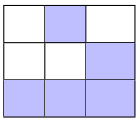
Lab 4: Conway's Game of Life

Due date: Sunday March 3, 11:59pm

The task is to construct a working Game of life (for a overview check out:
http://en.wikipedia.org/wiki/Conway's_Game_of_Life
(there are several implementations of this game on the internet, do not use those because they all interface with some modules that we did not talk about.)

Overview pseudocode

1. Create a 2D array that represents the playing field or map (or world) for the game
2. Create a start configuration and place it on the map, for example a glider:



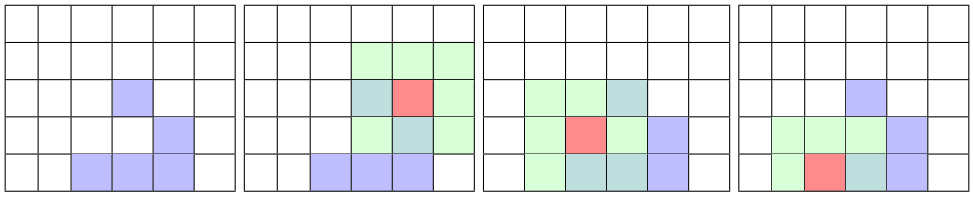
   The dark color means alive, white is dead, you can either use a fancy character (for example
   `XX = unichr(0x2588) # this prints a black rectangle (=asci(219))`
   but you can also use a 'X' for 'alive' and a space for 'dead'.
3. START
4. Evolve every cell in the map according to these rules:
   (a) if cell is alive and has 2 or 3 neighbors $\Rightarrow$ alive
   (b) if cell is dead and has 3 neighbors $\Rightarrow$ alive
   (c) otherwise cell $\Rightarrow$ dead
5. Display the map and use a sleep function so that the map stays up for x seconds
6. Goto back to START

The MAP

Displaying the 2D list is not difficult, look at the "".join example in class (text processing); but to make this a good show/pseudomovie you will need to clear the screen before you display the map. On a mac/linux this can be done by using `os.system('clear')`
on windows use
`os.system('cls')`

The EVOLVE function

for every cell you will need to interrogate the neighbors whether they are alive or not Lets assume we have a glider at the boundary and we evaluate the state for next timeclick and we evaluate in the second figure from left for the red cell, then we need to count the neighbors (green), this will result in 2 'alive' neighbors, thus the cell will stay dead, whereas the third example shows a currently dead cell will become alive (set to 'X') because it has exactly 3 neighbors. Last figure: Be careful at the boundary, you are not allowed to query cells that are not in your 2D map (because the index will be out of range); this cell dies because it has only 1 neighbor.



Deliverable

- Write the program and at least one other start configuration.
- upload the code to CANVAS, it should be executable as `python conway.py`

if you want to use optional arguments tell Tara how to call those, for example my test program can be called like this: `python conway.py`
`python conway.py glider glidergun simple rand glider glider`
This last call results in a start configuration that looks like this