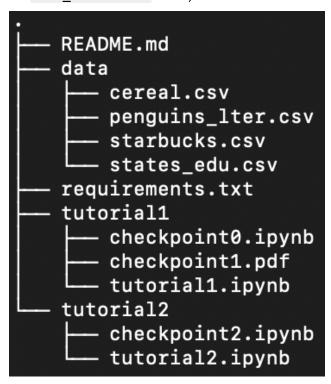
Checkpoint 1

1. Set up checkpoint directory

The tutorial folder you downloaded should have the following structure (run tree in the mdst tutorials folder):



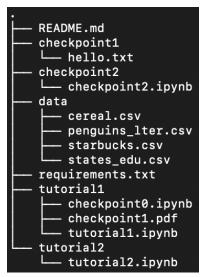
Note -- after completing the <u>setup</u>, you will also have your virtual environment in env/ Run tree -I env to see your structure without the environment

Note 2 -- on macOS, you may need to install tree using brew install tree

You are going to update this structure to store your checkpoints. Use Unix commands to do the following (see <u>documentation</u> or this <u>cheat sheet</u> for help):

- 1.1 Create two new folders for the checkpoints -- checkpoint1 and checkpoint2
- **1.2** Create an empty file called "hello.txt" in checkpoint1/
 - In this file, introduce yourself! Tell us your name, year, school/major, and a fun fact about yourself:)
- **1.3** Move the checkpoint files from the tutorial folders to the checkpoint folders
 - From tutorial2/, move checkpoint2.ipynb to checkpoint2/

Your final setup should have the structure:



2. Make a new local repository

The way you will be turning in the checkpoints is by uploading your work to your own github repo and linking that repo to us.

2.1 Move to the folder mdst_tutorials (the base folder you downloaded for the

tutorials) 2.2 Run git init

This creates a new git repository in the mdst_tutorials folder. You can always tell when you're in a git repo because there will be a hidden .git/ folder (verify with 1s -a)

The git repository you made here is saved only to your computer. Next, we want to connect it to a remote (online) repo.

3. Make a remote repo

If you don't already have a github account, make one! You can use either your umich or personal email.

- **3.1** On github.com, find the button to make a new repository. Click on it.
- **3.2** Give the repo a name and description and make it public. Do not initialize the repo with a README or .gitignore (we will be importing an existing repository)
- **3.3** Finish creating the repository

You should be able to see your new repository on your github profile, but there's nothing there! We need to link the local repo we created in step 2 to this repo.

4. Link repos

- **4.1** Open your repo from step 3 and copy the link (should have format https://github.com/<username>/<repo name>.git)
- 4.2 Navigate back to the first folder, where you ran git init
- 4.3 Run git remote add origin LINKFROMABOVE

Now your repositories are linked! You can now commit and push freely to your repo.

5. Commit checkpoints

When you turn your checkpoints in, we will only want to see the actual checkpoint files. The tutorials are the same for everyone (also we wrote them!) so you can just leave those out of the repository.

5.0 (optional) Create a .gitignore file to ignore the files you don't want to push (see documentation)

You will want to ignore the env, data, tutorial1, and tutorial2 folders You may also want to ignore any .ipynb_checkpoints or __pycache__ folders

5.1 Add, commit, and push the checkpoint folders and files to your git repository

See the Git refresher below for how to do that

Make sure to update your repo periodically as you work through the rest of the checkpoints! Even though we will only check the final version of the repo when you are finished, this is good practice and the whole reason git exists!

Git refresher

Since we're already here, let's review how to commit and push to git. Recall that a commit is like a checkpoint -- it saves the state of your files at a given instance. Pushing uploads your local changes/commits to your online repository.

In your own git repo, run git status to check your status. There will be 2-3 categories:

Changes not staged for commit -- these are files which have been changed and not committed

- Changes to be committed -- these are also files which have been changed and not committed, but you have marked them as 'to-be-committed'
- Untracked files -- these are files which have never been committed

If there are files in the untracked files section which you know you will never commit, you can have them stop showing up with a .gitignore file. Create this file inside your repo (using any editor) and include each pattern to ignore on its own line. For example, to ignore all .txt files, your gitignore would include the line *.txt (the * symbol matches anything that comes before a .txt)

To mark files as 'to-be-committed', run git add FILENAME to add them to the staging area. This tracks which file(s) are being updated with a single commit. These files you add will move from changes not staged for commit to changes to be committed.

Once you have added all the files you want to save at once, run git commit. This will open a terminal text editor, where you will write a commit message (should be a short description of changes made).

Vim: type i, type your message, ESC, then :x
Nano: type your message, CTRL-0, ENTER, then CTRL-X

After committing, if you run git status again, the 'changes to be committed' section will disappear (those files have now been committed). The rest will look the same.

Finally, to push your changes, run git push. After providing your login information, you will be able to refresh the github page and see your changes!

If you have authentication errors, you may need to create an SSH key. See this quide for help.