Teamwork with Git and GitHub

Setup

One team member will:

1. Create an organization.
2. Create a repository that belongs to the organization (***not*** your personal account) (but don't push anything).
3. Invite the other team member, make them an owner of the organization, and a collaborator on the shared repository.
4. Clone the repository down locally
5. Then create a a README file with an editor. Then `add`, `commit`, and `push` that README.

The other team member(s) will:

1. Accept the invitation to the organization.
2. Clone the repository

   ```
   git clone REPOSITORY_URL
   ```

Workflow

1. Add, commit, and push like normal.

   ```
   git add myfile.py
   git commit
   git push origin main
   ```

   If you get an error when trying to push, it most likely means you haven't pulled recently.

2. To get up to date with whats on GitHub, do a **pull**.

   ```
   git pull origin main
   ```

   Make sure you have a **clean working directory** before pulling.

Merge Conflicts

When both team members change the same line in the same file, a **merge conflict** will occur. This is how git says that it doesn't know how to merge the changes together.

A merge conflict will happen after you run `git pull`. You will need to handle the merge conflict before continuing work.

When a merge conflict occurs:

1. Use `git status` frequently to check out what's going on.
2. Fix the conflicting file, either:
   - Manually edit the file and put it into the state that you want.
   - Use one of the commands below to use one version of the file.
3. `git add` the conflicting file(s).
4. `git commit` to conclude the merge.

   Git will prepopulate a commit message for the merge

Helpful Commands

During a merge conflict (Replace `FILE` with your specific file):

- Use your version of the file:

  ```
  git checkout --ours FILE
  ```

- Use the version of the file from GitHub:

  ```
  git checkout --theirs FILE
  ```

- Return the file to the original conflicted state:

  ```
  git checkout -m FILE
  ```

Avoiding Merge Conflicts

- Each team member should have their own "working" notebook so that two people aren't editing the same notebook.
    - For example, Jane works in jane_explore.ipynb and Pat works in pat_explore.ipynb, if both are working on different exploration tasks.
    - Once both team members are happy with their work, one person will combine elements of both notebooks into explore.ipynb.
- Eventually the individual works will be merged to a final notebook.
- Communicate clearly about who is working in this final notebook, only one person should be adding, comitting, and pushing this notebook at a time.