Minesweeper, Part1: Burrying mines

SHAPE Summer 19, Introductory Computer Science

July 1, 2019

Write all your code in a new file `minesweeper_firstname1_firstname2.py`.

Model View Controller Architecture

MVC (Model-View-Controller) is a software design architecture [1] that separates the data, the way data is displayed to the user, and the way in which the user can manipulate the data. This division of labor has many advantages, including better reusability, clarity and efficiency. Therefore, this project will follow MVC design architecture. In part 1, we will focus on data representation.

Representing the Game Board

You already started thinking about what kind of data should be represented in the grame. In this sheet, we will focus on the game board only, which includes the positioning of mines.

- Step 1: write a function `def create_board(width, height)` that returns an empty game board. The board should be represented as a list of rows. Each row should initally just contain the value `None` for each field. For example, a 3×3 board should be represented as

```
[[None, None, None],
 [None, None, None],
 [None, None, None]]
```

- Step 2: write a function `def bury_mines(gameboard, n)` that takes as a game board in the format described above and places n mines in a random position on the board. The function does not have to return the game board because you are modifying the board object itself. Take a look at the random module (in the Python documentation). Make sure there are actually n distinct mines, so you cannot place two mines in the same position. To place a mine, replace the None entry in the corresponding position with another object (for example the string "*").

Counting Mines

Write a function `def get_mine_count(gameboard, x,y)` that returns the number of mines adjacent to this x and y coordinate (in either direction or diagonal, so there are up to 8 possible cells that can have a mine). Pay attention to the boundary of the game board. Do not count the cell x,y itself.

Printing the Game Board

Later we will write a graphical user interface for the game. For now it is useful to be able to display the board on the console.

[1] `https://en.wikipedia.org/wiki/Model-view-controller`

- Write a function `def print_mines(gameboard)` that displays the board and the position of all mines. For example, for a 6x6 board it might look something like this

```
. . . . . .
. * . . * .
. . * . * .
. * . * . .
. * . . * .
. . . . . .
```

  `*` represents a mine and `.` represents an empty cell.

- Then write a function `def print_board(gameboard)` that displays the board, including the position of mines, but instead of `.` to represent empty cells print out the integer number of mines adjacent to that cell.