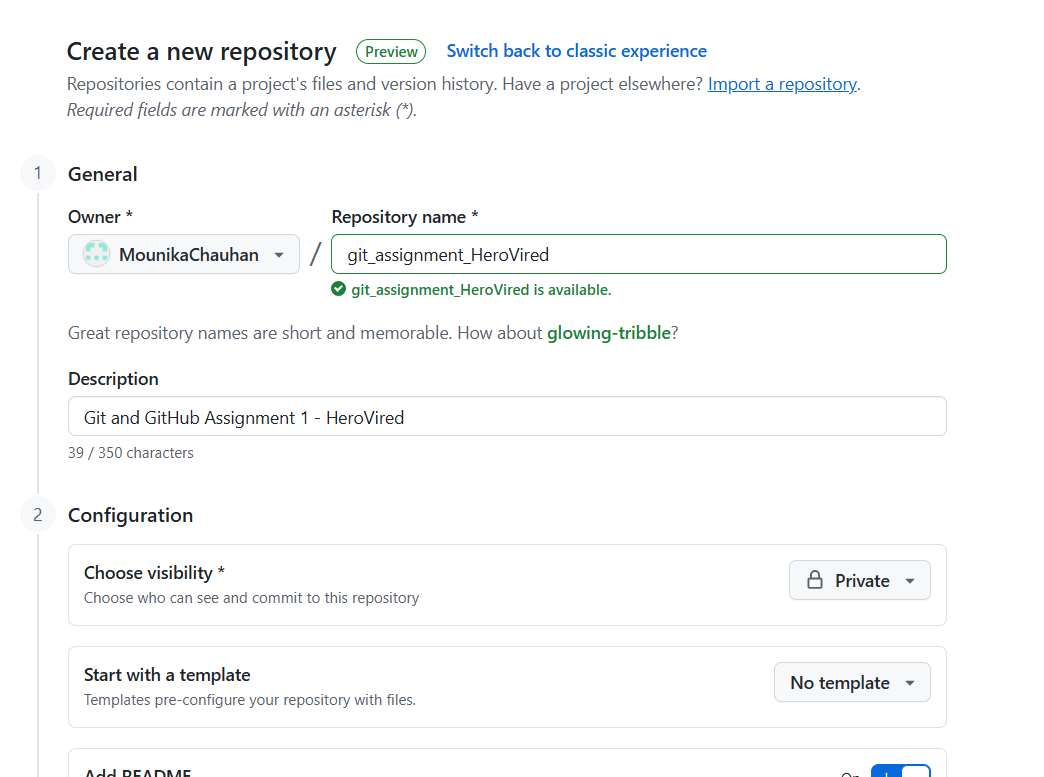
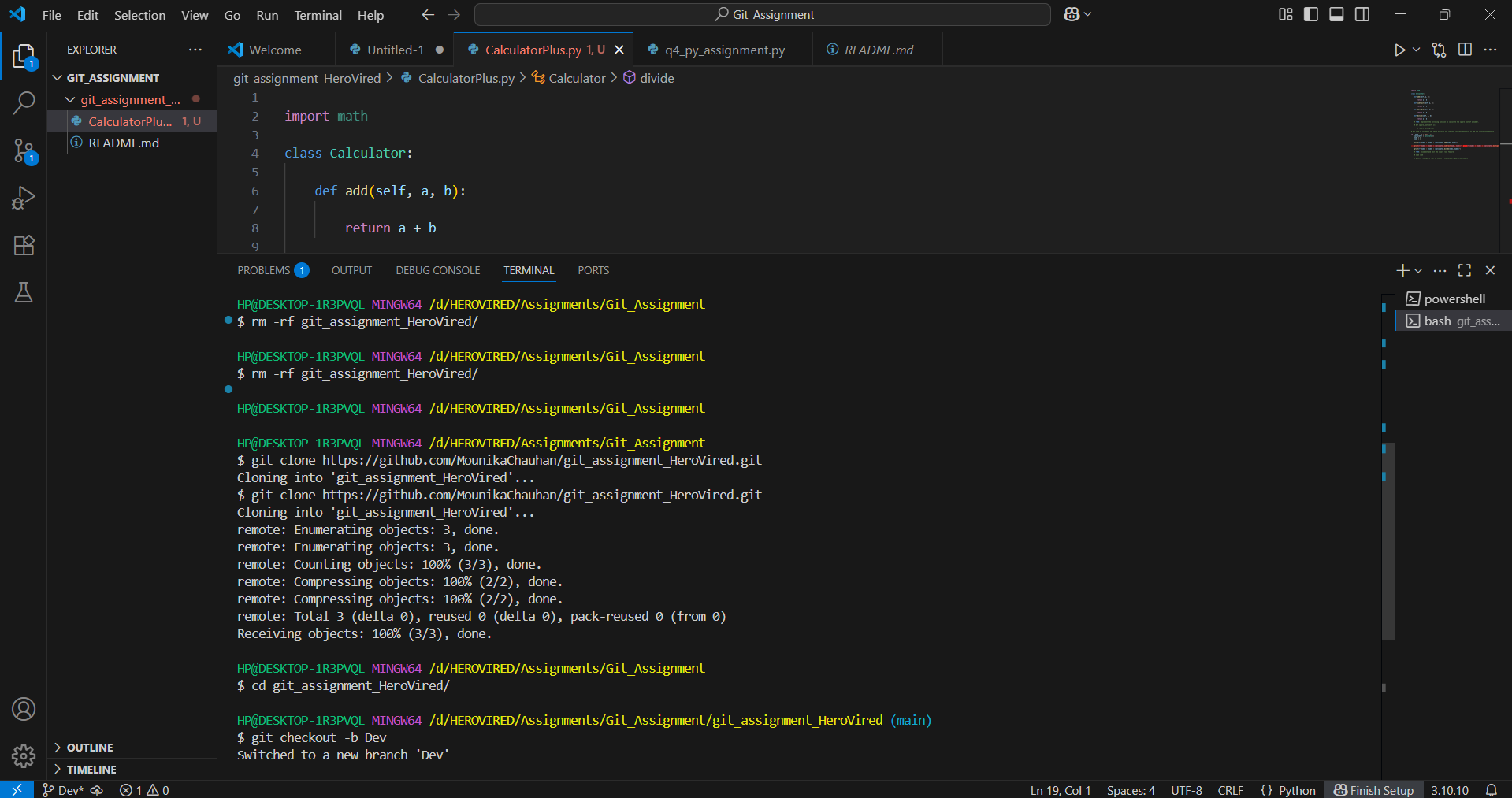
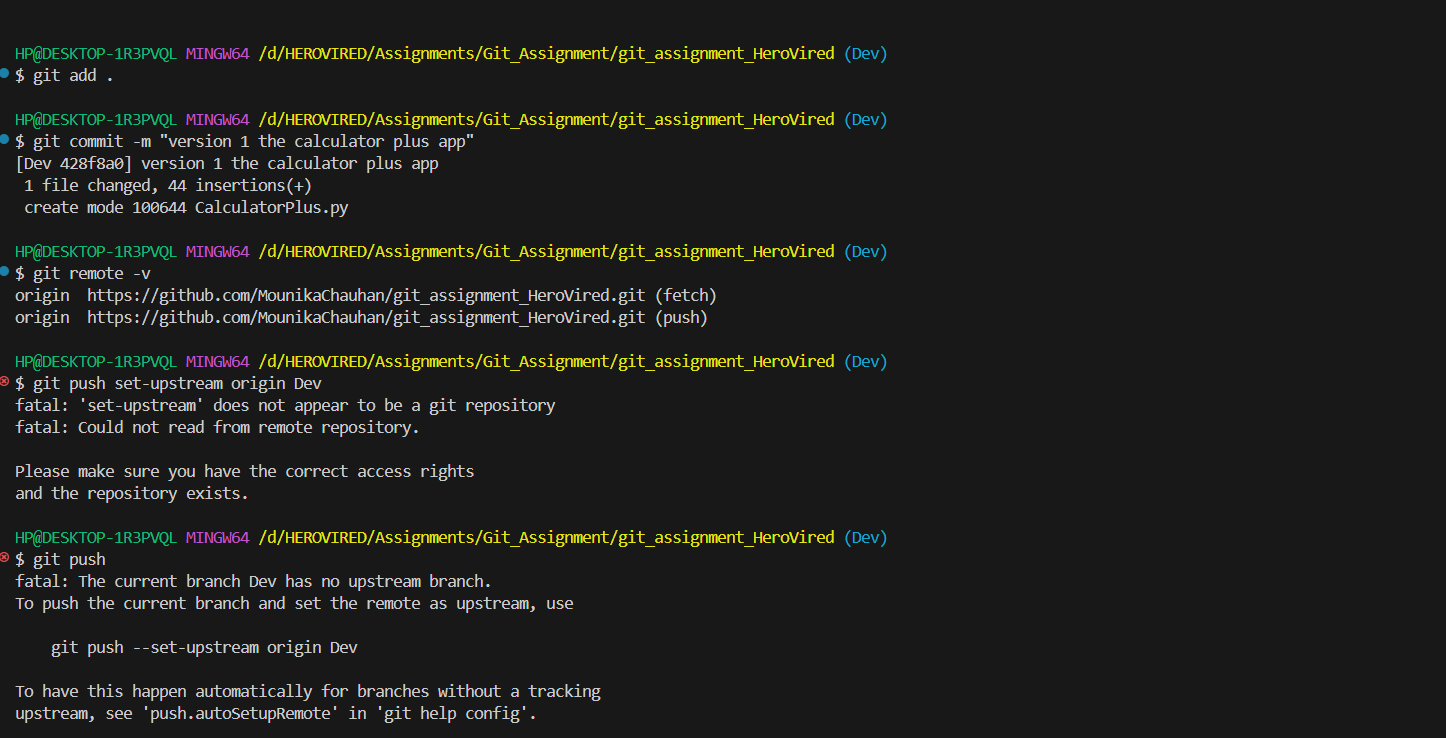
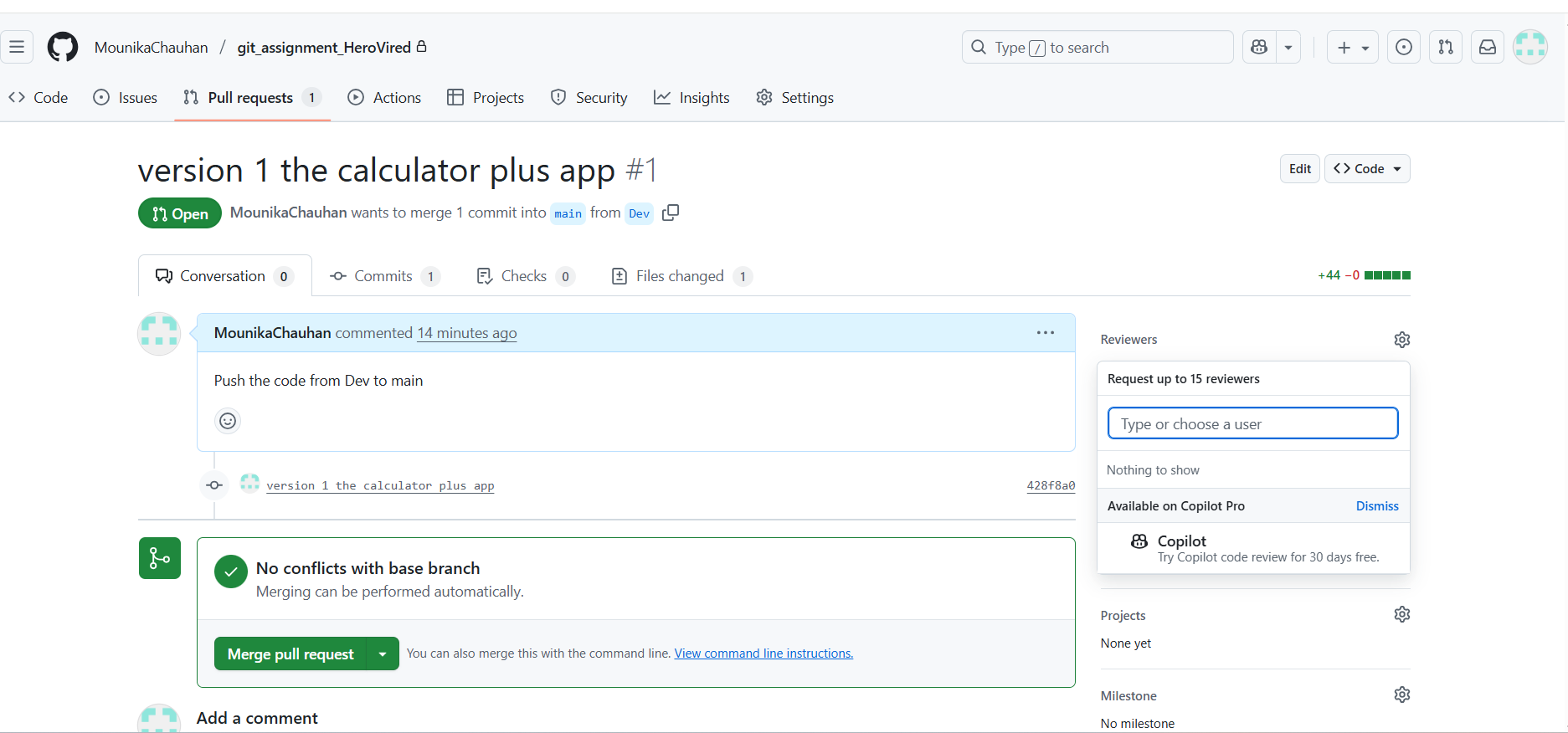
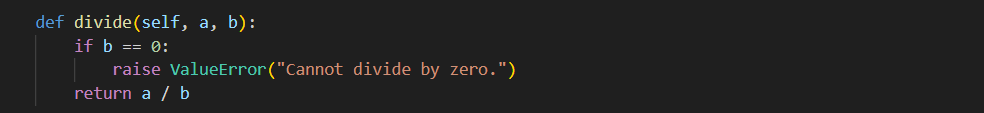
Q1: You are part of a development team working on a Python application called "CalculatorPlus." The application provides basic arithmetic operations, such as addition, subtraction, multiplication, and division. Your task is to implement a new feature that adds support for calculating the square root of a number.

Step 1: Create a repo with private permission and add a readme file to update the steps followed in taks.

  
  
step2: As shown in the below screenshot, follow the below steps  
a. Clone the github in your local by opening the terminal   
b. in the clone repos, go the path in terminal and run below commands  
git checkout –b “Dev” #to create a new local branch   
c. create a new python file and add the given code in the file   
  


Step3 : Once the code it add to the Dev branch, we need to push the code to Remote Dev branch  
a. git add . #to add the files for commit  
b. git commit -m "version 1 the calculator plus app" #committing the files with commit message  
c. git push –set-upstream origin Dev #push the committed files to the specific branch “Dev” as the branch is present in local but not in remote we need to use the –set-upstream



Step4: Create a PR from Dev to main and merge the code  
To add anyone of your class member as reviewer, we need to add them as collaborator   
Go to the Settings -> Collaborators and add your classmate github\_id  
Go to the GitHub and create a PR as shown in the screenshot and reviewer.  
  
  
  
Once the reviewer approves the PR, your main branch would be update with the pushed changes in Dev  
  
Once the code is merged to main create release v1,   
Go to GitHub → Go to **Releases** → **New Release** → Tag version v1.0 → Publish release.  
  
Ste5: Feature branch code  
a. In your local create a branch called “feature/sqrt”  
git checkout –b “feature/sqrt”  
b. uncomment the square root code and save the code.   
Run the python code to check if there are any errors.  
now run the commands to push the code to feature/sqrt  
git add .  
git commit -m "feature branch to add the sqrt code"  
git push --set-upstream origin feature/sqrt  
  
Step 6: Bug Fix   
As the division by 0 is not possible, we need to fix this bug from Dev  
Update the code in the python file and save the file   
  
git checkout Dev  
git add .

git commit -m "Fix: Added divide by zero check"

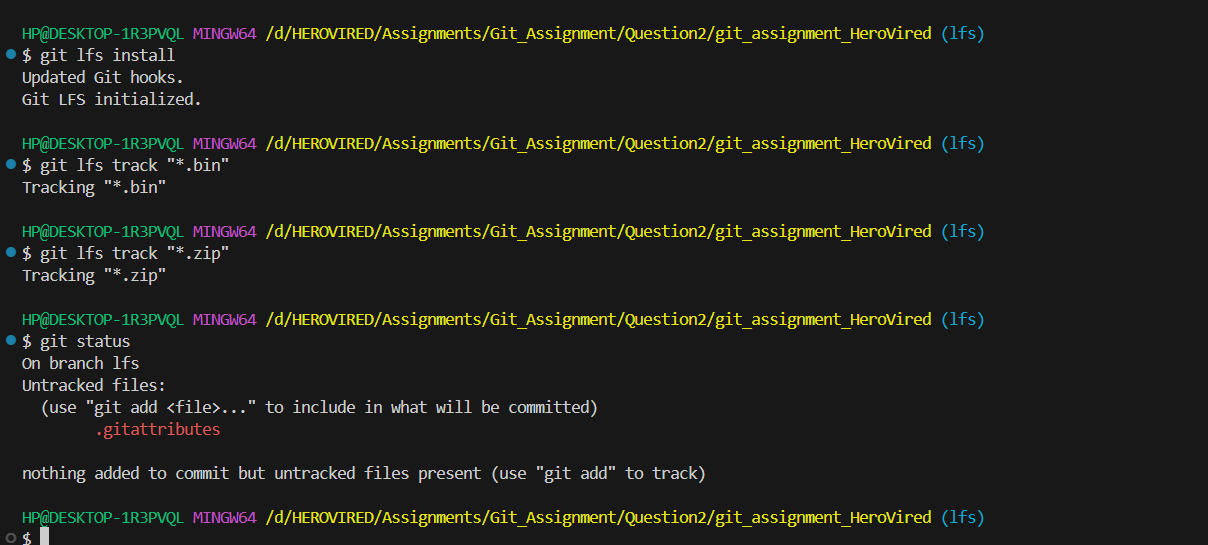
git push origin Dev  
  
now the Dev branch is update with the bug fix  
  
Step 7. Update the feature branch with Dev changes  
git checkout feature/sqrt

git pull origin dev #this will open the editor file for the merge changes, save the changes push the code to feature branch

git push origin feature/sqrt  
  
Step8: Create a pull request from feature/sqrt to main and add classmate to review,  
Once approved merge the PR  
as my classmates are not available for approval, I haven’t added anyone for review  
  
Step 9: Once the changes are merged, create a new release v2.0  
Go to GitHub → Go to Releases → New Release → Tag version v2.0 → Publish release  
  
Q2: For a project that deals with large binary files, integrate Git LFS (Large File Storage) to handle these files efficiently. Demonstrate how to add, commit, and push binary files to the repository, ensuring they are tracked by Git LFS correctly. Clone the repository on another machine to verify that the binary files are downloaded correctly.

In the repository ‘git\_assignment\_HeroVired’, create a branch ‘lfs’. Upload any large file whose size is over ‘200mb’ and try to push this file into the repository.

To perform this, clone the repo into a new folder   
  
Step 1: Initialize the LFS   
Command : git lfs install  
output:   
Updated Git hooks.  
Git LFS initialized.  
  
Step2: Create a new branch named “lfs”  
cd git\_assignment\_HeroVired  
git checkout -b lfs  
  
Step3: Select the binary files which you want to track   
example :   
git lfs track "\*.bin"  
git lfs track "\*.zip"  
  
If you type the status command it shows that there is update in the .gitattributes file   
  
add this file and make a commit   
git add .

git commit –m “configuring lfs”  
  
Step4: Add any .bin or .zip file to the lfs branch  
  
I have added dummy\_file.zip   
  
once the file is added  
run the commands   
  
git add .  
  
git commit -m "Add large binary file with Git LFS"  
  
git push origin lfs  
  
  
Q3: In this same GitHub repository, create a new branch ‘geometry-calculator’, we'll work on a simple Python program that calculates the area of a circle and the area of a rectangle. We'll use Git stash to switch between working on multiple features (calculating circle area and calculating rectangle area) without committing incomplete changes.  
  
Create a branch for Circle Area

git checkout -b feature/circle-area

uncomment the code in geometry\_calculator.py   
  
radius = 5

print(f"The area of the circle with radius {radius} = {calculator.calculate\_circle\_area(radius)}")

b. Stash Circle Area changes

git stash save "WIP: circle area feature"

git status # should show clean working directory

c. Create branch for Rectangle Area

git checkout geometry-calculator

git checkout -b feature/rectangle-area

#Uncomment rectangle code only:

length = 10

width = 6

print(f"The area of the rectangle with length {length} and width {width} = {calculator.calculate\_rectangle\_area(length, width)}")

d. Stash Rectangle Area changes

git stash save "WIP: rectangle area feature"

git status # clean again

e. Switch back to Circle Area branch

git checkout feature/circle-area

git stash pop # brings back the stashed circle code

Complete and test the feature:

python geometry\_calculator.py

# should print: The area of the circle with radius 5 = 78.53981633974483

f. Commit and Push Circle Area

git add geometry\_calculator.py

git commit -m "Implement circle area feature"

git push origin feature/circle-area

g. Switch to Rectangle Area branch

git checkout feature/rectangle-area

git stash pop # brings back rectangle code

Complete and test:

python geometry\_calculator.py

# should print: The area of the rectangle with length 10 and width 6 = 60

h. Commit and Push Rectangle Area

git add geometry\_calculator.py

git commit -m "Implement rectangle area feature"

git push origin feature/rectangle-area

i. Create Pull Requests

Go to GitHub repo → Create PR from feature/circle-area → target = dev

Then create another PR from feature/rectangle-area → target = dev  
  
As I created both PR and approved one after the other, I got merge conflicts  
which I have resolved and merge the PR.