GitLab: online storage for git repos.

Git: tracks local changes.

Prerequisites:

* obtain a Git Lab account,
* Install Git,
* Decide on a text editor, and
* Decided on a command line.

1. Show use case for using Git and Git Lab. Present a demo of local machine vs Git Lab account.
   1. Working area (contains untracked files)

git add <filename>

* 1. Staging area (contains files to be committed)

git commit -m “commit message”

* 1. Local repo or history
  2. Push to remote

git push origin main

1. Create folder CS330e
2. Create subfolder git
3. On GitLAb
   1. Create a folder, Project1
   2. Visibility: public/private
4. On your Laptop
   1. Switch to the “git” folder
   2. Copy the contents of the remote GitLab folder “project1” using “git clone …”
   3. Create a file called Hello.py (shortcut: echo "print(\"Hello World! \")" > Hello.py)
   4. In a git folder, any newly created file will be untracked. An untracked file is not a part of any previous commit. To stage the file, you need to use the command “git add .”. Staging a file or a set of files will add them to a snapshot of a set of changes that will be committed later. Finally, to commit the changes, you use the command ‘git commit -m “….” ‘
   5. Finally, to send the local changes to remote, you need to use the command “git push origin main”
5. On your Laptop, whenever you need to update a working project, it is recommended to create a dev branch. In the dev branch, you keep working on your modifications and when your code works well, you can integrate these changes back to the working project.
   1. To create a dev branch, you use the commands “git branch dev” and “git checkout dev”. Alternatively, you can do both in one step “git checkout -b dev”.
6. Merge Conflict:
   1. Create a folder on Git Lab call it merge\_conflict.
   2. Clone it to your local machine.
   3. Add a file, project1.py, to it and add, commit and push to remote
   4. On Git Lab, update the file.
   5. On local machine, pull the changes.
   6. Simultaneously, update project1. Py
      1. on your machine, by adding

Def f1():

Pass

* + 1. on Git Lab, by adding

Def f2():

Pass

* 1. At this point, local and remote are different.
  2. On local, add, commit and push, you will receive an error message.

To https://gitlab.com/fareszf/merge\_conflict.git

! [rejected] main -> main (non-fast-forward)

error: failed to push some refs to 'https://gitlab.com/fareszf/merge\_conflict.git'

hint: Updates were rejected because the tip of your current branch is behind

hint: its remote counterpart. Integrate the remote changes (e.g.

hint: 'git pull ...') before pushing again.

hint: See the 'Note about fast-forwards' in 'git push --help' for details.

* 1. Pull changes, you receive the following.

Auto-merging project1.py

CONFLICT (content): Merge conflict in project1.py

Automatic merge failed; fix conflicts and then commit the result.

Note: git is unsure if there should be one function, i.e., f1 or f2, or two separate functions.

* 1. To resolve the conflict, open the file and remove all of the unnecessary tags. Save, add, commit and push.
  2. Ask your partner to pull