MagmaLabs Mini-Test

Goal

The main goal of this mini-test is to get a sense of how the applicant can leverage previous experience to learn new technologies and build applications using Ruby with them.

Game of Life

Project description

The Game of Life, also known simply as Life, is a cellular automaton devised by the British mathematician John Horton Conway in 1970. 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.

Functional Requirements

The universe of the Game of Life is an infinite two-dimensional orthogonal grid of square cells, each of which is in one of two possible states, alive or dead.

Every cell interacts with its eight neighbors, which are the cells that are directly horizontally, vertically, or diagonally adjacent.

At each step in time, the following transitions occur:

- Any living cell with fewer than two live neighbours dies, as if caused by underpopulation.
- Any living cell with more than three live neighbours dies, as if by overcrowding.
- Any living cell with two or three live neighbours lives on to the next generation.
- Any dead cell with exactly three live neighbours becomes a live cell.

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 happen simultaneously, and the discrete moment at which this happens is sometimes called a tick (in other words, each generation is a pure function of the one before).

The rules continue to be applied repeatedly to create further generations.

The game should be run in command line and print the data to standard output.

Non-Functional Requirements

- Performance and scalability
  - Must be able to run in read only filesystems(12 factor)
- Quality
  - Code style defined by the community
  - Unit Tests
  - Rubycritic score above 90
- Runtime
  - Can be executed via command line and print the output to the console
- Source code
  - OOP
  - Small methods
  - Single responsibility principle

Technologies

- Ruby Programming Language(preferred)
- MiniTest/Rspec testing frameworks(preferred)
- Command line

Deliverables

- Source code hosted in GitHub
  - Must include a README about how to setup the project
  - Git Commits as atomics as possible
- Do not invest more than 10 hours on the mini-test, and deliver the app before 7 natural days.

How review is done

We will review the code taking in consideration the following items:

1. Object-oriented
2. File structure
3. Testing
4. Duplicated code
5. Separation of concerns

6. Language naming conventions
7. Good variables and methods naming
8. Clear solution
9. Use of advance features of the language
10. Abstraction
11. Use of Git, atomic and well defined commits

References

- http://tutorials.jumpstartlab.com/projects/ruby_in_100_minutes.html
- https://www.coderetreat.org/pages/facilitating/gol/
- https://github.com/rbenv/rbenv
- https://github.com/rbenv/ruby-build#readme
- https://github.com/whitesmith/rubycritic
- https://github.com/rubocop-hq/ruby-style-guide
- http://www.betterspecs.org/