Day 1.

1. Concept of version control system.
    a. Version control, also known as source control, is the practice of tracking and managing changes to software code.
    b. Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.
    c. https://www.atlassian.com/git/tutorials/what-is-version-control
2. Concept of Git.
    a. It is a version control system.
    b. Having a distributed architecture, Git is an example of a DVCS (hence Distributed Version Control System). Rather than have only one single place for the full version history of the software as is common in once-popular version control systems like CVS or Subversion (also known as SVN), in Git, every developer's working copy of the code is also a repository that can contain the full history of all changes.
    c. https://www.atlassian.com/git/tutorials/what-is-git
3. Some basic commands.
    a. git init – Initializes a new Git repository. It creates a local git repository for us in our store folder.
    b. git add <file or directory name> – Adds files in to the staging area for Git.
    c. git commit -m “message” – Record the changes made to the files to a local repository.
    d. git status – It gives us all the necessary information about the current branch.
    e. git log – It shows log of commit history.
4. Setting up git.
    a. Download link: https://git-scm.com/
    b. Configure:
        - git config --global user.name "Name"
        - git config --global user.email email@example.com
5. Getting started.
    a. Create a repo in github named demo.
        i. echo "# Demo" >> README.md
        ii. git init
        iii. git add README.md
        iv. git commit -m "first commit"
        v. git branch -M main
        vi. git remote add origin https://github.com/jenishrijalWFT/demo.git
        vii. git push -u origin main

    b. push an existing repository from the command line
        i. git remote add origin https://github.com/jenishrijalWFT/demo.git
        ii. git branch -M main
        iii. git push -u origin main

6. Semantic Commit Messages.
    a. https://gist.github.com/joshbuchea/6f47e86d2510bce28f8e7f42ae84c716
7. Atomic commits.
    a. https://dev.to/this-is-learning/the-power-of-atomic-commits-in-git-how-and-why-to-do-it-54mn
8. Exercise for day: https://learngitbranching.js.org/ (Day 1 and 2)

Day 2.

1. Branching. (Demo)
    a. Branching means you diverge from the main line of development and continue to do work without messing with that main line.
    b. git branch <branch name> – Creates a new branch.
    c. git checkout <branch name> –It is used to switch branches, whenever the work is to be started on a different branch.
    d. git checkout -b <branch name> - creates a branch of that name and switch to it.
    e. https://www.atlassian.com/git/tutorials/using-branches
2. Collaborating with git
    a. git clone <remote_URL> – Makes an identical copy of the latest version of a project in a repository and saves it to your computer.
    b. git pull <remote> – Used to get updates from the remote repository.
    c. git push -u origin <branch name> – Sends local commits to the remote repository.
3. Merge and solving conflict. (Demo)
    a. Merging is Git's way of putting a forked history back together again. The git merge command lets you take the independent lines of development created by git branch and integrate them into a single branch.
    b. https://www.atlassian.com/git/tutorials/using-branches/git
4. Exercise for day: https://gitexercises.fracz.com/ (Day 2 and 3)

Extra : https://www.atlassian.com/git
Download Book:
https://github.com/progit/progit2/releases/download/2.1.426/progit.pdf
Git CheatSheet: https://www.atlassian.com/git/tutorials/atlassian-git-cheatsheet