2D Array and Matrix Application: Game of Life

Conway's Game of Life Resources

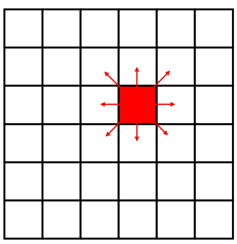
- A short video on Game of Life
  - https://www.youtube.com/watch?v=CgOcEZinQ2I
- Other applications of the Game
  - Two plain text versions from conwaylife.com: http://www.conwaylife.com/wiki/Plaintext
  - Game of Life Clock:
    - https://www.youtube.com/watch?v=3NDAZ5g4EuU
    - The original challenge: https://codegolf.stackexchange.com/questions/88783/build-a-digital-clock-in-conways-game-of-life

HW 8, Lab -- "Life"

John Conway

Grid World

red cells are alive



white cells are empty

Evolutionary rules

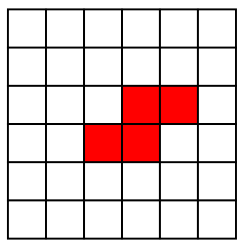
- Everything depends on a cell's eight neighbors
- Exactly 3 neighbors give birth to a new, live cell!
- Exactly 2 or 3 neighbors keep an existing cell alive
- Any other number of neighbors kill the central cell (or keep it dead)

3

Problem 1 -- Life

Grid World

red cells are alive



white cells are empty

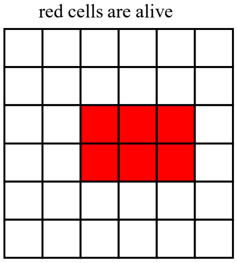
Evolutionary rules

- Everything depends on a cell's eight neighbors
- Exactly 3 neighbors give birth to a new, live cell!
- Exactly 2 or 3 neighbors keep an existing cell alive
- Any other number of neighbors kill the central cell (or keep it dead)

Problem 1 -- Life

Grid World

red cells are alive



white cells are empty

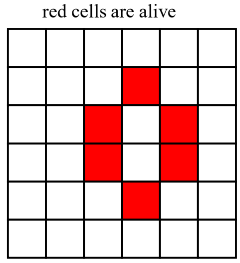
Evolutionary rules

- Everything depends on a cell's eight neighbors
- Exactly 3 neighbors give birth to a new, live cell!
- Exactly 2 or 3 neighbors keep an existing cell alive
- Any other number of neighbors kill the central cell (or keep it dead)

Problem 1 -- Life

Grid World

red cells are alive



white cells are empty

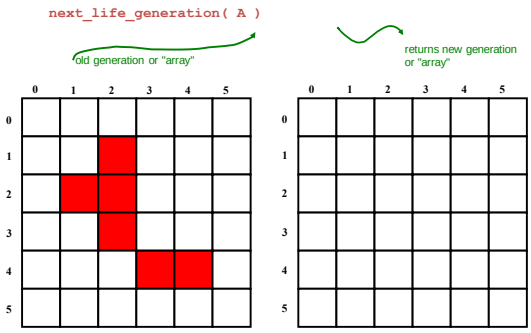
Evolutionary rules

- Everything depends on a cell's eight neighbors
- Exactly 3 neighbors give birth to a new, live cell!
- Exactly 2 or 3 neighbors keep an existing cell alive
- Any other number of neighbors kill the central cell (or keep it dead)

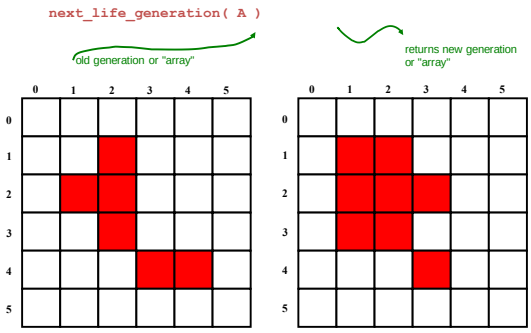
Keep going!

life out there...

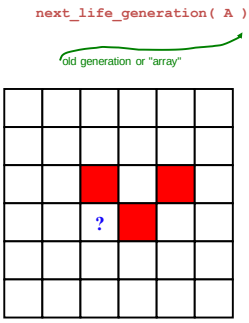
Problem 1 -- Creating Life



Problem 1 -- Creating Life



Problem 1 -- Details



For each generation…

- 0 represents an empty cell
- 1 represents a living cell
- outermost edge should ***always*** be left empty (even if there are 3 neighbors)
- compute *all* cells based on their *previous* neighbors before updating any of them

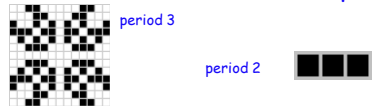
http://www.math.com/students/wonders/life/life.html

life out there...

Problem 1 – to ∞ and beyond!

- Are there stable life configurations? "rocks"

- Are there oscillating life configurations? "plants"



- Are there self-propagating life configurations? "animals"

Let us work out a couple of problems together.

1. Find the neighbors for some special cells in Conway's Game of Life, return them as a list

a) Upper left corner: The cell index is [0][0]. It doesn't have any upper or left neighbors, only the ones on right or below.

0,0	0,1
1,0	1,1

Python code:

```
neighbors = [m[1][0]] + [m[0][1]] + [m[1][1]]
```

b) Your work: Finding neighbors for

- **upper right corner**,
- **lower left corner**, and
- **lower right corner**

2. Place integers 1..9 in a 3x3 matrix, no repetition is allowed, similar to Sudoku. Some cells may have been initially filled correctly. The unfilled cells are marked by -1.

```
[[1, -1, -1],
 [-1, -1, 4],
 [5, 6, -1]]
```

results in

```
[[1, 2, 3],
 [7, 8, 4],
 [5, 6, 9]]
```

```
[[-1, -1, -1],
 [-1, -1, -1],
 [-1, -1, -1]]
```

results in

```
[[1, 2, 3],
 [4, 5, 6],
 [7, 8, 9]]
```

Python list tools needed remove() and pop():

```
n = [i for i in range(1, 10)]
for k in range(len(n)):
   v = randint(1,10)
   if v in n:
      n.remove(v)

m = []
for k in range(len(n)):
   m = m + [n.pop()]
#  m = m + [n.pop(0)]
```

Try *list_remove_pop.py*