Assignment #4—It's Git-ing Hot in Here

Due: February 15, 2023 at 11:59pm

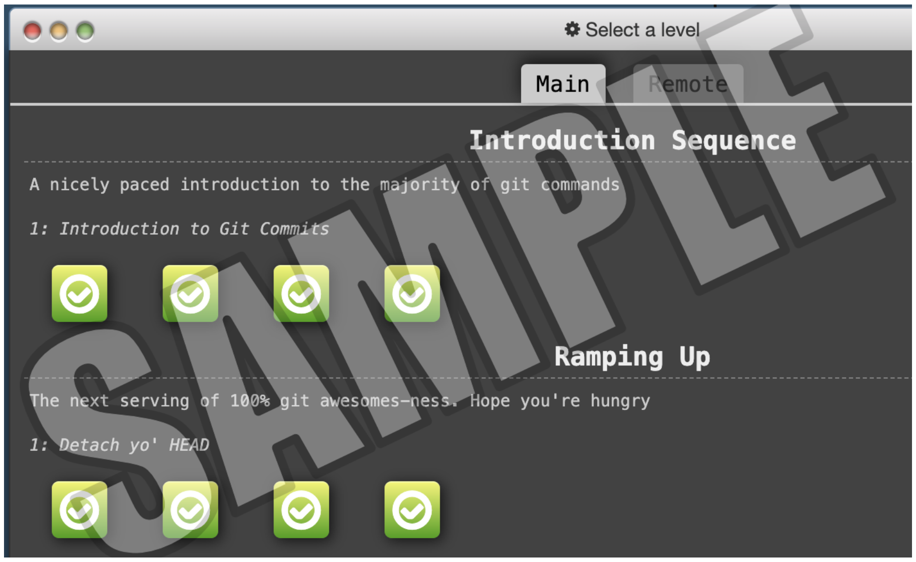
This assignment consists of three different components:

1. Do a Git tutorial
2. Practice using Git from the command line
3. Practice using GitHub

We expect this assignment to take 1-3 hours depending on your proficiency level with the tools. If you find yourself unproductively stuck or unproductively struggling, ask on Ed and/or go to office hours!

Part I: Master Git (1 point)

In this part of the assignment, you will complete a tutorial to help you master Git. The tutorial can be found here. You should complete the entire Introduction Sequence and Ramping Up sections.[1] Once you are done, submit a screenshot of your progress card.


Select a level
Main
Remote
Introduction Sequence
A nicely paced introduction to the majority of git commands
1: Introduction to Git Commits
Ramping Up
The next serving of 100% git awesomes-ness. Hope you're hungry
1: Detach yo' HEAD
SAMPLE

Part II: Your First Git Repo (1 point)

Let's create your first git repository together—we'll be making a simple **rock paper scissors** game in Python, and using a **git repository** to track changes as we add features.

A Git Repository is Born!

The first step is to create a directory where you'll store your files. Go into your Terminal and create a new directory, then **initialize your git repository**.

[1] You can do more if you want to, it's good practice! But we're only requiring the first two sections.

Then, let's create our file– let's call it **`rock_paper_scissors.py`**. Here's what we want you to do:

1. **While working on the game, commit your changes each time you finish a unit of functionality.** For example, a good first commit might be setting up a main function and greeting the user (your "initial commit.") Another one might be adding an explanation of the rules and how to play; etc. Remember, that your code must compile at each commit.
2. Your application should meet the following requirements:
    a. It should greet the user when it starts
    b. It should explain the rules of Rock Paper Scissors, which (for those of you who don't know) are as follows: two players secretly pick one of "rock," "paper," or "scissors." Both players reveal their selection to the other player at once; the winner is chosen based on what the selections are. Rock beats scissors (by crushing them); scissors beats paper (by cutting it); and paper beats rock (by covering it). If both players select the same one, it is a tie.
    c. It should allow the player to enter in their selection (you can use the **`input`** method). You can assume the user will type in **`rock`**, **`paper`**, or **`scissors`** (all lower-case).
    d. It should print out the computer's selection, which should be random (hint: you can use **`import random`** then do **`random.choice(list)`** to select a random element from a given list)
    e. It should declare a winner, or a tie.

Extending Our Game

Now that you've got a basic game working, let's extend it to make it just a bit fancier– when we tie, the computer should rematch us until one of us wins or loses! **To extend with our experimental new feature, let's create a new branch called `no-ties`!**

After creating this branch, implement your new feature. Feel free to commit as many times as makes sense (e.g. for a first pass implementation, and then again as you fix any bugs). Try to have at least two commits by the end of this.

Once you finish, **let's rebase this feature back to our `main` branch,** while also **squashing all the commits into a single one** (this will keep the history linear– a nice aspect of rebasing– while merging all our extra bugfixes and work into one go!). For this, let's use the **interactive rebase** function. Keeping your first commit of your no-ties feature, tell git to `squash` all the ones that came after. Then, exit your **`$EDITOR`** and watch git do its magic!

Part III: Practice with GitHub (1 point)

For this part of the assignment, you will get practice with using GitHub in order to collaborate with other students in the class. We will be using [this repository] to collaborate together.

Student Hobbies

First, we want to compile a list of hobbies for all of the students in the course.

You can do this part from the GitHub online or desktop interface, then by cloning the directory locally.

If you are working locally: You will want to **fork** this repository, then **clone** it on your computer so you can make edits. Before you start making edits, you will want to pull any new changes from the remote repository on Github. Remember to create a GitHub account if you don't already have one, and log in on the shell with the `gh auth login` command.

You can fork the repository and clone it to your computer all in one go from the github CLI tool; simply do `gh repo fork stanford-cs45/win23-assign4 --clone=true`

Once you have navigated to the repository, you should add a hobby to the list of hobbies found in `student_hobbies.txt` and include your SUNet in parentheses. You should also find a hobby that you share with someone else (i.e. a hobby someone else added that you also enjoy). For the hobby you share, you should add your SUNet in parentheses, separated by a single space.

Based on the number of people who share the hobby, move the line to the appropriate part of the file where the hobbies shared by the most people are at the bottom of the file and the hobbies shared by the fewest number of people are at the top of the file.

Once you are done, push these changes to your fork of the repository.

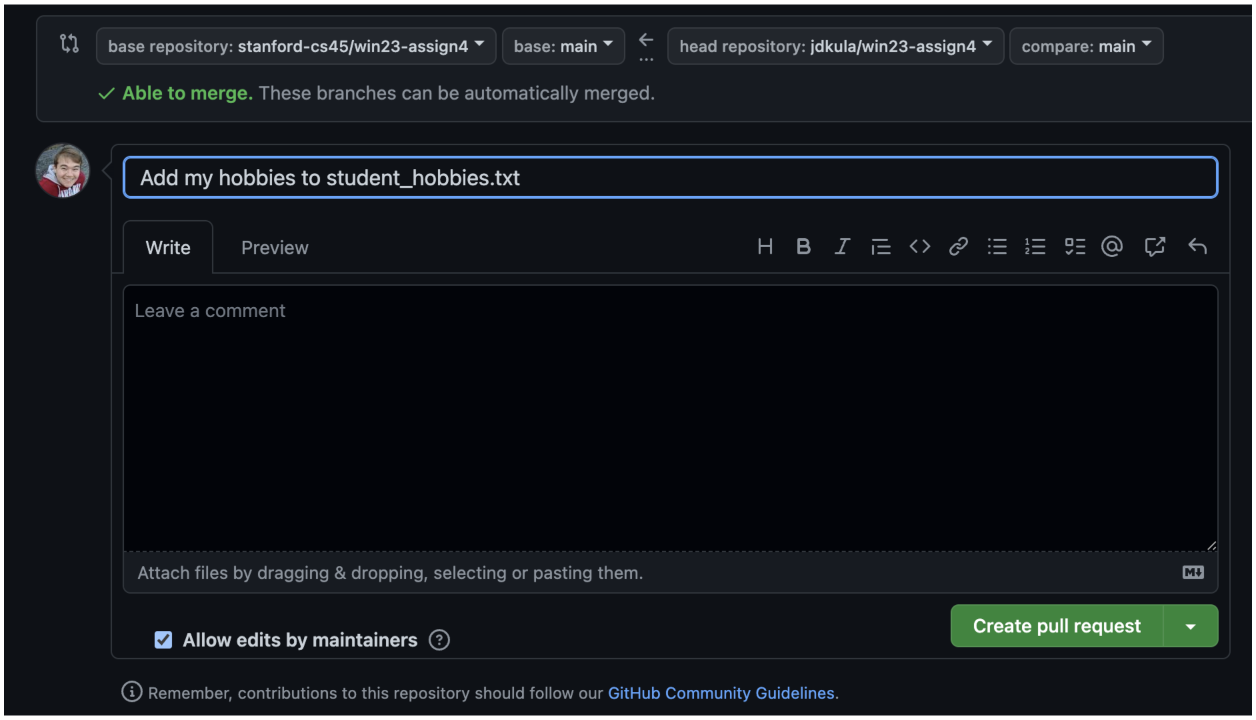
For example, the `student_hobbies.txt` file may look as follows when you first download it:

```
Unset
Surfing (adrazen)
Languages (akshay01)
Drawing (jdkula)
```

Consider a student whose SUNet is `karel` who enjoys coding and also enjoys languages. After the student has completed this part of the assignment, the file would look as follows:

```
Unset
Surfing (adrazen)
Coding (karel)
Drawing (jdkula)
Languages (akshay01 karel)
```

Finally, you'll be **submitting a pull request to the shared repository**. You can do this from the GitHub Website by going to your forked repository and click the "Contribute" then "Open Pull Request" button. You'll get a screen that looks like the following– fill out a title and optionally a comment below, and then click "Create Pull Request"!



base repository: stanford-cs45/win23-assign4 · base: main ← head repository: jdkula/win23-assign4 · compare: main

✓ Able to merge. These branches can be automatically merged.

Add my hobbies to student_hobbies.txt

Write · Preview

Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

☑ Allow edits by maintainers

Create pull request

Remember, contributions to this repository should follow our GitHub Community Guidelines.

[Here are some additional instructions from GitHub about creating pull requests, if useful.]()

Sleuthing for a Fact

We have a text file listing a bunch of facts called `facts.txt` on the repository—but oh no!! There are supposed to be 3090 fun facts in the file, but there are only 3089 currently. One must have been deleted at some point!

Your goal is to **find this missing fact** and **who deleted it!** To do this, you can use the `git log` and `git diff` commands to inspect the repository.

To start your sleuthing, *you can provide a file name to* `git log` *to show only the commits that affected that file.* This will give you the **commit hashes** of all commits that affect it. Do you see any strange or suspicious commit messages?

Once you identify a suspicious commit (or pair), you can use `git diff` to see what files changed between two commits. Recall from Part I, if you want to see what a single commit changed, you can compare a commit with the commit right before it. Recall that you can use `a1b2c3^` to refer to "the commit just before `a1b2c3`".

Once you've found the missing fun fact, put it in `missing_fact.txt` – then, place it in with your rock-paper-scissors game and commit it as an additional file.

Feedback Survey (0.5 points)

Once you have completed the assignment, you can earn an additional 0.5 points by completing our anonymous feedback survey. Given this is the first offering of the course, we want to collect as much feedback as possible to improve the course in the future. You can complete the survey here. Once you complete the survey, you will receive a completion code which you should place in a text file named `survey.txt`. The survey is anonymous so submitting the completion code is the only way to verify that you completed the survey. *Please do not share this code with anyone, as that would constitute a breach of the honor code.*

Submitting Your Assignment

Once you have finished this assignment, you will need to upload your files to Gradescope. Make sure to upload all files to the Assignment 4 submission page. You should also upload `survey.txt` if you completed the survey.

Here is what you need to submit for each part:

- Part I: A screenshot of your completion of the required parts of the tutorial. This screenshot can either be submitted separately or committed into your Rock-Paper-Scissors repo.
- Part II: Your entire directory containing the Rock Paper Scissors game
- Part III: On GitHub, make sure you've created a pull request with the appropriate changes to the main repository. On GradeScope, submit a `missing_fact.txt` file that contains the missing fact (in your rock-paper-scissors repo).
- Survey: `survey.txt`