As Art said, commit messages are essential for communicating why your code was changed. This is for your coworkers or collaborators, and also for your future self. Let’s go through some best practices for writing commit messages.

Here is the message format we use here at Udacity. You can find it [here](http://udacity.github.io/git-styleguide/) for future reference.

type: subject

body

footer

The first line is the subject. This should be a short description of what changed. Nothing like “fixed it” or “did something,” these need to be clear and informative, and try to avoid profanity. The subject should be 50 characters or less, with the first letter capitalized, and end without a period. At Udacity, we also include a short annotation about the type of the commit, if it is a bug fix, a feature, change to the documentation, etc.

The body is next, this is where you give a more detailed description of why you made the change. The body should typically have around 72 characters per line. This is to ensure that the message fits into a terminal window when using git on the command line. You’ll also need to make sure there is a blank line between the subject line and the body. You can also add bullet points, using asterisks or dashes, when you need to make a list.

Some commits don’t require a body in the message. If you fix a typo for example, it’s okay to only have a subject line.

You can also include a footer, typically this will be used to indicate which issues or bugs the commit addresses.

A more fleshed out example looks like this:

feat: Summarize changes in around 50 characters or less

More detailed explanatory text, if necessary. Wrap it to about 72

characters or so. In some contexts, the first line is treated as the

subject of the commit and the rest of the text as the body. The

blank line separating the summary from the body is critical (unless

you omit the body entirely); various tools like `log`, `shortlog`

and `rebase` can get confused if you run the two together.

Explain the problem that this commit is solving. Focus on why you

are making this change as opposed to how (the code explains that).

Are there side effects or other unintuitive consequences of this

change? Here's the place to explain them.

Further paragraphs come after blank lines.

- Bullet points are okay, too

- Typically a hyphen or asterisk is used for the bullet, preceded

by a single space, with blank lines in between, but conventions

vary here

If you use an issue tracker, put references to them at the bottom,

like this:

Resolves: #123

See also: #456, #789

This does come with an exception of course. If you are working on an open source project, be sure to follow the message format for that project. This will make the maintainers happy and increase the chance your pull request is accepted.

NEXT

**Apply these best GitHub practices moving forward and attract the attention of recruiters and senior software engineers.**

You'll want to:

* Commit every week, at minimum
* Get 2-4 weeks streaks

A personal side-project may grow into something bigger. You can't always predict what will catch the eye of a recruiter or senior software engineer, but when it does, you want to showcase your best work.

Continue building your knowledge of using GitHub and improving your profile, be sure to check out the "GitHub Professional Profile Resources" found in the Classroom folder under Lesson Resources.

Once you've implemented the necessary changes to make your profile more professional, remember to maintain activity by regularly making commits on GitHub.

Good luck!