**CLI Commands:**

* Run the pwd command and explain to students that it stand for “print working directory” and tells you what directory you’re currently in
* Run the cd command to switch to a new directory and explain to students that it stands for “change directory”
* Run the ls command and explain to students that it stands for “list” and lists all the visible files and directories within your current directory
* Run the mkdir command and explain to students that is stands for “make directory” and creates a directory within your current directory
* Run the touch command and explain to students that it creates a new file in the working directory
* Run the clear command and explain to students that it clears your previous inputs and outputs from the CLI

**Git Commands:**

Having an example repository already on your machine to show these would probably save time, but if you don’t already have one then:

* In whatever folder you choose on your computer, make a directory with mkdir examplerepo
* Change the directory to examplerepo with cd examplerepo
* Type git init to initialize a repository
* If you’re using git bash or a Mac, then touch example.html
* Open example.html in Visual Studio Code
* Add in boilerplate HTML (if you type ! then press Tab, it’ll put it in for you)
* Add a <p> tag with some text
* In the CLI, type git status to show that there have been changes made
* If it says that everything is clean, check to make sure you’ve saved the changes in VS Code
* Do a git add . then git commit –m “<message>”
* After the commit, git log to show the commit message history
* Add another <p> to the HTML file
* git status, add, commit again
* git log to see your first commit’s hash code
* Do git revert –n <first 7 digits of the code>
* git commit –m “Just reverted back to my first commit”
* Check the VS Code files to see that the changes from the second commit are no longer there
* git log to see the second commit’s hash code
* Do git reset <first 7 digits of the second commit’s code> --hard
* Check the VS Code files to see that the changes from the second commit are back
* git log to see that the revert commit is no longer there

**Branching:**

* Create a new branch off of master (or main) called develop and switch to it
* Make a change to example.html
* Commit the change
* Switch back to the main branch and run git log to demonstrate that this branch does not have that commit
* Run git branch to see what branches exist
* Run git branch –r to see what remote branches exist

**Merge Conflicts:**

* Open examplerepo again
* Remind students that a change was made to example.html in the develop branch that does not exist on this branch
* Make a different change to example.html
* git merge develop
* See how the IDE shows you the differences between the branches
* Keep which change you want and delete the other one plus any comments the IDE made about the conflict
* git add . and git commit
* git log to show that it’s been fixed

**GitHub Tour:**

* Create a new branch on the example repository you just made and cloned
* Create an index.html file with boilerplate code
* git add, git commit, git push
* Show that the new branch and file shows up on GitHub
* Create a pull request on that branch