



**Computer science fundamentals and career pathways**

**B.TECH CSE CORE SECTION (A) SEMESTER - 1**

**Course code : ETCCcp105**

**ASSIGNMENT TITLE : git assignment**

**SUBMITTED BY : MANPREET KAUR**

**Roll number : 2501010070**

**SUBMITTED TO : rajesh kumar sir**

**DATE OF SUBMISSION : 23 november 2025**

Git

A logo on a red background

AI-generated content may be incorrect.

Git is a distributed version control system (DVCS) designed to track changes in source code or data, enabling multiple developers to collaborate efficiently on software projects. Created by Linus Torvalds in 2005, Git allows developers to manage versions, track changes, and maintain history locally on their machines as well as synchronize with remote repositories. It supports non-linear workflows by enabling thousands of parallel branches and provides tools for committing snapshots, branching, merging changes, and collaborating asynchronously across different locations. Git is free, open-source, and widely adopted for both open-source and commercial software development due to its speed, data integrity, and distributed nature.​

Key concepts of Git include repositories (local and remote), commits (snapshots of project states), and branches to work on different features simultaneously without affecting the main codebase. Users stage changes to prepare them for a commit, which permanently records those changes in the repository history. Git also supports powerful collaboration features common in modern development workflows.​

In summary, Git enables developers to keep a detailed history of their code, work collaboratively in a distributed manner, and manage multiple versions and branches safely and efficiently throughout the software development lifecycle.​

**GITHUB**

GitHub is a website and cloud service where developers store and share their code projects using Git. It helps people work together on software from anywhere by keeping track of all code changes. GitHub also offers tools for managing projects, reviewing code, and collaborating easily. It is popular for hosting open-source and private projects, making teamwork easier

**VISUAL STUDIO CODE [ VS CODE ]**

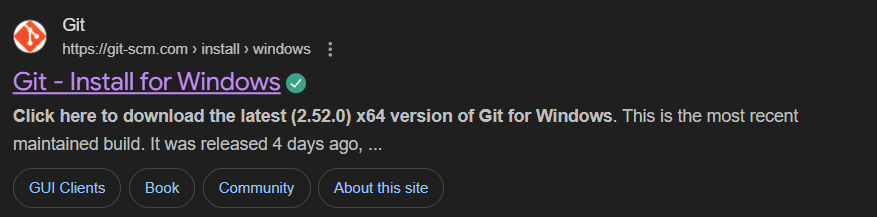
VS Code, short for Visual Studio Code, is a free and lightweight code editor made by Microsoft. It supports many programming languages and helps with writing, debugging, and managing code. VS Code includes features like syntax highlighting, code completion with IntelliSense, built-in Git control, and customizable extensions, making coding easier and faster across different operating systems

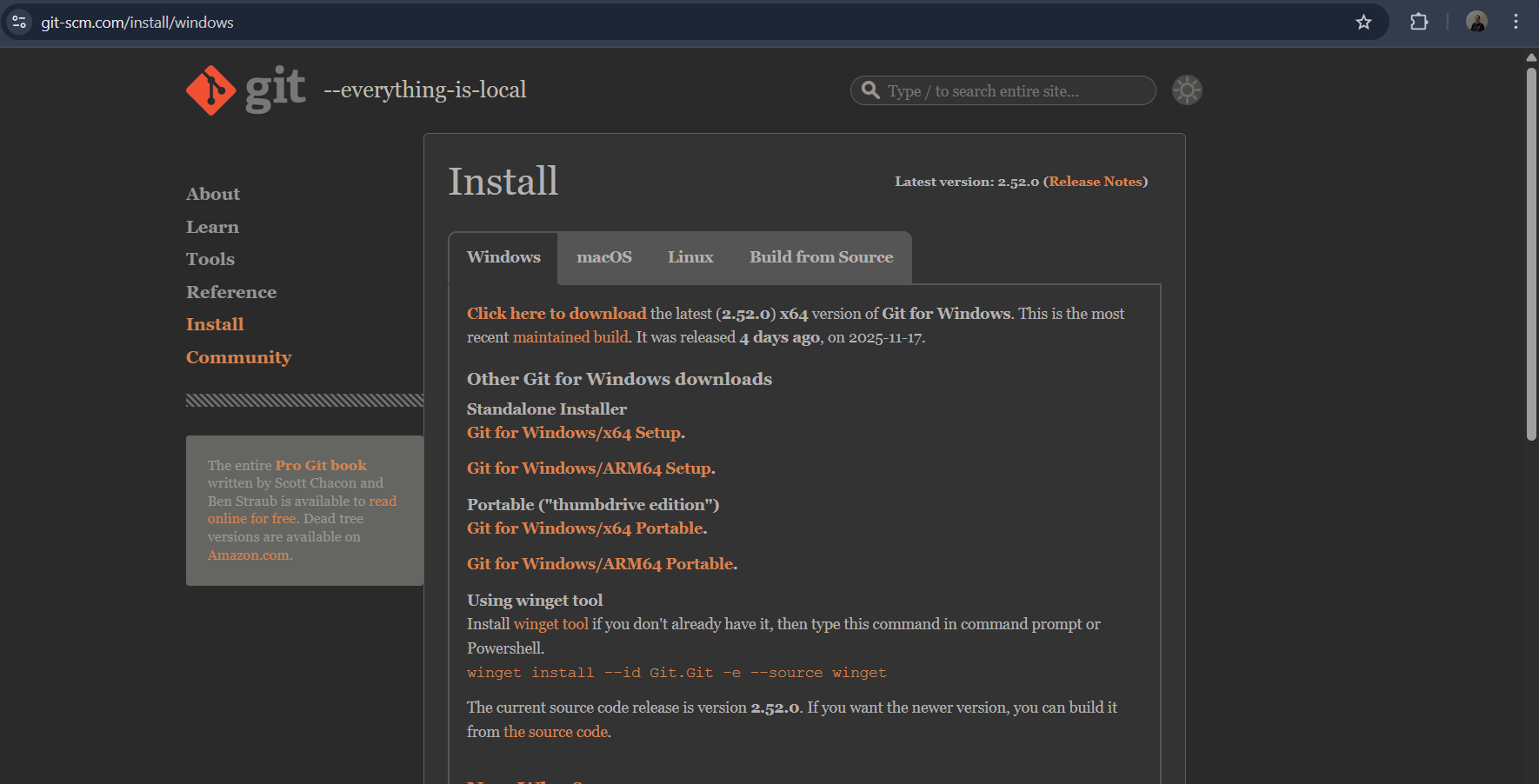
**ABOUT THIS PROJECT**

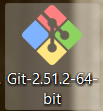
For this assignment, I made a “Basic Calculator” using “Python” programming language in “VS Code” . I used “Git” for “version control” and “Github” for “hosting my files online” and uploaded this on Github with multiple commits , with proper documentation along with the installation steps in this file.

INSTALLATION STEPS

First you have to install the Git for this visit this link :- <https://git-scm.com/install/> and select your operating system. I am working on windows so I have selected the “windows” option and then download the “latest version of Git” from their .

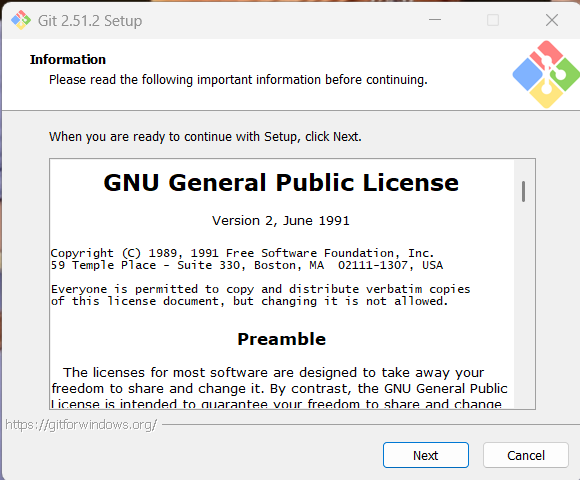






After the installation you will see this file in the “downloads” sections, now open it and

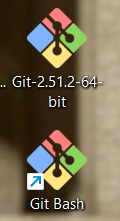
Select the next option and just go with the default settings as shown in the snapshots below.



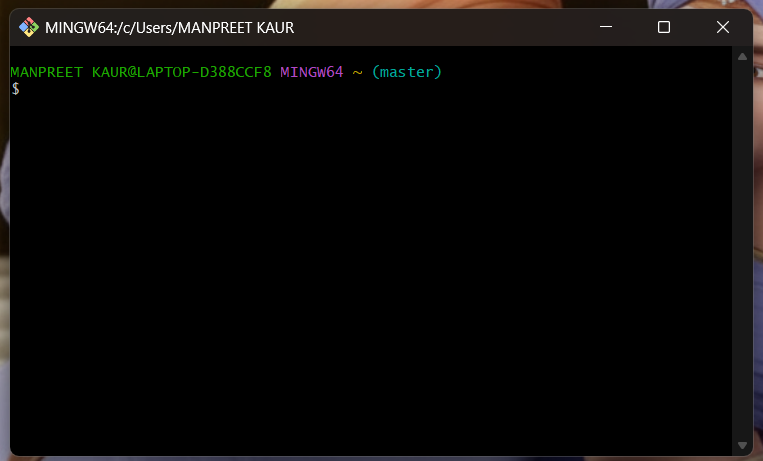
After doing this click on the install option and Git Bash will be installed in your system as shown in the snapshots below.

A screenshot of a computer

AI-generated content may be incorrect.



After all this you will see the “Git Bash” icon on your desktop. “Git Bash” is just like a terminal as shown below. You can run the commands “**git config user.name”** to check the user name and “**git config user.email**” command to check the email id of the user as shown in the snapshot below.



**GIT COMMANDS USED**

**GIT CONFIG USER.NAME**

The **git config user.name** command sets your name in Git. This name appears on your future commits to show who made them. Use **git config --global user.name "Your Name"** to set it for all projects on your computer. You can also set it for one project only without --global. This helps others know who made changes. You can check the current name with git config user.name. It does not change past commits, only new ones.

You can see its implementation part in the snapshot given below.

A computer screen shot of a black screen

AI-generated content may be incorrect.

**GIT CONFIG USER.EMAIL**

The **git config user.email** command sets your email address in Git. This email appears in your future commits to show who made the changes. When you run  **git config --global user.email** **"your\_email@example.com"**, it sets this email for all your projects on your computer. You can also set it only for one project without --global. This helps people see which email is linked to the commits. You can check the email set by running git config user.email. Like user.name, it only affects new commits, not old ones. You can use any email, real or placeholder, but it should match your identity if you want others to recognize you properly.

You can see its implementation part in the snapshot given below.

A computer screen shot of a black screen

AI-generated content may be incorrect.

**GIT - - VERSION**

The command **git --version** shows which version of Git is installed on your computer. When you type this command in the terminal, it tells you the Git program's version number. This helps you know if Git is installed and which update you have. It is a quick check to ensure Git is ready to use.

You can see its implementation part in the snapshot given below.

**A screenshot of a computer

AI-generated content may be incorrect.**

**GIT INIT**

The **git init** command is used **to start a new Git repository**. When you run it in a project folder, it creates a hidden **.git folder** inside that folder. This .git folder holds all the information and history Git needs to track changes in your project. After using git init, the folder becomes a Git repository, and you can start adding and saving changes to it with other Git commands. It's usually the first step when you want to use Git for a new or existing project.

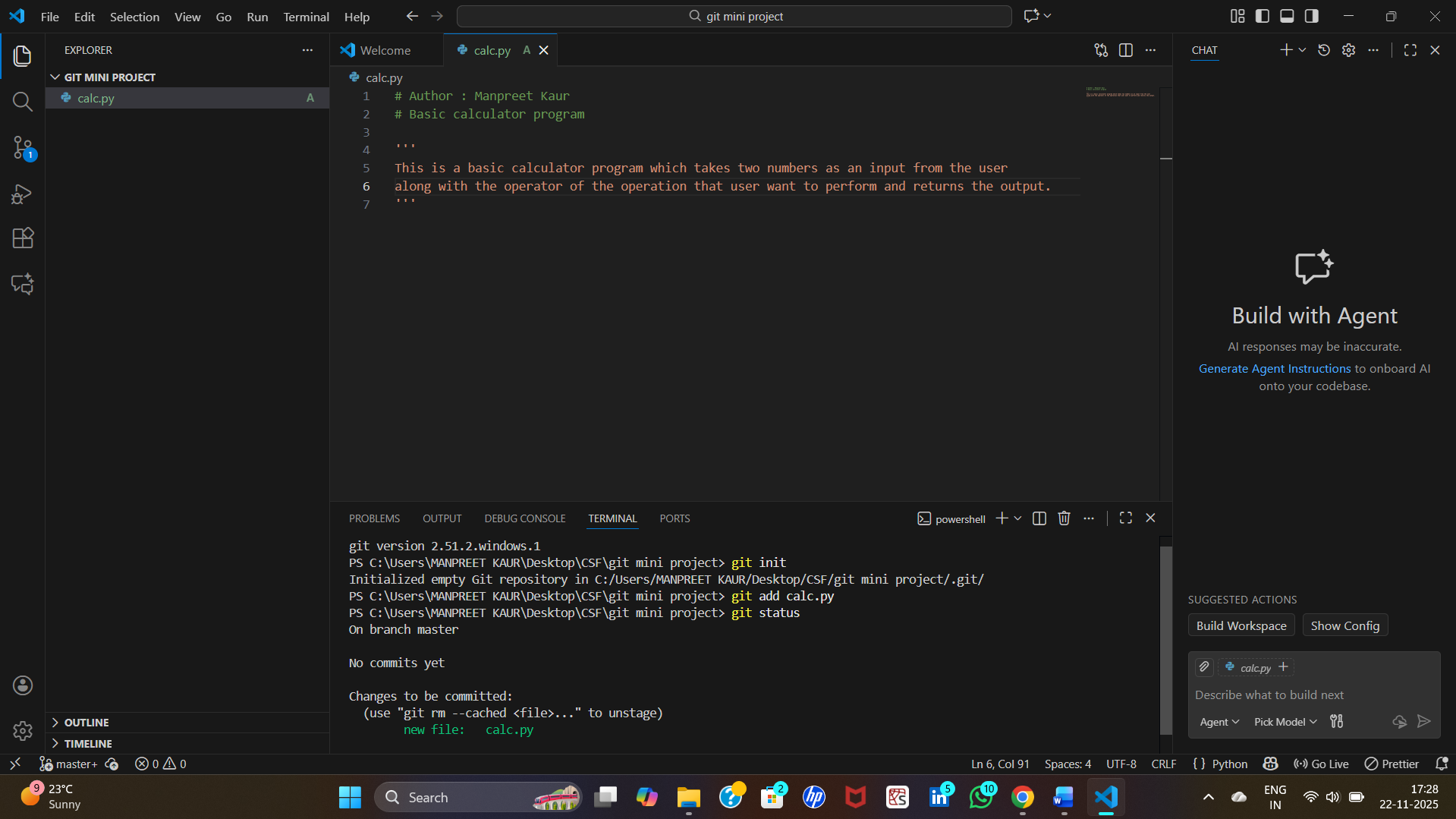
You can see its implementation part in the snapshot given below.

****

**GIT ADD**

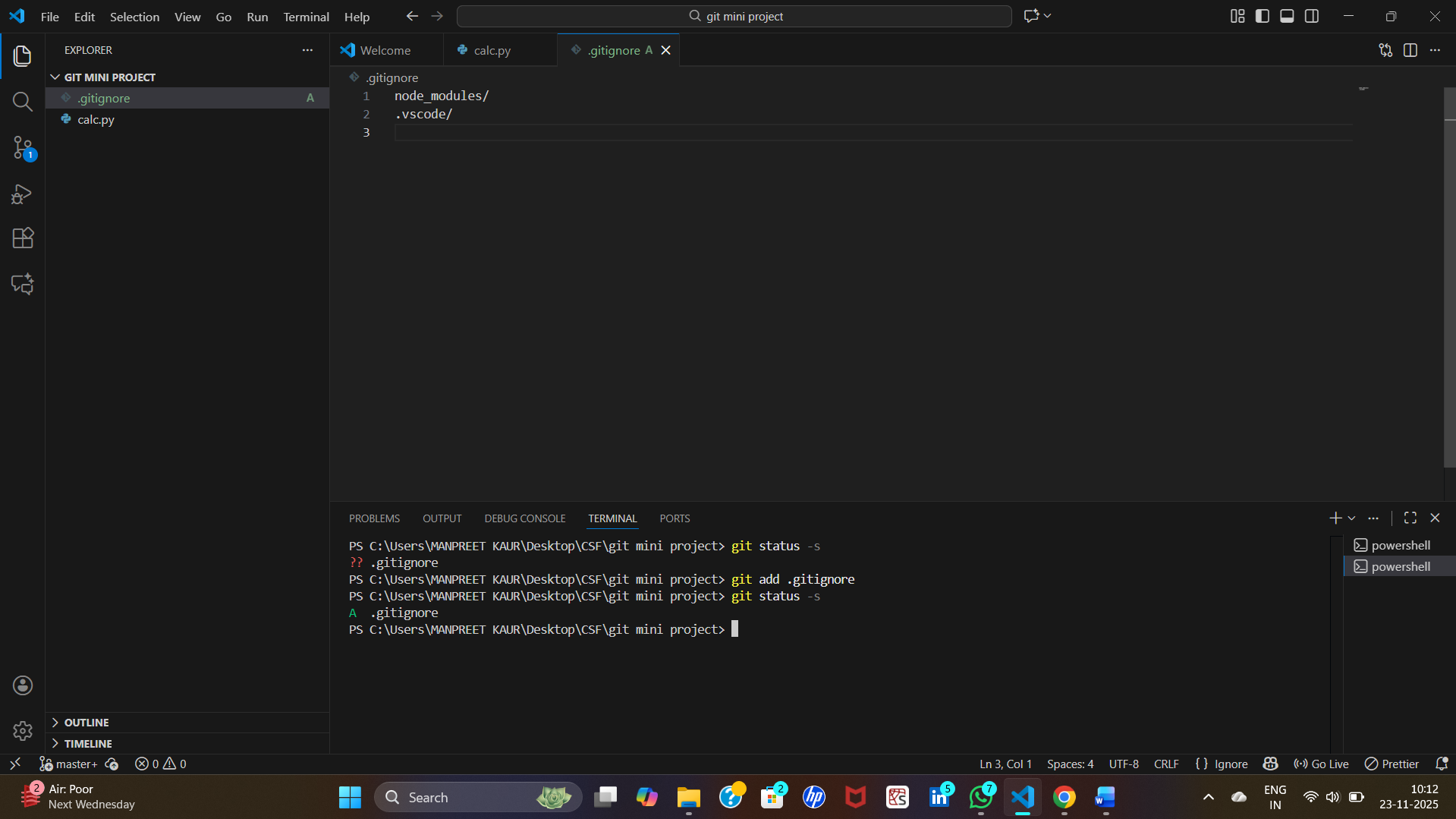
The **git add** command is used to move changes in your files from the working area to the staging area. It prepares these changes to be saved in the next commit. You can add one file, many files, or all files using this command. Without git add, Git won't include your changes when you commit. It’s like telling Git “**these files are ready to be saved now**” before actually saving them with a commit.

You can see its implementation part in the snapshot given below.

****

**A screenshot of a computer

AI-generated content may be incorrect.**

****

**GIT STATUS**

The **git status** command shows **the current state of your Git project**. It tells you which files are ready to be saved (staged), which files have changes but are not yet staged, and which files Git is not tracking. This command helps you understand what work is done and what still needs to be added or committed. It does not show past commits, only the current status of files in your project.

The **git status -s** command shows a short and simple list of the state of files in your project. It tells you which files are changed, added, or not tracked but uses short codes instead of full sentences. This helps you quickly see what has changed without a lot of text. It’s like a quick summary of your project status in a simple format.

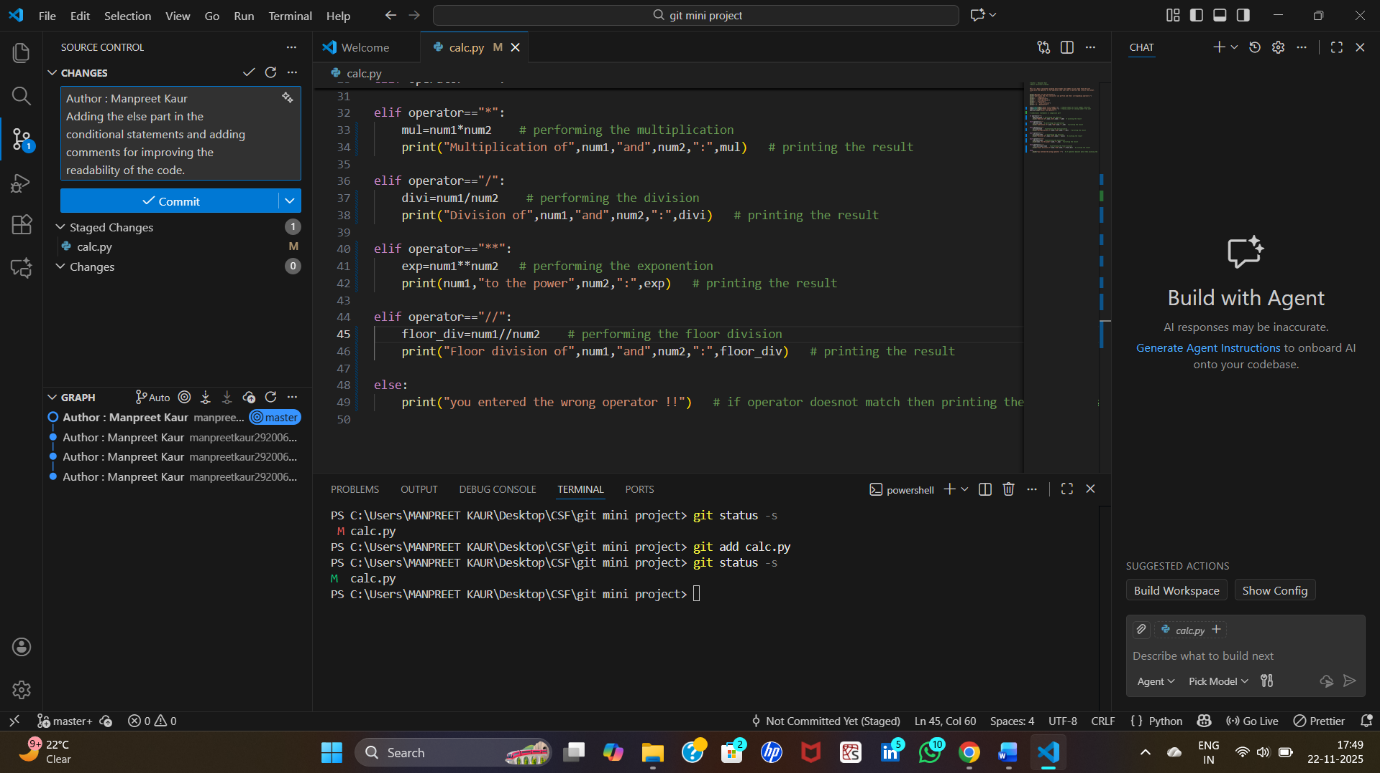
You can see its implementation part in the snapshot given below.

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

****

**GIT COMMIT**

The **git commit** command **saves your staged changes in Git**. It creates a snapshot of your project at that time with a message describing what changed. This snapshot is saved in the repository’s history, so you can go back to it later if needed. It’s like saving your work officially in Git.

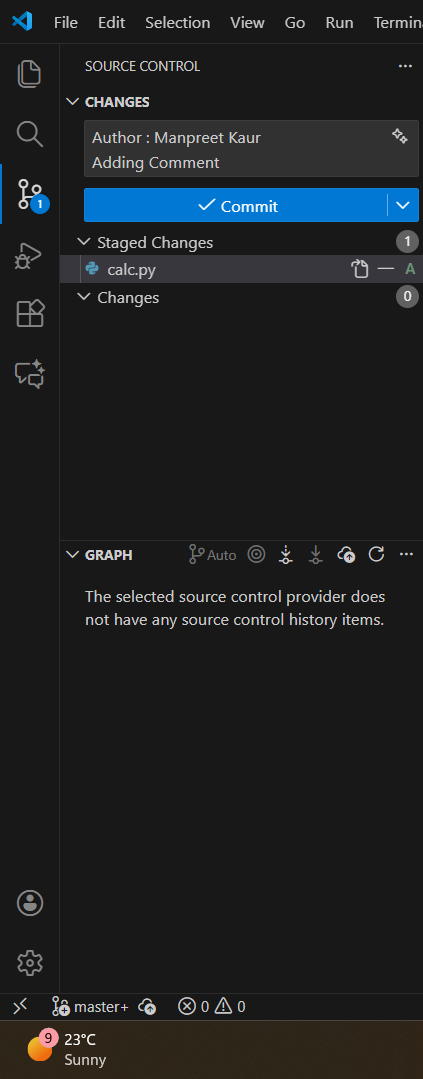
I have listed all the 6 commits that I have done in the snapshots below.

**First commit :-**

In this commit I have added the comments in my code.

**A screenshot of a computer

AI-generated content may be incorrect.**

****

**Second commit :-**

In this commit I have added the welcome message and the user instructions in my code.

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Third commit :-**

In this commit I have included the code for taking the input from the user.

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Fourth commit :-**

In this commit I have added the conditional statements and the calculations part.

**Fifth commit :-**

In this commit I have added the else part of the conditional statements and comments so that code become readable.

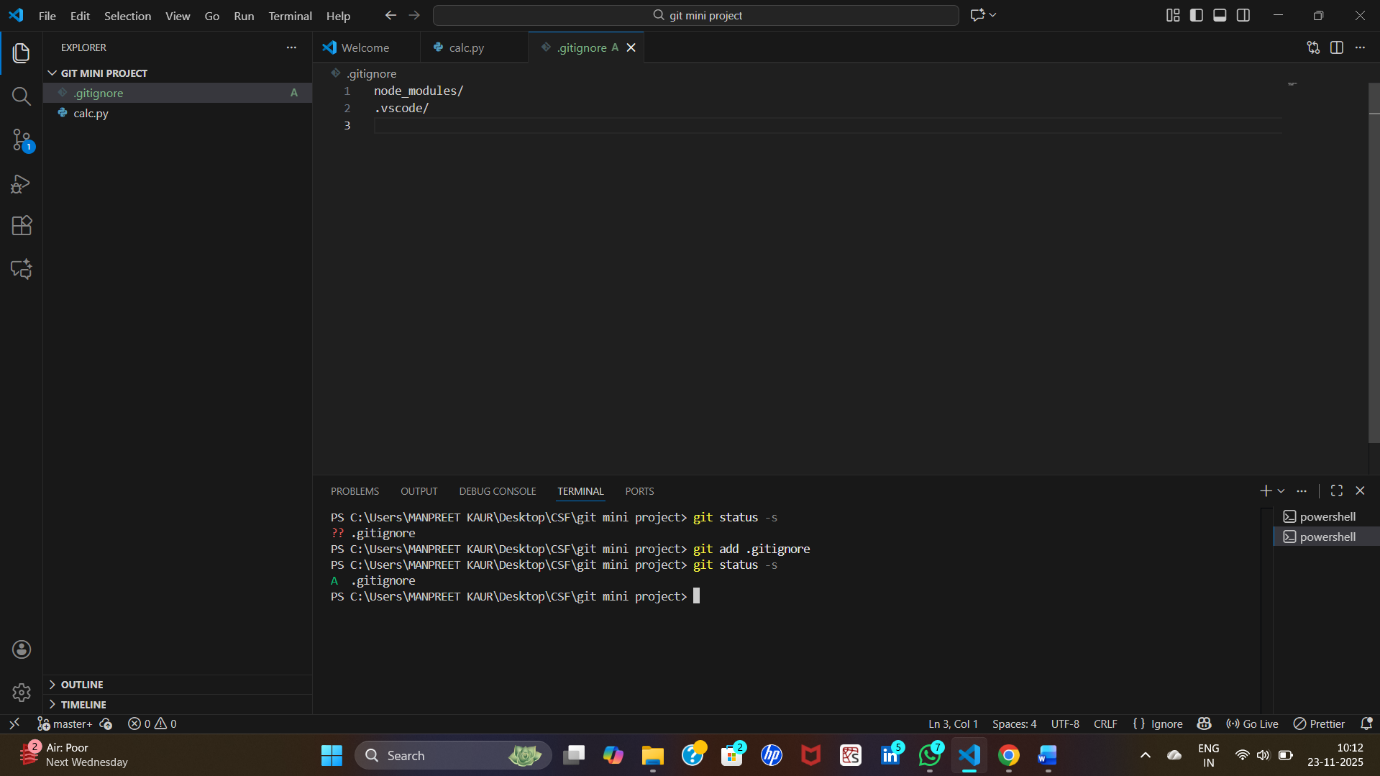
THE IMPLEMENTATION OF THE FOURTH AND THE FIFTH COMMIT IS SHOWN IN THE FOLLOWING SNAPSHOT.

**A screenshot of a computer

AI-generated content may be incorrect.**

**Sixth commit :-**

In this commit I have added the .gitignore file.

****

**A screenshot of a computer

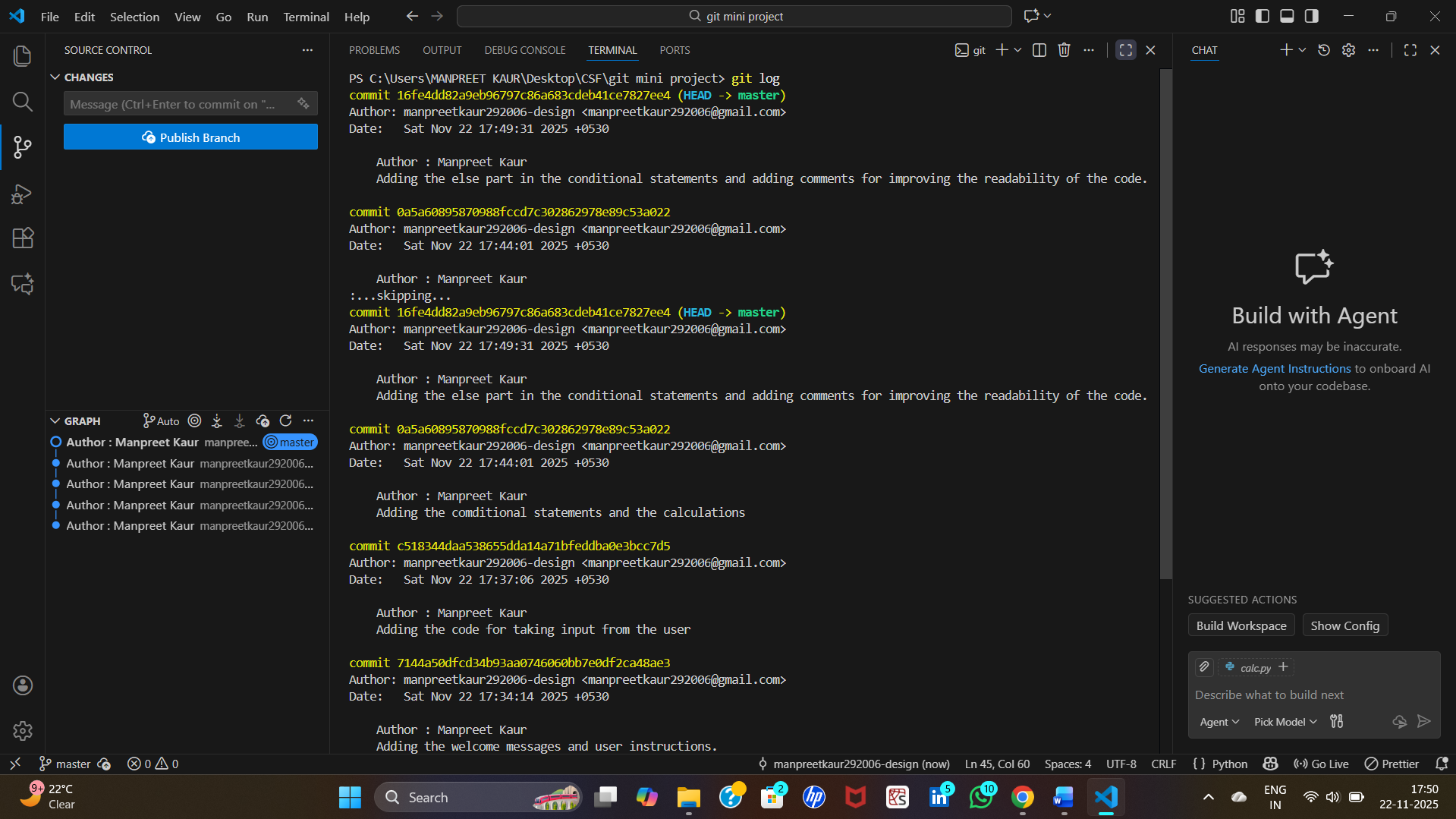
AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.**

**GIT LOG**

The **git log**command shows the history of commits in your project. It lists all saved changes with details like the commit ID, author, date, and message. This helps you see what was changed, who changed it, and when. It works like a timeline for your project's saved work.

You can see its implementation part in the snapshot given below.

****

**A screenshot of a computer screen

AI-generated content may be incorrect.**

**GIT LS – FILES**

The **git ls-files** command shows a list of all files that Git is currently tracking in your project. It looks at the files staged and the working directory, and shows which files are tracked, untracked, or ignored based on the rules. It's like a detailed list of files Git knows about or manages in your project folder.

You can see its implementation part in the snapshot given below.

**A screen shot of a computer

AI-generated content may be incorrect.**

**GIT REMOTE ADD ORIGIN <URL>**

The command **git remote add origin <url>** is used to connect your local Git project to a remote repository. "**origin**" is the name given to this remote (you can use other names, but "origin" is common). The <url> is the web address of the remote repository, like on GitHub. This command lets you push your local changes to the remote and pull updates from it. It's usually used once when setting up a new remote connection for your project.

You can see its implementation part in the snapshot given below.

For this I have first created a repositoy named “**GIT ASSIGNMENT**” on the “**github**” . After this I have copied its URL and run the command “**git remote add origin <URL>**” in <URL> I have pasted the URL of the repository that I have created.

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**GIT PUSH ORIGIN MASTER**

The command **git push origin master** sends your **local master branch changes to the remote** **repository called origin**. It updates the master branch on the remote with your latest commits from your local computer. This is commonly used to share your main development work with others or back it up remotely.

**A computer screen shot of a computer screen

AI-generated content may be incorrect.**

You can see in the snapshots that I have executed this command and after this all the files that I have created were moved to my github repository “GIT ASSIGNMENT” as mentioned in the sanpshots below.

**A screenshot of a computer

AI-generated content may be incorrect.**

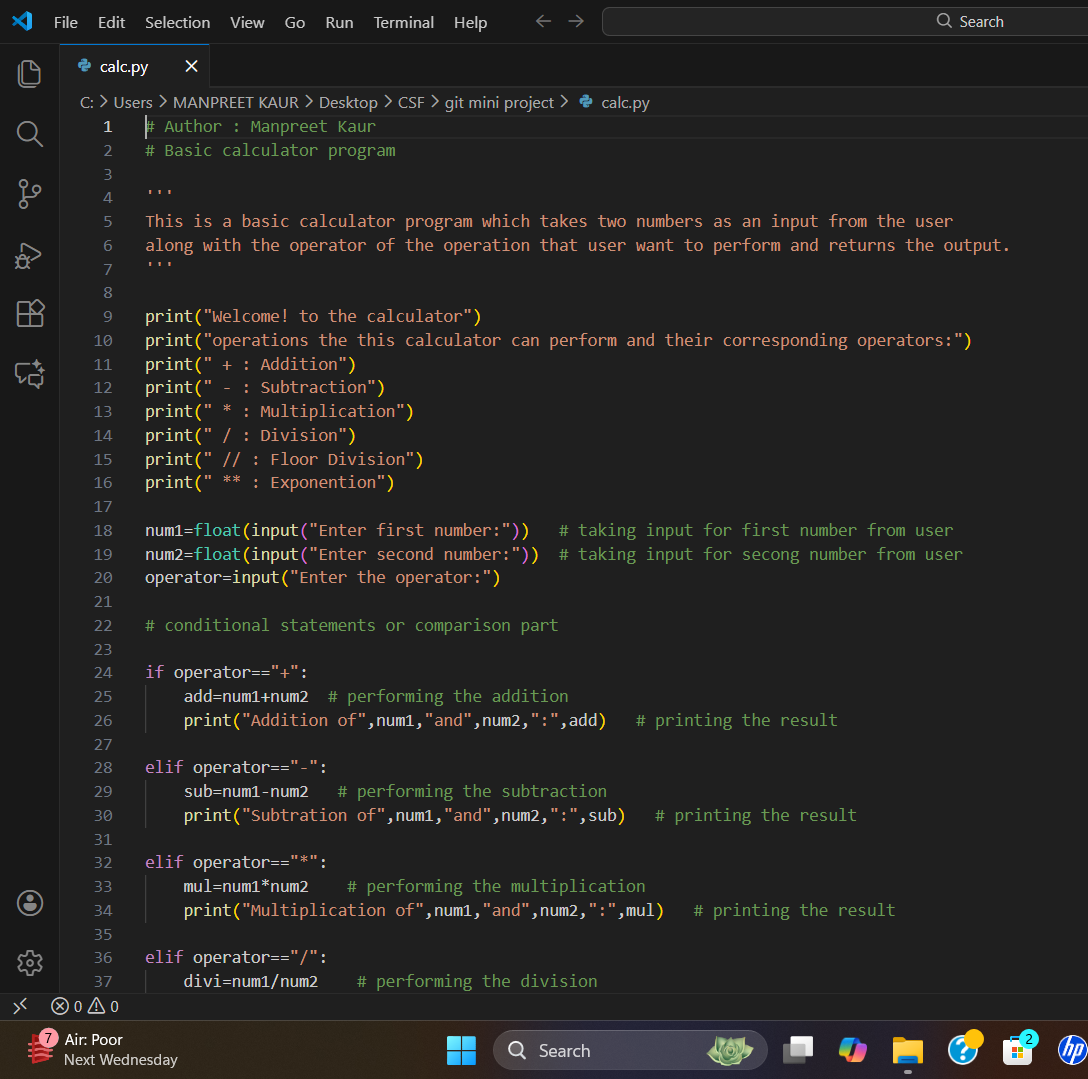
In the snapshot given below you can see the history of all the commits that I have made. Here I have made 6 six commits all these commits are listed here in order.

1. Adding the comments
2. Adding the welcome message and the user instructions
3. Adding the code taking the input from the user
4. Adding the conditional statements and the calculation part
5. Adding the else part of the conditional statements and the comments for easy readability
6. Adding the .gitignore file

**A screenshot of a computer

AI-generated content may be incorrect.**

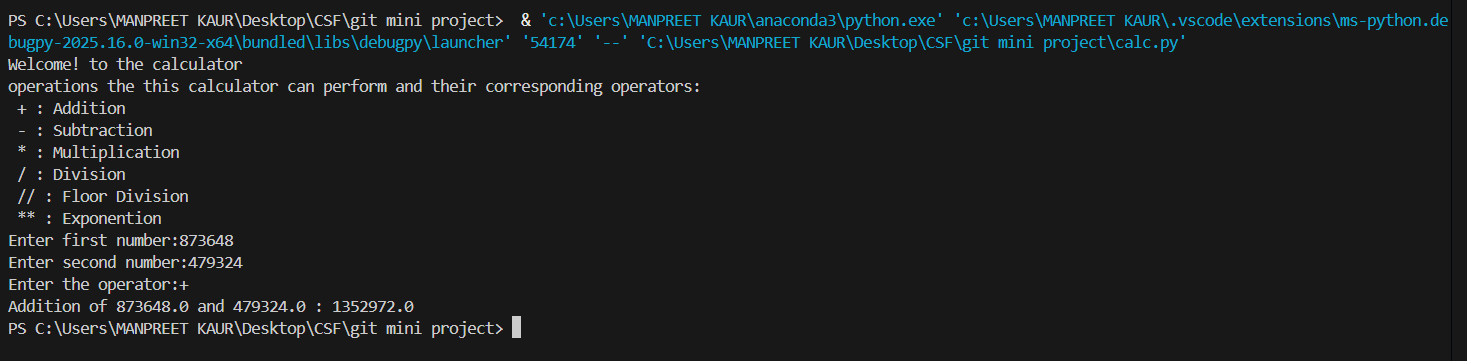
**PYTHON CODE**

****

A screenshot of a computer program

AI-generated content may be incorrect.

**OUTPUT**

****

**GIT HUB REPOSITORY**

<https://github.com/manpreetkaur292006-design/GIT-ASSIGNMENT>

**REFLECTION**

From this project, I have learned how to use Git for version control. I understood how Git helps to save and track every change I make in my code. I learned how to create repositories, add and commit changes, and push the code to GitHub. I also experienced how Git and GitHub help in managing versions and collaborating with others. Using Visual Studio Code with Git made coding and project management easier. Overall, this project taught me the basic tools and steps for working with Git and GitHub in software development.