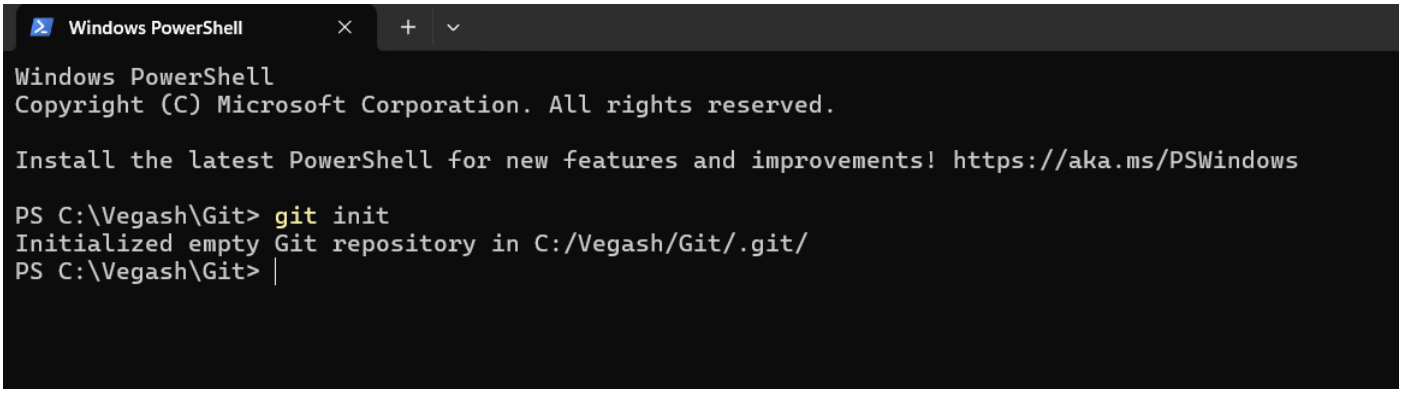
GIT AND GITHUB

ASSIGNMENT QUESTION

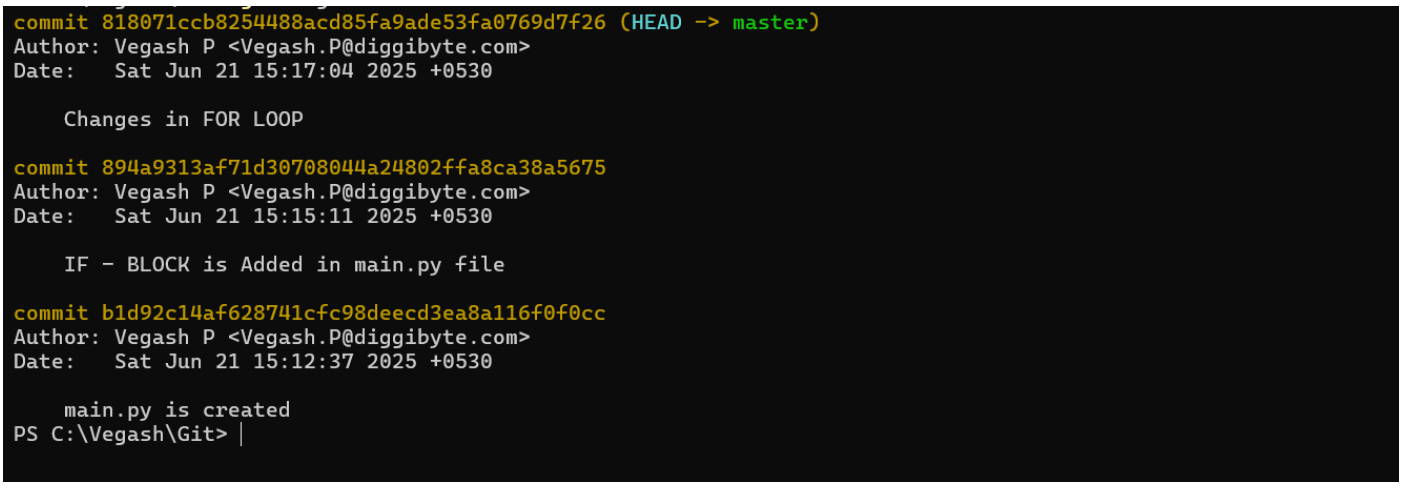
Question 1

Step 1: Create a new Git repository.

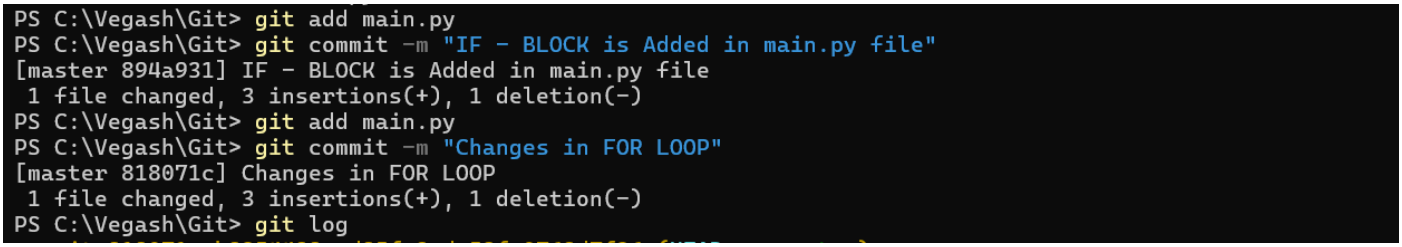


Step 2: Create a file and commit changes.

Step 3: View the commit history of your repository.



Step 4: Open the file you created earlier and make some changes to it.



Step 5: Check the file you modified is now marked as "modified" and unstaged. Hint (git status)

```
    main.py is created
PS C:\Vegash\Git> 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:   main.py
```

Step 6: Stage the changes you made to the file and commit the changes to the repository.

```
no changes added to commit (use "git add" and/or "git commit -a")
PS C:\Vegash\Git> git add main.py
PS C:\Vegash\Git> git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   main.py

PS C:\Vegash\Git> git commit -m "one more print statement is added"
[master 00b1a68] one more print statement is added
 1 file changed, 3 insertions(+), 1 deletion(-)
PS C:\Vegash\Git> git status
On branch master
nothing to commit, working tree clean
PS C:\Vegash\Git>
```

Step 7: Clone the repository you have created in GitHub.

```
PS C:\Vegash\Cloning for Assignment> git clone https://github.com/Vegash09/Git_Submodule.git
Cloning into 'Git_Submodule'...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 9 (delta 0), reused 6 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (9/9), done.
PS C:\Vegash\Cloning for Assignment>
```

Step 8: Fetch the changes, navigate into the cloned repository using the command line, and use the command git fetch to fetch any changes that have been made to the original repository since you cloned it.

```
PS C:\Vegash\Cloning for Assignment\Git_Submodule> git fetch
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 1.03 KiB | 66.00 KiB/s, done.
From https://github.com/Vegash09/Git_Submodule
   93173f9..9c42877  main       -> origin/main
PS C:\Vegash\Cloning for Assignment\Git_Submodule>
```

Step 9: Pull changes, merge the changes you just fetched into your local copy of the repository, and use the command git pull.

```
PS C:\Vegash\Cloning for Assignment\Git_Submodule> git fetch
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 943 bytes | 104.00 KiB/s, done.
From https://github.com/Vegash09/Git_Submodule
   9c42877..428ef36  main       -> origin/main
```

```
PS C:\Vegash\Cloning for Assignment\Git_Submodule> git merge
Updating 93173f9..428ef36
Fast-forward
 First.py | 12 ++++++++++++
 1 file changed, 12 insertions(+)
PS C:\Vegash\Cloning for Assignment\Git_Submodule>
```

Question 2

Step 1: Clone the repository you have created in GitHub.

```
PS C:\Vegash\Cloning for Assignment> git clone https://github.com/Vegash09/Git_Submodule.git
Cloning into 'Git_Submodule'...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 9 (delta 0), reused 6 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (9/9), done.
PS C:\Vegash\Cloning for Assignment>
```

Step 2: Create a new branch using the command.

```
PS C:\Vegash\Cloning for Assignment\Git_Submodule> git branch dev
PS C:\Vegash\Cloning for Assignment\Git_Submodule> git branch
  dev
* main
PS C:\Vegash\Cloning for Assignment\Git_Submodule>
```

Step 3: Switch to the new branch.

```
PS C:\Vegash\Cloning for Assignment\Git_Submodule> git checkout dev
Switched to branch 'dev'
PS C:\Vegash\Cloning for Assignment\Git_Submodule> git branch
* dev
  main
PS C:\Vegash\Cloning for Assignment\Git_Submodule>
```

Step 4: Make some changes to the code in your local copy of the repository.

```
PS C:\Vegash\Cloning for Assignment\Git_Submodule> git add First.py
PS C:\Vegash\Cloning for Assignment\Git_Submodule> git status
On branch dev
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   First.py

PS C:\Vegash\Cloning for Assignment\Git_Submodule>
```

Step 5: Commit changes to the new branch.

```
PS C:\Vegash\Cloning for Assignment\Git_Submodule> git commit -m "Made the Changes through dev branch"
[dev 6251c00] Made the Changes through dev branch
 1 file changed, 2 insertions(+)
PS C:\Vegash\Cloning for Assignment\Git_Submodule>
```

Step 6: Switch back to the original branch

```
PS C:\Vegash\Cloning for Assignment\Git_Submodule> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS C:\Vegash\Cloning for Assignment\Git_Submodule> git branch
  dev
* main
PS C:\Vegash\Cloning for Assignment\Git_Submodule>
```

Step 7: Merge the new branch.

```
PS C:\Vegash\Cloning for Assignment\Git_Submodule> git merge dev
Updating 428ef36..0275d4a
Fast-forward
 First.py | 4 ++++
 1 file changed, 4 insertions(+)
PS C:\Vegash\Cloning for Assignment\Git_Submodule>
```

Step 8: Push changes to the original branch

```
PS C:\Vegash\Cloning for Assignment\Git_Submodule> git push origin main
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 12 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 623 bytes | 623.00 KiB/s, done.
Total 6 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To https://github.com/Vegash09/Git_Submodule.git
   428ef36..0275d4a  main -> main
PS C:\Vegash\Cloning for Assignment\Git_Submodule>
```

Question 3

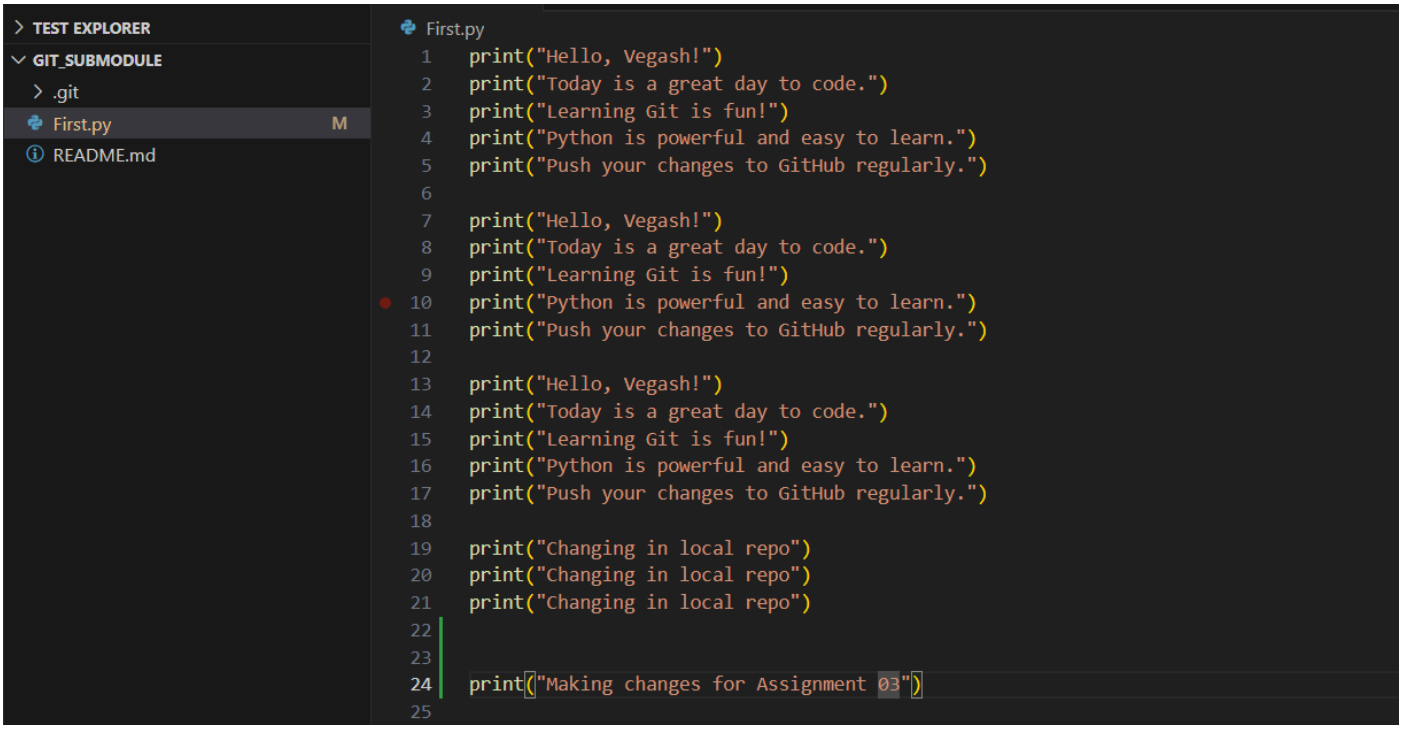
Step 1: Create a feature branch.

```
PS C:\Vegash\Cloning for Assignment\Git_Submodule> git checkout -b feature
Switched to a new branch 'feature'
PS C:\Vegash\Cloning for Assignment\Git_Submodule>
```

Step 2: Switch to the new branch.

```
PS C:\Vegash\Cloning for Assignment\Git_Submodule> git checkout -b feature
Switched to a new branch 'feature'
PS C:\Vegash\Cloning for Assignment\Git_Submodule>
```

Step 3: open the file and make some changes to it.



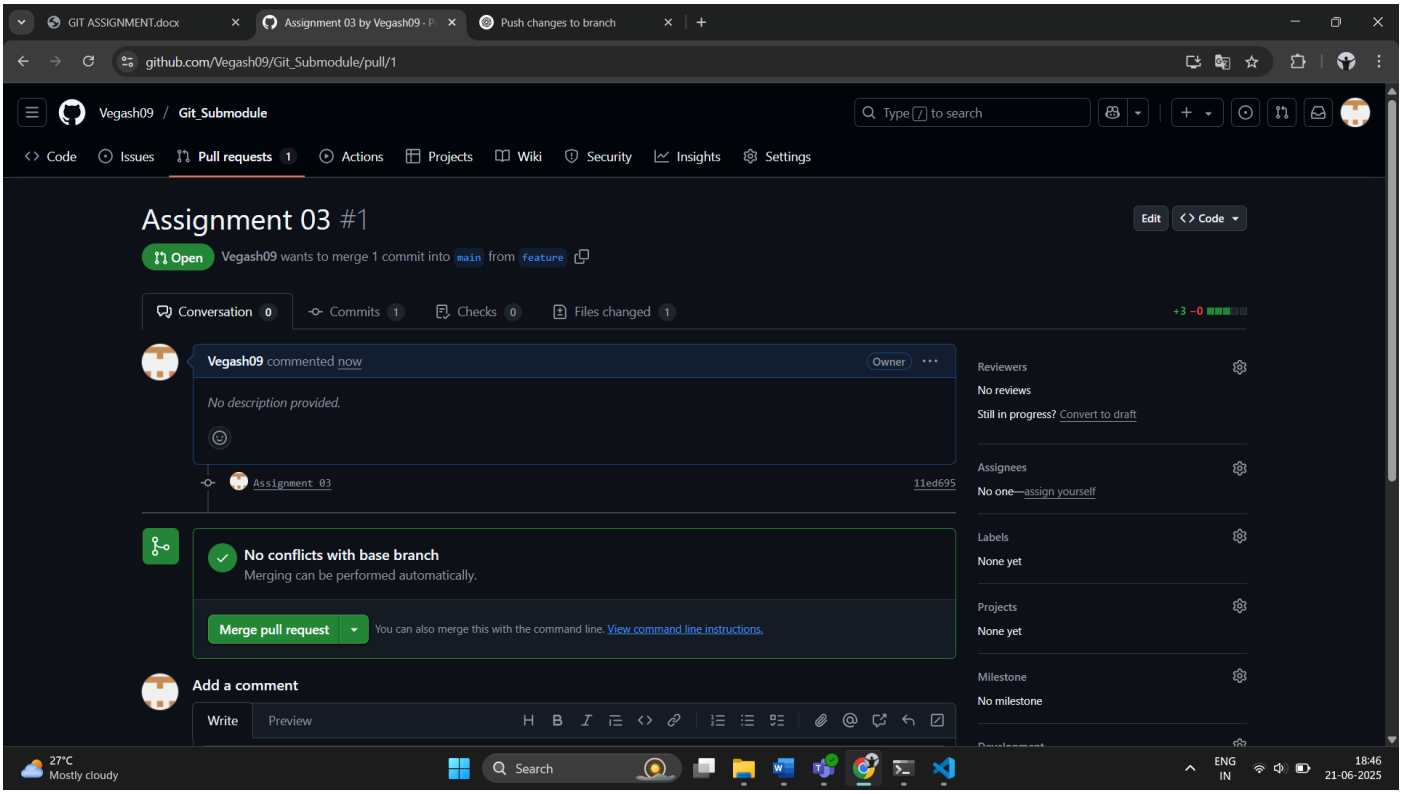
Step 4: Add and commit the changes to the new branch.

```
PS C:\Vegash\Cloning for Assignment\Git_Submodule> git add .
PS C:\Vegash\Cloning for Assignment\Git_Submodule> git commit -m "Assignment 03"
[feature 11ed695] Assignment 03
 1 file changed, 3 insertions(+)
PS C:\Vegash\Cloning for Assignment\Git_Submodule>
```

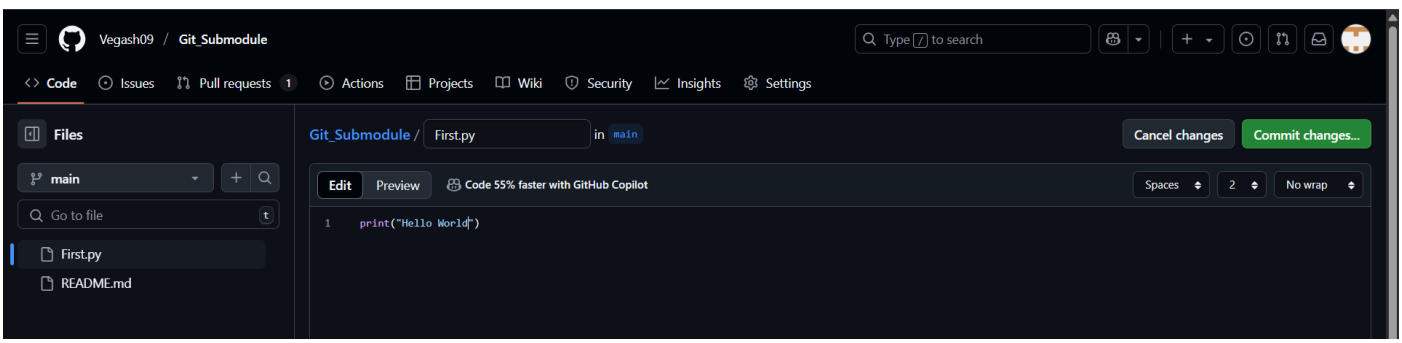
Step 5: Push the changes to the new feature branch.

```
PS C:\Vegash\Cloning for Assignment\Git_Submodule> git push origin feature
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 353 bytes | 176.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'feature' on GitHub by visiting:
remote:      https://github.com/Vegash09/Git_Submodule/pull/new/feature
remote:
To https://github.com/Vegash09/Git_Submodule.git
 * [new branch]      feature -> feature
PS C:\Vegash\Cloning for Assignment\Git_Submodule>
```

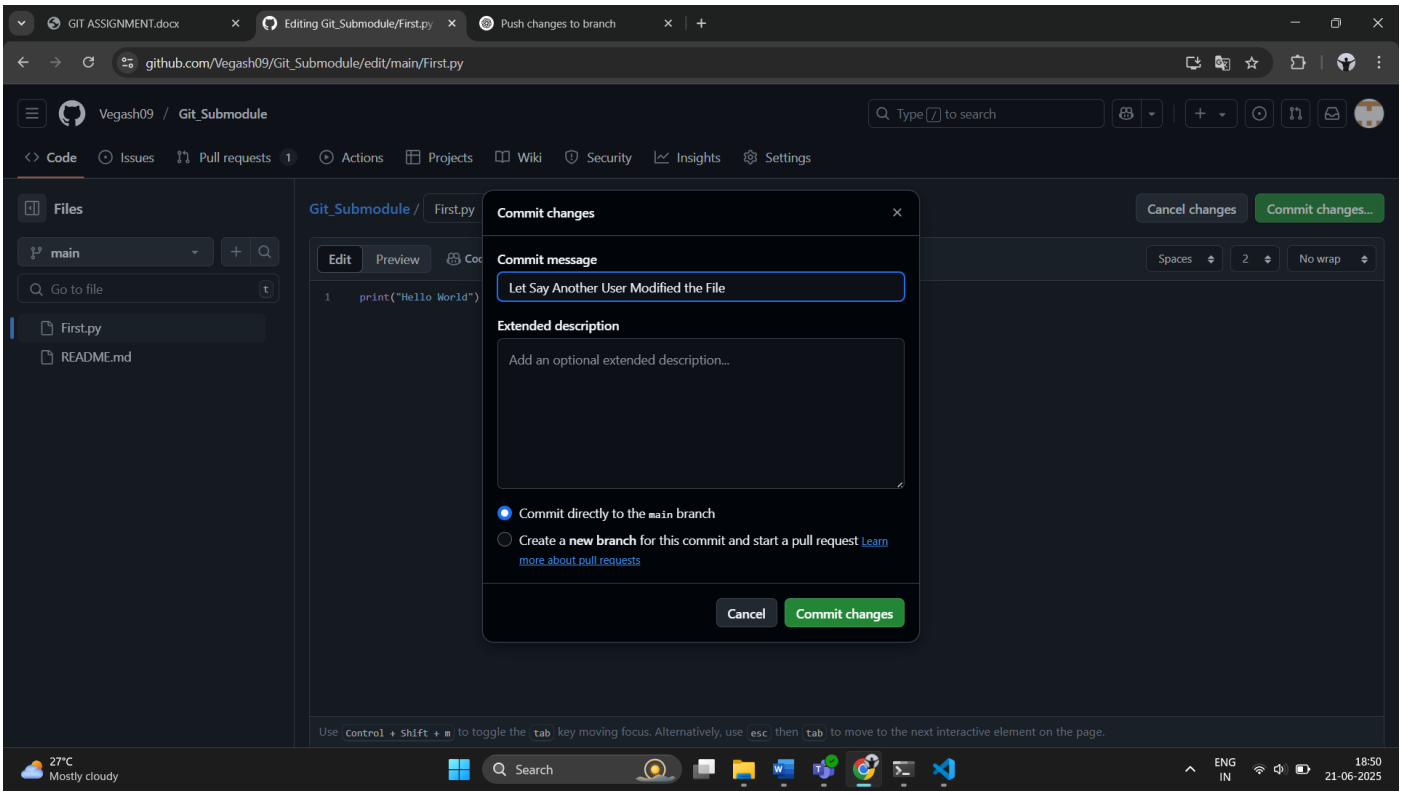
Step 6: Create a pull request.



Step 6: As another user in the master branch make some changes to the same file.



Step 7: Add and commit the changes to the master branch.

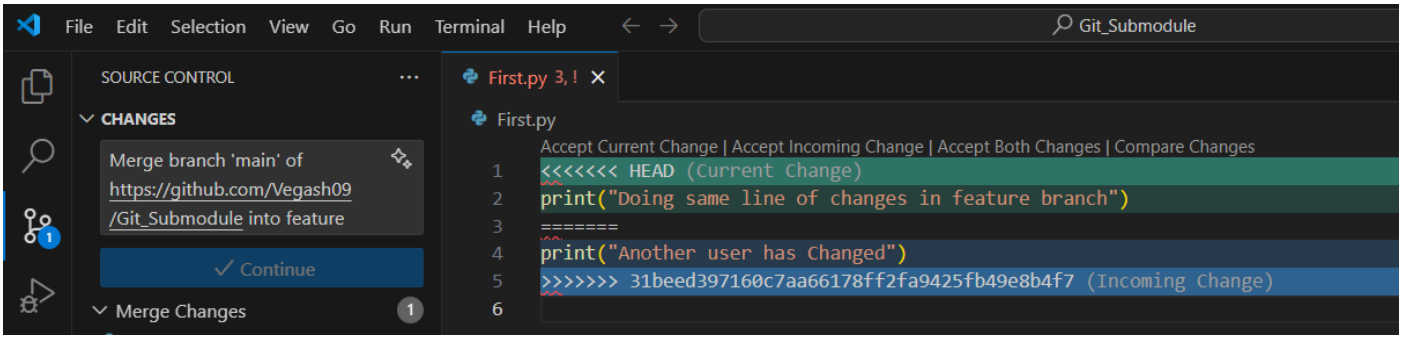


Step 8: Push the changes to the master branch.

Note: There will be a conflict in the pull request, how do we resolve it?? Hint: git rebase

```
PS C:\Vegash\Cloning for Assignment\Git_Submodule> git push origin main
To https://github.com/Vegash09/Git_Submodule.git
 ! [rejected]        main -> main (fetch first)
error: failed to push some refs to 'https://github.com/Vegash09/Git_Submodule.git'
hint: Updates were rejected because the remote contains work that you do not
hint: have locally. This is usually caused by another repository pushing to
hint: the same ref. If you want to integrate the remote changes, use
hint: 'git pull' before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

```
PS C:\Vegash\Cloning for Assignment\Git_Submodule> git pull origin main
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 953 bytes | 27.00 KiB/s, done.
From https://github.com/Vegash09/Git_Submodule
 * branch            main       -> FETCH_HEAD
   aebc024..31beed3  main       -> origin/main
Auto-merging First.py
CONFLICT (content): Merge conflict in First.py
Automatic merge failed; fix conflicts and then commit the result.
```



```
PS C:\Vegash\Cloning for Assignment\Git_Submodule> git add .
PS C:\Vegash\Cloning for Assignment\Git_Submodule> git rebase --continue
[detached HEAD b6738e8] Some another person is ediited the main
 Author: Vegash09 <vegash.p@diggibyte.com>
 1 file changed, 2 insertions(+), 1 deletion(-)
Successfully rebased and updated refs/heads/feature.
```

```
PS C:\Vegash\Cloning for Assignment\Git_Submodule> git push origin main -f
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Vegash09/Git_Submodule.git
 + e01bf43...a808675 main -> main (forced update)
PS C:\Vegash\Cloning for Assignment\Git_Submodule>
```

I resolved the conflict manually by opening the file, identifying the conflicting sections marked by Git, and editing them to keep the correct content. After resolving the conflict, I added the file using git add and continued the rebase with git rebase --continue. Finally, I used git push --force to update the remote branch after the rebase.

Question 4

Step 1: Step 1: Create a feature branch.

```
PS C:\Vegash\Cloning for Assignment\Git_Submodule> git checkout -b temp
Switched to a new branch 'temp'
PS C:\Vegash\Cloning for Assignment\Git_Submodule>
```

Step 2: Switch to the new branch.

```
PS C:\Vegash\Cloning for Assignment\Git_Submodule> git checkout -b temp
Switched to a new branch 'temp'
PS C:\Vegash\Cloning for Assignment\Git_Submodule>
```

open the file and make some changes to it.

Add and commit the changes to the new branch.

open the same file and make some changes to it.

Add and commit the changes to the new branch.

open the same file and make some changes to it.

Add and commit the changes to the new branch.

```
commit 28f7920bf5424d9cec07361be7ed0b93d2234cc7 (HEAD -> temp)
Author: Vegash P <Vegash.P@diggibyte.com>
Date:   Sun Jun 22 08:26:07 2025 +0530

    commit 03

commit a081aa180c79ca05d375f52de2875c58c47f183c
Author: Vegash P <Vegash.P@diggibyte.com>
Date:   Sun Jun 22 08:25:27 2025 +0530

    commit 02

commit 1d7a3c45ef8d102c2d9a3f02ed6596c96a9d58b1
Author: Vegash P <Vegash.P@diggibyte.com>
Date:   Sun Jun 22 08:24:59 2025 +0530

    commit 01
```

Step 3: Identify the commit or commits that you want to "cherry-pick"(Note the hash of the commit or commits that you want to "cherry-pick")

```
commit a081aa180c79ca05d375f52de2875c58c47f183c
Author: Vegash P <Vegash.P@diggibyte.com>
Date:   Sun Jun 22 08:25:27 2025 +0530

    commit 02
```

Step 4: Use the "git checkout" command to switch to the branch where you want to apply the changes.

```
PS C:\Vegash\Cloning for Assignment\Git_Submodule> git add .
PS C:\Vegash\Cloning for Assignment\Git_Submodule> git commit -m "Assignment 03"
[feature 11ed695] Assignment 03
 1 file changed, 3 insertions(+)
PS C:\Vegash\Cloning for Assignment\Git_Submodule>
```

Step 5: Use the "git cherry-pick" command followed by the commit hash(es) that you want to apply.

```
PS C:\Vegash\Cloning for Assignment\Git_Submodule> git cherry-pick 30479cd2c154bfac5d393f79fe96a65ee934884b
error: Cherry-picking is not possible because you have unmerged files.
hint: Fix them up in the work tree, and then use 'git add/rm <file>'
hint: as appropriate to mark resolution and make a commit.
fatal: cherry-pick failed
PS C:\Vegash\Cloning for Assignment\Git_Submodule> git add .
PS C:\Vegash\Cloning for Assignment\Git_Submodule> git commit -m "Merge Conflict rasied via cherry pick"
[main ae4bb1c] Merge Conflict rasied via cherry pick
 Date: Sun Jun 22 09:15:19 2025 +0530
 1 file changed, 1 insertion(+)
 create mode 100644 cherry.txt
PS C:\Vegash\Cloning for Assignment\Git_Submodule>
```

Question 5

Step 1: Step 1: Create a feature branch.

```
PS C:\Vegash\Cloning for Assignment\Git_Submodule> git branch undo
PS C:\Vegash\Cloning for Assignment\Git_Submodule> git checkout undo
D       First.py
D       Rebase.py
D       cherry.txt
Switched to branch 'undo'
PS C:\Vegash\Cloning for Assignment\Git_Submodule>
```

Step 2: Switch to the new branch.


```
PS C:\Vegash\Cloning for Assignment\Git_Submodule> git branch undo
PS C:\Vegash\Cloning for Assignment\Git_Submodule> git checkout undo
D       First.py
D       Rebase.py
D       cherry.txt
Switched to branch 'undo'
PS C:\Vegash\Cloning for Assignment\Git_Submodule>
```


open the file and make some changes to it.

Add and commit the changes to the new branch.

open the same file and make some changes to it.

Add and commit the changes to the new branch.

open the same file and make some changes to it.

Add and commit the changes to the new branch.

```
nothing added to commit but untracked files present (use "git add" to track)
PS C:\Vegash\Cloning for Assignment\Git_Submodule> git add .
PS C:\Vegash\Cloning for Assignment\Git_Submodule> git commit -m "Commit 1"
[undo b382764] Commit 1
 1 file changed, 1 insertion(+)
 create mode 100644 Files/undo.txt
PS C:\Vegash\Cloning for Assignment\Git_Submodule> git add .
PS C:\Vegash\Cloning for Assignment\Git_Submodule> git commit -m "Commit 2"
[undo 2f26f79] Commit 2
 1 file changed, 2 insertions(+), 1 deletion(-)
PS C:\Vegash\Cloning for Assignment\Git_Submodule> git add .
PS C:\Vegash\Cloning for Assignment\Git_Submodule> git commit -m "Commit 3"
[undo 3bf6b8f] Commit 3
 1 file changed, 2 insertions(+), 1 deletion(-)
PS C:\Vegash\Cloning for Assignment\Git_Submodule>
```

Step 3: Use the "git log" command to view the commit history and identify the commit to which you want to reset.

```
commit 2f26f79884bcfdaa58a27850e354dd5d11fa6993
Author: Vegash P <Vegash.P@diggibyte.com>
Date:   Sun Jun 22 15:21:05 2025 +0530

    Commit 2
```

Step 4: Use the "git reset" command followed by the desired reset type and the commit hash

```
PS C:\Vegash\Cloning for Assignment\Git_Submodule> git reset --hard 2f26f79884bcfdaa58a27850e354dd5d11fa6993
HEAD is now at 2f26f79 Commit 2
PS C:\Vegash\Cloning for Assignment\Git_Submodule>
```

Step 5: Verify that the reset was successful by using the "git log" command again.

```
commit 2f26f79884bcfdaa58a27850e354dd5d11fa6993 (HEAD -> undo)
Author: Vegash P <Vegash.P@diggibyte.com>
Date:   Sun Jun 22 15:21:05 2025 +0530

    Commit 2

commit b382764bf50997bb9dc4415a550a492e7c7526b4
Author: Vegash P <Vegash.P@diggibyte.com>
Date:   Sun Jun 22 15:20:41 2025 +0530

    Commit 1
```

Step 6: Use the "git log" command to view the commit history and identify the commit that you want to reverse.

```
246f60a HEAD@{1}: commit: commit 03
4eacdf2 HEAD@{2}: commit: commit 02
93d6dd7 HEAD@{3}: commit: commit 01
```

```
4eacdf2 HEAD@{2}: commit: commit 02
```

Step 7: Use the "git revert" command followed by the commit hash or reference to which you want to revert. (Hint: git revert <commit hash>)

```
PS C:\Vegash\Cloning for Assignment\Git_Submodule\Files> git revert 4eacdf2d3cdfffd4fdd2725f530359ec74abbad6
[undo 007f3db] Revert "commit 02"
 1 file changed, 1 deletion(-)
 delete mode 100644 Files/revert2.txt
PS C:\Vegash\Cloning for Assignment\Git_Submodule\Files> git log
commit 007f3dbaef4ce9a887754ba75049f844627c0d51 (HEAD -> undo)
Author: Vegash P <Vegash.P@diggibyte.com>
Date:   Sun Jun 22 15:33:24 2025 +0530

    Revert "commit 02"

    This reverts commit 4eacdf2d3cdfffd4fdd2725f530359ec74abbad6.
```

Step 8: Verify that the revert was successful by using the "git log" command again.

Note: Identify the difference between git log after git reset and git r evert.

```
commit ae4bb1c6039e7a34dcbbdb88f054ff91f8aa3abc
Author: Vegash P <Vegash.P@diggibyte.com>
PS C:\Vegash\Cloning for Assignment\Git_Submodule\Files> git reflog
007f3db (HEAD -> undo) HEAD@{0}: revert: Revert "commit 02"
```

Point	git reset	git revert
What it does	Moves HEAD to an earlier commit	Creates a new commit to undo a previous one
History changes?	Yes (can remove commits)	No (keeps all commits)
Safe for team use?	❌ No (changes history)	✅ Yes (keeps history clean)
Creates new commit?	❌ No	✅ Yes

`