Tutorial 06 - Collaborating with Github

git commands vs. shell commands

- git commands are different from shell commands
- shell commands are components of the operating system `Unix`
- someone wrote code (C, shell, Perl, and Python) that created a collection of functions, including `init`, `add`, `commit`, `push`, and `pull`
- we can execute these functions inside the `Unix` operating system using the program `git`
- next week you will see some other examples of programs written in other languages that we can execute in `Unix` to do bioinformatic analyses

add vs. commit

- `add` includes the current version of files in the staging area
- `commit` creates a snapshot or time capsule of all files in the staging area
- this two step process allows us to only record information abou files we've made changes to and added to the staging area in a commit call, rather than always keeping track of all files
- it also allows us to be sure what changes to what files we want to commit before doing so

commit messages

- commit messages are like comments in your code
- commit messages are for you or future users of the repository to understand the changes that were made to the code over its development
- `git` forces you to "comment" each of your commits because it is a good practice

origin & master

Why do I always need to type origin and master when interacting with Github?

- origin is the "nickname" for the github repository that the code is linked to; it would be annoying to type out the full web address for the github repository each time you pushed or pulled information and so we can use that shorthand name instead
  - you can see this if you run `git remote -v` from inside a git repository that is associated with a github repository
- master specifies the branch you are working with and most of the time you'll be on the main branch, which is called "master"

branches

The syntax and details of working with branches is less important than a conceptual understanding at this point.

- Branches are useful because it allows use to have a “stable”, functioning version of the code that anyone can use, but at the same time be working on improving or developing the code (on a different branch).
- It also allows for multiple collaborators to work on different portions of the code simultaneously because the main branch maintains a functioning version of all the code, but individuals can make, potentially bad, changes to their own versions of the code on their own branch.

How to do an assignment from now on

1) fork the TAs repository

2) clone the github repo to your local machine

3) do your work; during which time you can add and commit changes locally as you wish; you can also push changes to your repository as you wish

4) do one last add and commit to record all of your changes

5) push your local git repository to your github repository

6) submit a pull request by 10:25 on Fridays to "turn in" your answers