# Solutions to the Exercises in Lecture 3, ASTR 400B

The goal of these exercises is to familiarize you with the usage of git and give you a sense of how distributed version control systems work.

Some notes first according to the feedbacks I got in class:

A. How the file system works in Linux (on Nimoy) can be confusing. If you have little experience with Linux, checkout this tutorial (https://www.tutorialspoint.com/unix/unix-file-system.htm) on "Unix / Linux - File System Basics".
B. Say your username on Nimoy is `stars`, then "your home directory" means "`/home/stars`". Note that "`/home`" contains the home directories for all users, not just yours.
C. Use "`pwd`" to check which directory you are working with now. Usually you don't have the permissions to create/modify files outside your home directory.
**D. To clear some confusions about Repository**
   a. In your first homework, you were instructed to clone[1] your GitHub homework repository (a remote repository)[2] into your local machine (e.g., Nimoy). Thus, there should be a local repository that you can work with. Say your GitHub homework repository is named "`400B_stars`", then after you clone it, say, at your home directory, you should get a new folder named "`400B_stars`" in your home directory.
   b. This new folder, "`400B_stars`", is the working folder that contains your local repository, and people often interchangeably use "working folder" and "local repo" to refer this folder**. So why is this folder so special that "`git`" command knows it is a repository and other folders are not?** Because a local repository will contains a hidden folder named "`.git`", which is a mini-file-system that keeps all the file operation history for this repository.
   c. Therefore, all the git commands only work inside your local repository. And only file operations inside the local repository are meaningful to your version control system, that is, will be kept in the history.
   d. Because git is a distributed version control system, don't worry if you somehow mess up your local repository. You can always delete it and clone a new local repo at any time for a fresh start from the status of the remote repository on GitHub. Actually, you can also have multiple local repositories on your local machine at different locations.
   e. Usually, it is not recommended to make your home directory a local repository (so if you find you have a hidden folder named "`.git`" under your home directory, delete it). Also, do not clone a repository inside another local repository, which will make things quite complicated.

Now onto solutions.

## Exercise 1. Clone the class repository in your computer so you can pull updates in the future to get slides and homework assignments easily.

**Command**: `git clone https://github.com/gurtina/ASTR400B_2020.git`

[1] "Clone" in this content always means clone a remote repository to a local repository.
[2] Read the 7th slide of Lecture 3 for these concepts.

**Solution**: Execute the command listed above. The following example runs this command in my home directory. You should see a similar output below and there will be a new folder named "`ASTR400B_2020`" under the current working directory.

```
rixin@nimoy:~$ ls
400B_RixinLi  anaconda3  data  phys105a
rixin@nimoy:~$ git clone https://github.com/gurtina/ASTR400B_2020.git
Cloning into 'ASTR400B_2020'...
remote: Enumerating objects: 42, done.
remote: Counting objects: 100% (42/42), done.
remote: Compressing objects: 100% (28/28), done.
remote: Total 42 (delta 5), reused 26 (delta 3), pack-reused 0
Unpacking objects: 100% (42/42), done.
Checking connectivity... done.
rixin@nimoy:~$ ls
400B_RixinLi  anaconda3  ASTR400B_2020  data  phys105a
```

## Exercise 2. Now let's work on your homework repository.

(1) Create a folder named "`git_test`" under your home directory
(2) Clone your own HW repo in the directory "`git_test`" you just created.

**Solution**: Most of us already have a local homework repository by doing HW1. This exercise requires us to clone our GitHub repository into another local repository inside the new folder "`git_test`". The following example shows how to do this task. Now you have multiple local repositories in one local machine (you can also have multiple local repos at different computers).

```
rixin@nimoy:~$ ls
400B_RixinLi  anaconda3  ASTR400B_2020  data  phys105a
rixin@nimoy:~$ mkdir git_test
rixin@nimoy:~$ cd git_test/
rixin@nimoy:~/git_test$ git clone https://github.com/astroboylrx/400B_RixinLi.git
Cloning into '400B_RixinLi'...
Username for 'https://github.com': astroboylrx@gmail.com
Password for 'https://astroboylrx@gmail.com@github.com':
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 8 (delta 0), reused 5 (delta 0), pack-reused 0
Unpacking objects: 100% (8/8), done.
Checking connectivity... done.
rixin@nimoy:~/git_test$ ls
400B_RixinLi
rixin@nimoy:~/git_test$ pwd
/home/rixin/git_test
rixin@nimoy:~/git_test$ ls 400B_RixinLi
Homeworks  README.md
```

(3) Now you have a new working repo, "`cd`" into this copy of repo.
(4) Create a text file "`lecture3.txt`" and write "`Lecture 3 test`" into it.
(5) Add, commit, and push your changes ("`git add .`" will add all changes)

**Solution**: The following example shows how to do this task. Keep in mind that you can constantly use "git status" to check the status of the current working folder to make sure you have committed or have pushed your changes. And if you make multiple changes to your local repository, "`git add .`" may speed up your workflow. Here "`.`" is meaningful to the "`git`" command as an argument for the "`add`" option (not the same as the one used in the file path).

```
rixin@nimoy:~/git_test$ cd 400B_RixinLi
rixin@nimoy:~/git_test/400B_RixinLi$ echo "Lecture 3 test" > lecture3.txt
rixin@nimoy:~/git_test/400B_RixinLi$ ls
Homeworks  lecture3.txt  README.md
rixin@nimoy:~/git_test/400B_RixinLi$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        lecture3.txt

nothing added to commit but untracked files present (use "git add" to track)
rixin@nimoy:~/git_test/400B_RixinLi$ git add .
rixin@nimoy:~/git_test/400B_RixinLi$ git commit -m "RL: add a text file for lecture 3"
[master b3e8000] RL: add a text file for lecture 3
 1 file changed, 1 insertion(+)
 create mode 100644 lecture3.txt
rixin@nimoy:~/git_test/400B_RixinLi$ git push
Username for 'https://github.com': astroboylrx@gmail.com
Password for 'https://astroboylrx@gmail.com@github.com':
Counting objects: 3, done.
Delta compression using up to 12 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 348 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/astroboylrx/400B_RixinLi.git
   4f72f6e..b3e8000  master -> master
rixin@nimoy:~/git_test/400B_RixinLi$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working directory clean
```

(6) Create a directory “L3” and move “lecture3.txt” into it
(7) Again, add, commit, and push your changes

**Solution**: The following example shows how to do this task. This exercise repeats the typical git workflow to help you memorize it.

```
rixin@nimoy:~/git_test/400B_RixinLi$ mkdir L3
rixin@nimoy:~/git_test/400B_RixinLi$ git mv lecture3.txt ./L3/
rixin@nimoy:~/git_test/400B_RixinLi$ ls
Homeworks  L3  README.md
rixin@nimoy:~/git_test/400B_RixinLi$ git add .
rixin@nimoy:~/git_test/400B_RixinLi$ git commit -m "RL: mv the text file into L3 (lecture
3)"
[master 12f0bd9] RL: mv the text file into L3 (lecture 3)
 1 file changed, 0 insertions(+), 0 deletions(-)
 rename lecture3.txt => L3/lecture3.txt (100%)
rixin@nimoy:~/git_test/400B_RixinLi$ git push
Username for 'https://github.com': astroboylrx@gmail.com
Password for 'https://astroboylrx@gmail.com@github.com':
Counting objects: 3, done.
Delta compression using up to 12 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 371 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/astroboylrx/400B_RixinLi.git
   b3e8000..12f0bd9  master -> master
rixin@nimoy:~/git_test/400B_RixinLi$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working directory clean
```

## Exercise 3. Now go to your original homework repository folder that you did your homework and pull the changes you made inside "`git_test`".

**Solution**: This exercise aims to teach you how to manage multiple local repositories. For example, you have a local repo on both Nimoy and your laptop. Say if you did your homework on Nimoy and finished the entire git workflow. This exercise provides an example of how to keep the local repo on your laptop up to date so that you may use your laptop to do your next homework. Using the command "`git pull`" in an outdated local repository will update it from the remote repository if anything has changed there. In the example figure below, I went outside "`git_test`" and `cd` into the original homework repository, where the folder "`L3`" didn't exist (since it is one commit behind the remote repository). After using "`git pull`", I kept this homework repository up to date ("`L3`" appeared) with the remote GitHub repository which had modifications "`push`"-ed from the other local repository.

```
rixin@nimoy:~/git_test/400B_RixinLi$ ls
Homeworks  L3  README.md
rixin@nimoy:~/git_test/400B_RixinLi$ pwd
/home/rixin/git_test/400B_RixinLi
rixin@nimoy:~/git_test/400B_RixinLi$ cd
rixin@nimoy:~$ pwd
/home/rixin
rixin@nimoy:~$ cd 400B_RixinLi/
rixin@nimoy:~/400B_RixinLi$ pwd
/home/rixin/400B_RixinLi
rixin@nimoy:~/400B_RixinLi$ ls
Homeworks  README.md
rixin@nimoy:~/400B_RixinLi$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working directory clean
rixin@nimoy:~/400B_RixinLi$ git pull
Username for 'https://github.com': astroboylrx@gmail.com
Password for 'https://astroboylrx@gmail.com@github.com':
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 6 (delta 0), reused 6 (delta 0), pack-reused 0
Unpacking objects: 100% (6/6), done.
From https://github.com/astroboylrx/400B_RixinLi
   4f72f6e..12f0bd9  master     -> origin/master
Updating 4f72f6e..12f0bd9
Fast-forward
 L3/lecture3.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 L3/lecture3.txt
rixin@nimoy:~/400B_RixinLi$ ls
Homeworks  L3  README.md
```

(1) Try to revert the last commit you just made if you have time.

**Solution**: Using the command "`git revert HEAD`" to revert the last commit you just made and "`git push`" to update the remote repository with one commit reverted[3]. As you can see in the example figure below, the folder "`L3`" disappeared and the file "`lecture3.txt`" was back. You may revert as many times as you want, though it should be a rare case. Now you get the idea that the remote repository on GitHub can receive updates from any local repository that you might work with.

```
rixin@nimoy:~/400B_RixinLi$ ls
Homeworks  L3  README.md
rixin@nimoy:~/400B_RixinLi$ git revert HEAD
[master 22ec671] Revert "RL: mv the text file into L3 (lecture 3)"
 1 file changed, 0 insertions(+), 0 deletions(-)
 rename L3/lecture3.txt => lecture3.txt (100%)
rixin@nimoy:~/400B_RixinLi$ git push
Username for 'https://github.com': astroboylrx@gmail.com
Password for 'https://astroboylrx@gmail.com@github.com':
Counting objects: 2, done.
Delta compression using up to 12 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 355 bytes | 0 bytes/s, done.
Total 2 (delta 0), reused 0 (delta 0)
To https://github.com/astroboylrx/400B_RixinLi.git
   12f0bd9..22ec671  master -> master
rixin@nimoy:~/400B_RixinLi$ ls
Homeworks  lecture3.txt  README.md
rixin@nimoy:~/400B_RixinLi$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working directory clean
```

[3] Read the 13th slide of Lecture 3 for these operations.

Now, if you use the command "`git log`" to list all the history of file changes, you'll see the revert operation in fact created another commit. As mentioned in the class, anything that you committed will be kept in the repository history, so "`revert`" won't erase the last commit, it just changes the file content back to the status before last commit. In the meantime, "`revert`" will add itself into the history in the form of a commit.

```
rixin@nimoy:~/400B_RixinLi$ git log
commit 22ec67137d9d536f1d37f326994493e9a62eff00
Author: Rixin Li <astroboylrx@gmail.com>
Date:   Thu Jan 23 19:36:20 2020 -0700

    Revert "RL: mv the text file into L3 (lecture 3)"

    This reverts commit 12f0bd96a29fc7a6719051cbccbd132807db78c2.

commit 12f0bd96a29fc7a6719051cbccbd132807db78c2
Author: Rixin Li <astroboylrx@gmail.com>
Date:   Thu Jan 23 19:31:12 2020 -0700

    RL: mv the text file into L3 (lecture 3)

commit b3e8000435a276e8da0474308105a6b63d2ffcb2
Author: Rixin Li <astroboylrx@gmail.com>
Date:   Thu Jan 23 17:57:45 2020 -0700

    RL: add a text file for lecture 3

commit 4f72f6e58728c7bb6935c26c3af182eff8ec982f
Author: Rixin Li <astroboylrx@gmail.com>
Date:   Thu Jan 23 16:20:54 2020 -0700

    RL: push my first commit as HW1

commit 376b1e32834f9e11471d672f3f35c39b0de00e1c
Author: Rixin Li <astroboylrx@gmail.com>
Date:   Thu Jan 23 16:05:50 2020 -0700

    Initial commit
```

Because "`revert`" create another commit, a text editor will be presented to you after the command "`git revert HEAD`" for you to write the commit message. By default, on Nimoy, a text editor named "GNU nano" was used. You may just press Ctrl+X to exit it and answer Y to any questions afterward.