sec)

5. running the code for 180 iterations with a ***size_of_board***=*100,* ***rules***=*B3/S23,* ***rle***=*" 5b3o11b3o5b$4bo3bo9bo3bo4b$3b2o4bo7bo4b2o3b$2bobob2ob2o5b2ob2obobo2b$b2obo4bob2ob2obo4bob2ob$o4bo3bo2bobo2bo3bo4bo$12bobo12b$2o7b2obobob2o7b2o$12bobo12b$6b3o9b3o6b$6bo3bo9bo6b$6bobo4b3o11b$12bo2bo4b2o5b$15bo11b$11bo3bo11b$11bo3bo11b$15bo11b$12bobo!",* ***pattern_position***=*(40,40). will give us:*



(run time was around 8 sec)

REFERENCES:

- https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life
- http://www.conwaylife.com/wiki/Main_Page
- http://conwaylife.appspot.com/
- http://www.conwaylife.com/