Process:

- ❖Create a folder on the desktop/documents.
- ❖Open vs code
- ❖File-open folder-selected the created folder.
- ❖Create a new file in that folder in vscode
- ❖Write your source code.
- ❖Run and debug
- ❖Run these commands:
    - ➢git –version
    - ➢git –global –list
    - ➢git init
    - ➢git add filename along with the extension
    - ➢Git commit

OUTPUT:

```
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
#
# Initial commit
#
# Changes to be committed:
#       new file:   name.py
.git/COMMIT_EDITMSG [unix] (20:32 04/02/2025)
```

1,0-1 Top"~/Desktop/git practice/.git/COMMIT_EDITMSG" [unix] 11L, 229B

- Open a new terminal - beside runcode-...- u can select the new terminal.

The commands that should be executed are highlighted below:

- PS C:\Users\Administrator\Desktop\git practice> git commit -m "MY FIRST COMMIT"
- [master (root-commit) 3ac0309] MY FIRST COMMIT
- 1 file changed, 1 insertion(+)
- create mode 100644 name.py
- PS C:\Users\Administrator\Desktop\git practice> git init -b main
- warning: re-init: ignored --initial-branch=main
- Reinitialized existing Git repository in C:/Users/Administrator/Desktop/git practice/.git/
- PS C:\Users\Administrator\Desktop\git practice> git branch master

GITHUB:

- Create repository
- Name the remote repo same as the local repo
- 2 types:
  - Private: URL will be generated. If someone should access that then that person should have the permissions which are granted by the owner.
  - Public: Can Be accessible by everyone, they can be changed without the permissions.
- Desc: optional the description about the application will be described.
- Then a new global repo will be created.
- If we need to edit the desc(ReadMe file) can be done after creating the global repository.
- Inorder to access the local repo files should be done by pairing the SSH key.

SSH	HTTPS
SSH is a public key cryptography protocol that ensures no one can change the data during transfer	HTTPS uses SLS/TLS protocol and it is easy to configure.
More difficult to setup and is not that wide spread	Easy to set up and is wide spread.
Offers high data integrity and security	Offers low data security level.
Some Firewalls refuse to take the SSH connections on the default port.	Uses token based authentication to create connections on port 443
No need to provide the username and password for every action	Need to provide the username and password for every action.
Benefits: One time Setup Improved Security Time Saving	Benefits: Simple setup Availability Portability

GIT STAGES:

Working Directory: The untracked file storage area(vscode/local repo)
git init file will be created(.py/.txt)

Staging Area: acts as a cache where the changes has been made before committing into the repo.
git add filename:- to add single file
git add all/ git f1 f2 / git add –all /git -a:- to add multiple files.

Local Repository: Own laptops
git commit:- files move from local repo - remote repo

Remote Repository: Git Server
Files will move to the local repo

UNDERSTANDING OF GIT COMMANDS:

- “ git –version ” : checks the version of the git that is installed.
- “git config –global –list ” : check’s the identities present other than the author
- “ git config –global user.name ‘your name’ ” : set author’s name

- “ git config –global user.email ‘your email’ ” : set author’s email
- “ git config –list ”: checks the current configuration.
- “ git init ”: Initialize a new git repo.
- “ git clone url_repo ”: clone a repo which is existing by using an URL.
- “ git status”: this command shows all the files which need to be committed.
- “ git add filename”:adds the specific file to the staging area.
- “git add all/ git f1 f2 / git add –all /git -a”: adds multiple files to the staging area.
- “git commit -m “commit message:commits the fille/files which are present in the staging area.
- “git branch”: shows in which branch the file/files were committed.
- “git branch <branch name>”: creates a new branch.
- “git checkout <branch name>”: switches the file from one branch to another branch.
- “git checkout -b <branch name>”: creates and switches the files from one branch to another.
- “git merge <branch name>”: merges the branch history into the current one.
- “git branch -d <branch name>”: deletes a branch.
- “ git diff”: this command shows the files which are not yet staged/ which are in the working directory.

- “git diff -staged”: this command shows the difference between the files that are staged and latest committed version.
- “git diff [first branch] [second branch]”: shows the difference between the two branches.
- “git reset [file]”: un stages the file which is present in the staging area but does not delete the file in the working directory.
- “git reset [commit] ”: resets the current commit to the previous commit.
- 3 modes:
- “git reset –soft [commit] ”: resets the current commit to the previous commit but keeps the changes in the staging area.
- “git reset –mixed [commit] ”: resets the current commit to the previous commit and un stages the file from the staging area.
- “git reset –hard [commit] ”: resets the current commit to the previous commit and discards all the history.
- “git rm [file] ”: delete the file from working directory and the staging area.
- “git rm -f [file] ”: forces the deletion of the file from working directory and the staging area.
- “git rm -cached [file] ”: remove the file from git tracking but keeps it locally.
- “git rm -r [directory] ”: remove the directory and all it’s files.