

1. Make a wrong commit intentionally

2. Undo it using:

- `git reset --soft` OR
- `git revert`

3. Explain (in PR comment or README):

- Which command you used
- Why

```
hp@hp-HP-EliteBook-840-G3:~/GitAssignment5$ git clone https://github.com/Ankita-Advitot/GitAssignment5.git
Cloning into 'GitAssignment5'...
warning: You appear to have cloned an empty repository.
```

```
hp@hp-HP-EliteBook-840-G3:~/GitAssignment5$ touch readme.md
hp@hp-HP-EliteBook-840-G3:~/GitAssignment5$ nano readme.md
```

```
hp@hp-HP-EliteBook-840-G3:~/GitAssignment5$ git add .
hp@hp-HP-EliteBook-840-G3:~/GitAssignment5$ git commit -m "Wrong commit"
[main (root-commit) 9f9dc52] Wrong commit
 1 file changed, 1 insertion(+)
  create mode 100644 readme.md
hp@hp-HP-EliteBook-840-G3:~/GitAssignment5$ git push origin main
Username for 'https://github.com': Ankita-Advitot
Password for 'https://Ankita-Advitot@github.com':
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 259 bytes | 86.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Ankita-Advitot/GitAssignment5.git
 * [new branch]      main -> main
```

```
hp@hp-HP-EliteBook-840-G3:~/GitAssignment5$ git revert HEAD
[main e802b16] Revert "Wrong commit"
 1 file changed, 1 deletion(-)
 delete mode 100644 readme.md
hp@hp-HP-EliteBook-840-G3:~/GitAssignment5$ git log --oneline
e802b16 (HEAD -> main) Revert "Wrong commit"
9f9dc52 (origin/main) Wrong commit
hp@hp-HP-EliteBook-840-G3:~/GitAssignment5$ nano readme.md
hp@hp-HP-EliteBook-840-G3:~/GitAssignment5$ nano readme.md
hp@hp-HP-EliteBook-840-G3:~/GitAssignment5$ git add .
hp@hp-HP-EliteBook-840-G3:~/GitAssignment5$ git commit -m "wrong commit for soft reset"
[main 6d20127] wrong commit for soft reset
 1 file changed, 5 insertions(+)
 create mode 100644 readme.md
hp@hp-HP-EliteBook-840-G3:~/GitAssignment5$ git log --oneline
6d20127 (HEAD -> main) wrong commit for soft reset
e802b16 Revert "Wrong commit"
9f9dc52 (origin/main) Wrong commit
```

```
GNU nano 7.2                                         readme.md *
```

This is doing for soft reset

```
^G Help      ^O Write Out   ^W Where Is    ^K Cut        ^T Execute    ^C Location   M-U Undo     M-A Set Mark
^X Exit      ^R Read File   ^\ Replace    ^U Paste     ^J Justify    ^/ Go To Line M-E Redo     M-G Copy
```

```
hp@hp-HP-EliteBook-840-G3:~/GitAssignment5$ nano readme.md
hp@hp-HP-EliteBook-840-G3:~/GitAssignment5$ nano readme.md
hp@hp-HP-EliteBook-840-G3:~/GitAssignment5$ git add .
hp@hp-HP-EliteBook-840-G3:~/GitAssignment5$ git commit -m "Committing after fixing issue using reset --soft "
[main 31d7dbd] Committing after fixing issue using reset --soft
 1 file changed, 7 insertions(+)
 create mode 100644 readme.md
hp@hp-HP-EliteBook-840-G3:~/GitAssignment5$ git push origin main
Username for 'https://github.com': Ankita-Advitot
Password for 'https://Ankita-Advitot@github.com':
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (5/5), 553 bytes | 276.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Ankita-Advitot/GitAssignment5.git
 9f9dc52..31d7dbd  main -> main
hp@hp-HP-EliteBook-840-G3:~/GitAssignment5$
```

```
GNU nano 7.2                                         readme.md *
```

This is doing for soft reset

```
^G Help      ^O Write Out   ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo    M-A Set Mark
^X Exit     ^R Read File   ^\ Replace    ^U Paste     ^J Justify    ^/ Go To Line M-E Redo    M-G Copy
```

```
9f9dc52 (origin/main) Wrong commit
hp@hp-HP-EliteBook-840-G3:~/GitAssignment5$ nano readme.md
hp@hp-HP-EliteBook-840-G3:~/GitAssignment5$ nano readme.md
hp@hp-HP-EliteBook-840-G3:~/GitAssignment5$ git add .
hp@hp-HP-EliteBook-840-G3:~/GitAssignment5$ git commit -m "wrong commit for soft reset"
[main 6d20127] wrong commit for soft reset
 1 file changed, 5 insertions(+)
 create mode 100644 readme.md
hp@hp-HP-EliteBook-840-G3:~/GitAssignment5$ git log --oneline
6d20127 (HEAD -> main) wrong commit for soft reset
e802b16 Revert "Wrong commit"
9f9dc52 (origin/main) Wrong commit
```

```
9f9dc52 (origin/main) Wrong commit
hp@hp-HP-EliteBook-840-G3:~/GitAssignment5$ git reset --soft
hp@hp-HP-EliteBook-840-G3:~/GitAssignment5$ git log --oneline
6d20127 (HEAD -> main) wrong commit for soft reset
e802b16 Revert "Wrong commit"
9f9dc52 (origin/main) Wrong commit
hp@hp-HP-EliteBook-840-G3:~/GitAssignment5$ git reset --soft HEAD-1
hp@hp-HP-EliteBook-840-G3:~/GitAssignment5$ git log --oneline
e802b16 (HEAD -> main) Revert "Wrong commit"
9f9dc52 (origin/main) Wrong commit
hp@hp-HP-EliteBook-840-G3:~/GitAssignment5$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   readme.md
```

```
hp@hp-HP-EliteBook-840-G3:~/GitAssignment5$ nano readme.md
hp@hp-HP-EliteBook-840-G3:~/GitAssignment5$ nano readme.md
hp@hp-HP-EliteBook-840-G3:~/GitAssignment5$ git add .
hp@hp-HP-EliteBook-840-G3:~/GitAssignment5$ git commit -m "Committing after fixing issue using reset --soft"
[main 31d7dbd] Committing after fixing issue using reset --soft
 1 file changed, 7 insertions(+)
 create mode 100644 readme.md
hp@hp-HP-EliteBook-840-G3:~/GitAssignment5$ git push origin main
Username for 'https://github.com': Ankita-Advitot
Password for 'https://Ankita-Advitot@github.com':
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (5/5), 553 bytes | 276.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Ankita-Advitot/GitAssignment5.git
 9f9dc52..31d7dbd  main -> main
hp@hp-HP-EliteBook-840-G3:~/GitAssignment5$
```

```
GNU nano 7.2                               README.md

This is doing for soft reset
i have done soft reset now
this is for fixing the wrong code and then again committing

[ Read 7 lines ]
^G Help      ^O Write Out    ^W Where Is    ^K Cut        ^T Execute    ^C Location   M-U Undo    M-A Set Mark
^X Exit      ^R Read File    ^\ Replace     ^U Paste      ^J Justify    ^/ Go To Line M-E Redo    M-G Copy
```

What `git revert` Does

- Keeps the wrong commit in history (so reviewers can see it)
- Adds a new commit that cancels its effect
- Makes the project correct again without confusion

Example:

Commit 1 → Wrong change ✗

Commit 2 → Revert commit ✓

1 Difference between `git revert HEAD` and `git revert`

- ◆ `git revert HEAD`

👉 What it means

- **HEAD = the latest commit**
- **So this command reverts only the last commit**

👉 In simple words

“Undo the most recent commit safely”

Example:

`git revert HEAD`

- ✓ Reverts only the latest commit
 - ✓ Creates a new commit
 - ✓ Safe for PRs and shared repos
-

- ◆ `git revert (without anything)`

👉 What happens

- Git will ask you which commit you want to revert
- It opens an editor or shows options

👉 Usually not used alone in practice
People normally specify:

- HEAD
 - or a commit hash
-

2 Revert a Specific Commit (not the latest)

- ◆ When you want to undo a particular commit

First, find the commit ID:

```
git log --oneline
```

Example output:

a1b2c3d Added wrong logic

e4f5g6h Added README

Now revert the specific commit:

```
git revert a1b2c3d
```

👉 This will:

- Undo only that commit
 - Keep all other commits safe
 - Create a new “revert” commit
- ✓ Best for shared repositories
-

3 What if you want to DELETE a commit?

⚠ Important: Deleting commits is dangerous after pushing

- ◆ If commit is NOT pushed (safe case)

You can delete it using:

```
git reset --hard HEAD~1
```

👉 This:

- Deletes the last commit completely
- Deletes changes also

⚠ Only use this before pushing

- ◆ Delete a specific older commit (advanced & risky)

This requires:

git rebase -i

Example:

git rebase -i HEAD~4

Then change:

pick a1b2c3d wrong commit

to:

drop a1b2c3d wrong commit

⚠ Never do this on a shared branch without permission

4 Simple Comparison Table

Task	Command	Safe for PR?
------	---------	--------------

Undo last commit	<code>git revert HEAD</code>	Yes
Undo specific commit	<code>git revert <commit-id></code>	Yes
Delete last commit (local only)	<code>git reset --hard HEAD~1</code>	No
Delete old commit	<code>git rebase -i</code>	No

5 Best Rule to Remember 🧠

- 👉 Already pushed? → Use `git revert`
- 👉 Not pushed yet? → You may use `git reset`

◆ What is `git reset --soft`?

`git reset --soft` undoes a commit but keeps all the changes in the staging area.

👉 In simple words:

“Remove the commit, but keep my code exactly as it is and ready to commit again.”

◆ Syntax

```
git reset --soft HEAD~1
```

◆ What this command does (step-by-step)

Assume you made a wrong commit.

When you run:

```
git reset --soft HEAD~1
```

✓ What happens:

- ✗ The **last commit is removed**
- ✗ All file changes are **still staged** (`git add` not lost)
- ✗ You can fix the mistake and **commit again**

✗ What does NOT happen:

- Files are **not deleted**
- Code changes are **not lost**
- Working directory stays the same

◆ When to Use `git reset --soft`

Use it when:

- You **haven't pushed** the commit yet
- You want to:
 - Change commit message
 - Add/remove files
 - Combine commits (squash)

◆ Example (Very Simple)

Step 1: Wrong commit

```
git commit -m "Wrong commit message"
```

Step 2: Undo commit but keep changes

```
git reset --soft HEAD~1
```

Step 3: Fix and commit again

```
git commit -m "Correct commit message"
```

◆ Difference Between Reset Types

Command	Commit Removed	Changes Kept	Safe after push
--soft	<input checked="" type="checkbox"/> Yes	<input checked="" type="checkbox"/> Staged	<input checked="" type="checkbox"/> No
--mixed	<input checked="" type="checkbox"/> Yes	<input checked="" type="checkbox"/> Unstaged	<input checked="" type="checkbox"/> No
--hard	<input checked="" type="checkbox"/> Yes	<input checked="" type="checkbox"/> Deleted	<input checked="" type="checkbox"/> No

◆ Important Warning !

 Do NOT use `git reset --soft` after pushing to GitHub

Why?

- It **rewrites commit history**
- Causes issues for teammates
- Breaks Pull Requests

 After push, always use:

```
git revert
```

- ◆ **One-Line Summary (Easy to Remember)**

`git reset --soft` removes the commit but keeps all changes staged.