CSC290. Project Code Commit

Each team member will submit one substantial commit, to show both their participation in the project, and to show the use of good coding practices. The commit should be decently sized, but not too large since each commit should be atomic. However, the commit should not be a simple typo fix or other trivial code.

Each submission will be graded on:

- whether the code is readable,
- whether the commit message is clear,
- whether the commit is consistent with the rest of the project.

Working Together

Although each team member submit a commit individually, your team need to work together to make sure that each team member is successful.

Each team member should be responsible for a large enough coding part of the game so that they have a nontrivial commit that they can submit.

Your team should also talk to each other about coding conventions that you will follow. For example, will you be using tabs or spaces? What naming strategy will you use for variables, functions, and methods? You should follow the style guidelines typical for the programming language that you chose, but we will also be looking for consistency among the team members.

Most importantly, your team should plan ahead and decide who/how to resolve code conflicts, especially if multiple team members are working on the same file at the same time. **Avoid the scenario where everyone is committing code at the same time right before the deadline, and creating conflicts.**

Team members must not use the `git push -f` feature to overwrite other students' commits. Any student who uses `git push -f` will receive a grade of 0 for the code commit. Instead, plan ahead and talk to your team members!

Commit Messages

Different organizations and projects will have different guidelines for writing commit messages. For our course, we will follow a commonly used set of guidelines described here: https://chris.beams.io/posts/git-commit/

1. Separate the commit message subject from body with a blank line.
2. Limit the commit message subject line to 50 characters.
3. Capitalize the commit message subject line.
4. Do not end the commit message subject line with a period.
5. Use the imperative mood in the commit message subject line: i.e. describe what the commit message does (in present tense) if you apply the commit.
6. Wrap the commit message body at 72 characters.
7. Use the commit message body to explain what and why vs. how.

It is difficult to change a commit message after you have committed the code. Instead, you should think about the commit message as you are writing code, or even before writing your code. Think about the changes you want to make, then make those changes, and make your commit.

If you want to become a git superuser, you can look up "interactive rebase" and the `git commit --amend` functionality.

Code Style

You should follow the style guidelines typical for the programming language that you chose. For Python, the PEP8 style guide is most commonly used https://www.python.org/dev/peps/pep-0008/ Here's a style guide for Java https://google.github.io/styleguide/javaguide.html

For your commit, we are looking for:

- names in the appropriate cases for your language, for example `CamelCase` for Java names and Python classes, and `pothole_case` for Python names.
- consistent use of spaces or tabs across your team
- consistent code width (no more than 80 characters)
- general consistency in the way code is commented across your entire team (e.g. use of docstrings)

Code Clarity

For clear code, we are looking for code that is easy to understand. You can make your code easier to understand by:

- naming your variables well
- using an appropriate amount of comments
- avoid repetition in code

How to submit

The code commits will be collected on MarkUs. Submit a link to your github commit in a plain text file called `link.txt`.

Please make sure that you are submitting a link to the **commit**, not a link to the main repository page, or a link to the branch. For example, this is a link to a github commit:

`https://github.com/kwpark23/Zeros_Matter/commit/467a5162abe12998545b9bcb3ebdd312e9b8d922`

Grading

Grading will be out of 5. With:

- 1 point for a non-trivial commit
- 1 point for a commit that is **atomic**
- 1 point for a commit message the follows the guidelines discussed in class, and outlined above.
- 1 point for code style
- 1 point for code clarity