CPS506 - Comparative Programming Languages - Fall 2017

Assignment 2 - Elixir

This assignment is a relatively simple program to capture various aspects of programming languages. This version is in Elixir

The application is a simple Snakes and Ladders game described here. In addition to the specifications there, the following Elixir-specific parameters will apply:

1. Do `mix new assign2` to create your package.
2. Your `Assign2` module must have a `readFrom/1` function that accepts a string and returns a "game object". Since there are no objects in Elixir, "game object" refers to data of some sort - possily a tuple, map or array - that you choose to represent your game.
3. Your `Assign2` module must have a `print/1` function that formats the "game object" as a string.
4. When run via `iex`, the above methods would return results.
5. While it should be possible to also run the code as an application, the actual way to do it appears to be very obscure. Therefore, it will not be part of the evaluation for this assignment.
6. Put your ownership information (see the assignment page) in the `assign2/README.md` file.
7. Your code will be tested via `iex -S mix`.
8. The marker should be able to run your program by entering the following code:

```
$ iex -S mix
Interactive Elixir (1.4.1) - press Ctrl+C to exit (type h() ENTER for help)
iex(1)> Assign2.print(Assign2.readFrom("board 3 4
players 2
dice 1
turns 5"))
```

The Elixir style guide is a good source of guidance on code structuring.

You should do your assignment in the your Git `CPS506` repository in a folder called `assign2` (created with the `mix` command above). Every time you have completed a part of the assignment, you should commit it to the repository. You shouldn't wait until everything is complete to do this, it's better to check in regularly. Remember to do `git status`, `git commit`, and `git commit` from somewhere within the repository periodically to make sure you're commiting all of your code. Also remember to **not** add binary files or other files that can be generated from the source. `mix` automatically creates a `.gitignore` file that excludes commonly created files that should be excluded; add to that file if you notice any undesirable files being staged for commiting to the repository. You can so a `git add .` as many times as you want, but you only have to do it once each time there are new files to be included in the repository. In a terminal/command window simply change to the working directory and check-in, for example:

```
cd gitlab/cps506
git commit -m "finished code and tests for snakes and ladders"
git push
```