# In-class assignment #5

**Instructions:** We're going to build on the pre-class assignment and lecture by working through GitHub's process for dealing with the creation of issues and pull requests, as well as branching, and forking. **Get together in a group of three, and try to do the following:**

1. One member of the group should create a new git repository on GitHub and give the other members access to it. Put the **bad code** named "string reversal.py" in the repository and push it to GitHub.

2. Create two GitHub issues associated with this repository: one describing the horrific formatting of the repository, and one detailing the two bugs. In general, it's best to be as informative and factual as possible in an issue as a courtesy to the developer, even pointing out exact lines of code, etc.

3. The other group member(s) should also clone the repository to their own machines. One of you should make a branch called cleaned-code, check out that branch (which you can verify with "git status"), run ruff format on the code in that branch, commit your changes, and push them back to GitHub (**don't forget to push the branch** – see the GitHub docs for instructions). Another group member should make a second branch called fixed-code, check out that branch, fix the code bug, and push those changes and new branch to GitHub as well. Coordinate with your group members to make sure that you've edited at least one of the same lines of code in a different way in each branch – **you want to deliberately create a merge conflict for a later step in this assignment!**

4. Now, each of the two people who created a new branch should issue a pull request (PR) **from their branch to the main branch** (so there will be two pull requests). Make sure to describe what you've done and **refer to the issue describing the problem by its number** (if you do it correctly it will automatically create a link). Don't do anything with this PR yet!

5. Then, the person/people who did NOT create that pull request should comment on the PR with one or more suggested changes (you can make general comments or comments associated with specific lines of code – try doing both!). The creator of the PR should update the branch on their own machine and push to GitHub, which should update the GitHub PR. The original author of the PR should respond to the comments. If the commentors feel that you have done things correctly, then they can approve the pull request on GitHub (look under the "Files Changed" tab for the "Review changes" button to approve). Once both commentors approve, **merge only one of the branches!**

6. Assuming your second (unmerged) branch also has a line of code edited that was also modified in the merged PR, you will now have a merge conflict. If it's small you can resolve it via the GitHub web interface, but if it's complicated you will have to do it via the command line using a text editor. **Fix the merge conflict, get two approvals on the PR, and merge!**

7. Now that your two issues have been addressed, make sure to close the issues. It's good practice to make a final comment saying something to the effect of "Fixed in PR#1234. Closing issue." This provides a historical record that the issue was actually fixed, and where!

8. You no longer need the two branches that you've created, so delete them both locally on your laptop and on GitHub.

**Congratulations, you're all done!** If you've finished the process described above and still have some time left in class, try replicating it but fix the changes by **forking the code on GitHub rather than making branches**. This will create entirely new repositories instead of just adding a branch to the existing one, but otherwise it's largely the same workflow.