

# Software Design for Data Science

Team Collaboration

*Melissa Winstanley  
University of Washington  
February 6, 2025*





# Standups

- What is a standup?
  - 1-2 minutes per person
- Why standups?
  - Communicate status and actions within and between teams
- What do I say in a standup?
  - Progress you've made since the last standup
    - How it compares with the plan
    - If behind plan, how to compensate to make plan end date
  - Deliverables for next period
  - Challenges to making next deliverables
    - Technology uncertainties and blockers
    - Team issues


# Standups

In class!

# GitHub Issues


 Search or jump to... 




[Pull requests](#) [Issues](#) [Codespaces](#) [Marketplace](#) [Explore](#)

 **UWDATA515 / lecture6-demo** [Public](#) [Edit Pins](#) [Watch](#)

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)

[main](#) [2 branches](#) [0 tags](#) [Go to file](#) [Add file](#) [Code](#)

 **mwinstanley** against 26b71e1 16 minutes ago [12 commits](#)

 .gitignore	Initial commit	1 hour ago
 LICENSE	Initial commit	1 hour ago
 README.md	against	16 minutes ago

# GitHub Issues

- Good way to keep track of
  - Milestones
  - Tasks
  - Bugs
- If you link to an issue in a PR, it will auto-link the two
- Use it if you like!

# GitHub Pull Requests

- Pull Request = “PR”
  - Yeah, it’s a confusing name: “hey repo owners, please PULL my change into main”
- What’s in the description of a PR?
  - Background
    - Describe what change you are making
    - Describe how that change affects the application
  - Test plan
    - How did you test the code?
    - Include specific commands that you ran, and the output
- ALWAYS!
- Only push directly to main in a true emergency (not in this class!)

# Good option: draft pull requests

If you're not fully ready to submit it for code review



Another update

Write Preview

H B I

Leave a comment

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

Create pull request



Remember, contributions to this repository should follow our [GitHub Community Guidelines](#)



## Create pull request

Open a pull request that is ready for review

## Create draft pull request

Cannot be merged until marked ready for review

# Code Reviews

- What is code review?
  - Someone looks at, comments on, and approves a PR - NOT one person
- Why code review?
  - To find bugs
  - To improve code quality
- What do you comment on in a code review?
  - Bugs or missing pieces
  - Style
    - Choice of variable and function names
    - Code readability
  - If the PR doesn't have a test plan
  - How to improve reuse and efficiency
  - How to use existing Python packages



# Branch Protection

- Protect `main` against accidents!
- Require pull requests and code reviews
- Common in the real world

```
remote: error: GH006: Protected branch update failed for refs/heads/main.  
remote: error: At least 1 approving review is required by reviewers with write access.  
To https://github.com/UWDATA515/lecture6-demo.git  
! [remote rejected] main -> main (protected branch hook declined)  
error: failed to push some refs to 'https://github.com/UWDATA515/lecture6-demo.git'
```

# Branch Protection

GitHub Repository Page > Settings > Branches > Add branch protection rule

Branch name pattern \*

main

Protect matching branches

☒ **Require a pull request before merging**

When enabled, all commits must be made to a non-protected branch and submitted via a pull request before they can be merged into a branch that matches this rule.

☒ **Require approvals**

When enabled, pull requests targeting a matching branch require a number of approvals and no changes requested before they can be merged.

Required number

☒ **Do not allow bypassing the above settings**

The above settings will apply to administrators and custom roles with the "bypass branch protections" permission.

Do this now for  
your exercise  
repository!  
Do this on your  
project repository!

# Communication strategies

- Use a chat mechanism for informal, quick communication
  - Email doesn't really work
  - Slack
  - Microsoft Teams
  - Discord
  - Google Chat
- Don't hesitate to hop on a Zoom to follow-up
- Documentation as communication
  - Code documentation
  - Pull request description

# Pair programming

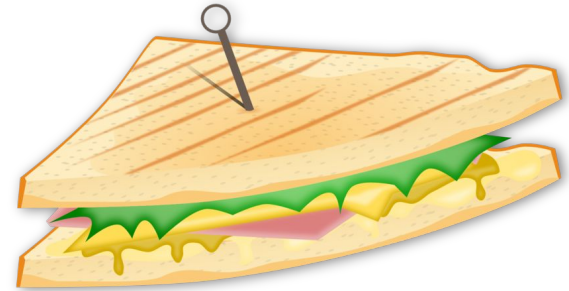
- Two people, one keyboard
- Fewer mistakes with more eyes
- Learning for both people

Consider trying it for difficult tasks!

- Swap who has hands on keyboard at least every 30 minutes
- Keep talking

# Fast feedback

- Quick responses to communication
  - Code reviews
  - Emails
  - Chat
- Constructive criticism
  - Consider (gently) giving teammates feedback
  - Make observations, not generalization
  - Sandwich approach
  - Listen to the response!
  - Don't wait until the survey at the end of the quarter!



# Collaboration Summary

- Standups
- GitHub issues
- Pull requests
- Code reviews
- Chat
- Pair programming
- Fast feedback