Key:

- Any fields in `<>` are required parameters.
- Any fields in `[]` are optional parameters. They may be omitted.
- Any field in `[]...` are optional parameters where more than one value can be assigned, separated by at least one space.
- Remember that “” can be used to separate parameters, such as inserting a filename that contains spaces.
- The `‘*’` acts as a wildcard, e.g. “`*.txt`” will match with any file in the current folder with the file extension, `txt`.

Commands:

- `git init <directory>`
  - Creates a new repository in the assigned directory.
  - Most people uses `‘.’` as the directory, indicating they want the repository created in the current folder, i.e. “`git init .`”
- `git add <filename1> [filename2]...`
  - Adds one or more files in the project to be versioned under the repository.
  - Files *must* be added first before the repository keeps track of their changes. Otherwise, they will be marked as “unversioned.”
- `git commit [folder-or-filename1]...`
  - Begins the process of creating a new revision in a repository. This command will bring up a text editor, where one can write up a message before closing it, finishing the commit process with the user-entered description explaining the changes.
  - If no folder or filename is provided, all changes in the project *except unversioned files* will be added in as a new revision. If a folder or filename is provided, a revision only containing the assigned folders or files with be versioned.
- `git status [folder-name1]...`
  - Displays a list of files in the project that changes from the last revision in the current branch, if no folder or file name is provided. Adding a folder name will display changes in that folder, only.
- `git diff <filename>`
  - Displays a list of files in the project that changes from the last revision in the current branch, if no folder or file name is provided. Adding a folder name will display changes in that folder, only.
- `git remove <filename1> [filename2]...`
  - Can be shortened to “`git rm`”
  - Deletes one or more files, both from the project, and in the repository.
- `git reset`
  - Reverts the project to the latest state stored for the current branch in the repository

- `git move <old-filename> <new-filename>`
  - Can be shortened to "`git mv`"
  - Renames a file, and adds that change in records to the repository.
  - This command can also move a file from one folder to another.
- `git revert <revision>`
  - Reverses a change from a revision in the repository. These changes needs to be manually committed.
- `git remote add <repo-name> <url>`
  - Bookmarks a URL under a given name. Commonly, this name is "origin."
- `git push [repo-name] [branch-name]`
  - Pushes local revisions into a remote repository. The repo-name parameter is a bookmark to a remote URL created from `git remote add` command.
- `git clone`
  - Downloads an online project into an empty folder.
- `git pull`
  - Actually two-commands-in-one: `git fetch` and `git merge`, run in that order.
  - Pulls changes from a remote repository, and merge them into the current project.
- `git fetch [repo-name] [branch-name]`
  - *Only* pulls changes from a remote repository into the local one. Changes do *not* get merged into the current project.
- `git merge [revision]`
  - Merges changes from a revision into the current one.
    - If a revision is not provided, it'll use "`HEAD`" (the most current revision in the current branch).