**GITHUB INTRODUCTION – Day Two**

Software developers use Git to collaborate all the time and so you will get practice with collaborating on Git today.

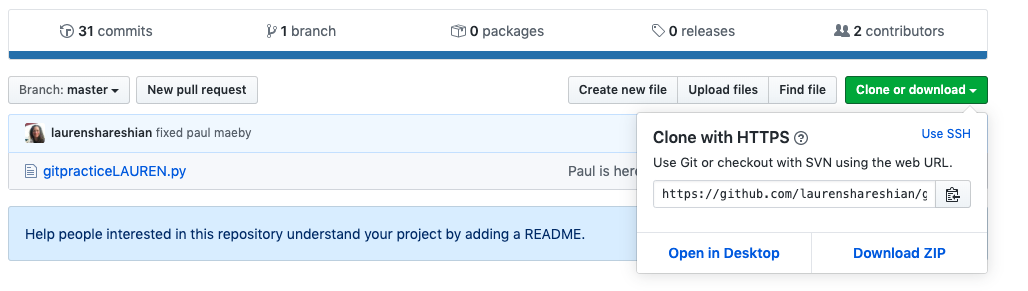
You and your partner should decide who is Partner A and who is Partner B in the steps below.

**Part I: Collaborating on the same repo**

1. Partner A should go to the GitHub website and click on their gitpractice repo that they added for their homework last night.

2. Partner A should go to Settings – Collaborators and enter Partner B’s GitHub username or email address.

3. Partner B should login to their GitHub account and click on the notification in the top right to accept this invitation and head to Partner A’s gitpractice webpage. Make sure it says “2 contributors” as it does below in the top right of the screenshot.



Partner B should select “Clone or download”. Partner B should copy the URL that is listed to use for later.

4. In a terminal, Partner B should navigate down into the Day4CollaboratingOnGit folder. They should type “pwd” to confirm that they are there.

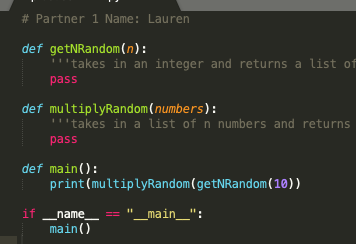
5. Partner B should then type “git clone” followed by the URL address copied from the above step. It will be something like:

“git clone <https://....gitpracticepartnername.git/>”

6. Both partners should navigate down into their gitpractice folders on their respective computers (Partner B will be in Day4 and Partner A will be in Day3). (Type ls to make sure that you see the gitpracticeNAME.py file.)

Now, here comes the fun part. You and your partner will be collaborating on the same code.

8. Partner A should add a new hashtagged comment at the top of their file that says “Partner 1” and then their first name. They can use a text editor like Sublime or vim. Here’s an example:



Now, this partner will commit their changes by typing:

git add -A

git commit -m “added partner 1 first name”

git push

9. If both partners refresh their GitHub webpages, they should now see Partner A’s name in the file on the website.

10. Partner B, however, does not have these updates on their local computer. To get these changes from master, Partner B should type “git pull” into their terminal. If they open the file back up, they should now see Partner A’s name. Now, they should add their name below and save it. They should also add in a comment with a bunch of hashtag symbols to denote separation between the header and the code below. It will look like this:



After saving, Partner B should type:

git add -A

git commit -m “added partner 2 first name”

git push

11. If both partners refresh their webpages, then they should see Partner 1 and 2 names now displayed.

12. Partner A should now type “git pull” to get Partner B’s name on their local computer. If they reopen the file, they should now see both names.

Alright, does using Git always work this smoothly? NO. You are bound to run into merge conflicts. Let’s run into a few now on purpose to figure out what to do.

13. Okay, let’s have Partner B add in the last names to both of the names. Make sure Partner B does NOT create any new lines – just add last names at the end of the current lines:



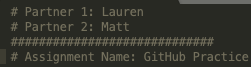
Save this and then type:

git add -A

git commit -m ‘added last names’

git push

14. HOWEVER, Partner A still only has the older version with just their first names saved, as they still haven’t typed “git pull” to pull down the most recent edits. Let’s suppose that they forgot to pull to get the last names and now tried editing the file to include an assignment name on the fourth line. Meaning, Partner A should NOT TOUCH THE FIRST THREE LINES BUT TYPE THIS ON THE FOURTH LINE:



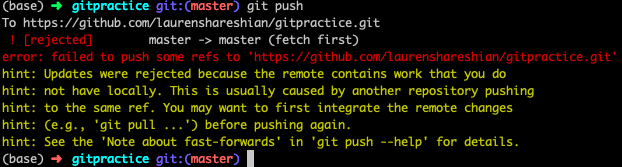
13. Now, have Partner A type:

git add -A

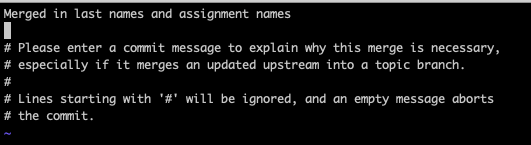
git commit -m “added assignment name”

git push

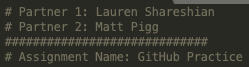
Partner A will get an error message that looks like this because their local version is a commit behind the origin version (the one with Partner B’s name in it as well).



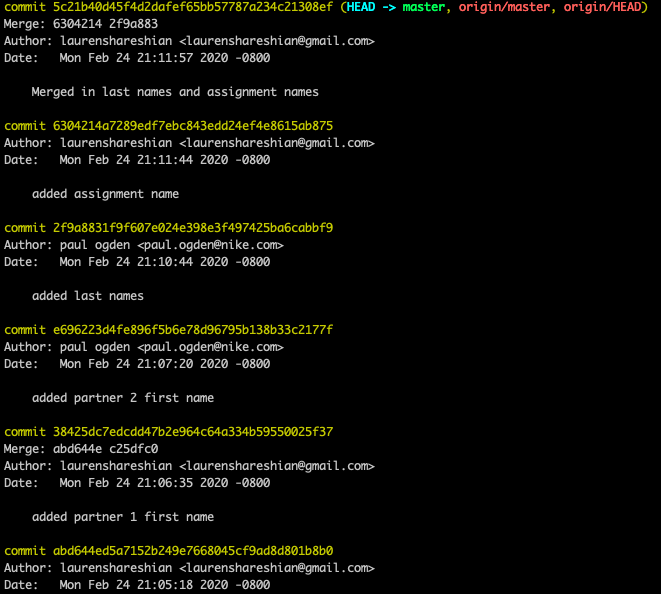
14. To remedy this, Partner A should type “git pull”. A vim merge commit pops up that will catch your local version up to origin. Change the very first line from “Merge branch ‘master of http…’ to your commit message “Merged in last names and assignment name” and type escape :w enter :q enter to save and quit:



15. Now Partner A should type git push, then Partner B should type git pull, and now if everyone refreshes their GitHub webpages, everyone should have the full names and the assignment name, as shown below:

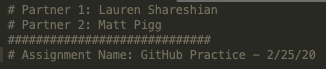


16. How can you both see the history of your revisions? Type “git log”. You should see each of your individual commits along with the most recent commit which helped to merge your changes together. If you guys ever really screwed stuff up and wanted to revert to a previous version of your code, you could revert to a previous version by referring to those unique yellow hashes assigned to each commit.



17. Okay, now let’s REALLY mess stuff up. Let’s have both partners messing with the same line. Before you start doing this, both of you should “git pull” once more to make sure that you really are starting from the same version.

18. Okay, let’s have Partner B put the date at the end of the assignment line:



Then, Partner B should type:

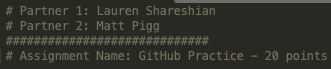
git add -A

git commit -m “added date”

git push

Partner B won’t run into problems because they were the first person to push.

19. Then, WITHOUT FIRST PULLING, let’s have Partner A type the assignment points at the end of the assignment name instead:



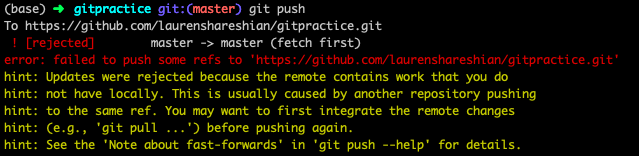
20. Alright, let’s have Partner A type the following

git add -A

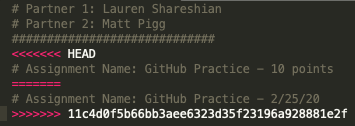
git commit -m “added points”

git push

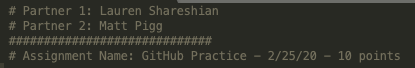
They will get a message that says merge conflict:



22. To fix this merge conflict, Partner B must go into the gitpracticeNAME.py file using vim or sublime. They’ll see <<<<< HEAD (which denotes the local version), ===== (which denotes the origin GitHub version), and a commit hash >>>>> with lots of letters and numbers, like this:



Partner B will need to delete four lines: the <<<< line, the second === line, the >>>> line, and one of the other two lines assignment name lines that are in conflict. You can manually edit the line you are keeping to reflect both partners’ changes as follows:



Then type escape :w enter :q to save and quit.

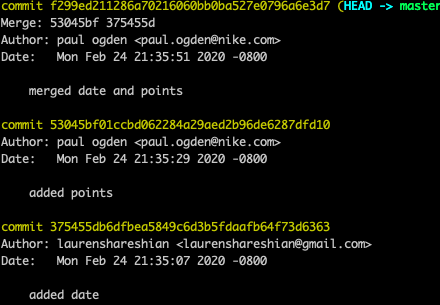
23. Partner A should now type:

git add -A

git commit -m “merged date and points”

git push

24. If Partner B pulls down again, everyone should now have all of the updates. We can type “git log” again to see the workflow once more:



**Homework Part I – 20 points**

One of you will fill in the getNRandom function and the other person will fill in the multiplyRandom function. The final product will be a program that generates 10 random numbers and returns the product of those 10 numbers. Fill in your part of the code on your local computer now. Then merge your changes together so that everyone has a complete working program on their computer that they can type “python gitpracticeNAME.py” from the terminal to run. Have one of the partners email Lauren with a link to their GitHub repo that displays the fully complete program.

**Homework Part II – 10 points extra credit**

Often, collaborators will not be pushing to the main branch of the repo. For example, consider all of the code that goes into making the Nike website. Would Nike want one disgruntled employee to be able to push a bunch of nonsense to the main code base? Instead, employees make branches for whatever features they are currently working on, which eventually get merged into the main branch after a pull request. This is also the process by which people contribute to open source software, a super cool thing to do.

Your homework will be to follow the steps on this website <https://codeburst.io/a-step-by-step-guide-to-making-your-first-github-contribution-5302260a2940> in order to contribute to your first open resource. More specifically, if you follow the steps correctly, your name will be featured here <https://github.com/firstcontributions/first-contributions/blob/master/Contributors.md>

as soon as the administrator approves your pull request. We don’t have time to get to this in class, and it is good practice having you learn to do something fully on your own using online resources. Therefore, we won’t be able to help you with this. But, if you get it working, email us to let us know that your name is displayed on <https://github.com/firstcontributions/first-contributions/blob/master/Contributors.md> and we’ll give you extra credit points.