Git & GitHub First Task Submission

This is my first task in **Git & GitHub** as part of the hands-on activity for debugging and restoring code.
In this project, I created and managed a simple Python calculator, intentionally introduced a bug, and then used Git to debug and restore a clean version of the code while preserving ongoing work.

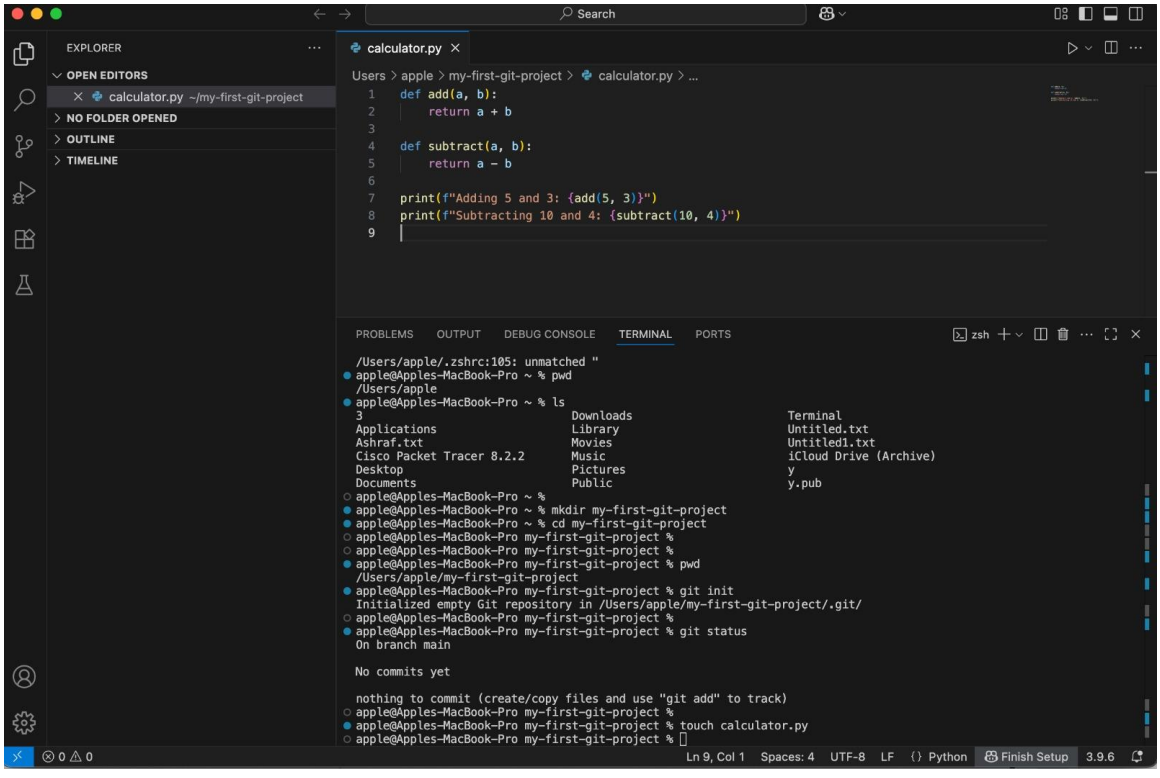
" **Explore the complete repository and the final version of the Python file here:**
My First Git Project"

1. Git Commands Used (in order)

```
git init
git add calculator.py
git commit -m "Initial commit"
git log
git diff
git stash
git revert <commit-id>
git stash pop
git push origin main
```

2. Screenshots of Key Commands

Insert the screenshots of the following commands at the placeholders below:



calculator.py

Users > apple > my-first-git-project > calculator.py > ...

```python
def add(a, b):
    return a + b

def subtract(a, b):
    return a - b

print(f"Adding 5 and 3: {add(5, 3)}")
print(f"Subtracting 10 and 4: {subtract(10, 4)}")

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS zsh

```
/Users/apple/my-first-git-project
apple@Apples-MacBook-Pro my-first-git-project % git init
Initialized empty Git repository in /Users/apple/my-first-git-project/.git/
apple@Apples-MacBook-Pro my-first-git-project %
apple@Apples-MacBook-Pro my-first-git-project % git status
On branch main

No commits yet

nothing to commit (create/copy files and use "git add" to track)
apple@Apples-MacBook-Pro my-first-git-project %
apple@Apples-MacBook-Pro my-first-git-project % touch calculator.py
apple@Apples-MacBook-Pro my-first-git-project % git add .
apple@Apples-MacBook-Pro my-first-git-project % git commit -m " Add basic calculator functions (add, subtract)"
[main (root-commit) 60eb7a7]  Add basic calculator functions (add, subtract)
 1 file changed, 8 insertions(+)
 create mode 100644 calculator.py
apple@Apples-MacBook-Pro my-first-git-project %
apple@Apples-MacBook-Pro my-first-git-project %
apple@Apples-MacBook-Pro my-first-git-project % git status
On branch main
nothing to commit, working tree clean
apple@Apples-MacBook-Pro my-first-git-project % git log
commit 60eb7a76e1874acd27d1eeed67acd7feb972be05 (HEAD -> main)
Author: Omar <omar.ashraf3011@gmail.com>
Date:   Wed Jul 30 06:13:21 2025 -0400

    Add basic calculator functions (add, subtract)
apple@Apples-MacBook-Pro my-first-git-project %
```

Ln 9, Col 1 Spaces: 4 UTF-8 LF {} Python Finish Setup 3.9.6

calculator.py

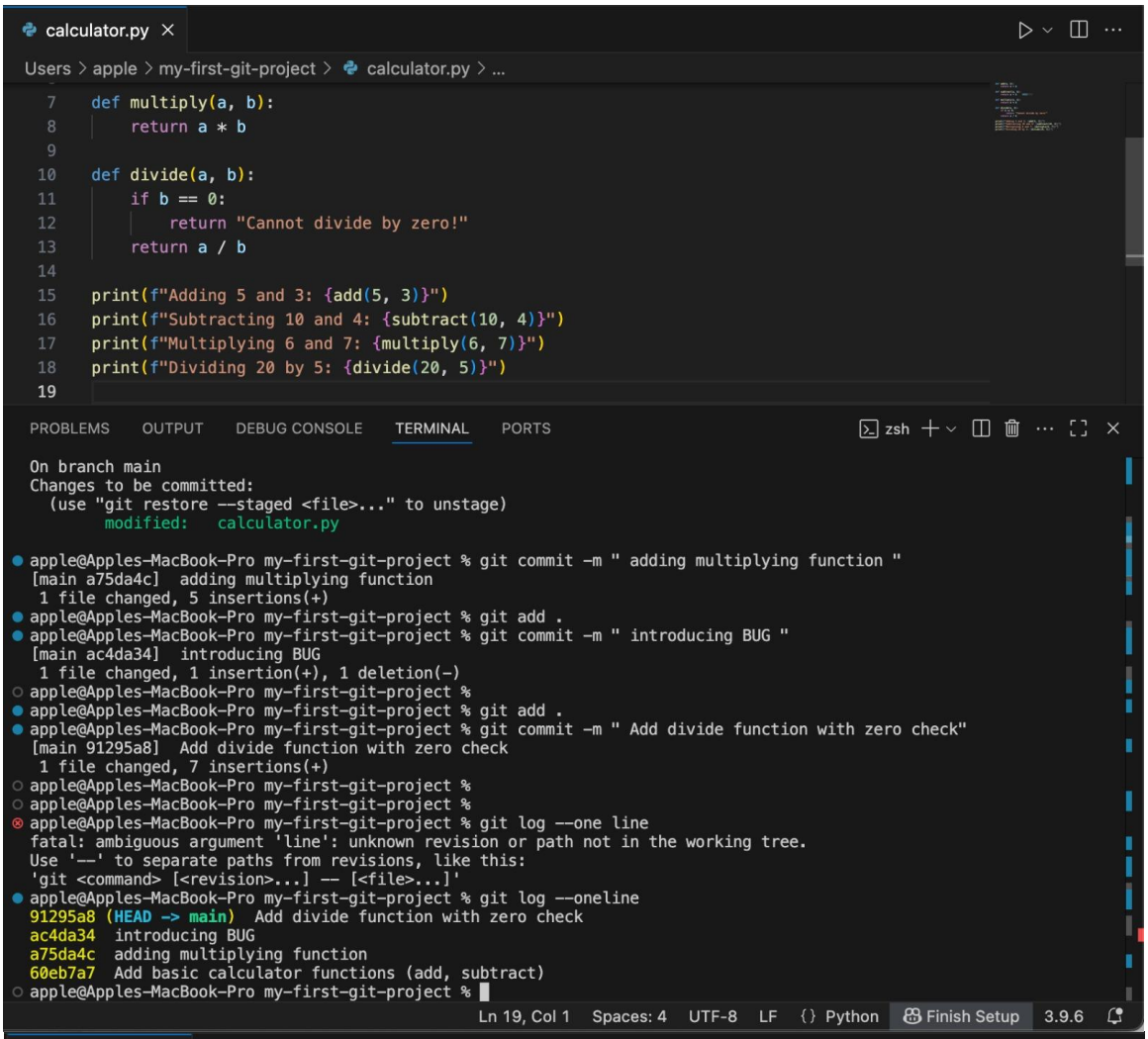
Users > apple > my-first-git-project > calculator.py > ...

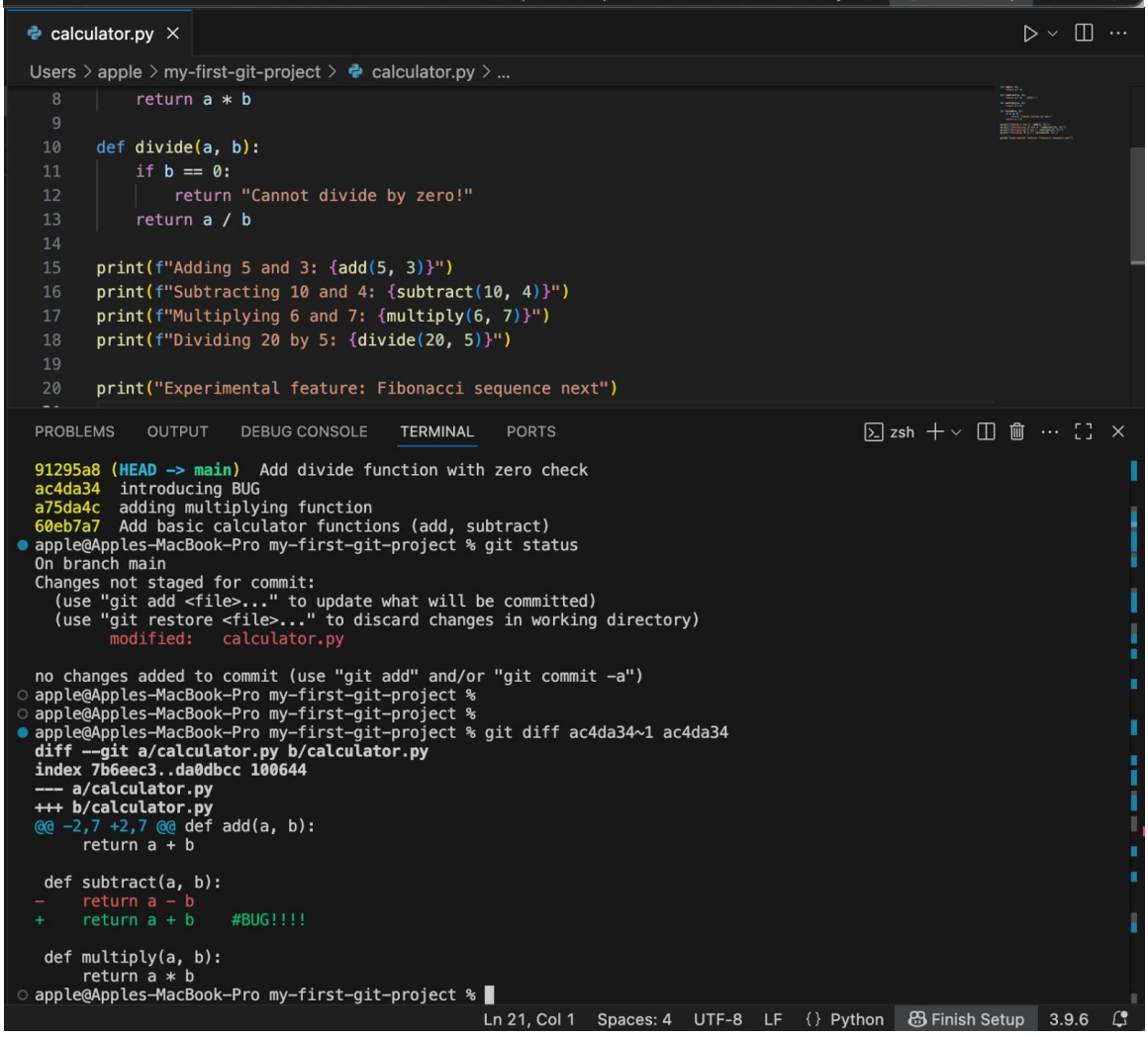
```python
    return a + b

def subtract(a, b):
    return a - b

def multiply(a, b):
    return a * b

print(f"Adding 5 and 3: {add(5, 3)}")
print(f"Subtracting 10 and 4: {subtract(10, 4)}")
print(f"Multiplying 6 and 7: {multiply(6, 7)}")

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS zsh

```
apple@Apples-MacBook-Pro my-first-git-project %
apple@Apples-MacBook-Pro my-first-git-project % git status
On branch main
nothing to commit, working tree clean
apple@Apples-MacBook-Pro my-first-git-project % git log
commit 60eb7a76e1874acd27d1eeed67acd7feb972be05 (HEAD -> main)
Author: Omar <omar.ashraf3011@gmail.com>
Date:   Wed Jul 30 06:13:21 2025 -0400

    Add basic calculator functions (add, subtract)
apple@Apples-MacBook-Pro my-first-git-project % git status
On branch main
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:   calculator.py

no changes added to commit (use "git add" and/or "git commit -a")
apple@Apples-MacBook-Pro my-first-git-project % git add .
apple@Apples-MacBook-Pro my-first-git-project % git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   calculator.py

apple@Apples-MacBook-Pro my-first-git-project % git commit -m " adding multiplying function "
[main a75da4c]  adding multiplying function
 1 file changed, 5 insertions(+)
apple@Apples-MacBook-Pro my-first-git-project %
```

Ln 13, Col 1 Spaces: 4 UTF-8 LF {} Python Finish Setup 3.9.6


calculator.py
Users > apple > my-first-git-project > calculator.py > ...
def multiply(a, b):
return a * b
def divide(a, b):
if b == 0:
return "Cannot divide by zero!"
return a / b
print(f"Adding 5 and 3: {add(5, 3)}")
print(f"Subtracting 10 and 4: {subtract(10, 4)}")
print(f"Multiplying 6 and 7: {multiply(6, 7)}")
print(f"Dividing 20 by 5: {divide(20, 5)}")
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
zsh
On branch main
Changes to be committed:
(use "git restore --staged <file>..." to unstage)
modified: calculator.py
apple@Apples-MacBook-Pro my-first-git-project % git commit -m " adding multiplying function "
[main a75da4c] adding multiplying function
1 file changed, 5 insertions(+)
apple@Apples-MacBook-Pro my-first-git-project % git add .
apple@Apples-MacBook-Pro my-first-git-project % git commit -m " introducing BUG "
[main ac4da34] introducing BUG
1 file changed, 1 insertion(+), 1 deletion(-)
apple@Apples-MacBook-Pro my-first-git-project %
apple@Apples-MacBook-Pro my-first-git-project % git add .
apple@Apples-MacBook-Pro my-first-git-project % git commit -m " Add divide function with zero check"
[main 91295a8] Add divide function with zero check
1 file changed, 7 insertions(+)
apple@Apples-MacBook-Pro my-first-git-project %
apple@Apples-MacBook-Pro my-first-git-project %
apple@Apples-MacBook-Pro my-first-git-project % git log --one line
fatal: ambiguous argument 'line': unknown revision or path not in the working tree.
Use '--' to separate paths from revisions, like this:
'git <command> [<revision>...] -- [<file>...]'
apple@Apples-MacBook-Pro my-first-git-project % git log --oneline
91295a8 (HEAD -> main) Add divide function with zero check
ac4da34 introducing BUG
a75da4c adding multiplying function
60eb7a7 Add basic calculator functions (add, subtract)
apple@Apples-MacBook-Pro my-first-git-project %
Ln 19, Col 1 Spaces: 4 UTF-8 LF {} Python Finish Setup 3.9.6


calculator.py
Users > apple > my-first-git-project > calculator.py > ...
return a * b
def divide(a, b):
if b == 0:
return "Cannot divide by zero!"
return a / b
print(f"Adding 5 and 3: {add(5, 3)}")
print(f"Subtracting 10 and 4: {subtract(10, 4)}")
print(f"Multiplying 6 and 7: {multiply(6, 7)}")
print(f"Dividing 20 by 5: {divide(20, 5)}")
print("Experimental feature: Fibonacci sequence next")
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
zsh
91295a8 (HEAD -> main) Add divide function with zero check
ac4da34 introducing BUG
a75da4c adding multiplying function
60eb7a7 Add basic calculator functions (add, subtract)
apple@Apples-MacBook-Pro my-first-git-project % git status
On branch main
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: calculator.py
no changes added to commit (use "git add" and/or "git commit -a")
apple@Apples-MacBook-Pro my-first-git-project %
apple@Apples-MacBook-Pro my-first-git-project %
apple@Apples-MacBook-Pro my-first-git-project % git diff ac4da34~1 ac4da34
diff --git a/calculator.py b/calculator.py
index 7b6eec3..da0dbcc 100644
--- a/calculator.py
+++ b/calculator.py
@@ -2,7 +2,7 @@ def add(a, b):
return a + b
def subtract(a, b):
- return a - b
+ return a + b #BUG!!!!
def multiply(a, b):
return a * b
apple@Apples-MacBook-Pro my-first-git-project %
Ln 21, Col 1 Spaces: 4 UTF-8 LF {} Python Finish Setup 3.9.6

```
    return a * b
apple@Apples-MacBook-Pro my-first-git-project % git stash push -m " experimental feature"
Saved working directory and index state On main:  experimental feature
apple@Apples-MacBook-Pro my-first-git-project % git status
On branch main
nothing to commit, working tree clean
apple@Apples-MacBook-Pro my-first-git-project % git stash list
stash@{0}: On main: experimental feature
apple@Apples-MacBook-Pro my-first-git-project %
```

calculator.py

Users > apple > my-first-git-project > calculator.py > ...

```python
def add(a, b):
    return a + b

def subtract(a, b):
    return a - b

def multiply(a, b):
    return a * b

def divide(a, b):
    if b == 0:
        return "Cannot divide by zero!"
    return a / b

print(f"Adding 5 and 3: {add(5, 3)}")
print(f"Subtracting 10 and 4: {subtract(10, 4)}")
print(f"Multiplying 6 and 7: {multiply(6, 7)}")
print(f"Dividing 20 by 5: {divide(20, 5)}")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

zsh

Python

```
 def multiply(a, b):
     return a * b
apple@Apples-MacBook-Pro my-first-git-project % git stash push -m " experimental feature"
Saved working directory and index state On main:  experimental feature
apple@Apples-MacBook-Pro my-first-git-project % git status
On branch main
nothing to commit, working tree clean
apple@Apples-MacBook-Pro my-first-git-project % git stash list
stash@{0}: On main: experimental feature
apple@Apples-MacBook-Pro my-first-git-project % git revert ac4da34
Auto-merging calculator.py
[main 9dfe1d1] Revert " introducing BUG"
 1 file changed, 1 insertion(+), 1 deletion(-)
apple@Apples-MacBook-Pro my-first-git-project % python calculator.py
zsh: command not found: python
apple@Apples-MacBook-Pro my-first-git-project % python3 calculator.py
Adding 5 and 3: 8
Subtracting 10 and 4: 6
Multiplying 6 and 7: 42
Dividing 20 by 5: 4.0
apple@Apples-MacBook-Pro my-first-git-project %
```

Ln 20, Col 1 Spaces: 4 UTF-8 LF {} Python Finish Setup 3.9.6

Python

```
apple@Apples-MacBook-Pro my-first-git-project % git log --oneline
9dfe1d1 (HEAD -> main) Revert " introducing BUG"
91295a8  Add divide function with zero check
ac4da34  introducing BUG
a75da4c  adding multiplying function
60eb7a7  Add basic calculator functions (add, subtract)
apple@Apples-MacBook-Pro my-first-git-project % git status
On branch main
nothing to commit, working tree clean
apple@Apples-MacBook-Pro my-first-git-project % git stash pop
Auto-merging calculator.py
On branch main
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:   calculator.py

no changes added to commit (use "git add" and/or "git commit -a")
Dropped refs/stash@{0} (58d58735abdd186b07d76340e16d19cdd9f9df58)
apple@Apples-MacBook-Pro my-first-git-project %
```

calculator.py

Users > apple > my-first-git-project > calculator.py > ...

```python
def multiply(a, b):
    return a * b

def divide(a, b):
    if b == 0:
        return "Cannot divide by zero!"
    return a / b

print(f"Adding 5 and 3: {add(5, 3)}")
print(f"Subtracting 10 and 4: {subtract(10, 4)}")
print(f"Multiplying 6 and 7: {multiply(6, 7)}")
print(f"Dividing 20 by 5: {divide(20, 5)}")

print("Experimental feature: Fibonacci sequence next")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Steps Overview:

1. Created `my-first-git-project`** folder and initialized Git with a simple `calculator.py` file.
2. Staged and committed** the initial calculator functions and verified using `git log`.
3. Added a multiply function** and committed the changes.
4. Added a divide function with zero check** and confirmed updates in commit history.
5. Introduced a bug** in the subtract function and used `git diff` to identify the changes.
6. Stashed uncommitted work** using `git stash` for safe temporary storage.
7. Reverted the buggy commit** using `git revert` and verified functionality by running the Python file.
8. Popped the stashed changes** back into the working directory using `git stash pop`.
9. Pushed all final changes** to the remote repository.

3. Final calculator.py Code

```
def add(a, b):
    return a + b

def subtract(a, b):
    return a - b

def multiply(a, b):
    return a * b

def divide(a, b):
    if b == 0:
        return "Cannot divide by zero!"
    return a / b

print(f"Adding 5 and 3: {add(5, 3)}")
print(f"Subtracting 10 and 4: {subtract(10, 4)}")
print(f"Multiplying 6 and 7: {multiply(6, 7)}")
print(f"Dividing 20 by 5: {divide(20, 5)}")

print("Experimental feature: Fibonacci sequence next")
```

4. Written Explanations

How did git log and git diff help identify the erroneous commit?

Using git log, I could review the commit history and locate the commit where the bug was introduced. git diff allowed me to compare the changes in that commit to previous ones and identify the exact line of code causing the error.

Why was git stash necessary in this scenario?

git stash allowed me to temporarily save uncommitted work. This made it possible to revert the repository to a clean state without losing ongoing changes.

Why did I choose git revert over git reset --hard?

I used git revert because it safely undoes the changes by creating a new commit, preserving the repository's history. In contrast, git reset --hard rewrites history and can cause problems when collaborating with others. Reset is useful in local-only scenarios, while revert is safer for shared repositories.

Difference between git stash pop and git stash apply:

git stash pop applies the stashed changes and removes them from the stash list. git stash apply applies the stashed changes but keeps them in the stash list for potential reuse.

Thanks for reviewing my work!

Omar Ashraf