Assignment 3

Overview

We will re-implement Conway's game of life and build upon the work we did for assignment two. This time we will use a library designed for vectorization to write an optimized simulation. We will compare its performance to our implementation for assignment two.

Assignment parts (100pts total)

- Your assignment will be run outside of a notebook! If you use a notebook to implement your assignment do not use it during the grading meeting. The instructor will run your script using Python via a command line (Linux shell). Failure to do this will result in a loss of all points.
- If necessary, modify your original implementation of Conway's Game of Life rules to completely decouple the simulation from the visualization. In assignment two you were to write a function which took at input a list of lists and return an updated list of lists representing the updated board state. Ensure that this function does not perform any visualization and rename this function to conway_assignment_two. (10pts)
- Write a function which calls conway_assignment_two. This function will contain a loop. Each iteration through this loop will represent a single time-step of our simulation. This function will take a parameter which causes it to save the board state for each time-step and return this board state history as a list. (15pts)
- A script which runs your simulation for a number of board sizes (starting at 1000x1000) and increasing in increments of 250x250 up to the largest size you can simulate for 100 time steps. This script will run both conway_assignment_two and conway_assignment_three functions. This script will output a graph of the time steps per millisecond for each board size you attempt for both the assignment three and assignment two implementations. (30pts)
- A script which runs your simulation using conway_assignment_three function for 500 time steps using a board of size 1000x1000 with a random initial board state where the cells are chosen to be alive with 50% probability. This script will plot the number of alive and dead cells (they should sum to the number of cells in your board). This plot will be dynamic and will update after each time step. This graph will be created and updated using the Matplotlib library. (30pts)
- A script that runs your simulation using the conway_assignment_three function. It will take command-line parameters specifying the number of time steps, the size of the board, and the starting configuration (blinker, glider, and random). This ought to be the function you use to

test your code. There will be an additional command-line parameter which animates the board states. (15pts)

Other conditions

- Use only Matplotlib's animation function or the plot function to show your simulation results.
- All options controlled via command-line parameters. I suggest the argparse library.
- Your board will wrap-around. This means that the last column is next to the first column and the first row is next to the last row.
- Your board will be represented as a numpy array. If need to use another format, please clear it with the instructor.

Academic dishonesty:

Academic dishonesty will not be tolerated. All materials submitted will be the work of your group and nobody else. If academic dishonesty is discovered, you will receive zero points for that portion of the project and you will be referred to the department where there may be additional repercussions.