Computer Science 97: Week 5 Worksheet

"Git Gud!"

1. Using Git for your projects.
    a. Make a private GitHub repository titled "cs97-assignments".
    b. Clone your newly created repository to your own computer.
    c. Create an "assignment3/" directory.
    d. Put your related assignment 3 work in this directory.
    e. Commit it and push it to GitHub.
    f. `ssh` into SEASnet. Create an SSH key on SEASnet and register it with GitHub.
    g. Clone your "cs97-assignments" directory on SEASnet.
2. Configuring Git
    a. Run these commands, replacing Gene Block's details with your own.
    b. `git config --global user.name "Gene Block"`
    c. `git config --global user.email "geneblock@ucla.edu"`
3. Git GUIs. Many students find it easier to work with Git, especially at first, by using a Git GUI program. Besides gitk, there is: GitKraken, Tower, GitHub desktop (all of which are free for student use via the GitHub Education Pack). Editors like Visual Studio Code and JetBrains IDEs also have built-in integration with Git. Try one of these out and visually explore a Git repository.
4. Git scavenger hunt! What do the following commands do?
    a. `git clone`
    b. `git init`
    c. `git help`
    d. `git status`
    e. `git add`
    f. `git commit`
    g. `git log`
    h. `git diff`
    i. `git checkout`
    j. `git push`
    k. `git pull`
    l. `git blame`
    m. `git show`
    n. `git reset`
    o. Now that you've explored these on your own, here's a great Git cheatsheet. :^)
5. Let's practice!
    a. Create a new local Git repository called "git-animals."
    b. Create a file called "git-dog.txt" that contains the text "Git me a dog!"
    c. Create a file called "git-cat.txt" that contains the text "Git me a cat!"
    d. What do you see when you run `git status`? Add "git-dog.txt" to the staging area. Did the output of `git status` change?
    e. Commit "git-dog.txt" with a helpful commit message.

    f. Create a new branch called “cats.” Switch to this branch.
    g. Add and commit “git-cat.txt” with a helpful commit message.
    h. Verify your changes with `git log`.
    i. Return to the master branch. Does the output of `git log` change?
    j. Bonus: Use `git merge` to merge “cats” into “master.” Why might this command be useful?
6. In the command `git diff HEAD~2 HEAD`, what do `HEAD` and `HEAD~2` refer to?
7. Discussion: Why do you think Git has the concept of a staging area? When would you use it?
8. Discussion 2: Many different people and companies have differing opinions on how to use git.
    a. How to write a commit message
    b. Originate Git Guide
    c. Git Flow
    d. GitFlow considered harmful
    e. GitHub Flow
    f. As you start to think about working in teams on your projects, what Git practices do you think it makes sense to follow as a team?

Recommended reading (optional!)

- *Astrolabe v. Olson*, in which Dr. Eggert was named a defendant for possible copyright infringement for tzdb, an open-source project he maintains. This is an inspiration for a part of the upcoming Assignment 4.
- *Pro Git*, a comprehensive book of all things Git-related.