Challenge #2: Conway's Game of Life

Master in Informatics and Computing Engineering
Programming Fundamentals
Instance: 2019/2020

1. Introduction

The challenge is extra work for more advanced Python programmers that easily solve the regular exercises.

*The **goal** of this challenge is for the students to implement a personal version of a zero-person game using the matplotlib library (see Plotting data with matplotlib)*

***Reference**: Wikipedia, the free encyclopedia at Conway's Game of Life page.*

***Advice:** Do not see any solution before trying.*

2. Introduction

The game is a zero-player game, meaning that its evolution is determined by its initial state, requiring no further input. One interacts with the Game of Life by creating an initial configuration and observing how it evolves, or, for advanced players, by creating patterns with particular properties.

3. Rules of the game

The universe of the *Game of Life* is a two-dimensional orthogonal grid of square *cells*, each of which is in one of two possible states, *alive* or *dead*, (or *populated* and *unpopulated*, respectively).

Every cell interacts with its eight *neighbours*, which are the cells that are horizontally, vertically, or diagonally adjacent. At each step in time, the following transitions occur:

1. Any live cell with fewer than two live neighbors dies, as if by *underpopulation*.
2. Any live cell with two or three live neighbors lives on to the *next generation*.
3. Any live cell with more than three live neighbors dies, as if by *overpopulation*.
4. Any dead cell with exactly three live neighbors becomes a live cell, as if by *reproduction*.

The initial pattern constitutes the *seed* of the system. The first generation is created by applying the above rules simultaneously to every cell in the seed; births and deaths occur simultaneously, and the discrete moment at which this happens is sometimes called a *tick*. Each generation is a pure function of the preceding one. The rules continue to be applied repeatedly to create further generations.

4. Let's do it in Python

Choose a suitable algorithm and code it in Python. You may want to use the `matplotlib` library to do the graphics.

Hint: Start small, get something small working and then add to it.

The end.

FPRO, 2019/20