GitHub Guide

What is GitHub?

GitHub is a version control system that is built on Git. It allows one to store code and make changes to an online repository. This allows one to easily change and update code as well as collaborate with others by sharing a common repository. We will be using GitHub Classroom to distribute all of the stencil code for assignments using a link on the course website, but let's learn a little more about GitHub and Git.

Git is the basis of GitHub and is a version control system you can use right from the command line, you can learn more about it on their website!

- Download the latest version of Git.
- Check out the installation guide.
- Take a look at this great cheat sheet of git commands that GitHub provides.

Why GitHub?

- Store your code and files on an online repository so you won't lose your code.
- If you commit (save) and push your code often:
  If your computer crashes or you make a mistake, you can revert to an earlier version of your code and keep working.
- You can create a record of changes and always check out a previous version
- Allows for easy collaboration. People can work on the same repository and update/change code on separate machines.

You can sign up for an account and get the GitHub Student Pack with your Mines Email for more tools and features

Some Essential Commands of Git

Below are some common git commands to get you started! The rest of the document will cover them in more detail. **(exclude the brackets for all commands)**

Command	Description
`git clone [URL-or-path-to-repo]`	Retrieve an entire repository from a hosted location via URL to your local machine
`git add [file]`	Add the given file to the repository. Use this when you create a new file and want to include it in a commit. Alternatively, include [--a] to add all files from your local repo
`git commit --m "[some message]"`	Commit your code to finalize and save changes to your current branch and repo on your local machine. Include `[--a]` to automatically add changed files that git is already tracking and `[--m "[some message]"]` to include a message about the commit (otherwise you will be kicked to an editor in which to type out your message).
`git push`	Push whatever commits you have made locally to the repository you cloned from to save your changes to the online repo. (You might have to pull first to sync with the remote repository)
`git pull`	Pull any changes from the remote repository you cloned from.
`git checkout -b [NAME_OF_BRANCH]`	Create a new branch called `NAME_OF_BRANCH`. The `-b` means you are creating a new branch.
`git branch`	List out all of the branches you have created.
`git merge [NAME_OF_BRANCH]`	Merge `NAME_OF_BRANCH` into your current branch.

Editing and committing changes

You can edit code on your personal machine. Once you're ready to save or backup your changes, you can follow these steps:

1. Open a terminal and `cd` into the project directory.
2. Add all the files to your working branch by running the command

   ```
   git add --a
   ```

   > Alternatively, if you only want to save a particular file to, you can run the command
   >
   > ```
   > git add [PATH_TO_FILE]
   > ```

3. Make a short description of the changes that you've made by running

   ```
   git commit --m "
   [SOME_MESSAGE]"
   ```

4. Push the changes to GitHub by running

   ```
   git push
   ```

Remember to commit often to save a record of your code and push to the remote repository to save it on GitHub as well.

Github Desktop

If you prefer GUIs we recommend using GitHub Desktop, this gives you a great visual way of interacting with GitHub.

- Download GitHub Desktop
- Follow the step-by-step guide for GitHub Desktop.

Branches

Branching is the way to work on different versions of a repository at one time. By default your repository has one branch named `main` which is considered to be the definitive branch.

For most projects in this class, you won't need to worry about creating different branches. However, when it comes time to work and collaborate with your peers on the final project, we highly recommend creating separate branches to experiment and make edits before

committing them to `main`. This will help save you from a lot of frustration when you're working on changes and want to save them to GitHub.

Create a new branch

You can create a copy of the current main branch by running the following command in the terminal once you `cd` into the project directory:

```
git checkout -b [NAME_OF_NEW_BRANCH]
```

The `-b` indicates that you are creating a new branch.

To switch back to the main branch, you can run:

```
git checkout main
```

You can see all of the branches you have created by running `git branch`.

Committing changes to a branch

Follow the same steps to commit changes to a branch as above. However, instead of writing `git push`, you need to write:

```
git push origin [NAME_OF_BRANCH]
```

Merging branches

To merge your changes from one branch into the main branch, make sure you've pushed your changes, then switch back to the main branch by running `git checkout main`. Then run the following command to merge the branches:

```
git merge [NAME_OF_BRANCH]
```

Your main branch should now have the new changes from the other branch and you can push to GitHub.