1. git init ---------> for initi git repository

2. git config --global user.name "username"

3. git config --global user.email "e-mail-id"

4. ssh-keygen -C "key name" --------> C for windows

5. git remote add origin <ssh of repo of github>

    a. origin--> your own account in GitHub (chetansomkuwar/master)

    b. any repository that starts with our own account = "Origin" or URL

6. git remote -v ------------> for check the ssh is connected or not

    git remote remove origin --------> REMOVE origin or SSH loging

7. git add file1 ------> from working area to staging area

    git add . (ALL)

   git reset file1 -------> remove file from staging area

    git reset . (ALL)

   git reset --hard --------> remove file from WORKING AREA + STAGING AREA

    gti reset <version id> -------> via reset pasted commit ID >> In git log above all commit ID are remove in history/logs

8. git commit -m "any comment" --------> staging are to local repo

9. git status

10. git push origin master --------> code goes to remote repo in GitHub

    git push upstream <branch_name> --------->

11. git pull origin master ----------> to pull code from remote repo to working directory

12. git branch ---------------> to list branches

13. git branch <branch_name> ----------> to create branch

14. git checkout <branch_name> --------> switch into another branch

15. git checkout -b "<branch_name>" ----------> to create & switch into that same branch

git branch -d <branch_name> (After Merging data will available in both "Master & Branch also" then we delete branch that's why we use -d)

git branch -D <branch_name> (Forcefully delete branch)

16. git log ---------> to check the logs

    git log -10 --------> to check recent 10 logs

    git log --oneline -----------> to see the logs in one line with version id

17. git revert <version id/commit id> ---------> go to previous version

18. git fetch -------> remote repo to local repo / file contents are not visible

19. git remote rm <fork repo name> ---------> to remove fork

20. git clone <URL of remote repo form GitHub> -----------> create exact replica in our working directory.

21. git stash ------> code go to stash area without commit id/logs, it will not saved log and commit ID

22. git stash list -------> list all incomplete items in stash.

23. git stash apply stash@{4} -------> 4th no file comes back for code editing (stash --> working directory)

    git add. + git commit -m "any comment"

24. git stash clear -----------> code is not committed and gone back.

For good work we should never commit from the "master" branch, because our code will not be finalized yet, it might be some error that's why work
with another branch then user will not be affected with it.

E.g. Create a branch and commit it from that created branch.

git branch jarvis
git checkout jarvis
git commit -m "" --> (out content are committed through branch by open source developer)

git branch spider
git checkout Jarvis
git commit -m "" --> (out content are committed through branch by open source developer)

now, our code is finalized by developer then we merge into master branch for open source use, for other people use.

git merge jarvis (now people can see coede in "master" branch)

now, we can upload in github via push and centrally stored in one repository in github cloud.

git push origin master (additional things is no extra commit created in master branch)

25. "fork" means create a copy of repository in our own acccount.
we can't make directly changes in anothers repository so we can fork it >> edit it >> PR it >> Review Verify & Approved it from forked repo owner.

- Forked repo
- create & checkout branch for editing code (this is good practice)
- [branch_1]: git add
- [branch_1]: git commit -m "commit new one"
- [branch_1]: git log
- [branch_1]: git push origin branch_1

26. Create a new branch in local repo for pull request and changes in code/content in file and then commit it, if we do it in master branch then
master branch will get lots of commit and commit id and this will gets difficult to review it and sorting it all this commits.

27. If i want to remove a commit id from master branch then, select commit id of "below one"

[branch_1]: git log (copy below commit id)
[branch_1]: git reset <commit id>
[branch_1]: git status (our file was untracked)
[branch_1]: git add .
[branch_1]: git stash (code saved into stash area/working dir)

Now my commit will remove in local repository.....................not in my remote repository, so i pushed it into origin

[branch_1]: git push origin branch_1 -f

Now commit will remove in remote repo and content was also removed.

28. git fetch --all --prune

Fork >> clone >> remote add origin >> upstream >> changes >> create branch >> PR >> chetan approve and merge and changes visible into main project

29. Merge conflict?

if Pravin make changes in line no 3 & rupam also make changes in line no 3 then GitHub will confused which one's changes will take, then both rupam & Pravin
make pull request to Chetan, then Chetan will review it "Manually write it >> resolved it >> commit merge"

Solution 2: - Pravin goes to master branch and edit that file, make some little changes and save it after that git add. + git commit -m