Game of Life Description

February 28, 2019

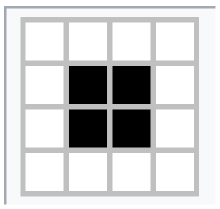
1 Question 1: Program for the Game Of Life

Imagine every point in a grid is a life. For a point A in the grid, there are 8 points around it. If there is a blackpoint, it means that the point is alive (otherwise it's dead). If 3 of the 8 points are alive, then the number of point A is 3 ("3" represents the number of living points within 8 surrounding nodes).

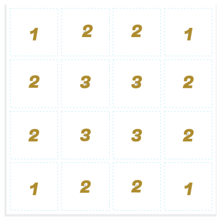
For example, Figure 1(a) is a grid with four rows and four columns. The corresponding number grid for Figure 1(a) is Figure 1(b).

The living nodes in a grid changes generation by generation according to the number of neighbors that are **alive**, as follows:

1. The neighbor of a given node are the eight nodes that touch it vertically, horizontally, or diagonally.
2. If a node is alive but either has no neighboring nodes alive or only one alive, then in the next generation the node dies of loneliness.
3. If a node is alive and has four or more neighboring nodes also alive, then in the next generation the node dies of overcrowding.
4. A living node with **either two or three** living neighbors remains alive in the next generation.



(a) status for living and dead nodes



1	2	2	1
2	3	3	2
2	3	3	2
1	2	2	1

(b) living number of current status

Figure 1: Living number for every point in a grid

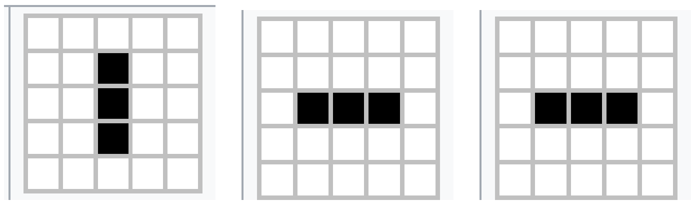

(a) initial status (b) first generation (c) second generation

Figure 2: Changes of generation

5. If a node is dead, then in the next generation it will become alive if it has **exactly three** neighboring nodes, no more, no fewer, that are already alive. All other dead nodes remain dead in the next generation.
6. All births and deaths take place at exactly the same time, so that dying nodes can help to give birth to another, but cannot prevent the death of others by reducing overcrowding; nor can nodes being born either preserve or kill nodes living in the previous generation.

An example is shown in Figure 2. Your goal is to write a python program that will show how an initial configuration will change from generation to generation.

Important——Sample input and Output

Your program should take input as:

5
3
5
0
2
1
2
2
2

The explanation for these input is:

5 — it's a 5*5 grid (the grid size)
3 — initial living node number (how much living node in an initial status)
5 — generation number (how many generations have gone through)
0 — The abscissa of the first living point
2 — The ordinate of the first living point
1 — The abscissa of the second living point
2 — The ordinate of the second living point

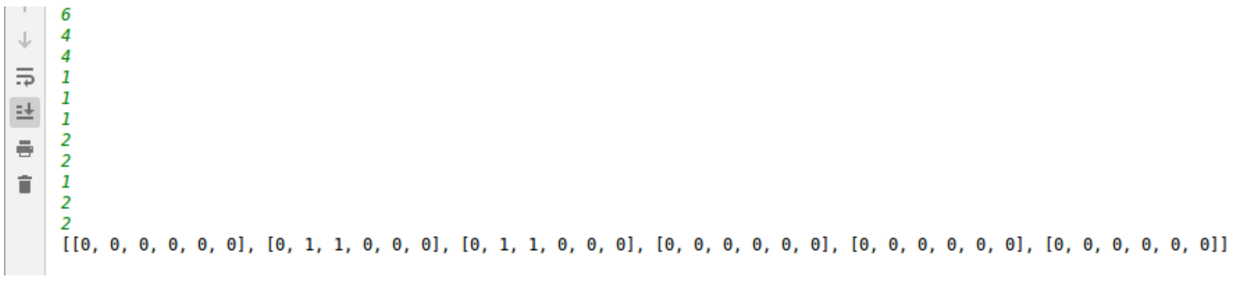
2 — The abscissa of the third living point
2 — The ordinate of the third living point

For the output, you have to output 1 if it's a living node, or 0 if it's dead. Your program should output **every line** of final generation as a **list**. For the above example, the list for the final generation should be:

[[0, 0, 0, 0, 0], [0, 1, 1, 1, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0]]

The output is a list, elements are lists as well!

To make sure you have understood the rule, another running process is shown in below



```
6
4
4
1
1
1
2
2
1
2
2
[[0, 0, 0, 0, 0, 0], [0, 1, 1, 0, 0, 0], [0, 1, 1, 0, 0, 0], [0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0]]
```

Hint

In order to put a list as an element into another list, you can use *list.append()* function.