About Git

Git is one of the most commonly used systems for version control which allows us to manage changes in our code and collaborate with others. This document will serve as a reference. For a more detailed guide, check out the Git documentation here.¹

Installing Git

Before installing Git, you first want to check if it is already installed on your machine. To do so, open terminal and type `git --version`. If the terminal echoes out the version, then Git is installed.

Otherwise, you can download the latest version of Git here. For more detailed installation instructions, check out this guide.

Fundamental Commands

`git clone URL-or-path-to-repo`
Clone a repository. This makes a new folder in the current directory containing the files in the repository in which you can work on the code.

`git add`
Add a given file to the repository so that it becomes tracked by git. To make things easier, you can run `git add .` to add all new files.

`git commit`
Finalize the current changes to your code as a commit to your current branch and repo on your local machine. A git repository is made up of a chain of commits, which mark small milestones in your project. We recommend you to commit frequently, with descriptive commit messages.

¹ The information of this document was sourced from the CS 32 Git Reference
Make sure to include -a to automatically add changes from tracked files, and -m “<message>” to include a message about the commit (otherwise you will be kicked to an editor in which to type out your message). For instance, `git commit -am “initial commit”`.

```
  git push
```

Push whatever commits you have made locally to the remote repository that you cloned previously. In this class, this effectively means uploading your code online to GitHub Classroom. Note that if you’re working in groups and someone else pushed something before you, you’ll have to `git pull` their changes first before pushing anything yourself.

```
  git pull
```

Pull any changes that have happened on the remote server you initially cloned and bring them into your local repository. You will need to have a clean repo so you will probably want to commit first if you have changes (or stash them).

```
  git status
```

See what changes are currently being tracked. (If you see files under “Changes not staged for commit” that you want to include in the next commit, run `git add .` to do so.)

```
  git log
```

See your history of commits, with the top being the most recent. Each commit is labelled with a long hexadecimal “hash”, which is its unique identifier. This command is helpful for specifying which commit you want to turn in as your submission.

```
  git branch
```

List all of the current branches.

```
  git checkout <branch>
```

Switch to a different branch called `<branch>` or create a new branch called `<branch>` if it does not exist yet. You can also use this to check out different commit hashes—very helpful for debugging if you’re trying to figure out when something broke!

```
  git merge <branch>
```

From your current branch, merge in the changes from `<branch>`. 
Branches (optional)

In Git, pieces of work are generally centered around branches. A branch is a working set of changes that will eventually be merged back into the application's codebase. When just 1 person is working in a codebase, as you will be for the first project, you may not need to branch. However, using branches is a good way to keep changes organized (and know which changes broke what later on!).

You can list all the branches on your local repo with the command `git branch`. When you do this at first, you should just see master which is our main branch in any git repo. You can see all the branches on your remote repo with the command `git branch -a`. One of the branches for the remote repo should be remotes/origin/stencil, which is where the stencil is! You can switch between branches with the command `git checkout <branch name>`.