EXPERIMENT No. 01: Setting Up and Basic Commands

AIM:

Initialize a new Git repository in a directory. Create a new file and add it to the staging area and commit

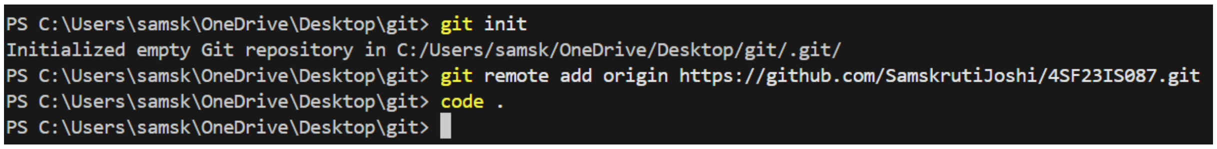
the changes with an appropriate commit message

COMMANDS:

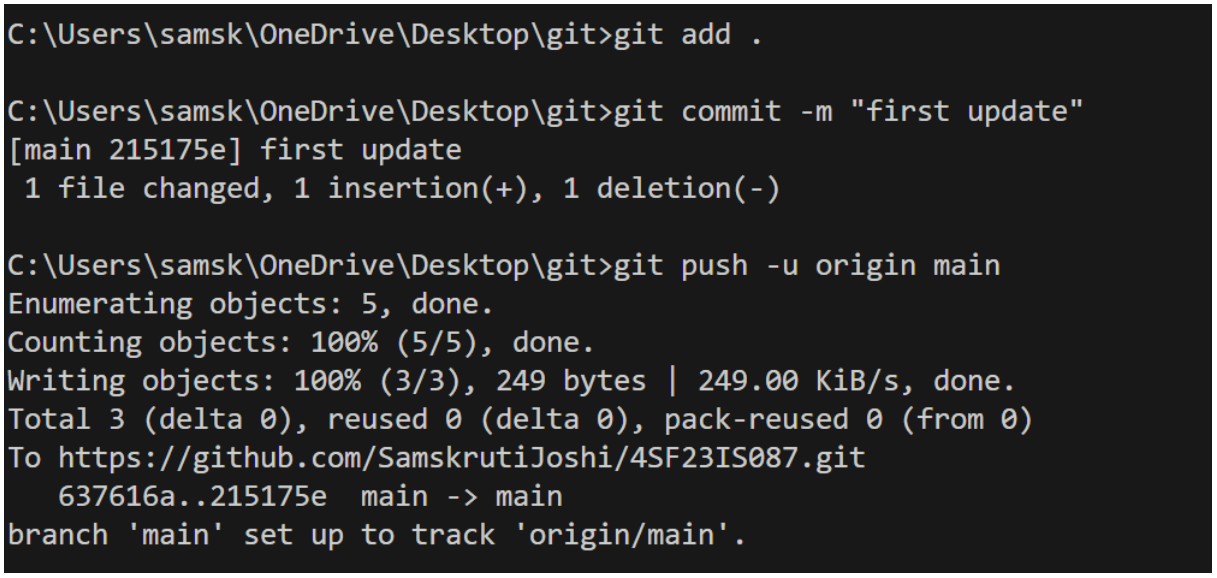
1. git init
2. git remote add origin “remote repository link”
3. git add .
4. git commit -m “message”
5. git push -u origin master

PROCEDURE AND RESULTS:

- Create a new directory and initialize a new Git repository in the current directory using the command - git init
- Clone a previously existing remote repository using the command - git clone “remote repository link”
- Navigate to the cloned repository using the command - cd “remote repository name”
- Make any required changes by adding a folder or a file Stage the changes applied using the command - git add .
- Record the changes applied that have been staged using the command - git commit -m “message” with an appropriate message
- Push the changes recorded into the remote repository using the command using the command –git push -u origin master



```
PS C:\Users\samsk\OneDrive\Desktop\git> git init
Initialized empty Git repository in C:/Users/samsk/OneDrive/Desktop/git/.git/
PS C:\Users\samsk\OneDrive\Desktop\git> git remote add origin https://github.com/SamskrutiJoshi/4SF23IS087.git
PS C:\Users\samsk\OneDrive\Desktop\git> code .
PS C:\Users\samsk\OneDrive\Desktop\git>
```



```
C:\Users\samsk\OneDrive\Desktop\git>git add .

C:\Users\samsk\OneDrive\Desktop\git>git commit -m "first update"
[main 215175e] first update
 1 file changed, 1 insertion(+), 1 deletion(-)

C:\Users\samsk\OneDrive\Desktop\git>git push -u origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 249 bytes | 249.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/SamskrutiJoshi/4SF23IS087.git
   637616a..215175e  main -> main
branch 'main' set up to track 'origin/main'.
```

EXPERIMENT No. 02: Creating and Managing Branches

AIM:

a. Create a new branch named “feature-branch”. Switch to the “master” branch. Merge the “feature-branch” into “master”.

b. Write the commands to stash your changes, switch branches, and then apply the stashed changes.

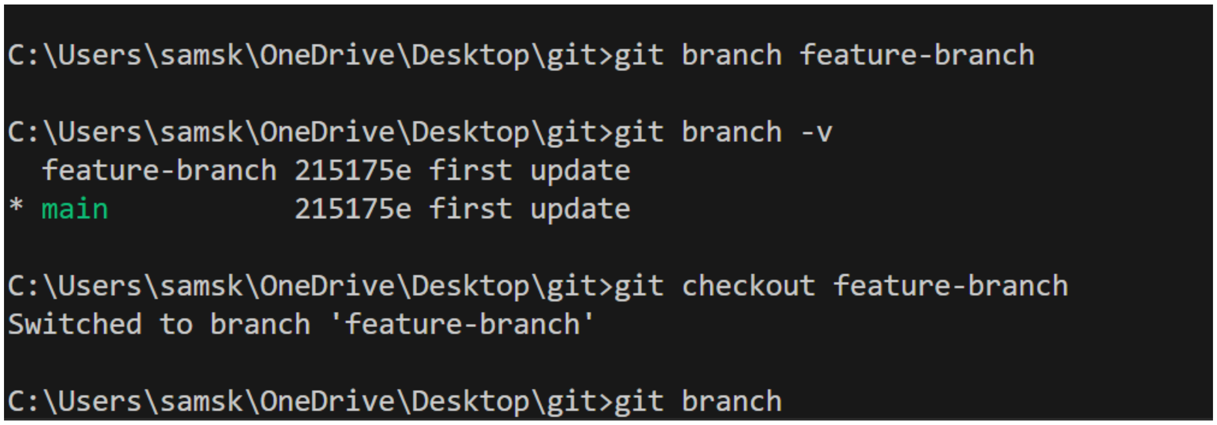
COMMANDS:

1. git branch feature-branch
2. git checkout master/main
3. git merge feature-branch
4. git checkout feature-branch
5. git stash
6. git stash apply

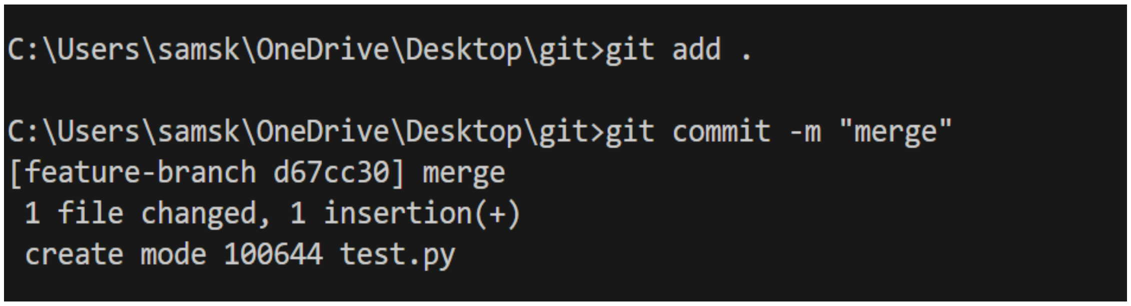
PROCEDURE AND RESULTS:

a.

- Create a new branch using the command - git branch feature-branch
- Navigate to the new branch created using the command – git checkout feature-branch
- Make any required changes, stage the changes and commit them with an appropriate message
- Navigate to the main branch using the command - git checkout master
- Merge the changes made into the main branch using the command – git merge feature-branch



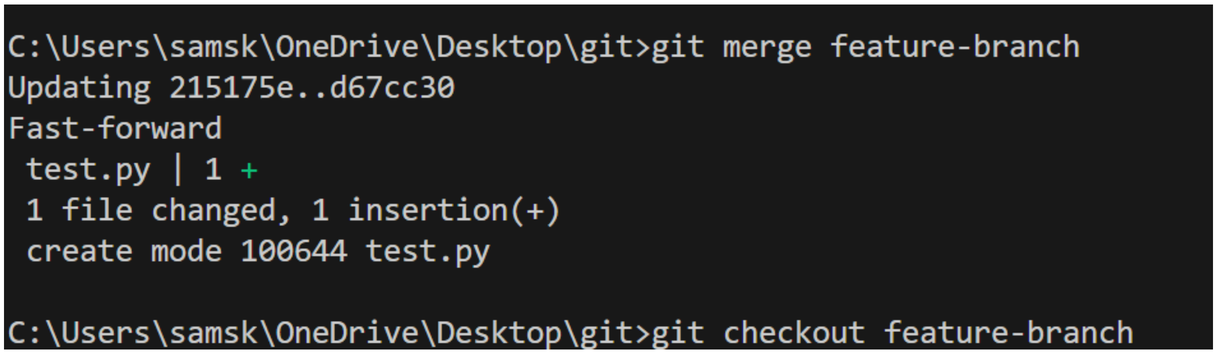
```
C:\Users\samsk\OneDrive\Desktop\git>git branch feature-branch

C:\Users\samsk\OneDrive\Desktop\git>git branch -v
  feature-branch 215175e first update
* main           215175e first update

C:\Users\samsk\OneDrive\Desktop\git>git checkout feature-branch
Switched to branch 'feature-branch'

C:\Users\samsk\OneDrive\Desktop\git>git branch
```



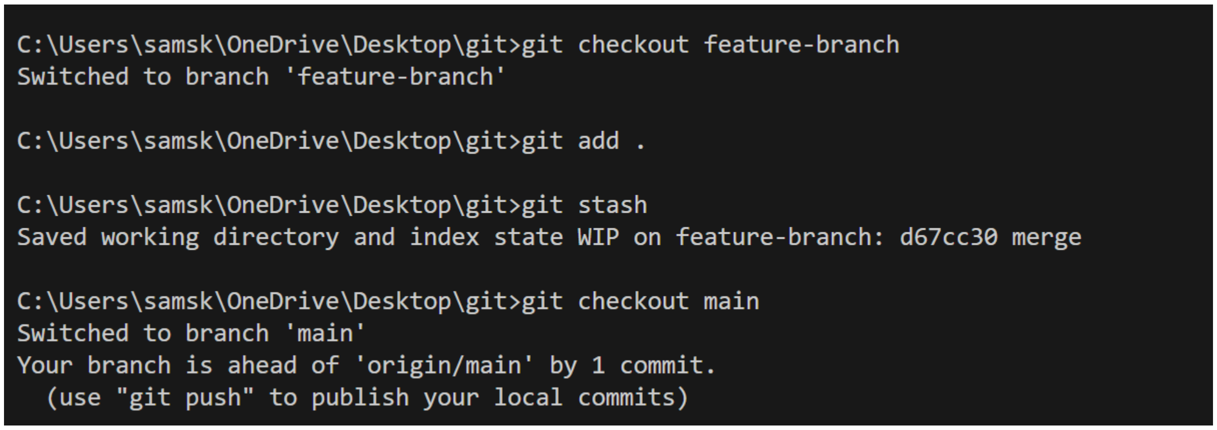
```
C:\Users\samsk\OneDrive\Desktop\git>git add .

C:\Users\samsk\OneDrive\Desktop\git>git commit -m "merge"
[feature-branch d67cc30] merge
 1 file changed, 1 insertion(+)
 create mode 100644 test.py
```

b.

- Navigate to the feature-branch
- Make any required changes, stash them using the using the command – git stash
- Navigate to the main branch and make any required changes and stage and commit them.
- Now go back to the feature-branch. If you want the stashed changes to be reflected, use the command – git stash apply.
- Later, these changes can be staged and committed.



```
C:\Users\samsk\OneDrive\Desktop\git>git merge feature-branch
Updating 215175e..d67cc30
Fast-forward
 test.py | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 test.py

C:\Users\samsk\OneDrive\Desktop\git>git checkout feature-branch
```



```
C:\Users\samsk\OneDrive\Desktop\git>git checkout feature-branch
Switched to branch 'feature-branch'

C:\Users\samsk\OneDrive\Desktop\git>git add .

C:\Users\samsk\OneDrive\Desktop\git>git stash
Saved working directory and index state WIP on feature-branch: d67cc30 merge

C:\Users\samsk\OneDrive\Desktop\git>git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)
```

EXPERIMENT No.03: Collaboration and Remote Repositories

AIM:

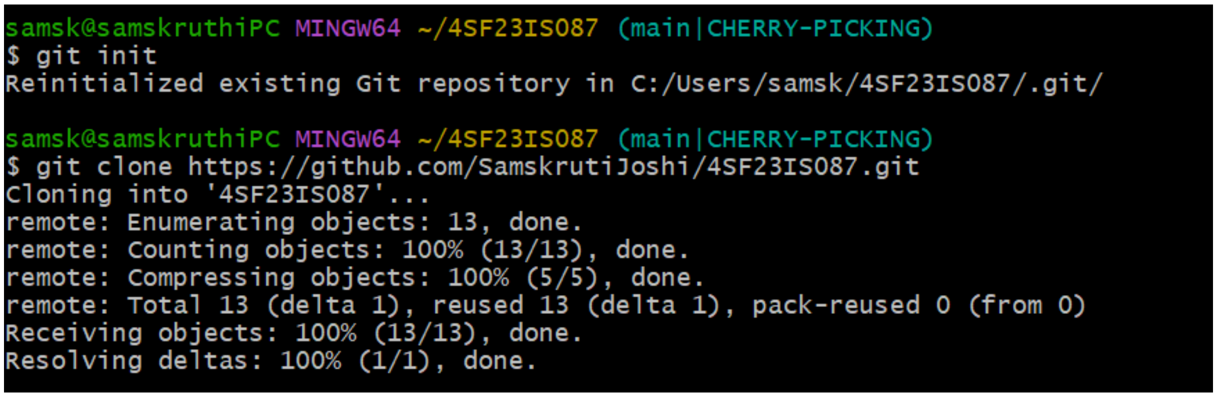
a. Clone a remote Git repository to your local machine.
b. Fetch the latest changes from a remote repository and rebase your local branch onto the updated remote branch.
c. Write a command to merge "feature-branch" into "master" while providing a custom commit message for the merge.

COMMANDS:

1. git clone <repository_url>
2. git checkout -b feature-branch
3. git fetch origin
4. git rebase origin
5. git merge --no-ff feature-branch -m "Custom commit message"

PROCEDURE AND RESULTS:

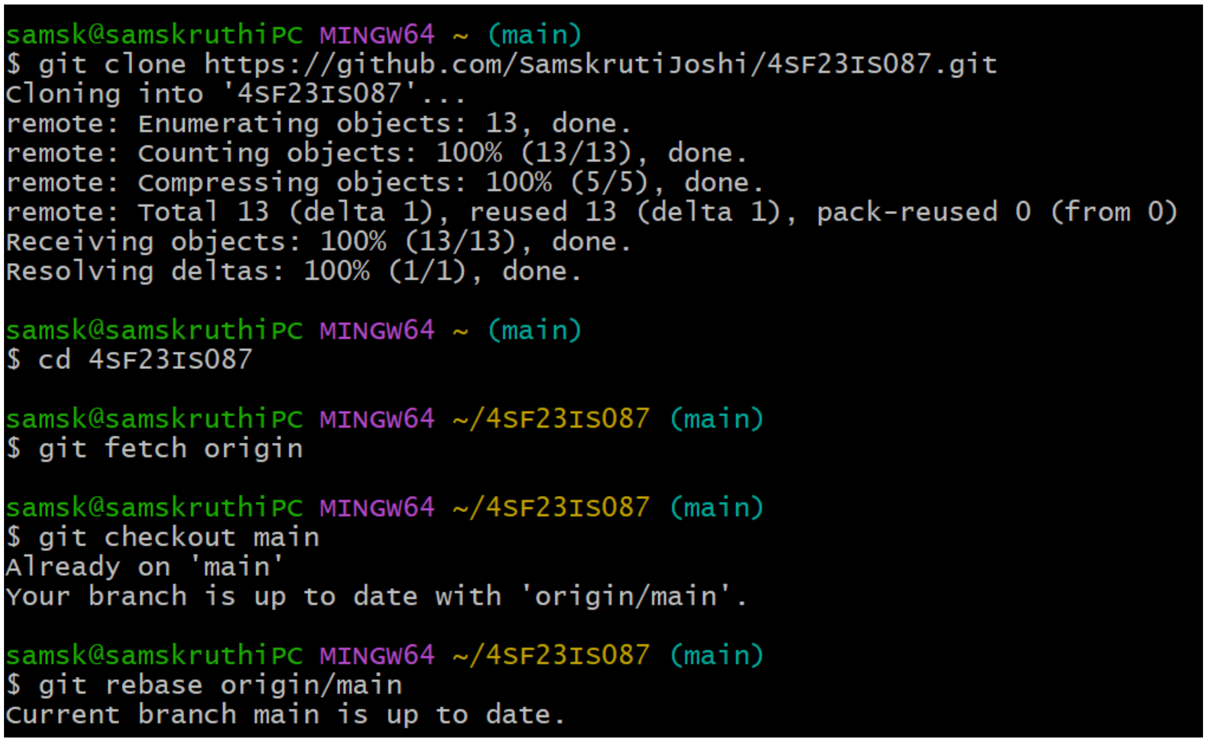
a. Clone
- To clone a remote repository, use the command – git clone <repository_url>



```
samsk@samskruthiPC MINGW64 ~/4SF23IS087 (main|CHERRY-PICKING)
$ git init
Reinitialized existing Git repository in C:/Users/samsk/4SF23IS087/.git/

samsk@samskruthiPC MINGW64 ~/4SF23IS087 (main|CHERRY-PICKING)
$ git clone https://github.com/SamskrutiJoshi/4SF23IS087.git
Cloning into '4SF23IS087'...
remote: Enumerating objects: 13, done.
remote: Counting objects: 100% (13/13), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 13 (delta 1), reused 13 (delta 1), pack-reused 0 (from 0)
Receiving objects: 100% (13/13), done.
Resolving deltas: 100% (1/1), done.
```

b. Fetch and Rebase

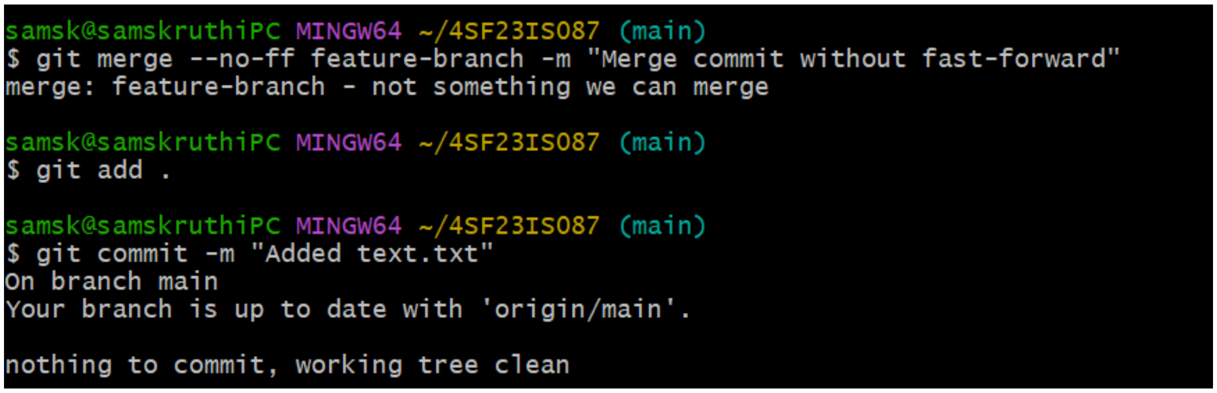
- To fetch the latest changes in the remote repository into your local repository, use the command - git fetch origin
- To merge the fetched changes onto your current working branch, use the command -git rebase origin.



```
samsk@samskruthiPC MINGW64 ~ (main)
$ git clone https://github.com/SamskrutiJoshi/4SF23IS087.git
Cloning into '4SF23IS087'...
remote: Enumerating objects: 13, done.
remote: Counting objects: 100% (13/13), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 13 (delta 1), reused 13 (delta 1), pack-reused 0 (from 0)
Receiving objects: 100% (13/13), done.
Resolving deltas: 100% (1/1), done.

samsk@samskruthiPC MINGW64 ~ (main)
$ cd 4SF23IS087

samsk@samskruthiPC MINGW64 ~/4SF23IS087 (main)
$ git fetch origin

samsk@samskruthiPC MINGW64 ~/4SF23IS087 (main)
$ git checkout main
Already on 'main'
Your branch is up to date with 'origin/main'.

samsk@samskruthiPC MINGW64 ~/4SF23IS087 (main)
$ git rebase origin/main
Current branch main is up to date.
```

c. Merge with message

- Navigate to the main branch
- To merge the feature-branch into the current branch without fast-forwarding, use the command – git merge –no-ff feature-branch -m “Commit message” with suitable message



```
samsk@samskruthiPC MINGW64 ~/4SF23IS087 (main)
$ git merge --no-ff feature-branch -m "Merge commit without fast-forward"
merge: feature-branch - not something we can merge

samsk@samskruthiPC MINGW64 ~/4SF23IS087 (main)
$ git add .

samsk@samskruthiPC MINGW64 ~/4SF23IS087 (main)
$ git commit -m "Added text.txt"
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

EXPERIMENT No.04: Git Tags and Releases

AIM:

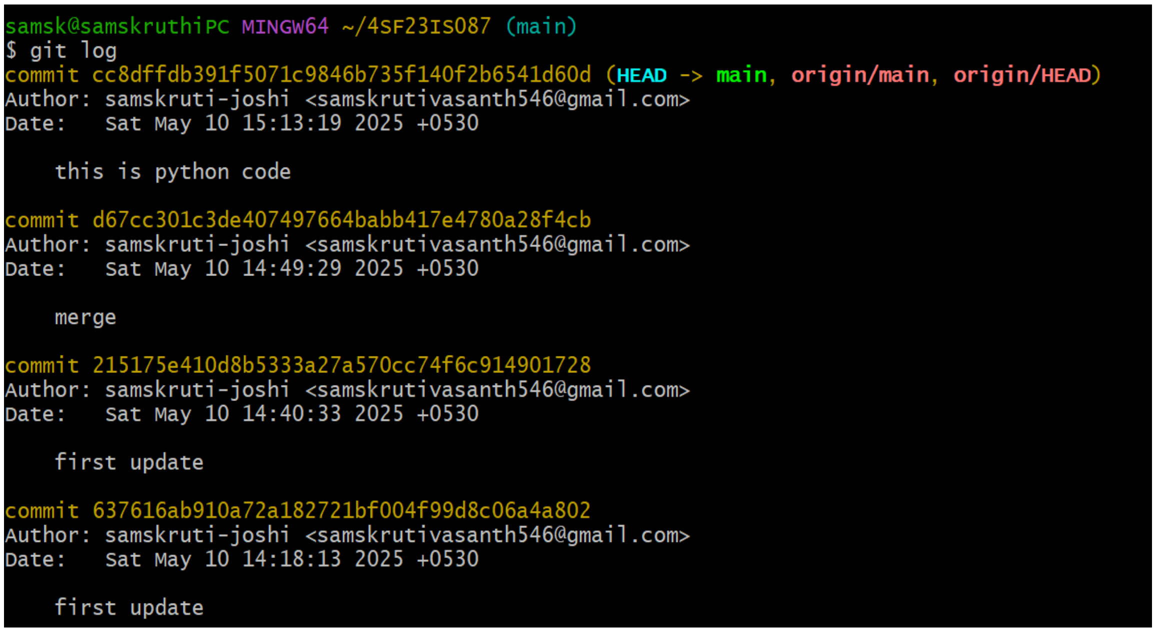
Write the command to create a lightweight Git tag named "v1.0" for a commit in your local repository.

COMMANDS:

1. git log
2. git tag <tag_name>
3. git show < tag_name >
4. git push origin < tag_name>

PROCEDURE AND RESULTS:

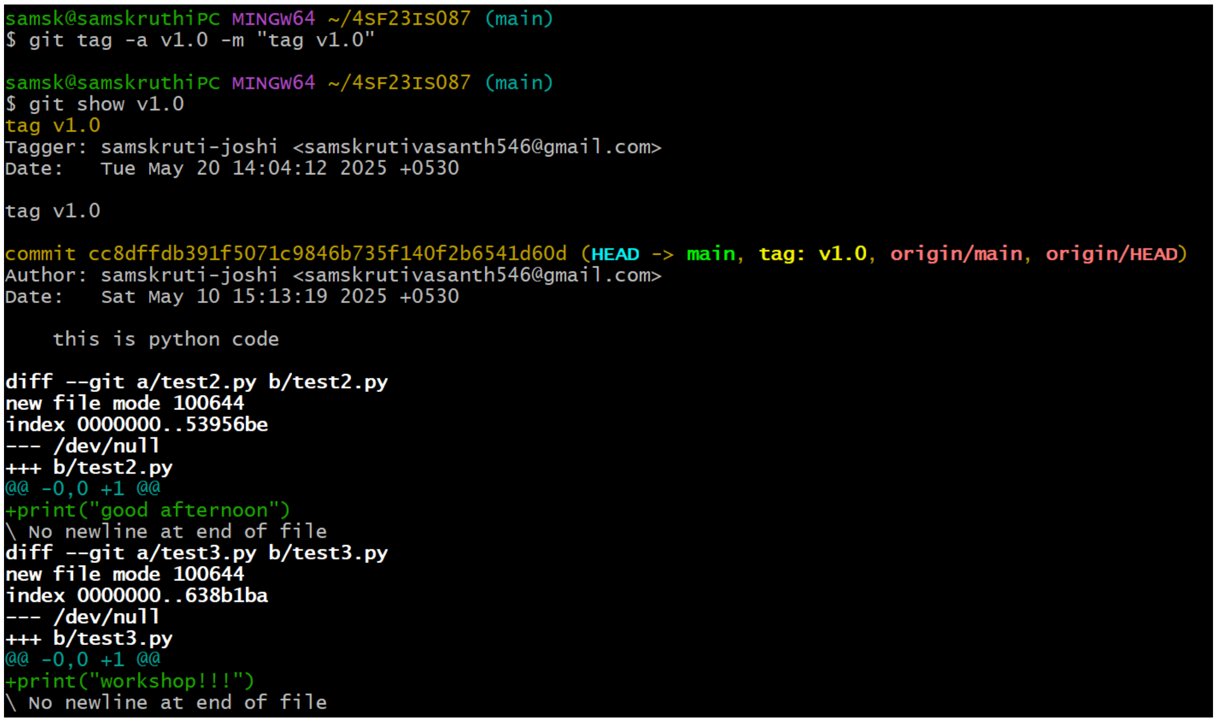
- To view the commit history, use the command - git log



```
samsk@samskruthiPC MINGW64 ~/4SF23IS087 (main)
$ git log
commit cc8dffdb391f5071c9846b735f140f2b6541d60d (HEAD -> main, origin/main, origin/HEAD)
Author: samskruti-joshi <samskrutivasanth546@gmail.com>
Date:   Sat May 10 15:13:19 2025 +0530

    this is python code

commit d67cc301c3de407497664babb417e4780a28f4cb
Author: samskruti-joshi <samskrutivasanth546@gmail.com>
Date:   Sat May 10 14:49:29 2025 +0530

    merge

commit 215175e410d8b5333a27a570cc74f6c914901728
Author: samskruti-joshi <samskrutivasanth546@gmail.com>
Date:   Sat May 10 14:40:33 2025 +0530

    first update

commit 637616ab910a72a182721bf004f99d8c06a4a802
Author: samskruti-joshi <samskrutivasanth546@gmail.com>
Date:   Sat May 10 14:18:13 2025 +0530

    first update
```

- To create a lightweight Git tag, use the command - git tag <tag_name>
- To verify that the tag was created correctly and is pointing to the desired commit, use the command – git show <tag_name>

- To push the tag onto the remote repository, use the command - git push origin < tag_name>



```
samsk@samskruthiPC MINGW64 ~/4SF23IS087 (main)
$ git tag -a v1.0 -m "tag v1.0"

samsk@samskruthiPC MINGW64 ~/4SF23IS087 (main)
$ git show v1.0
tag v1.0
Tagger: samskruti-joshi <samskrutivasanth546@gmail.com>
Date:   Tue May 20 14:04:12 2025 +0530

tag v1.0

commit cc8dffdb391f5071c9846b735f140f2b6541d60d (HEAD -> main, tag: v1.0, origin/main, origin/HEAD)
Author: samskruti-joshi <samskrutivasanth546@gmail.com>
Date:   Sat May 10 15:13:19 2025 +0530

    this is python code

diff --git a/test2.py b/test2.py
new file mode 100644
index 0000000..53956be
--- /dev/null
+++ b/test2.py
@@ -0,0 +1 @@
+print("good afternoon")
\ No newline at end of file
diff --git a/test3.py b/test3.py
new file mode 100644
index 0000000..638b1ba
--- /dev/null
+++ b/test3.py
@@ -0,0 +1 @@
+print("workshop!!!")
\ No newline at end of file
```

EXPERIMENT No.05: Advanced Git Operations

AIM:

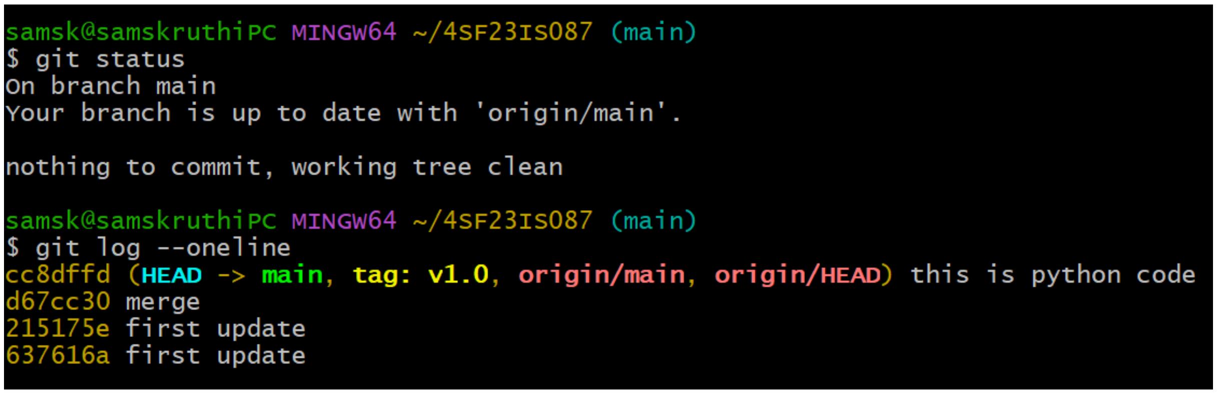
Write a command to cherry-pick a range of commits from "source-branch" to the current branch.

COMMANDS:

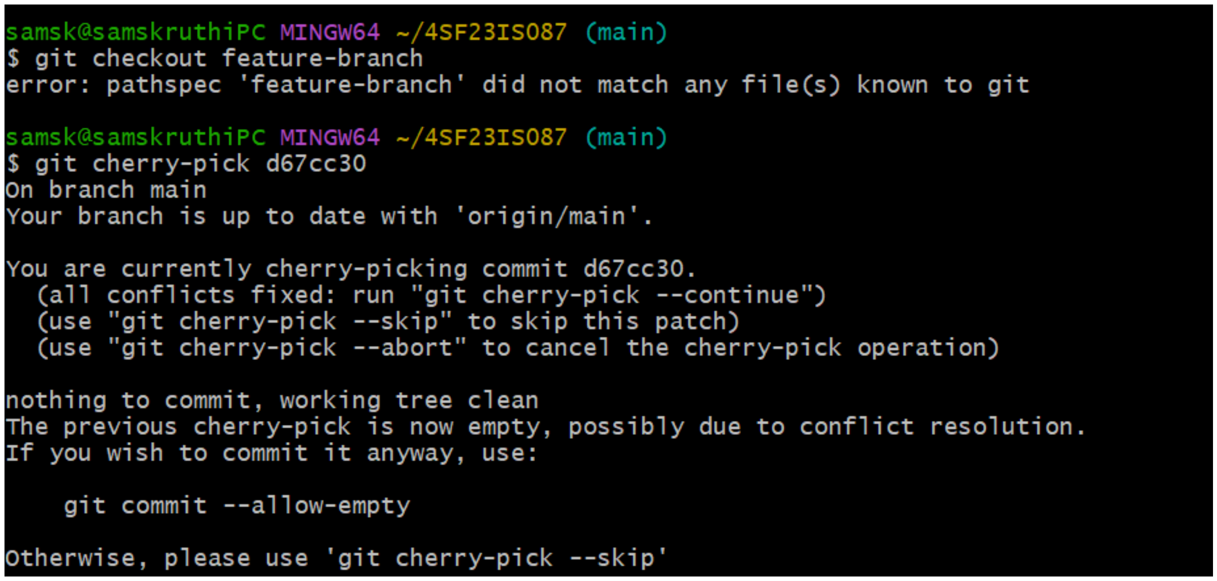
1. git status
2. git log –oneline
3. git checkout master
4. git cherry-pick <commit_id>

PROCEDURE AND RESULTS:

- Shows the current state of the working directory and staging area (which files are staged, unstaged, or untracked) - git status
- Displays the commit history in a compact, one-line-per-commit format. – git log --oneline
- Switches to the branch named feature-branch. – git checkout feature-branch
- Copy the commit_id to cherry pick. Navigate to the main branch.
- Now perform the cherry-pick operation by using the command git cherry-pick <commit id>



```
samsk@samskruthiPC MINGW64 ~/4SF23IS087 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

samsk@samskruthiPC MINGW64 ~/4SF23IS087 (main)
$ git log --oneline
cc8dffd (HEAD -> main, tag: v1.0, origin/main, origin/HEAD) this is python code
d67cc30 merge
215175e first update
637616a first update
```



```
samsk@samskruthiPC MINGW64 ~/4SF23IS087 (main)
$ git checkout feature-branch
error: pathspec 'feature-branch' did not match any file(s) known to git

samsk@samskruthiPC MINGW64 ~/4SF23IS087 (main)
$ git cherry-pick d67cc30
On branch main
Your branch is up to date with 'origin/main'.

You are currently cherry-picking commit d67cc30.
  (all conflicts fixed: run "git cherry-pick --continue")
  (use "git cherry-pick --skip" to skip this patch)
  (use "git cherry-pick --abort" to cancel the cherry-pick operation)

nothing to commit, working tree clean
The previous cherry-pick is now empty, possibly due to conflict resolution.
If you wish to commit it anyway, use:

    git commit --allow-empty

Otherwise, please use 'git cherry-pick --skip'
```

EXPERIMENT 6: Analysing and Changing Git History

AIM:

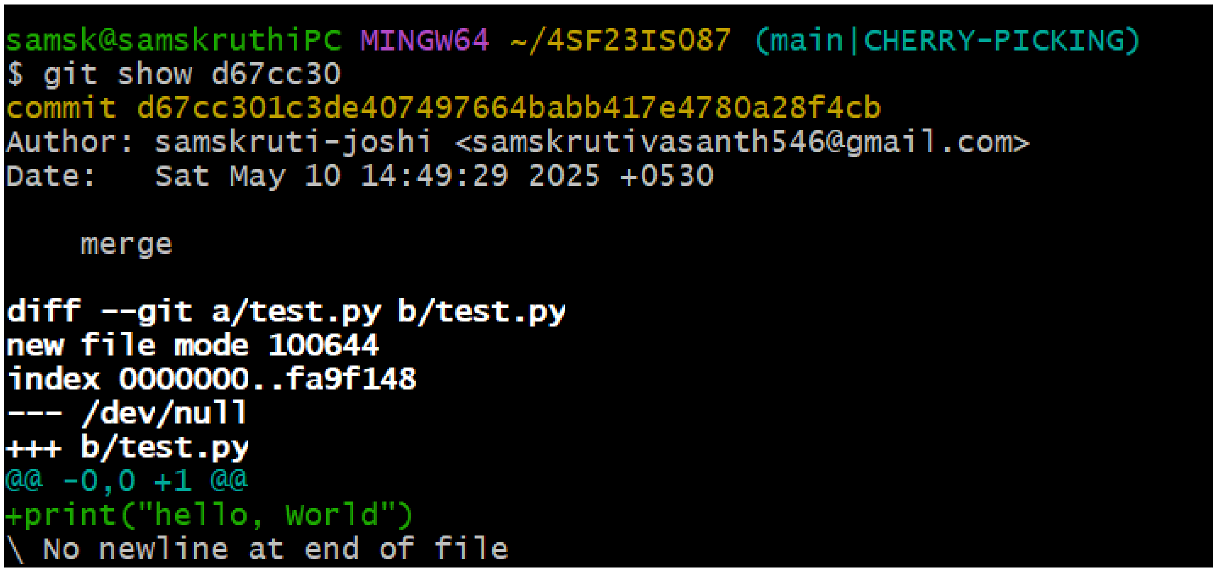
a. Given a commit ID, how would you use Git to view the details of that specific commit, including the author, date, and commit message?
b. Write the command to list all commits made by the author "JohnDoe" between "2023-01-01" and "2023-12-31."
c. Write the command to display the last five commits in the repository's history.
d. Write the command to undo the changes introduced by the commit with the ID "abc123".

COMMANDS:

1. git show <commit_id>
2. git log --author=<Username> --after=<start_date> --before<stop_date>.
3. git log -n
4. git revert abc123

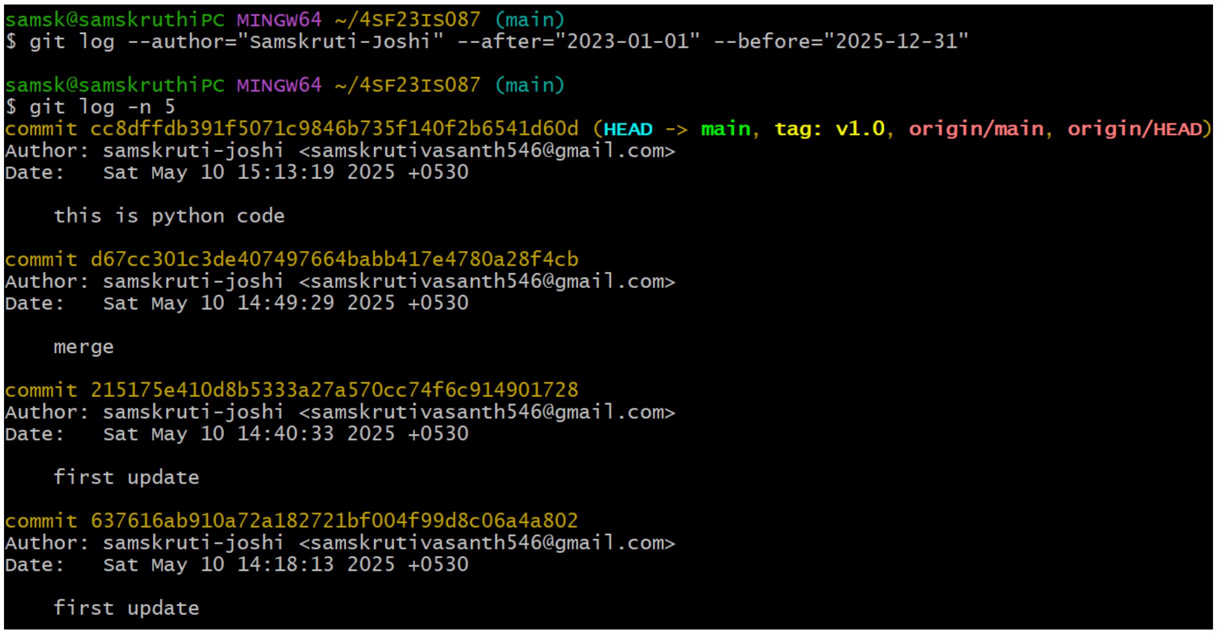
PROCEDURE AND RESULTS:

- To find the details of a particular commit with its associated commit id by using the command - git show <commit id>



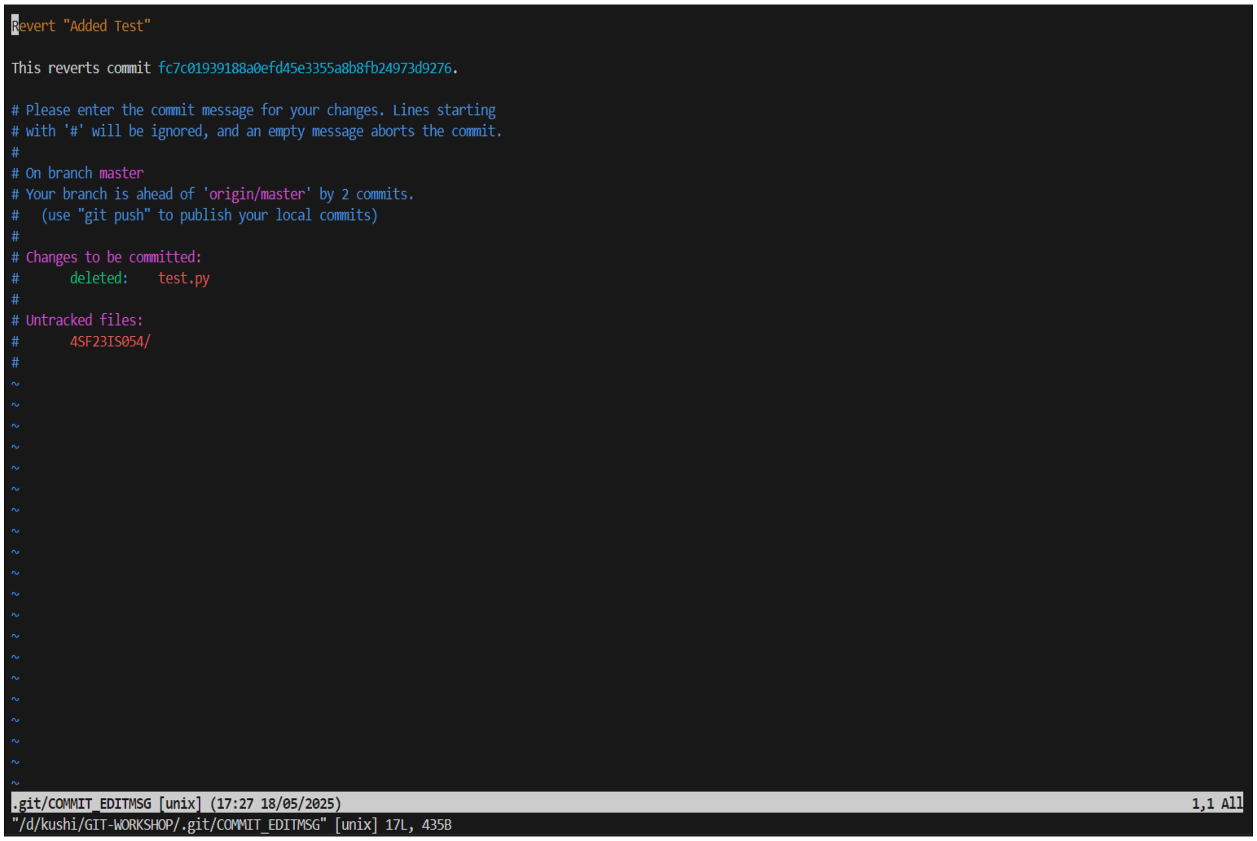
```
samsk@samskruthiPC MINGW64 ~/4SF23IS087 (main|CHERRY-PICKING)
$ git show d67cc30
commit d67cc301c3de407497664babb417e4780a28f4cb
Author: samskruti-joshi <samskrutivasanth546@gmail.com>
Date:   Sat May 10 14:49:29 2025 +0530

    merge

diff --git a/test.py b/test.py
new file mode 100644
index 0000000..fa9f148
--- /dev/null
+++ b/test.py
@@ -0,0 +1 @@
+print("hello, World")
\ No newline at end of file
```

To find the details of all the commits done by an author during a particular time interval, use the command - git log --author=Username --after=<start_date> --before<stop_date>.



```
samsk@samskruthiPC MINGW64 ~/4SF23IS087 (main)
$ git log --author="Samskruti-Joshi" --after="2023-01-01" --before="2025-12-31"

samsk@samskruthiPC MINGW64 ~/4SF23IS087 (main)
$ git log -n 5
commit cc8dffdb391f5071c9846b735f140f2b6541d60d (HEAD -> main, tag: v1.0, origin/main, origin/HEAD)
Author: samskruti-joshi <samskrutivasanth546@gmail.com>
Date:   Sat May 10 15:13:19 2025 +0530

    this is python code

commit d67cc301c3de407497664babb417e4780a28f4cb
Author: samskruti-joshi <samskrutivasanth546@gmail.com>
Date:   Sat May 10 14:49:29 2025 +0530

    merge

commit 215175e410d8b5333a27a570cc74f6c914901728
Author: samskruti-joshi <samskrutivasanth546@gmail.com>
Date:   Sat May 10 14:40:33 2025 +0530

    first update

commit 637616ab910a72a182721bf004f99d8c06a4a802
Author: samskruti-joshi <samskrutivasanth546@gmail.com>
Date:   Sat May 10 14:18:13 2025 +0530

    first update
```

- To find the list of the last m commits, staged by the user, use the git log -n m.
- To undo the changes introduced by the commit with a particular id, use the command - git revert <commit_id> , where <commit_id> is the id of the commit to be reverted.



```
Revert "Added Test"

This reverts commit fc7c01939188a0efd45e3355a8b8fb24973d9276.

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
# Your branch is ahead of 'origin/master' by 2 commits.
#   (use "git push" to publish your local commits)
#
# Changes to be committed:
#       deleted:    test.py
#
# Untracked files:
#       4SF23IS054/
#
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
.git/COMMIT_EDITMSG [unix] (17:27 18/05/2025)                                                              1,1 All
"/d/kushi/GIT-WORKSHOP/.git/COMMIT_EDITMSG" [unix] 17L, 435B
```