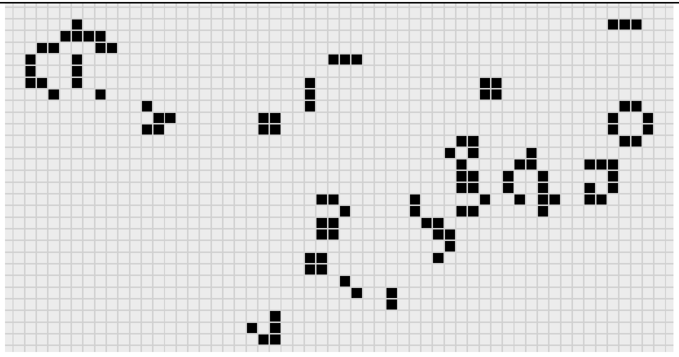
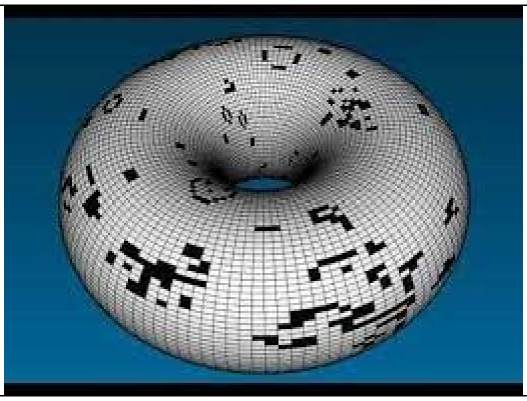
DSC 430: Python Programming

Assignment 0902: Game of Life

Read the Wikipedia article on **Conway's Game of Life**. We will implement this simulation on a **torus** because it will make the code easier and we won't need to deal with boundaries. This means that cells on the top are adjacent to cells on the bottom and the same is true for the left and right sides.

Write a function called **conway(s, p)** that generates a board, which is a square two dimensional NumPy array of size **s**. To simplify things, assume the board is a square, and s is length of the side. The board should be randomly populated with probability **p**. If p is set to 0, all cells shouldbe 0 (dead). If p is set to 1, all cells should be 1 (alive). Start with p=0.1; about 10 percent of cells should be 1.

Write a function **advance(b,t)** that accepts a Conway board and advances it **t time steps** according to the rules:

- Any live cell (marked as 1) with fewer than two live neighbors dies, as if by underpopulation.
- Any live cell (marked as 1) with two or three live neighbors lives on to the next generation.
- Any live cell (marked as 1) with more than three live neighbors dies, as if by overpopulation.
- Any dead cell (marked as 0) with exactly three live neighbors becomes a live cell, as if by reproduction.

Write a **function** to pleasantly **display the board**.

[3/1/2023] ADDITIONAL REQUIREMENTS:

1. When you "flip" a square in the middle of an iteration, don't let the flipped state impact its neighbors during that iteration. The neighbors should be impacted by the original state for that iteration. To ensure that, you must use two boards – the current one you are traversing over, and the new one which you store the new states of the cells. Think about how you can do that.

2. Write a function **simulate_gameoflife(s, p, n)**. This function is the top-level call to do the simulation. Create a board using **s** and **p**, and do a maximum of **n** iterations.

    - The simulation must terminate when it reaches the specified maximum number of iterations, or if the board configuration didn't change any cell at all at the end of an iteration (to prevent an infinite loop).

    - In the function, display the (current) board at the end of each iteration.

3. Show the result of a simulation with the maximum 10 iterations. Display should be something like this (using the board of 10 x 10):

```
Iteration 0:
_ * * _ _ _ _ * _ _
_ _ _ _ _ * _ _ _ _
_ _ * _ _ _ _ _ _ _
* * * _ _ _ _ _ _ _
_ * _ * * _ * * * *
* * _ * _ _ _ * * *
_ * _ * _ _ _ _ _ _
* _ _ _ _ _ * * _ _
_ * _ _ _ * _ _ _ _
_ _ _ _ _ * * _ * *

Iteration 1:
_ _ _ _ _ * _ * * _
_ * * _ _ _ _ _ _ _
_ _ * _ _ _ _ _ _ _
* _ _ _ _ _ _ * * *
_ _ _ * * _ * _ _ _
_ * _ * _ _ * _ _ _
_ * _ _ _ _ * _ _ _
* * * _ _ _ * _ _ _
* _ _ _ _ * _ _ * *
* * * _ _ * * * * _

Iteration 2:
* _ _ _ _ * _ _ * *
_ * * _ _ _ _ _ _ _
* _ * _ _ _ _ _ * *
_ _ _ * _ _ _ * * *
* _ * * * * * _ * *
_ _ _ * * _ * * _ _
_ _ _ _ _ * * * _ _
_ _ * _ _ * * * _ _
_ _ _ _ _ * _ _ * _
* * _ _ * * _ _ _ _
```

Ensure to provide a sufficient amount of documentation/comments in the code. Also write answers to these in a comment section at the end of the code file:

- your logic on how you updated the board.
- values for s and p which you found that created an interesting colony of cells.
- Verify the correctness of your code by checking at least 3 cells (from one iteration to the next iteration)

Submission: Submit the source file (.ipynb) and the exported html file (.html) to the D2L. Do not zip or archive the file. Your code must include comments at the top including your name, assignment number, and the honor statement, “I have not given or received any unauthorized assistance on this assignment.” Also, each function must include a docstring.