# Placement Empowerment Program

***Cloud Computing and DevOps Centre***

Use Git Hooks for Automation: Create a Git hook that automatically runs tests before committing code to your repository.

Name: Shahana.M.S Department:ADS



**Introduction**

Git hooks are automation scripts that run at specific points in the Git workflow. By setting up a Git hook, you can automatically enforce coding standards, run tests, or perform security checks before committing code. This helps prevent broken code from being committed and ensures a more stable codebase.

**Objective**

* Implement a **pre-commit Git hook** to automatically run tests before code is committed.
* Ensure only tested and validated code is added to the repository.
* Improve code quality and prevent errors from being committed.
* Automate repetitive tasks in the development workflow.

**Overview**

Git hooks are customizable scripts that execute when certain Git actions occur. A **pre-commit hook** runs before a commit is finalized, allowing you to automate checks like running unit tests, linting, or formatting code. If any test fails, the commit is blocked until the issue is resolved. This ensures that only well-tested code gets committed.

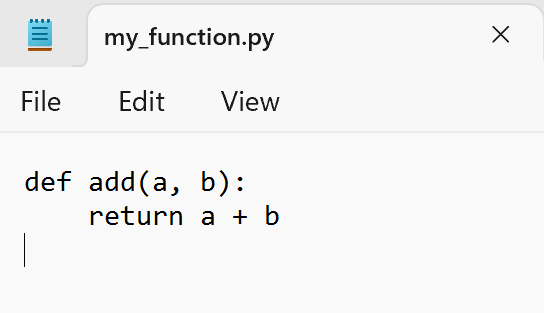
**Importance**

* **Prevents Broken Code:** Ensures that failing tests or syntax errors are caught before committing.
* **Improves Code Quality:** Enforces best practices like linting and formatting automatically.
* **Saves Time:** Reduces the need for manual testing and review.
* **Enhances Collaboration:** Keeps the repository clean and consistent for all developers.
* **Automates Workflows:** Integrates seamlessly with CI/CD pipelines for efficient development.

# Step-by-Step Overview

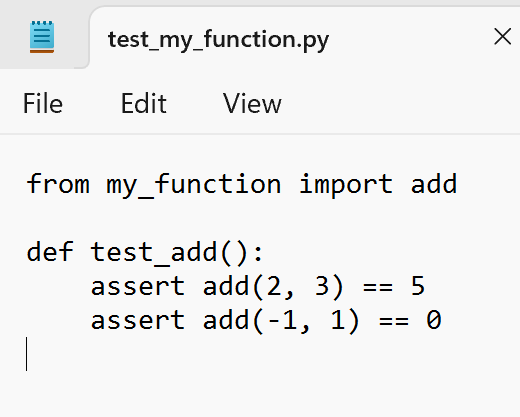
## Step 1:

Create a folder in Desktop and Open Notepad Write the Python function to add two numbers and Save the filein your folder.



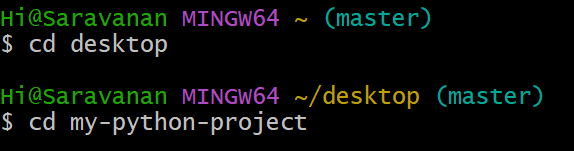
## Step 2:

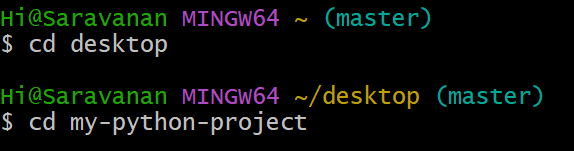
Open Notepad again to create the test file and Write the test case code to test the add function and Save the test file in the same folder where the first file is saved.



## Step 3:

Open GitBash and Navigate to the folder where the files are saved.





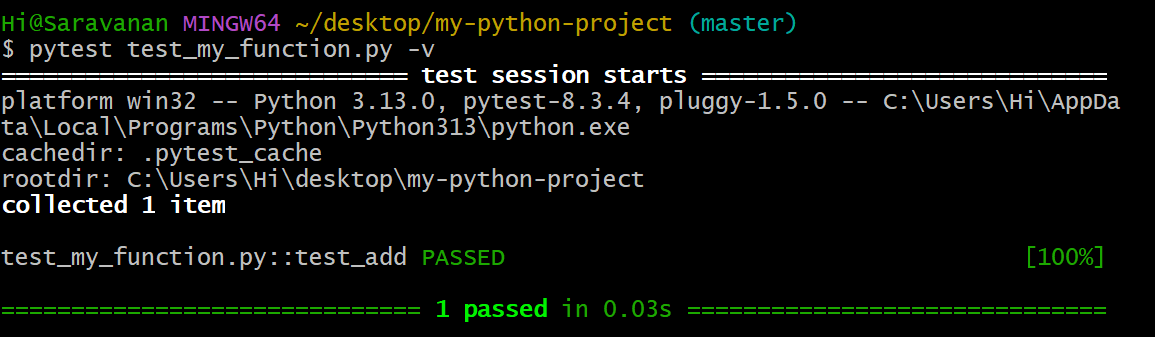
## Step 4:

Install pytest using pip by running the following command:

**pip install pytest**

****

## Step 5:

Now that we have the function and test cases, let’s run pytest to check if everything works.

## Step 6:

Run the following command to initialize Git in your project folder :

**git init(for initializing )**

****

## Step 7:

Copy the sample file to a new file called pre-commit (no .sample extension):

**cp .git/hooks/pre-commit.sample .git/hooks/pre-commit**

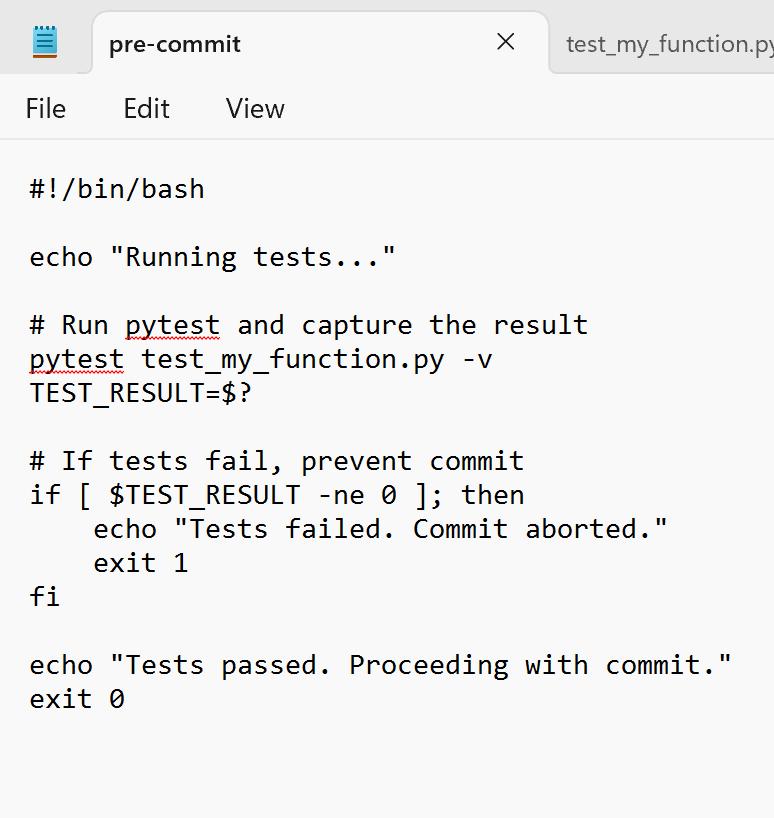
****

## Step 8:

Open Notepad and type the following code and save as **pre-commit**

in Inside the .git folder, **open the hooks folder**:

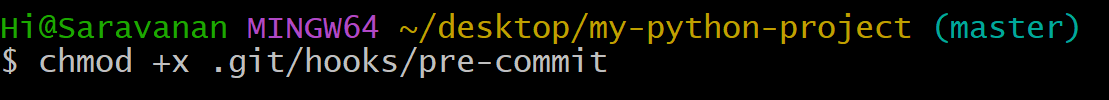
(Eg MyPath: C:\Users\Hi\desktop\my-python-project**\.git\hooks**)



## Step 9:

Once the file is saved, you need to make it executable. You can do this by running the following command in Git Bash:

**chmod +x .git/hooks/pre-commit**

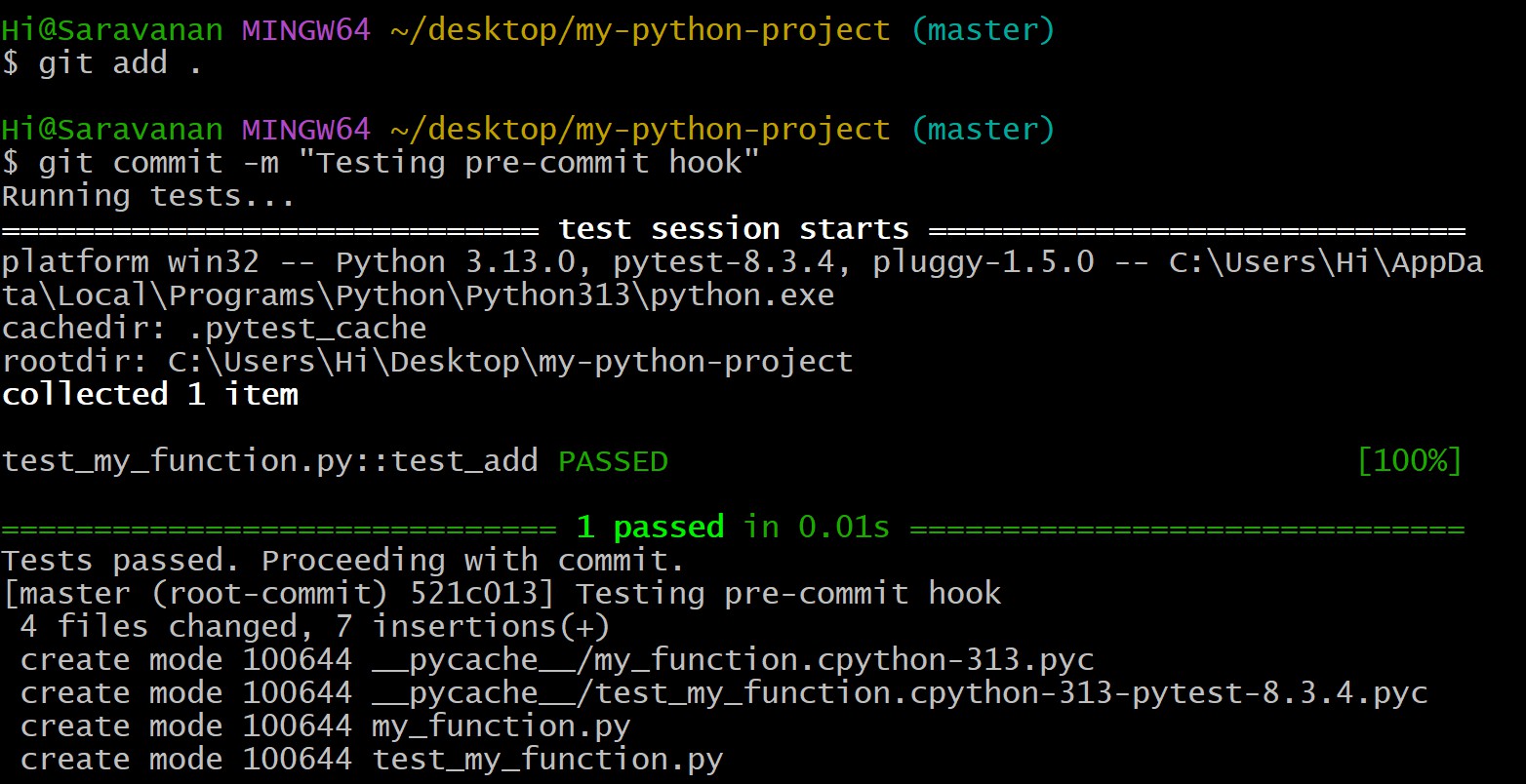
****

## Step 10:

You can test it by staging and committing some changes:

* Stage your changes**: git add .**
* Commit the changes**:**

**git commit -m "Testing pre-commit hook"**

****

**Outcome**

 A Git hook script that runs tests automatically before every commit.

 Reduced bugs and errors in the repository.

 Faster and more reliable development cycles.

 Improved code consistency and team collaboration.