Class Project

Objective

Combine all you have learned about Python thus far (except object-oriented design) to complete a functional program.

Due

11:59pm (midnight - 1) Tuesday Feb 14th.

Instructions

For this project we are going to program Conway's Game of Life. Please observe the following:

- Get started right away! Please do not wait until a few days before the project is due to get going.
- Do not look for source code on the web (you're likely to find some—honor system applies). That defeats your ability to learn. Use only the book, online Python documentation, the instructor or other classmates for help. Ask as many questions as you want on Slack.
- Use the newly created **#project** channel on slack to communicate about the project.
- *Do not share your project code on Slack.*
- You may work in groups of 2 or 3 to complete the project, but please do not share code electronically with each other. Do not cut-and-paste each other's code (again honor system). You must write your own code, but you can decide as a group how to implement the project.
- You must have your program reviewed by one other student in the class. Identify who your reviewer is by **Feb 7th** and plan to have your code to them no later than **Feb 12th** to give them time to look it over and for you to make corrections if need be. *You may send them your code via a private Slack message*. Once you have your reviewer please let Dr. Ficklin know. Each student must only review one other student's program, and group members cannot review other group member's code. If you have problems finding a reviewer let Dr. Ficklin know.

The Game of Life uses a 2-dimensional grid of cells that can either be in a state of "on" or "off". The program simulates a living system in which cells can interact with their immediate neighbors. At the beginning of the simulation, some of the cells are on and some are off, and as time progresses the cells will change states depending on the state of their neighbors at each time point. For more in-depth information please see the Wikipedia page here: https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life. Your job is to create a Python program that implements a simple game of life. Observe the following when writing your code:

1. Your script must be named **game_of_life.py**
2. Store the 2D grid using Python lists or a dictionary. The grid must be 30 cells high (rows) and 80 cells long (columns).
3. The program must allow the user to specify on the command-line the number of "ticks" (or time points) to run, and the set of cell indexes which should be "on" when the program starts.
    a. A cell index must be of the form **[row]:[col]**, where **[row]** is the row number and **[col]** is the column of the cell that should be turned on. The two are separated by a colon.

    For example, if you ran your script as:

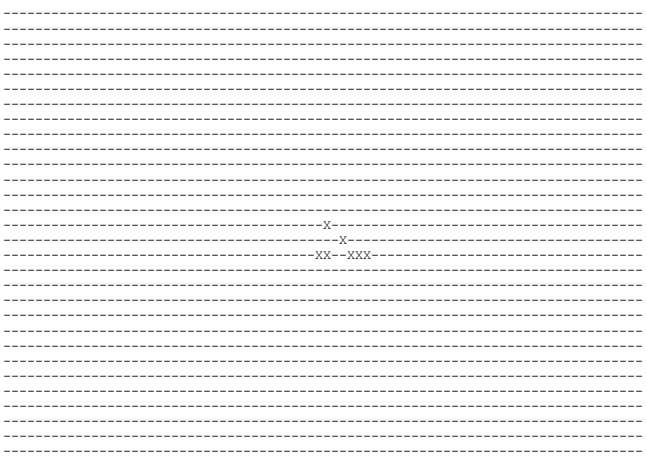
    ```
    python game_of_life.py 40 14:40 15:42
    ```

    then the program should expect to run for 40 ticks and start with cells (14,40) and (15,42) turned on. The user should be able to provide as many cell indexes as desired.
4. The numbering of rows and columns starts with 1 (not 0). That means the cell in the top-left has the index 1:1.

5. The user should be able to specify any number of cell indexes to the program.
6. The program should execute successfully with the following arguments:

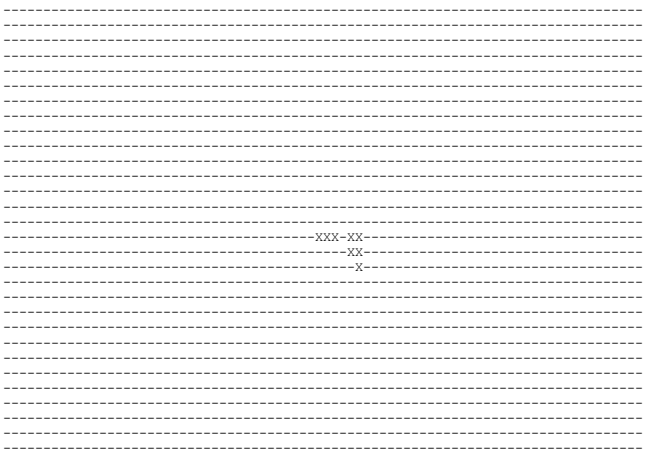
```
python game_of_life.py 50 14:40 15:42 16:39 16:40 16:43 16:44 16:45
```

7. At each step the program must print the contents of the grid to STDOUT. Each cell of every row is printed to the screen as either a dash character, **-**, or **X** character with a dash meaning the cell is "off" and an **X** meaning the cell is "on". Each row of the grid must be terminated with a carriage return, "\n". For example, the initial time point should look like the following:

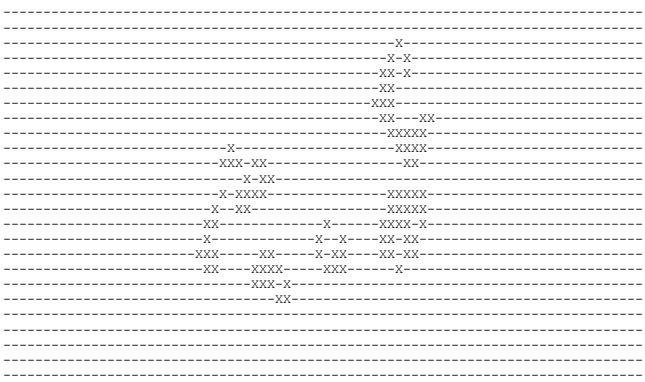


8. The program should follow these rules at each time point:
    a. Any "on" cell with fewer than two live neighbors is turned "off".
    b. Any "on" cell with two or three "on" neighbors remains "on".
    c. Any "on" cell with more than three "on" neighbors is turned "off".
    d. Any "off" cell with exactly three live neighbors is turned "on".

For example, after the first time point the printed output should look like the following:



The program should terminate after 50 ticks with the following pattern:



9. The program must use Sphinx Python docstring format for commenting and must include:
    a. A program header with the list of authors (group members), the name of your reviewer, the grade your reviewer provided, and a program synopsis.
    b. Each function must be fully commented with each functional parameter described using the **:param** and **:type** keywords.
    c. Every declared variable must have a comment describing what it stores.
10. Your program must have at least one function and should not use global variables.
11. When you review another student's code use the evaluation checklist below to provide a grade. If you provide fewer points than what is listed, you must indicate to the other student what was wrong so that they can correct it.

Evaluation Checklist (100 points possible)

1. The program:
    a. accepts the following command-line **(10 points).**

    ```
    python game_of_life.py 50 14:40 15:42 16:39 16:40 16:43 16:44 16:45
    ```

    b. executes to completion without presenting an error **(10 points).**
    c. terminates with the pattern shown in requirements #8 above **(15 points).**
2. The program uses the following:
    a. command-line arguments **(5 points).**
    b. a while loop **(5 points).**
    c. if statements **(5 points).**
    d. lists or dictionaries **(5 points).**
3. The program does not use global variables and has at least one function **(5 points).**
4. The program follows Sphinx docstring style documentation
    a. The program has a header **(5 points).**
    b. Each function has documentation for each parameter and its type is described **(5 points).**
    c. Each declared variable has a comment **(5 points).**
5. The printed grid is 30 x 80 with dashes for "off" cells and X for "on" cells **(5 points).**
6. The program runs for as many "ticks" as indicated in the command-line (e.g. 50) **(5 points).**
7. The program prints each time point (tick) to the screen following the rules above **(5 points).**
8. The program was reviewed by another student **(10 points).**