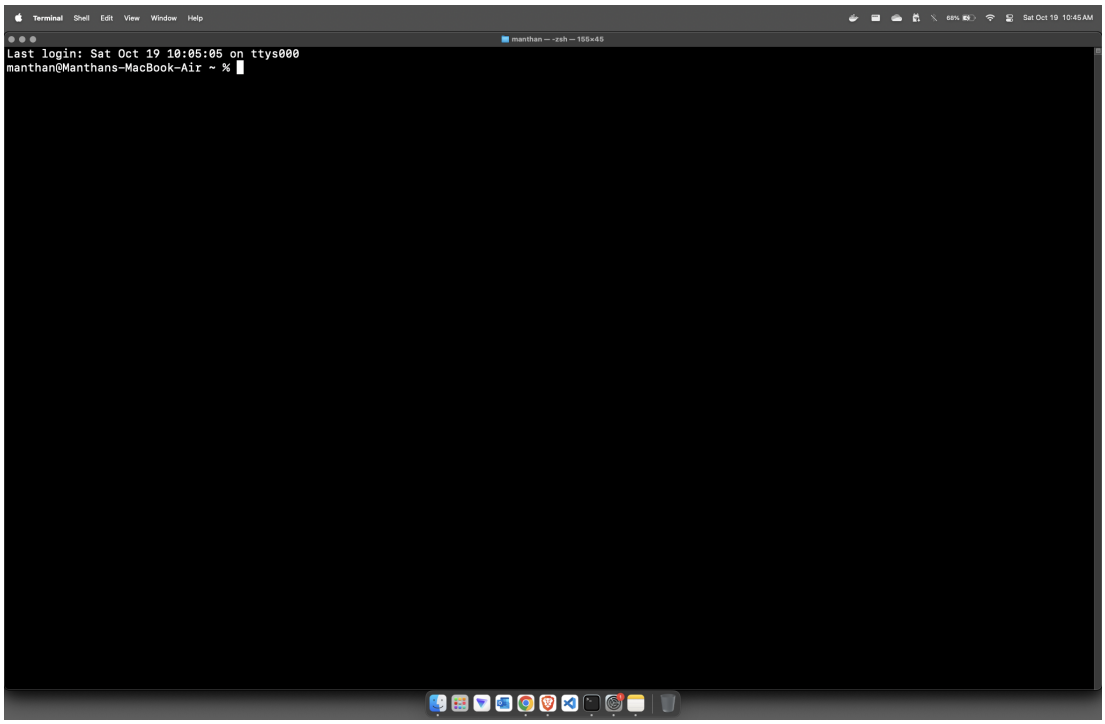
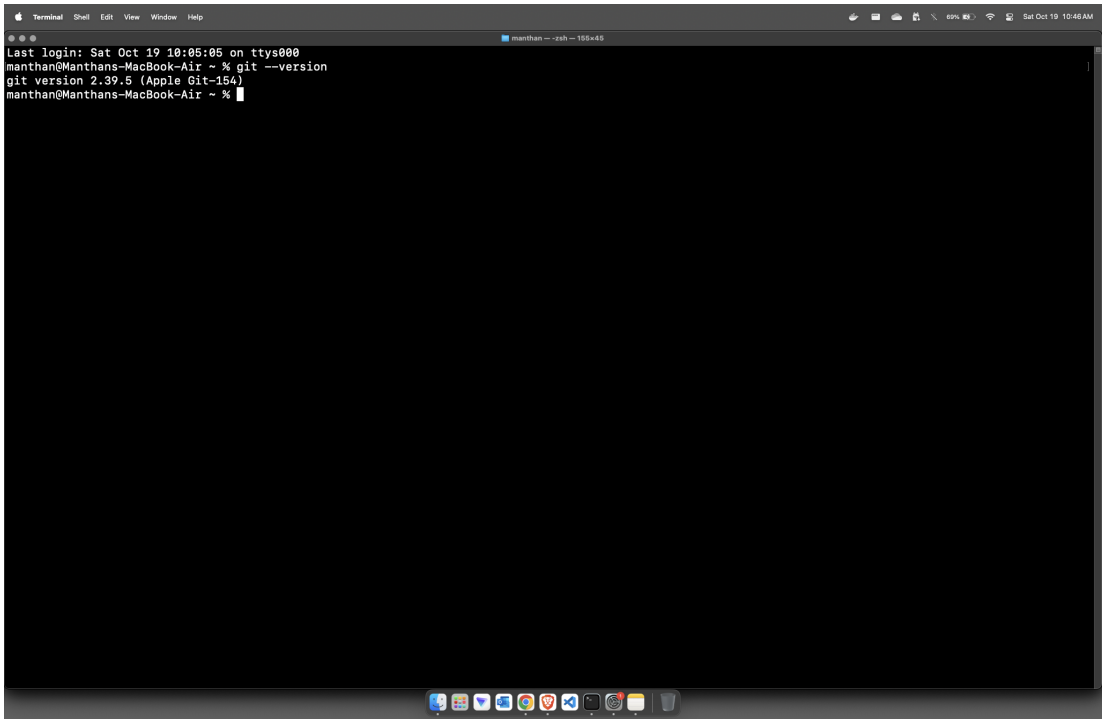
Git and Github - Lecture 3

1. Blank Terminal Screenshot

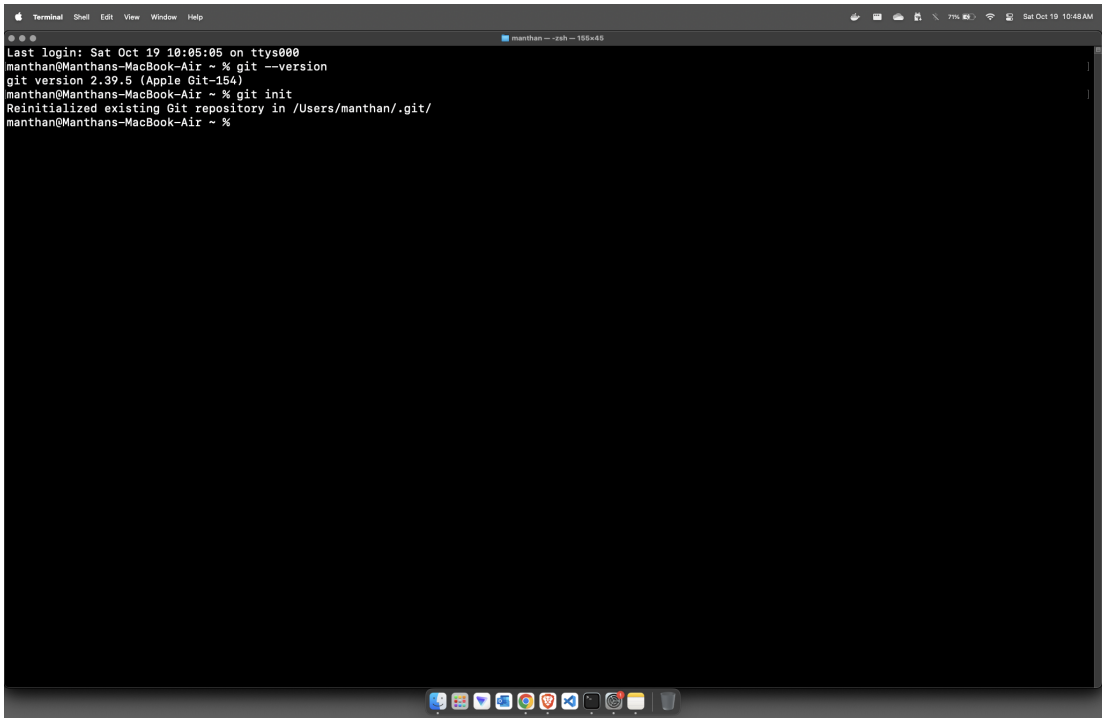
   Step - Open Terminal



2. To check git installation, check with git —version

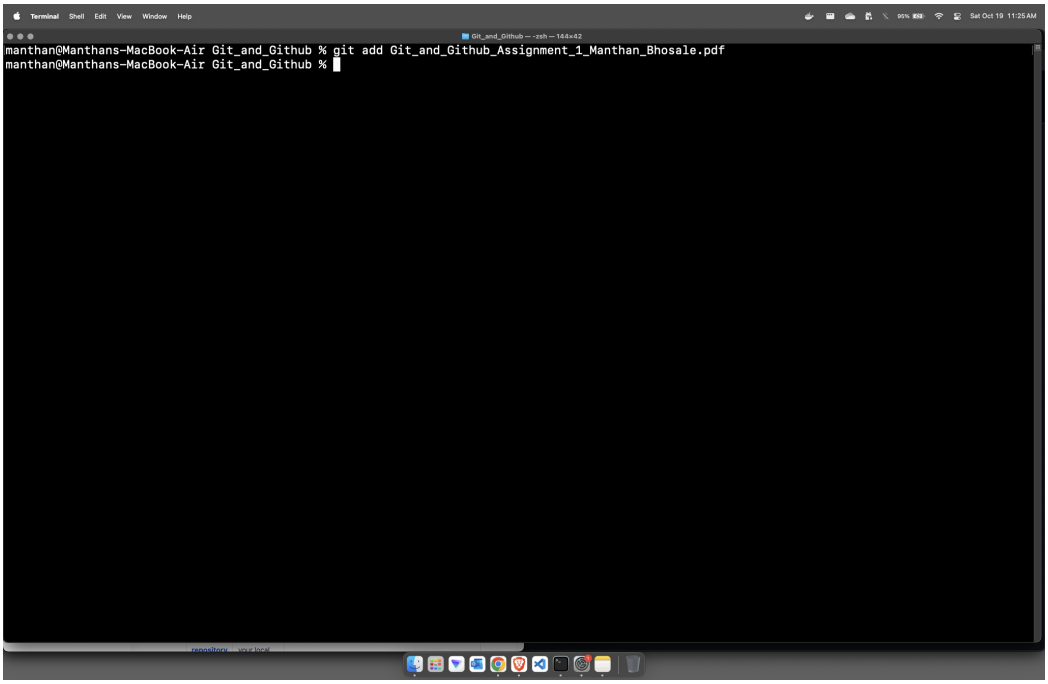


3. Initialization

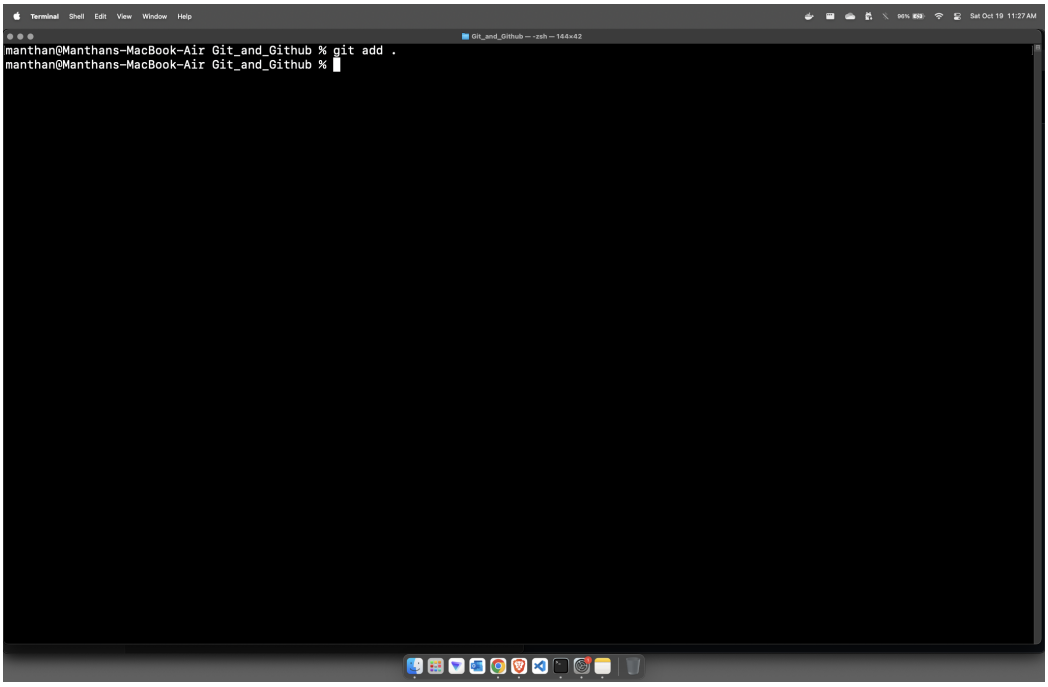


4. Basic Operations:

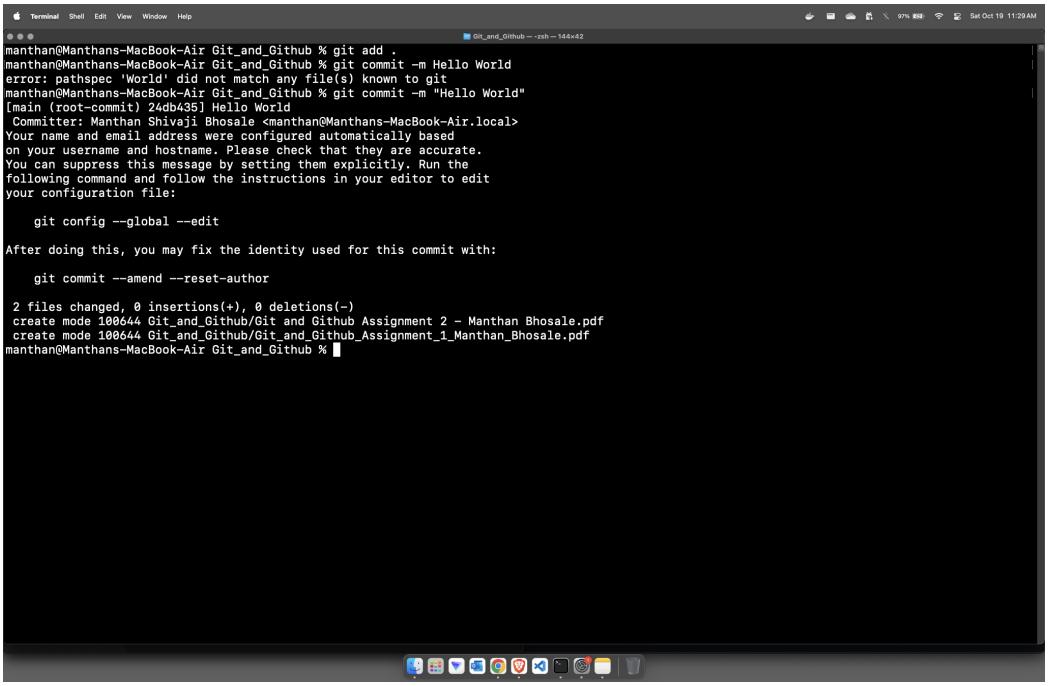
- git add [file]



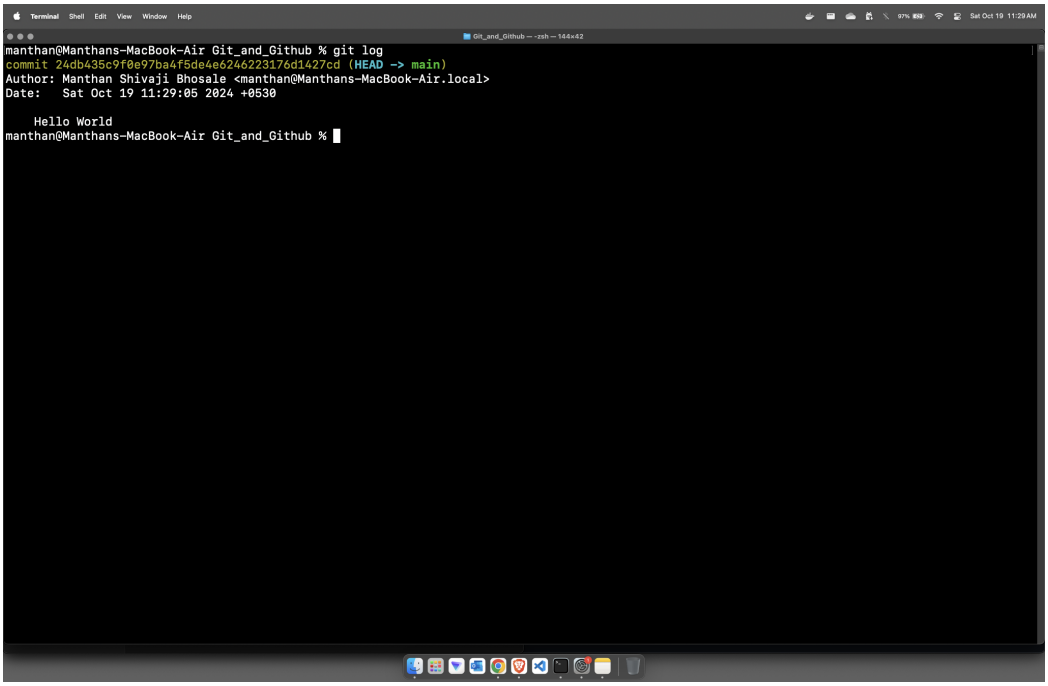
- git add .



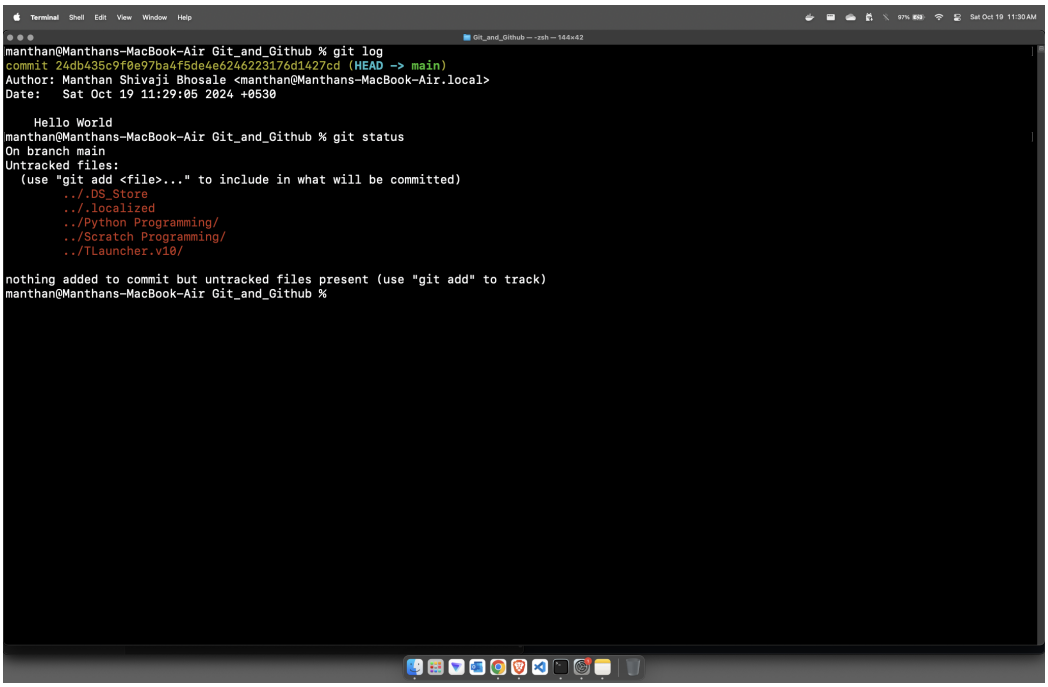
- git commit -m [message]



- git log



- git status



Punishment:

1. Git is a free and open-source distributed version control system designed for tracking changes in source code during software development. It allows developers to work on the same project without overwriting each other's files, and provides mechanisms for collaboration, branching, merging and rolling back changes. Git differs from other version control systems such as Subversion in that it is a distributed system rather than centralized, which means changes are stored locally and can be shared with others through a network.
2. The "git add" command is used to add files or directories to the staging area for further processing by Git's commit command. This allows developers to selectively choose what files they want to include in their commits without having to make changes directly on the repository itself.
3. The "git commit" command saves all local changes to the repository, while the "git push" command sends those changes to a remote repository for other developers to see and work with. This allows developers to collaborate and

work together on projects even when they are not in the same physical location.

4. To create a new branch in Git, you can use the "git branch" command followed by the name of the new branch you want to create. For example: "git branch newbranchname". Once the branch has been created, you can switch between it and your main branch using the "git checkout" command.
5. The purpose of Git branches is to allow developers to work on different features or enhancements simultaneously without affecting each other's progress. Branches are also useful for creating feature branches that can be merged into the master branch at a later date, allowing developers to test and validate changes before they are merged into production code.
6. To merge two Git branches together, you can use the "git merge" command followed by one of the branches you want to merge into another. For example: "git merge mainbranch". This will create a new branch that combines both sets of changes from the original branches.
7. A remote repository is an external location where your Git repository is stored and can be accessed by other developers. It allows multiple people to work on a project simultaneously without overwriting each other's files, since they are all working on different branches. Remote repositories are also useful for collaboration with other teams or organizations who may not have access to your local environment.
8. To revert a change in Git, you can use the "git reset" command followed by a commit hash. This will undo the changes made in that specific commit and leave your files untouched.
9. To collaborate with others using Git, you can share the codebase via a remote repository, allowing other developers to work on the same project without being in the same physical location. You can also use version control systems like Git to create pull requests, which allow other developers to review and comment on changes before they are merged into the main branch.
10. The difference between "git fetch" and "git pull" is that "git fetch" only downloads changes from a remote repository without merging them into your local codebase, while "git pull" also incorporates those changes into your local environment.