**Exercise 3**

There are a couple of ways to clean up your Git history before pushing to a remote repository, with the most common methods being git reset and git revert.

### Using git reset

The git reset command is a powerful tool for undoing changes in your local repository. It can be used to move the branch pointer to a previous commit, effectively discarding all commits that came after it.

1. **Find the commit hash**: Use git log to find the commit hash of the commit you want to revert to.
2. **Reset the branch**: Run git reset --hard <commit-hash>. This will discard all changes in your working directory and staging area, and reset the branch to the specified commit.
3. **Force push**: After resetting your local branch, you will need to force push the changes to the remote repository using git push --force. This overwrites the remote history with your local history.

### Using git revert

The git revert command creates a new commit that undoes the changes made in a previous commit. This is a safer option than git reset because it doesn't rewrite history and is a better choice when working on a shared branch.

1. **Find the commit hash**: Use git log to find the commit hash of the commit you want to revert.
2. **Revert the commit**: Run git revert <commit-hash>. Git will create a new commit that contains the exact opposite changes of the specified commit.
3. **Push the changes**: Once the revert commit is created, you can push it to the remote repository using git push origin <branch-name>.

### Other Clean-up Commands

* git clean: This command removes untracked files from your working directory. You can use git clean -n to perform a dry run and see which files would be removed, and then git clean -f to actually remove them.
* git rebase: This is a more advanced command that can be used to squash multiple commits into a single commit, reorder commits, or remove them entirely. It's an effective way to create a clean, linear history before pushing your changes.







