K.R. MANGALAM UNIVERSITY
THE COMPLETE WORLD OF EDUCATION

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

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]

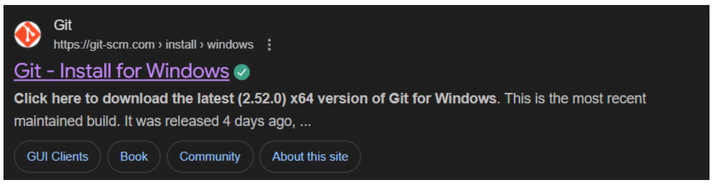
VS Code, short for Visual Studio Code, is a free and lightweight code editor made by Microsoft. It supports many programming languages and helps with writing, debugging, and managing code. VS Code includes features like syntax highlighting, code completion with IntelliSense, built-in Git control, and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

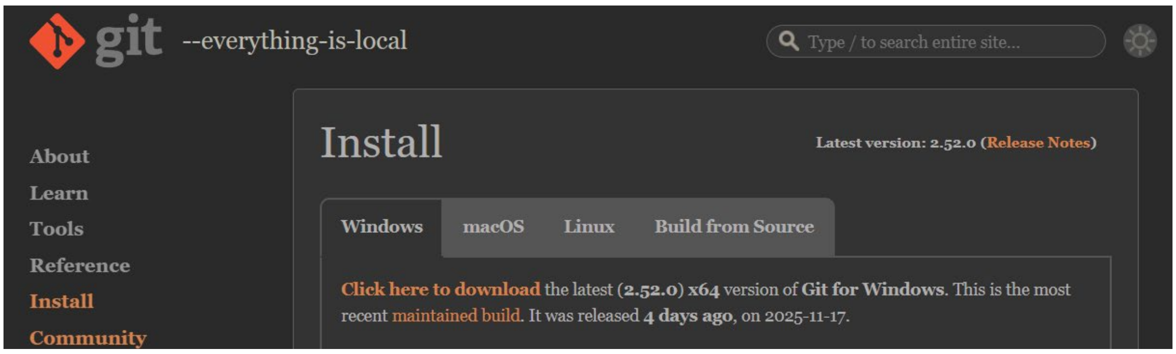
ABOUT THIS PROJECT

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 .

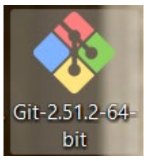

Git
https://git-scm.com › install › windows
Git - Install for Windows
Click here to download the latest (2.52.0) x64 version of Git for Windows. This is the most recent maintained build. It was released 4 days ago, ...



Install

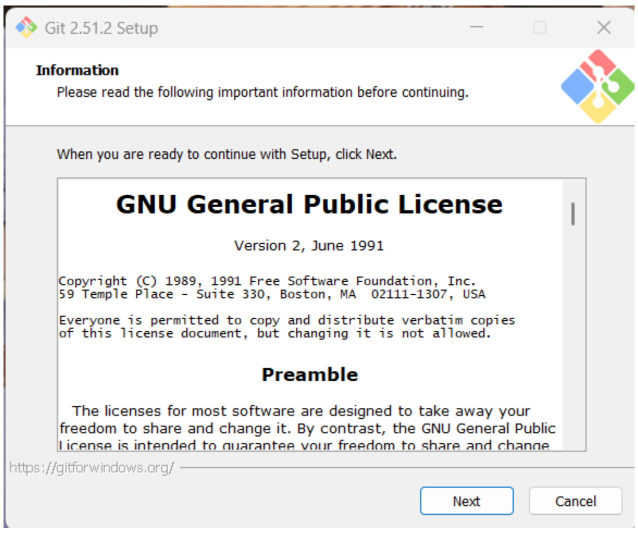
Latest version: 2.52.0 (Release Notes)

Click here to download the latest (2.52.0) x64 version of Git for Windows. This is the most recent maintained build. It was released 4 days ago, on 2025-11-17.



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.



Git 2.51.2 Setup

Information

Please read the following important information before continuing.

When you are ready to continue with Setup, click Next.

GNU General Public License

Version 2, June 1991

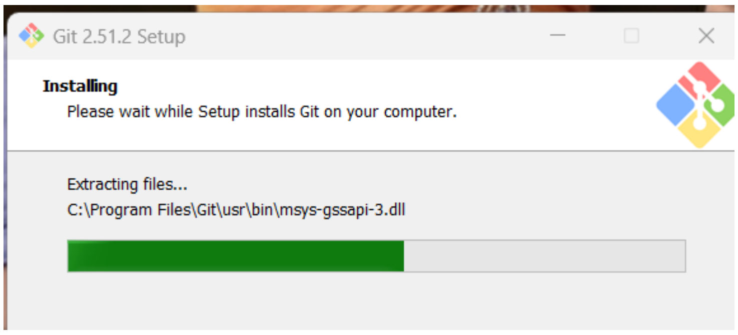
```
Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place - Suite 330, Boston, MA  02111-1307, USA

Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.
```

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change

https://gitforwindows.org/

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

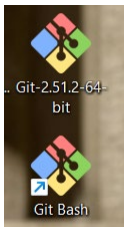


Git 2.51.2 Setup

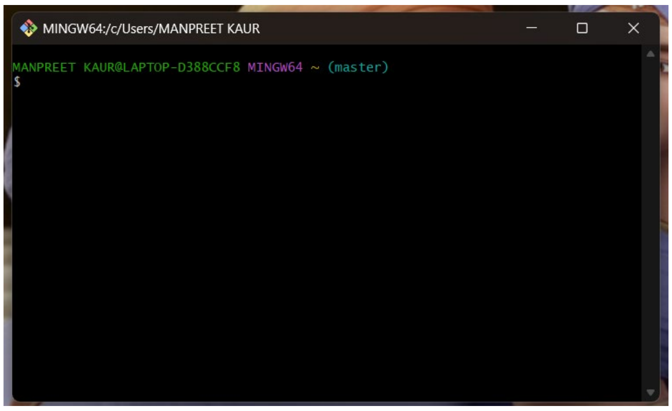
Installing

Please wait while Setup installs Git on your computer.

Extracting files...
C:\Program Files\Git\usr\bin\msys-gssapi-3.dll



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.



MINGW64:/c/Users/MANPREET KAUR

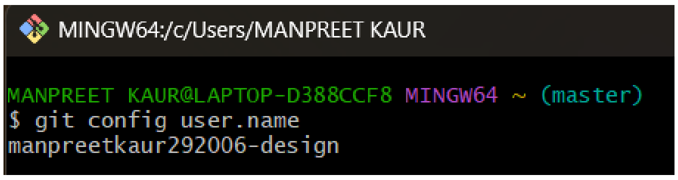
```
MANPREET KAUR@LAPTOP-D388CCF8 MINGW64 ~ (master)
$
```

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.

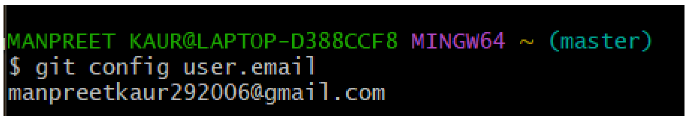


```
MINGW64:/c/Users/MANPREET KAUR

MANPREET KAUR@LAPTOP-D388CCF8 MINGW64 ~ (master)
$ git config user.name
manpreetkaur292006-design
```

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.

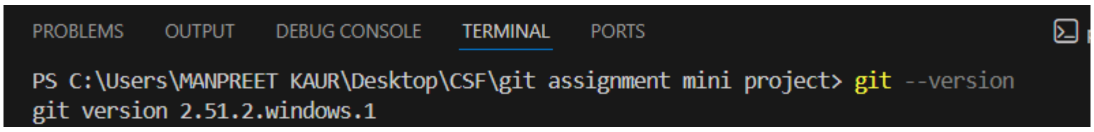


```
MANPREET KAUR@LAPTOP-D388CCF8 MINGW64 ~ (master)
$ git config user.email
manpreetkaur292006@gmail.com
```

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.

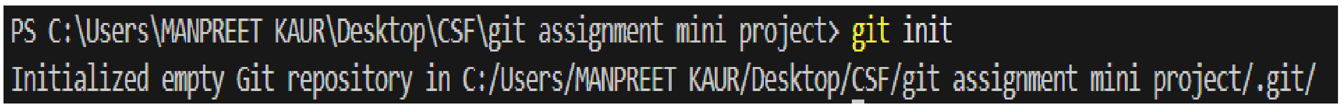


```
PS C:\Users\MANPREET KAUR\Desktop\CSF\git assignment mini project> git --version
git version 2.51.2.windows.1
```

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.

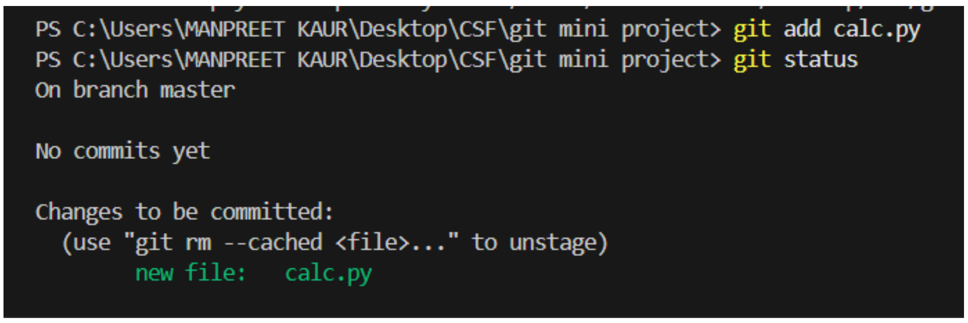


```
PS C:\Users\MANPREET KAUR\Desktop\CSF\git assignment mini project> git init
Initialized empty Git repository in C:/Users/MANPREET KAUR/Desktop/CSF/git assignment mini project/.git/
```

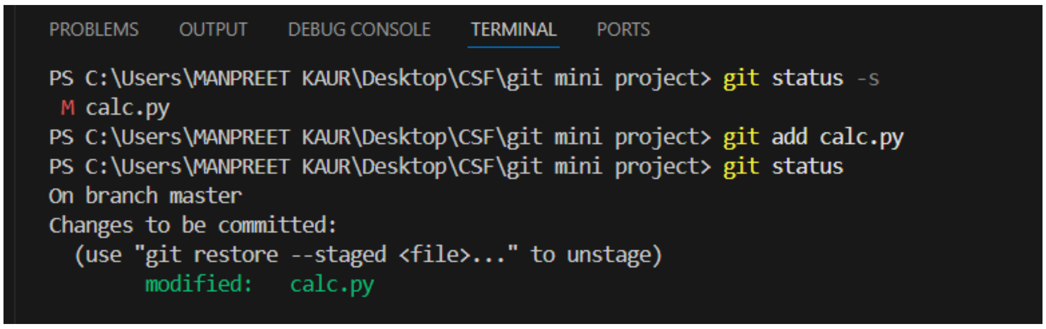
GIT ADD

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.

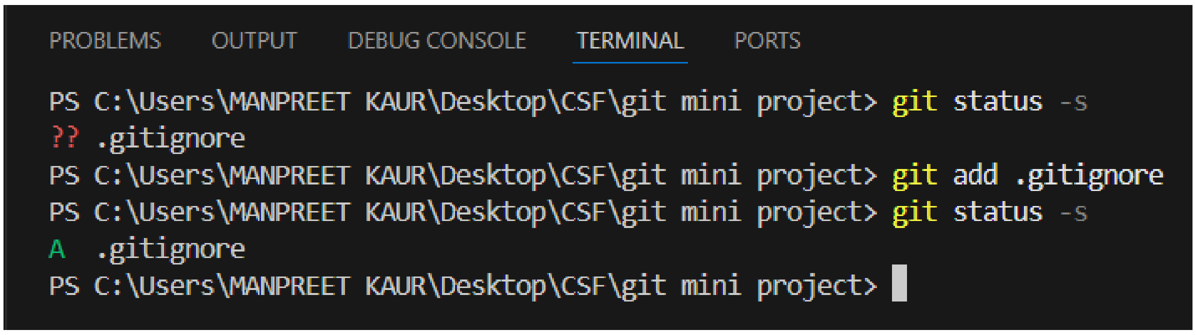
You can see its implementation part in the snapshot given below.



```
PS C:\Users\MANPREET KAUR\Desktop\CSF\git mini project> git add calc.py
PS C:\Users\MANPREET KAUR\Desktop\CSF\git mini project> git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   calc.py
```



```
PS C:\Users\MANPREET KAUR\Desktop\CSF\git mini project> git status -s
 M calc.py
PS C:\Users\MANPREET KAUR\Desktop\CSF\git mini project> git add calc.py
PS C:\Users\MANPREET KAUR\Desktop\CSF\git mini project> git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   calc.py
```



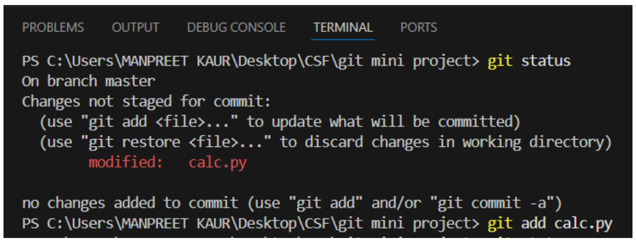
```
PS C:\Users\MANPREET KAUR\Desktop\CSF\git mini project> git status -s
?? .gitignore
PS C:\Users\MANPREET KAUR\Desktop\CSF\git mini project> git add .gitignore
PS C:\Users\MANPREET KAUR\Desktop\CSF\git mini project> git status -s
A  .gitignore
PS C:\Users\MANPREET KAUR\Desktop\CSF\git mini project>
```

GIT STATUS

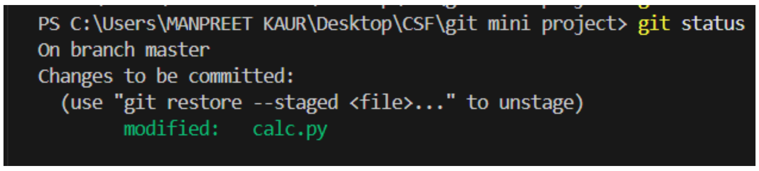
The **git status** command shows **the current state of your Git project**. It tells you which files are ready to be saved (staged), which files have changes but are not yet staged, and which files Git is not tracking. This command helps you understand what work is done and what still needs to be added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

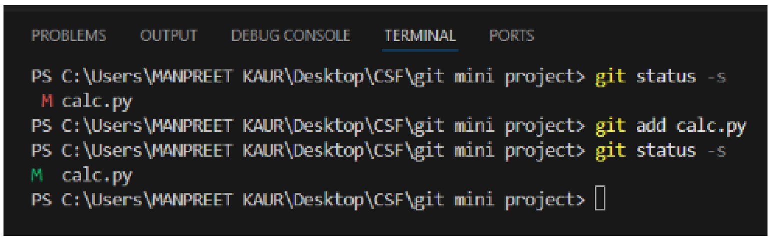
You can see its implementation part in the snapshot given below.



```
PS C:\Users\MANPREET KAUR\Desktop\CSF\git mini project> git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   calc.py

no changes added to commit (use "git add" and/or "git commit -a")
PS C:\Users\MANPREET KAUR\Desktop\CSF\git mini project> git add calc.py
```



```
PS C:\Users\MANPREET KAUR\Desktop\CSF\git mini project> git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   calc.py
```



```
PS C:\Users\MANPREET KAUR\Desktop\CSF\git mini project> git status -s
 M calc.py
PS C:\Users\MANPREET KAUR\Desktop\CSF\git mini project> git add calc.py
PS C:\Users\MANPREET KAUR\Desktop\CSF\git mini project> git status -s
M  calc.py
PS C:\Users\MANPREET KAUR\Desktop\CSF\git mini project>
```

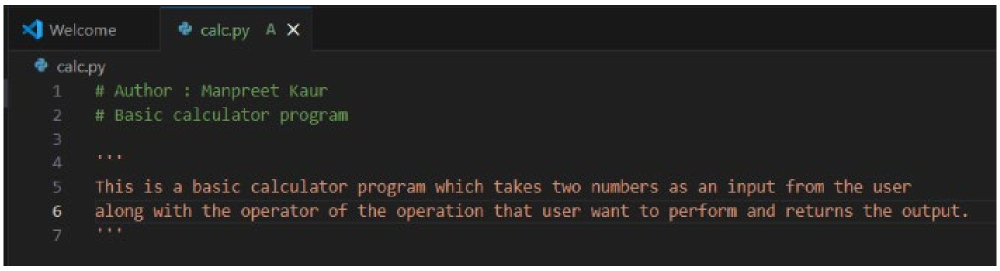
GIT COMMIT

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.

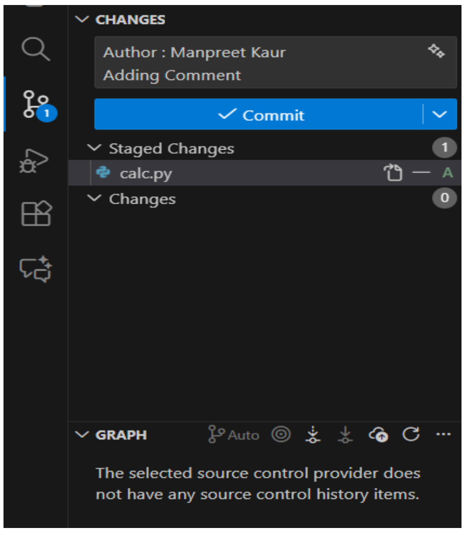
I have listed all the 6 commits that I have done in the snapshots below.

First commit :-

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



```
# Author : Manpreet Kaur
# Basic calculator program

'''
This is a basic calculator program which takes two numbers as an input from the user
along with the operator of the operation that user want to perform and returns the output.
'''
```



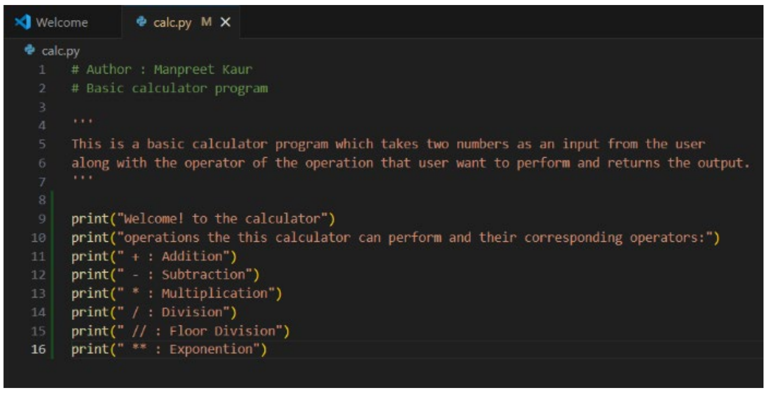
```
Author : Manpreet Kaur
Adding Comment
```

calc.py A

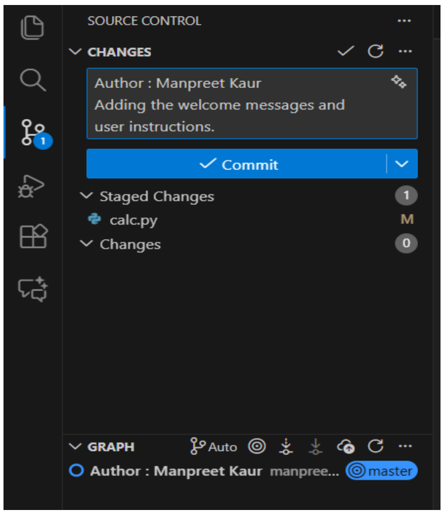
The selected source control provider does not have any source control history items.

Second commit :-

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



```python
# Author : Manpreet Kaur
# Basic calculator program

'''
This is a basic calculator program which takes two numbers as an input from the user
along with the operator of the operation that user want to perform and returns the output.
'''

print("Welcome! to the calculator")
print("operations the this calculator can perform and their corresponding operators:")
print(" + : Addition")
print(" - : Subtraction")
print(" * : Multiplication")
print(" / : Division")
print(" // : Floor Division")
print(" ** : Exponention")
```



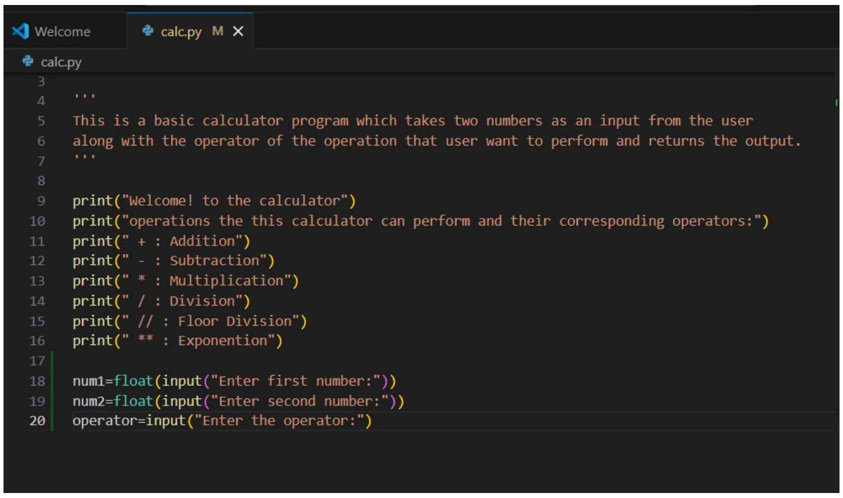
```
Author : Manpreet Kaur
Adding the welcome messages and
user instructions.
```

calc.py M

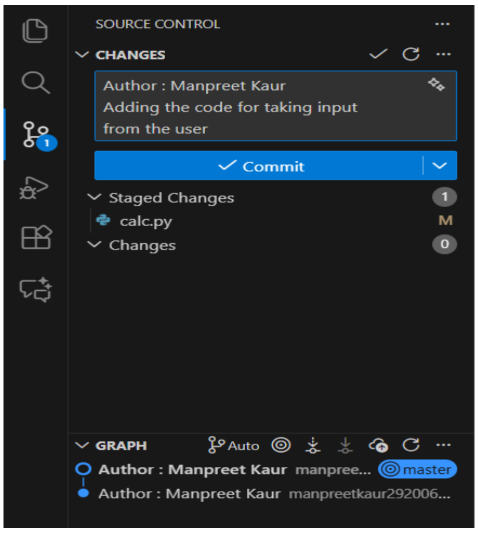
Author : Manpreet Kaur manpree... master

Third commit :-

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



```python
'''
This is a basic calculator program which takes two numbers as an input from the user
along with the operator of the operation that user want to perform and returns the output.
'''

print("Welcome! to the calculator")
print("operations the this calculator can perform and their corresponding operators:")
print(" + : Addition")
print(" - : Subtraction")
print(" * : Multiplication")
print(" / : Division")
print(" // : Floor Division")
print(" ** : Exponention")

num1=float(input("Enter first number:"))
num2=float(input("Enter second number:"))
operator=input("Enter the operator:")
```



Author : Manpreet Kaur
Adding the code for taking input
from the user

SOURCE CONTROL
CHANGES
Commit
Staged Changes 1
calc.py M
Changes 0
GRAPH
Author : Manpreet Kaur manpree... master
Author : Manpreet Kaur manpreetkaur292006...

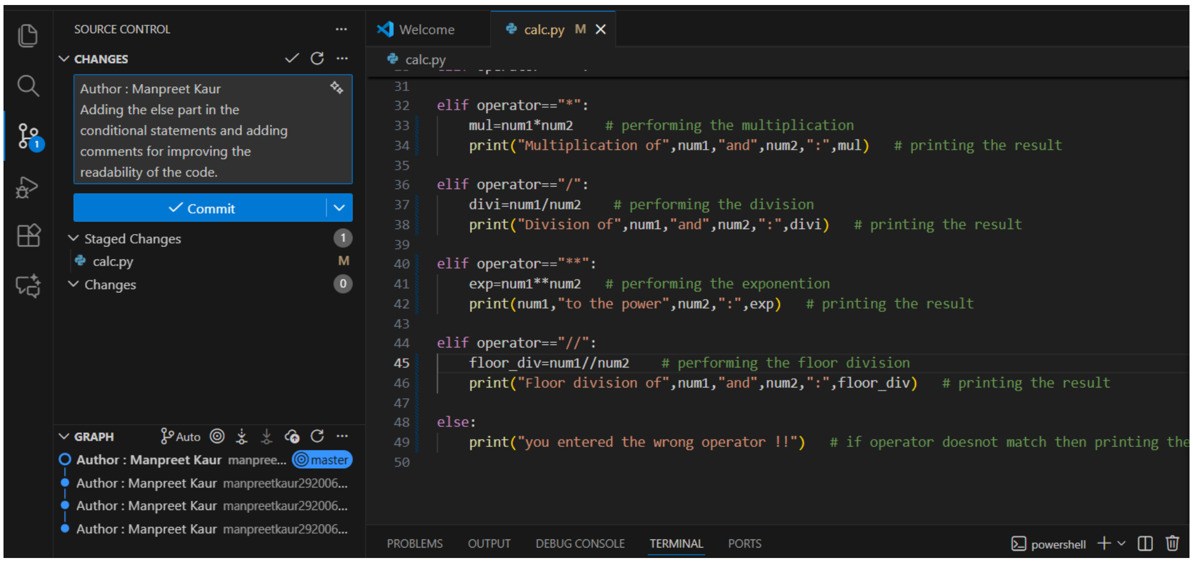
Fourth commit :-

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.



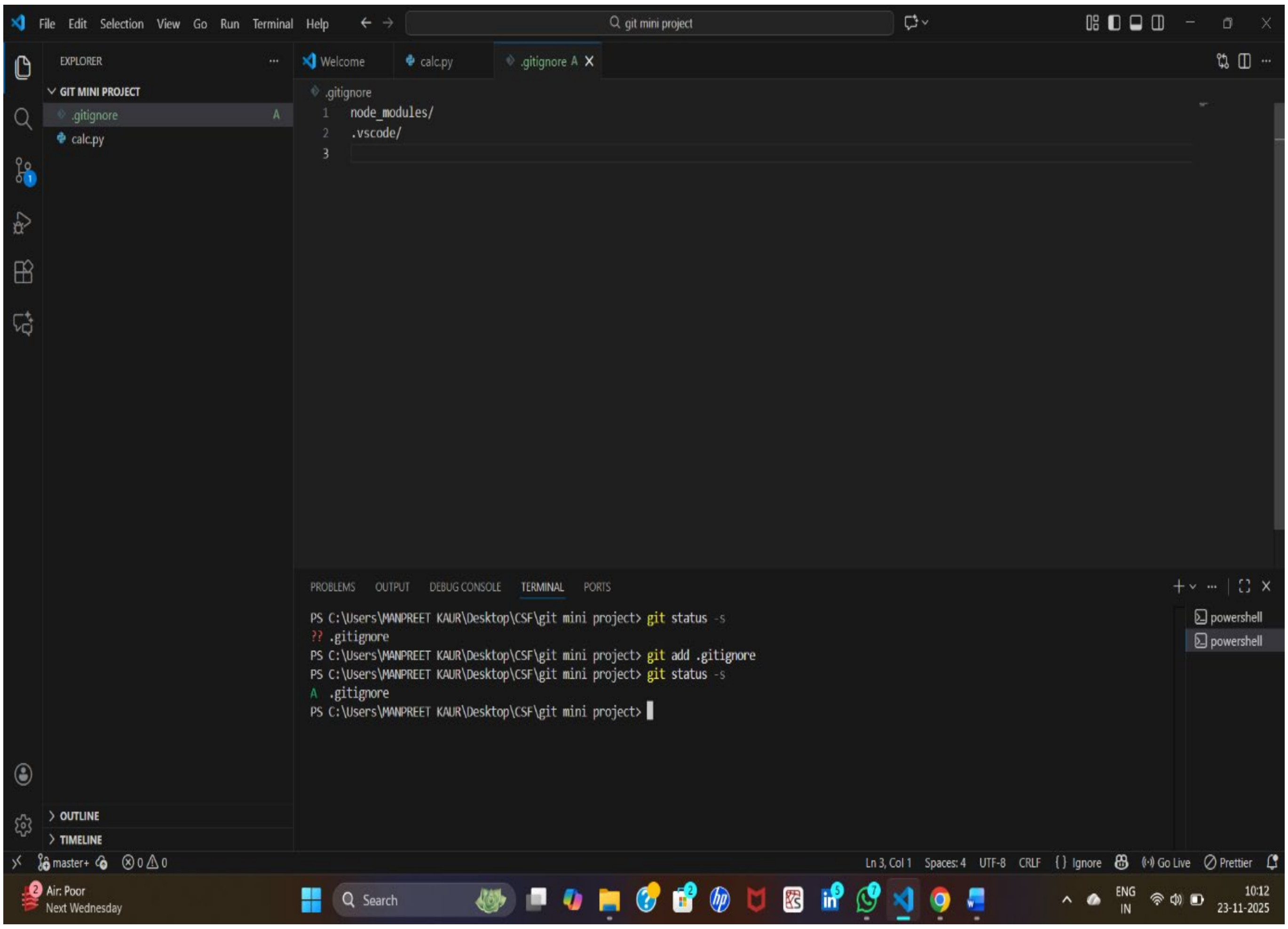
Author : Manpreet Kaur
Adding the else part in the
conditional statements and adding
comments for improving the
readability of the code.

```python

elif operator=="*":
    mul=num1*num2    # performing the multiplication
    print("Multiplication of",num1,"and",num2,":",mul)   # printing the result

elif operator=="/":
    divi=num1/num2    # performing the division
    print("Division of",num1,"and",num2,":",divi)   # printing the result

elif operator=="**":
    exp=num1**num2   # performing the exponention
    print(num1,"to the power",num2,":",exp)   # printing the result

elif operator=="//":
    floor_div=num1//num2    # performing the floor division
    print("Floor division of",num1,"and",num2,":",floor_div)   # printing the result

else:
    print("you entered the wrong operator !!")   # if operator doesnot match then printing the

```

SOURCE CONTROL
CHANGES
Commit
Staged Changes 1
calc.py M
Changes 0
GRAPH
Author : Manpreet Kaur manpree... master
Author : Manpreet Kaur manpreetkaur292006...
Author : Manpreet Kaur manpreetkaur292006...
Author : Manpreet Kaur manpreetkaur292006...
Welcome
calc.py M
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
powershell

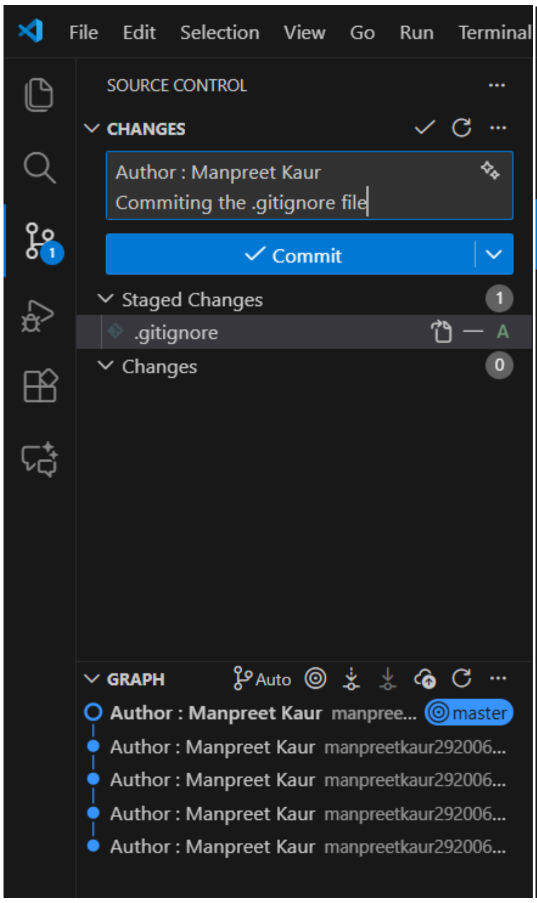
Sixth commit :-

In this commit I have added the .gitignore file.



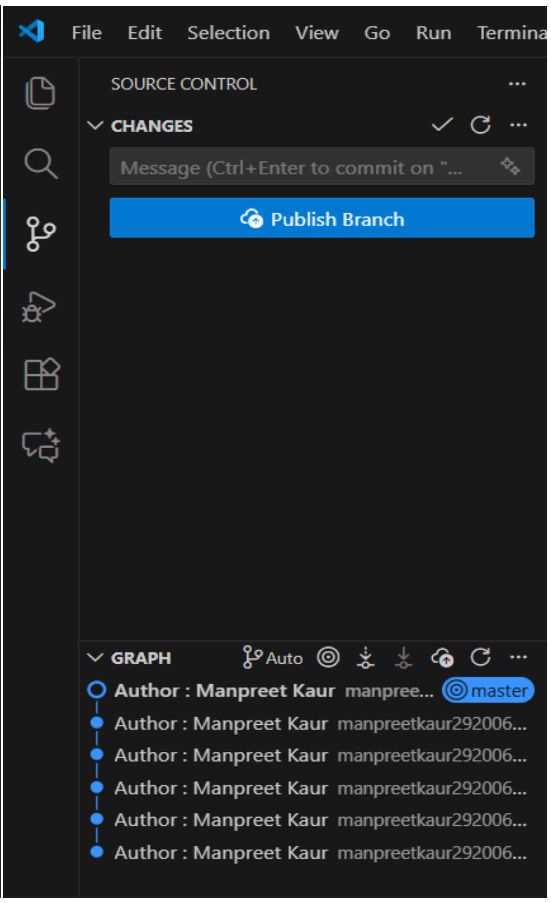
```
node_modules/
.vscode/
```

```
PS C:\Users\MANPREET KAUR\Desktop\CSF\git mini project> git status -s
?? .gitignore
PS C:\Users\MANPREET KAUR\Desktop\CSF\git mini project> git add .gitignore
PS C:\Users\MANPREET KAUR\Desktop\CSF\git mini project> git status -s
A  .gitignore
PS C:\Users\MANPREET KAUR\Desktop\CSF\git mini project>
```



```
Author : Manpreet Kaur
Commiting the .gitignore file
```

SOURCE CONTROL
CHANGES
Commit
Staged Changes 1
.gitignore A
Changes 0
GRAPH
Auto
Author : Manpreet Kaur manpree... master
Author : Manpreet Kaur manpreetkaur292006...
Author : Manpreet Kaur manpreetkaur292006...
Author : Manpreet Kaur manpreetkaur292006...
Author : Manpreet Kaur manpreetkaur292006...

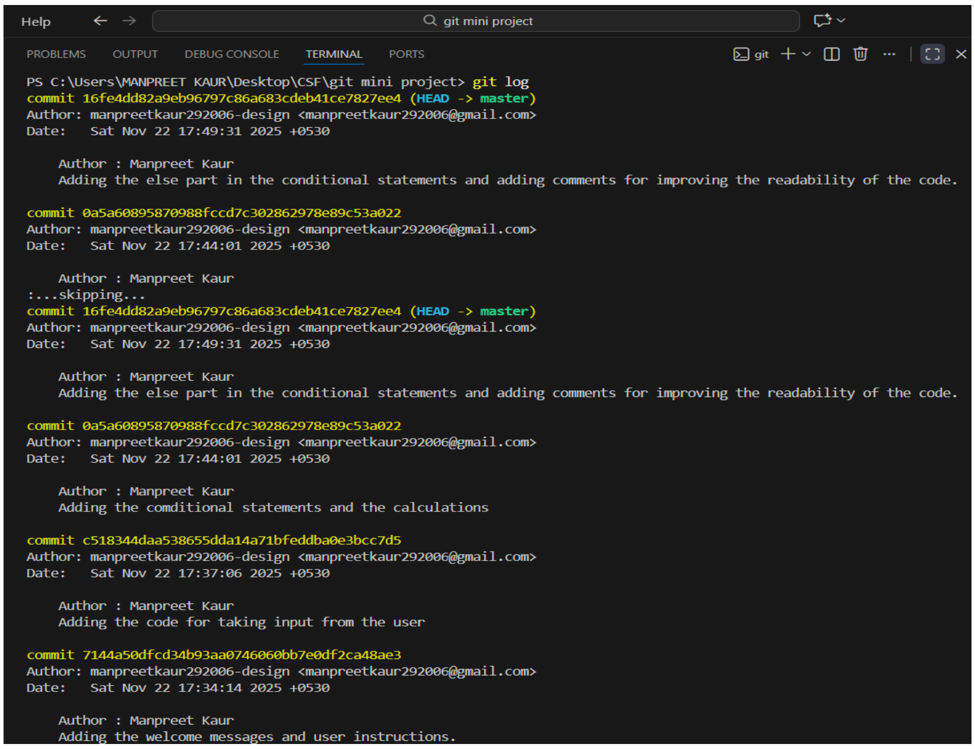


SOURCE CONTROL
CHANGES
Message (Ctrl+Enter to commit on "...
Publish Branch
GRAPH
Auto
Author : Manpreet Kaur manpree... master
Author : Manpreet Kaur manpreetkaur292006...
Author : Manpreet Kaur manpreetkaur292006...
Author : Manpreet Kaur manpreetkaur292006...
Author : Manpreet Kaur manpreetkaur292006...
Author : Manpreet Kaur manpreetkaur292006...

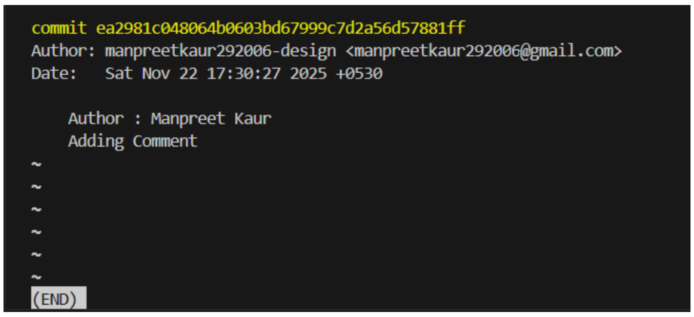
GIT LOG

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.



```
PS C:\Users\MANPREET KAUR\Desktop\CSF\git mini project> git log
commit 16fe4dd82a9eb96797c86a683cdeb41ce7827ee4 (HEAD -> master)
Author: manpreetkaur292006-design <manpreetkaur292006@gmail.com>
Date:   Sat Nov 22 17:49:31 2025 +0530

    Author : Manpreet Kaur
    Adding the else part in the conditional statements and adding comments for improving the readability of the code.

commit 0a5a60895870988fccd7c302862978e89c53a022
Author: manpreetkaur292006-design <manpreetkaur292006@gmail.com>
Date:   Sat Nov 22 17:44:01 2025 +0530

    Author : Manpreet Kaur
:...skipping...
commit 16fe4dd82a9eb96797c86a683cdeb41ce7827ee4 (HEAD -> master)
Author: manpreetkaur292006-design <manpreetkaur292006@gmail.com>
Date:   Sat Nov 22 17:49:31 2025 +0530

    Author : Manpreet Kaur
    Adding the else part in the conditional statements and adding comments for improving the readability of the code.

commit 0a5a60895870988fccd7c302862978e89c53a022
Author: manpreetkaur292006-design <manpreetkaur292006@gmail.com>
Date:   Sat Nov 22 17:44:01 2025 +0530

    Author : Manpreet Kaur
    Adding the comditional statements and the calculations

commit c518344daa538655dda14a71bfeddba0e3bcc7d5
Author: manpreetkaur292006-design <manpreetkaur292006@gmail.com>
Date:   Sat Nov 22 17:37:06 2025 +0530

    Author : Manpreet Kaur
    Adding the code for taking input from the user

commit 7144a50dfcd34b93aa0746060bb7e0df2ca48ae3
Author: manpreetkaur292006-design <manpreetkaur292006@gmail.com>
Date:   Sat Nov 22 17:34:14 2025 +0530

    Author : Manpreet Kaur
    Adding the welcome messages and user instructions.
```

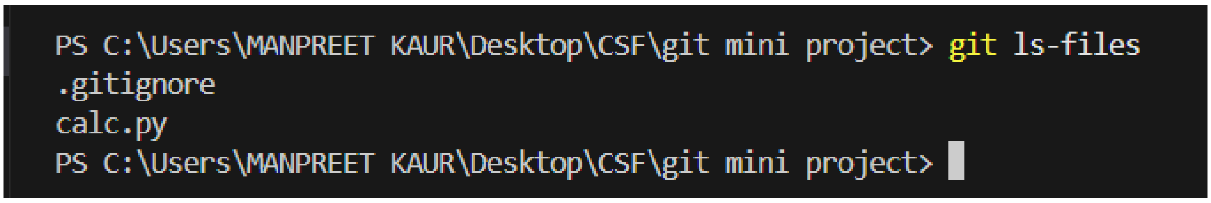


```
commit ea2981c048064b0603bd67999c7d2a56d57881ff
Author: manpreetkaur292006-design <manpreetkaur292006@gmail.com>
Date:   Sat Nov 22 17:30:27 2025 +0530

    Author : Manpreet Kaur
    Adding Comment
~
~
~
~
~
~
(END)
```

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.



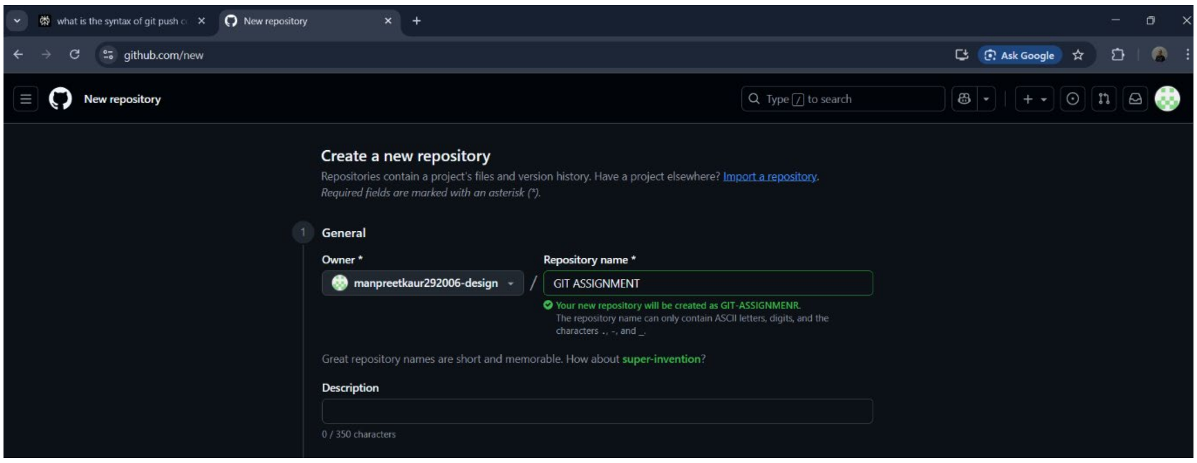
```
PS C:\Users\MANPREET KAUR\Desktop\CSF\git mini project> git ls-files
.gitignore
calc.py
PS C:\Users\MANPREET KAUR\Desktop\CSF\git mini project> 
```

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.



Create a new repository

Repositories contain a project's files and version history. Have a project elsewhere? Import a repository.

Required fields are marked with an asterisk (*).

General

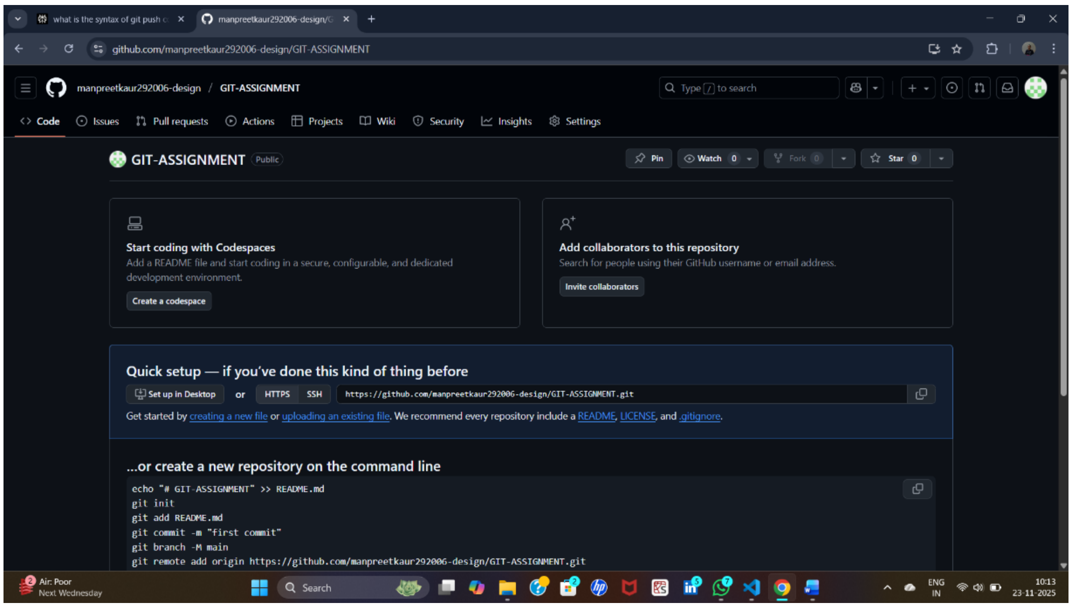
Owner *: manpreetkaur292006-design / Repository name *: GIT ASSIGNMENT

Your new repository will be created as GIT-ASSIGNMENR.
The repository name can only contain ASCII letters, digits, and the characters ., -, and _.

Great repository names are short and memorable. How about super-invention?

Description

0 / 350 characters



GIT-ASSIGNMENT Public

Start coding with Codespaces
Add a README file and start coding in a secure, configurable, and dedicated development environment.

Add collaborators to this repository
Search for people using their GitHub username or email address.

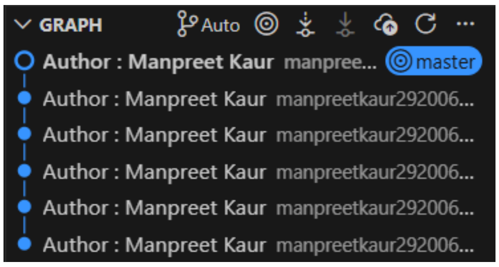
Quick setup — if you've done this kind of thing before

https://github.com/manpreetkaur292006-design/GIT-ASSIGNMENT.git

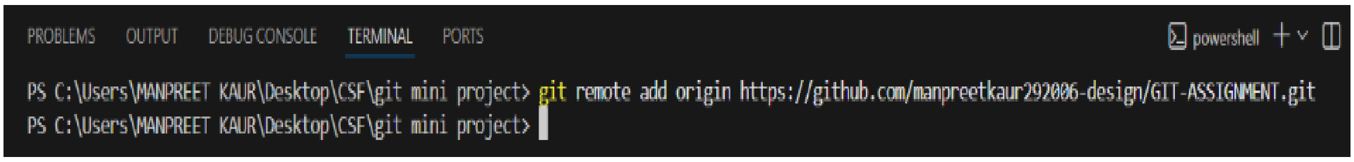
Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# GIT-ASSIGNMENT" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/manpreetkaur292006-design/GIT-ASSIGNMENT.git
```



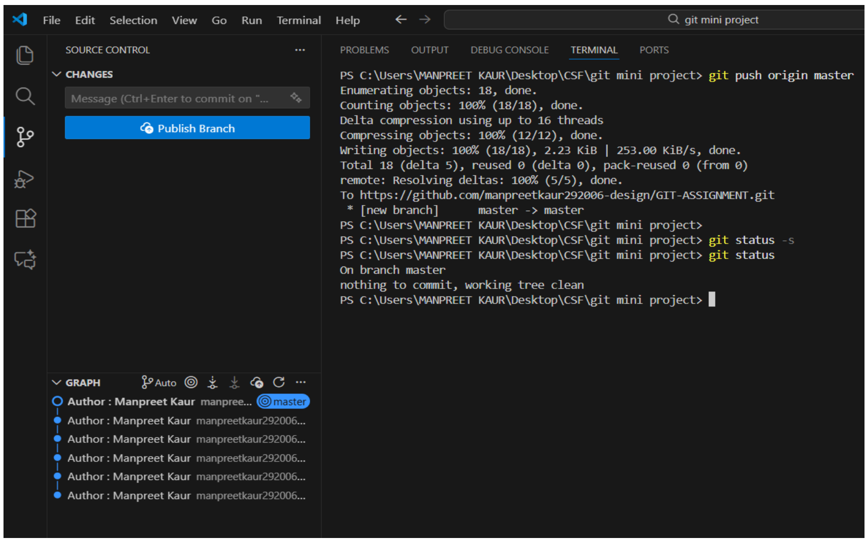
```
GRAPH  Auto
Author : Manpreet Kaur  manpree...  master
Author : Manpreet Kaur  manpreetkaur292006...
Author : Manpreet Kaur  manpreetkaur292006...
Author : Manpreet Kaur  manpreetkaur292006...
Author : Manpreet Kaur  manpreetkaur292006...
Author : Manpreet Kaur  manpreetkaur292006...
```



```
PS C:\Users\MANPREET KAUR\Desktop\CSF\git mini project> git remote add origin https://github.com/manpreetkaur292006-design/GIT-ASSIGNMENT.git
PS C:\Users\MANPREET KAUR\Desktop\CSF\git mini project>
```

GIT PUSH ORIGIN MASTER

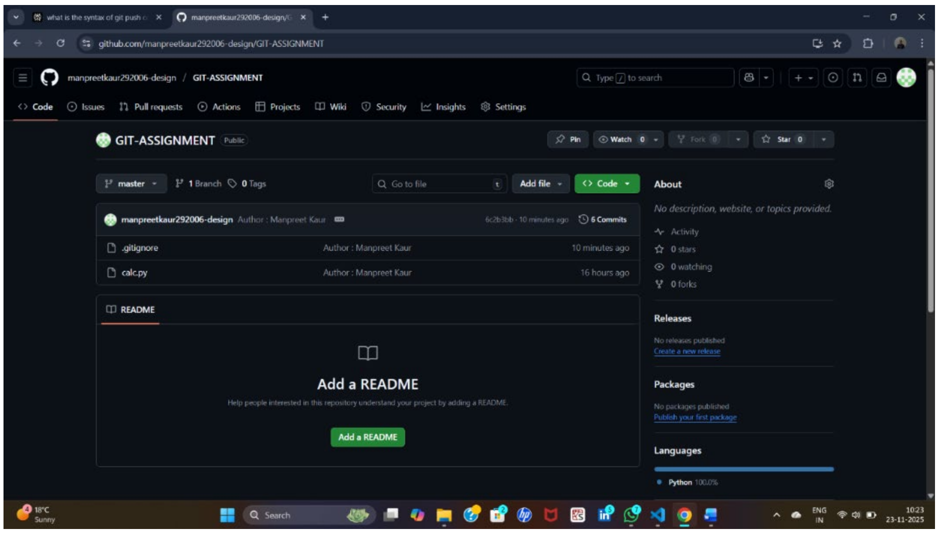
The command **git push origin master** sends your **local master branch changes to the remote repository called origin**. It updates the master branch on the remote with your latest commits from your local computer. This is commonly used to share your main development work with others or back it up remotely.



File Edit Selection View Go Run Terminal Help
git mini project
SOURCE CONTROL
CHANGES
Message (Ctrl+Enter to commit on "...
Publish Branch
GRAPH
Auto
Author : Manpreet Kaur manpree... master
Author : Manpreet Kaur manpreetkaur292006...
Author : Manpreet Kaur manpreetkaur292006...
Author : Manpreet Kaur manpreetkaur292006...
Author : Manpreet Kaur manpreetkaur292006...
Author : Manpreet Kaur manpreetkaur292006...
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS

```
PS C:\Users\MANPREET KAUR\Desktop\CSF\git mini project> git push origin master
Enumerating objects: 18, done.
Counting objects: 100% (18/18), done.
Delta compression using up to 16 threads
Compressing objects: 100% (12/12), done.
Writing objects: 100% (18/18), 2.23 KiB | 253.00 KiB/s, done.
Total 18 (delta 5), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (5/5), done.
To https://github.com/manpreetkaur292006-design/GIT-ASSIGNMENT.git
 * [new branch]      master -> master
PS C:\Users\MANPREET KAUR\Desktop\CSF\git mini project>
PS C:\Users\MANPREET KAUR\Desktop\CSF\git mini project> git status -s
PS C:\Users\MANPREET KAUR\Desktop\CSF\git mini project> git status
On branch master
nothing to commit, working tree clean
PS C:\Users\MANPREET KAUR\Desktop\CSF\git mini project>
```

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.

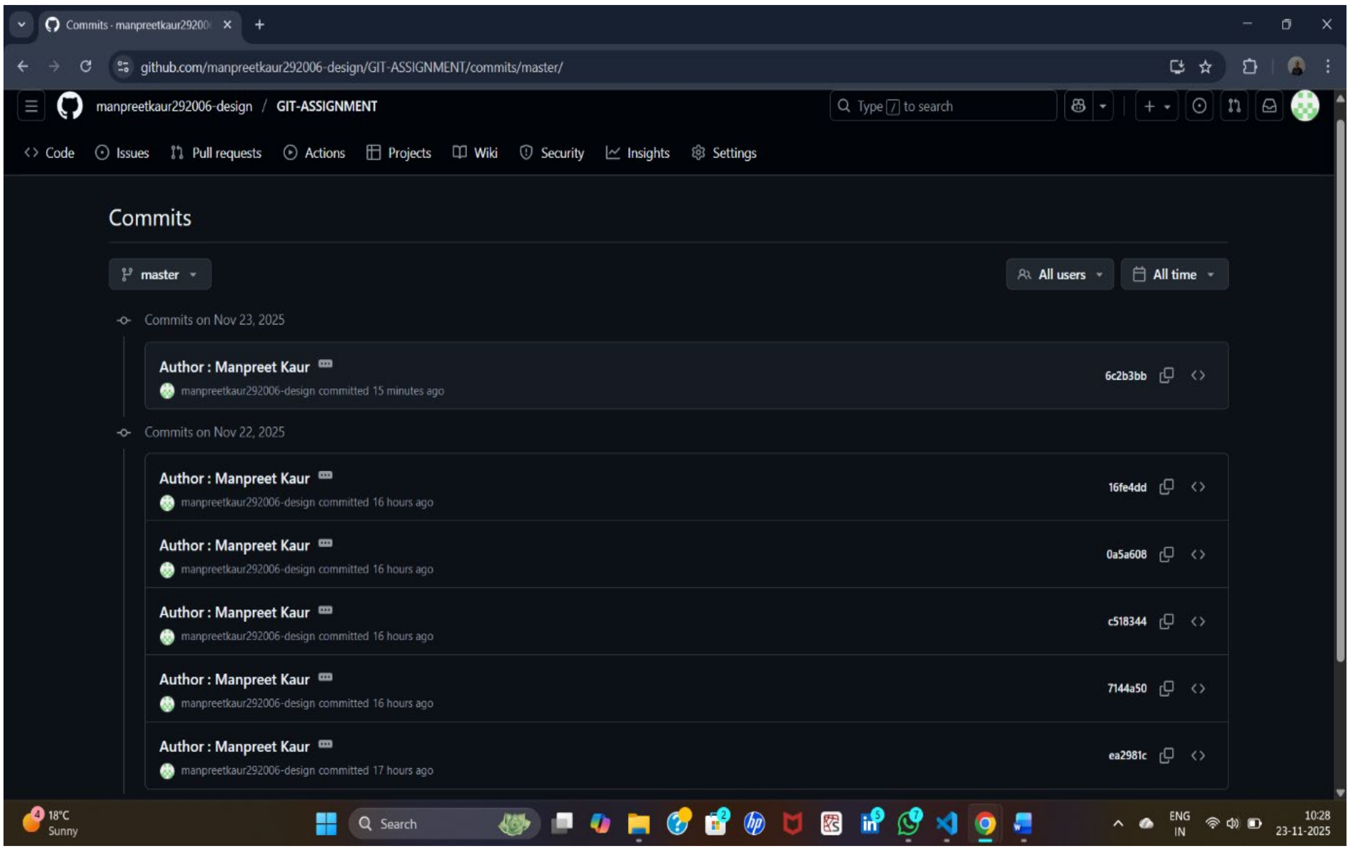


manpreetkaur292006-design Author : Manpreet Kaur	6c2b3bb · 10 minutes ago	6 Commits
.gitignore	Author : Manpreet Kaur	10 minutes ago
calc.py	Author : Manpreet Kaur	16 hours ago

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



github.com/manpreetkaur292006-design/GIT-ASSIGNMENT/commits/master/

manpreetkaur292006-design / GIT-ASSIGNMENT

Commits

master

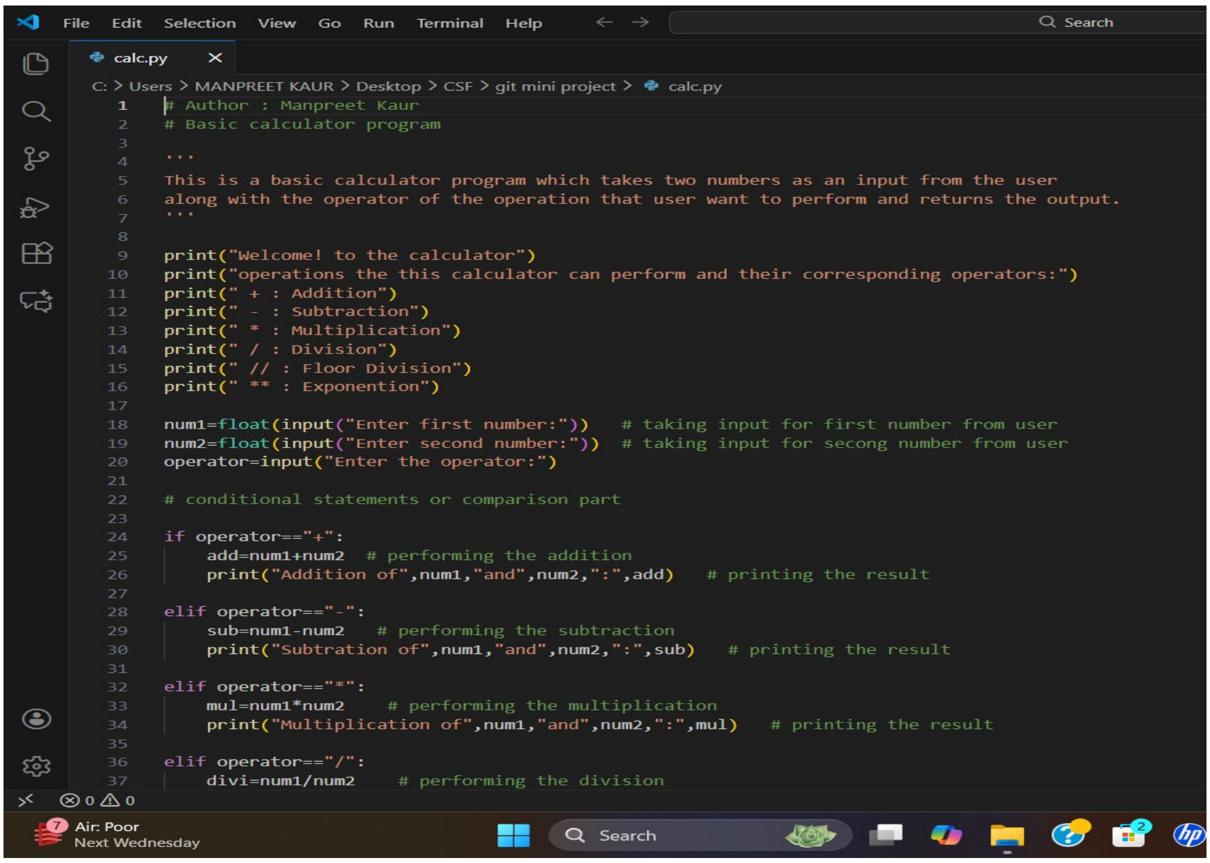
Commits on Nov 23, 2025

Author : Manpreet Kaur manpreetkaur292006-design committed 15 minutes ago	6c2b3bb

Commits on Nov 22, 2025

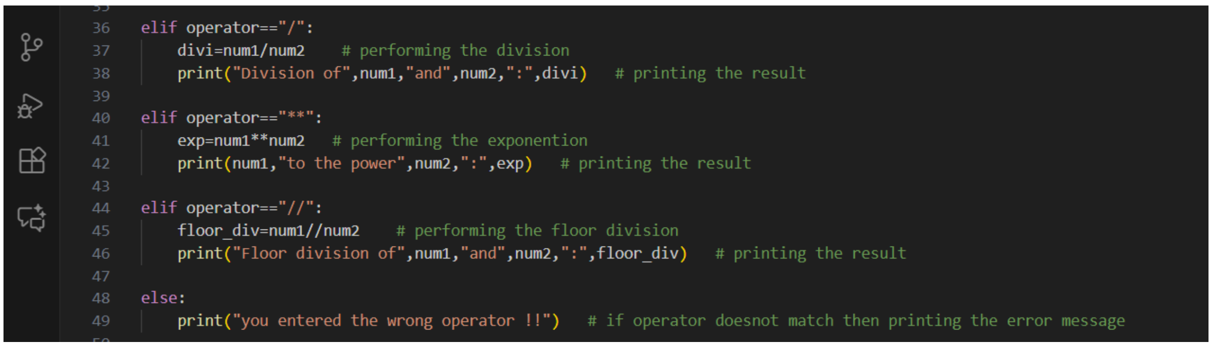
Author : Manpreet Kaur manpreetkaur292006-design committed 16 hours ago	16fe4dd
Author : Manpreet Kaur manpreetkaur292006-design committed 16 hours ago	0a5a608
Author : Manpreet Kaur manpreetkaur292006-design committed 16 hours ago	c518344
Author : Manpreet Kaur manpreetkaur292006-design committed 16 hours ago	7144a50
Author : Manpreet Kaur manpreetkaur292006-design committed 17 hours ago	ea2981c

PYTHON CODE



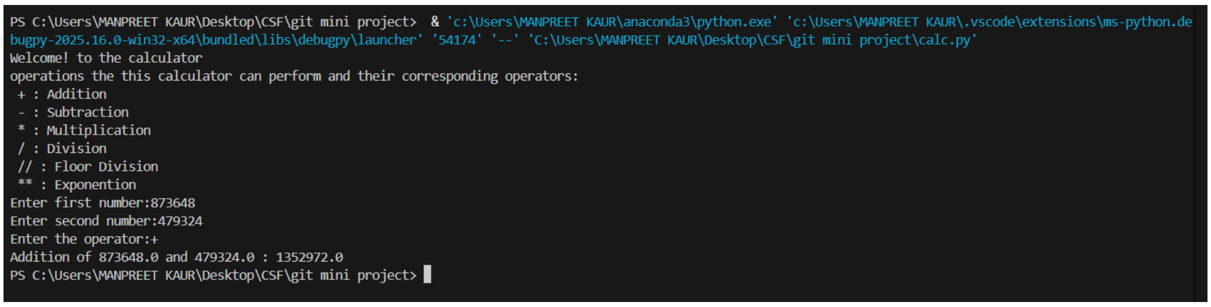
C: > Users > MANPREET KAUR > Desktop > CSF > git mini project > calc.py

```python
# Author : Manpreet Kaur
# Basic calculator program

'''
This is a basic calculator program which takes two numbers as an input from the user
along with the operator of the operation that user want to perform and returns the output.
'''

print("Welcome! to the calculator")
print("operations the this calculator can perform and their corresponding operators:")
print(" + : Addition")
print(" - : Subtraction")
print(" * : Multiplication")
print(" / : Division")
print(" // : Floor Division")
print(" ** : Exponention")

num1=float(input("Enter first number:"))   # taking input for first number from user
num2=float(input("Enter second number:"))  # taking input for secong number from user
operator=input("Enter the operator:")

# conditional statements or comparison part

if operator=="+":
    add=num1+num2  # performing the addition
    print("Addition of",num1,"and",num2,":",add)   # printing the result

elif operator=="-":
    sub=num1-num2   # performing the subtraction
    print("Subtration of",num1,"and",num2,":",sub)   # printing the result

elif operator=="*":
    mul=num1*num2    # performing the multiplication
    print("Multiplication of",num1,"and",num2,":",mul)   # printing the result

elif operator=="/":
    divi=num1/num2    # performing the division
```



```
    elif operator=="/":
        divi=num1/num2    # performing the division
        print("Division of",num1,"and",num2,":",divi)    # printing the result

    elif operator=="**":
        exp=num1**num2    # performing the exponention
        print(num1,"to the power",num2,":",exp)    # printing the result

    elif operator=="//":
        floor_div=num1//num2    # performing the floor division
        print("Floor division of",num1,"and",num2,":",floor_div)    # printing the result

    else:
        print("you entered the wrong operator !!")    # if operator doesnot match then printing the error message
```

OUTPUT



```
PS C:\Users\MANPREET KAUR\Desktop\CSF\git mini project>  & 'c:\Users\MANPREET KAUR\anaconda3\python.exe' 'c:\Users\MANPREET KAUR\.vscode\extensions\ms-python.de
bugpy-2025.16.0-win32-x64\bundled\libs\debugpy\launcher' '54174' '--' 'C:\Users\MANPREET KAUR\Desktop\CSF\git mini project\calc.py'
Welcome! to the calculator
operations the this calculator can perform and their corresponding operators:
 + : Addition
 - : Subtraction
 * : Multiplication
 / : Division
 // : Floor Division
 ** : Exponention
Enter first number:873648
Enter second number:479324
Enter the operator:+
Addition of 873648.0 and 479324.0 : 1352972.0
PS C:\Users\MANPREET KAUR\Desktop\CSF\git mini project> 
```

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.