Please note!

- By attending this class, you consent to being recorded.

- This recording will be placed in an online folder accessible to the students of this class. It may also be distributed to other DTU students for education purposes.

46120: Scientific Programming for Wind Energy

Git and GitHub Teaser

Jenni Rinker

Agenda for today.

- Pull new course material ✔
- Discuss week00 solutions ✔

- Meet the 46120 Teaching Team. ✔
- Course introduction: Jenni. ✔
- What is good code: Ju Feng. ✔
- Teaser for git/GitHub: Jenni.

- Begin groupwork on Week 1 homework.

New groups and introductions.

- **Physical students,** sit in a NEW group of 3, where all three numbers are represented.
  - No overlaps with previous group, please!
  - You can do a group of 2 if needed.
- **Online students**: screenshot this or open locally. Accept BOR invitation.

In your groups:

1. Introduce yourselves.
2. What do you know about git/GitHub?
3. How have you collaborated on code previously?

Teaser for Git and GitHub

Version control is great.

Scientific programming requires version control.

Scientists and engineers develop code to load, generate, analyse, model, and/or visualize data.

This code is often extremely *dynamic*.

- Fixing bugs, implementing new features, etc.

Common but suboptimal way to track these changes:

```
project_code/
    analyse_data.py
    analyse_data_v2.py
    other_analysis.py
    other_analysis_v2.py
    make_plots_FINAL.py
```

Difficult to track history, revert changes. Further, does not allow for collaboration.

- Git and GitHub (or GitLab/BitBucket) are tools that address these issues.

Git is a distributed version control software.

- "Version control software":
  - A program that tracks who made what changes to which parts of files.
- "Distributed":
  - Copies of the files & history can be located on different computers.
  - Changes in different computers can be transferred and merged.
- Git is by far the dominant software in use. You will use it throughout the semester in this course.
- Unlike OneDrive/Dropbox, checkpoints are done manually.
- A collection of files whose history is tracked is called a "**repo**" (repository).

Remember: these slides are a "teaser"! Your homework includes more tutorials and exercises.

Overview of the git process.

working directory — *a bunch of files with changes, but only a few are ready to be added to history.*

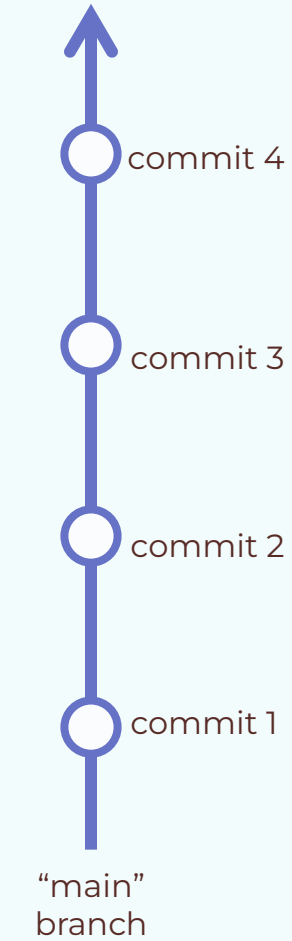
git add ← *tell git what files are ready*

staging area — *the files in the staging area are ready to be saved to repo history.*

take a snapshot! → **git commit**

repository — *the repository has the history of the files.*

Image from here



The beauty of branching.

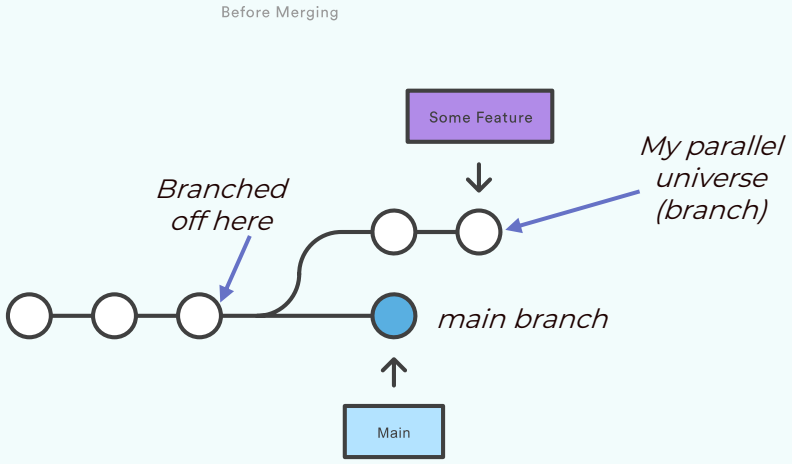
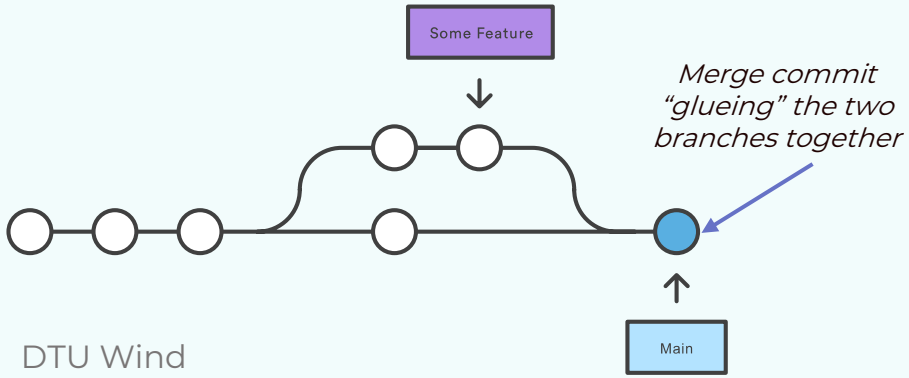


Image from Git Merge | Atlassian Git Tutorial

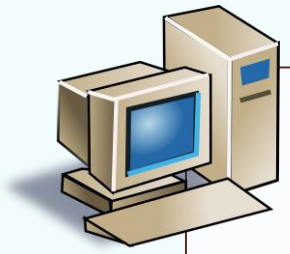


Be aware of merge conflicts ☺

- Sometimes, you need a parallel universe to develop your changes.
  - Maybe others are using code on main branch, or it's live deployed code.
- In this case, use a ***branch***.
- When you're ready, ***merge*** that branch back into main.
  - Different merge types possible. We will generally use "3-way merge".
  - Good tutorial: Git Merge | Atlassian

GitHub is a hosting service, an online version of your repo.

Git versus GitHub.



Git

ORIGIN – your computer

Git terminal

GitHub

REMOTE – online server

https://github.com/

status	*check current status*
add	*add files to staging area*
commit	*save "snapshot" of repo state*
clone	*download repo from server*
pull	*update local copy from server*
push	*update server from local copy*
branch	*create a new branch*

Excellent git cheatsheet: here

Reference of common git commands.

Run these in the Terminal, Anaconda Prompt or git-scm terminal.

- Download a repository from GitLab (“cloning”):

```
cd <directory where you want the repository>
git clone <path to github/gitlab url>
```

- Update your local copy from the cloud (“pulling changes”):

```
git pull [origin main]
```

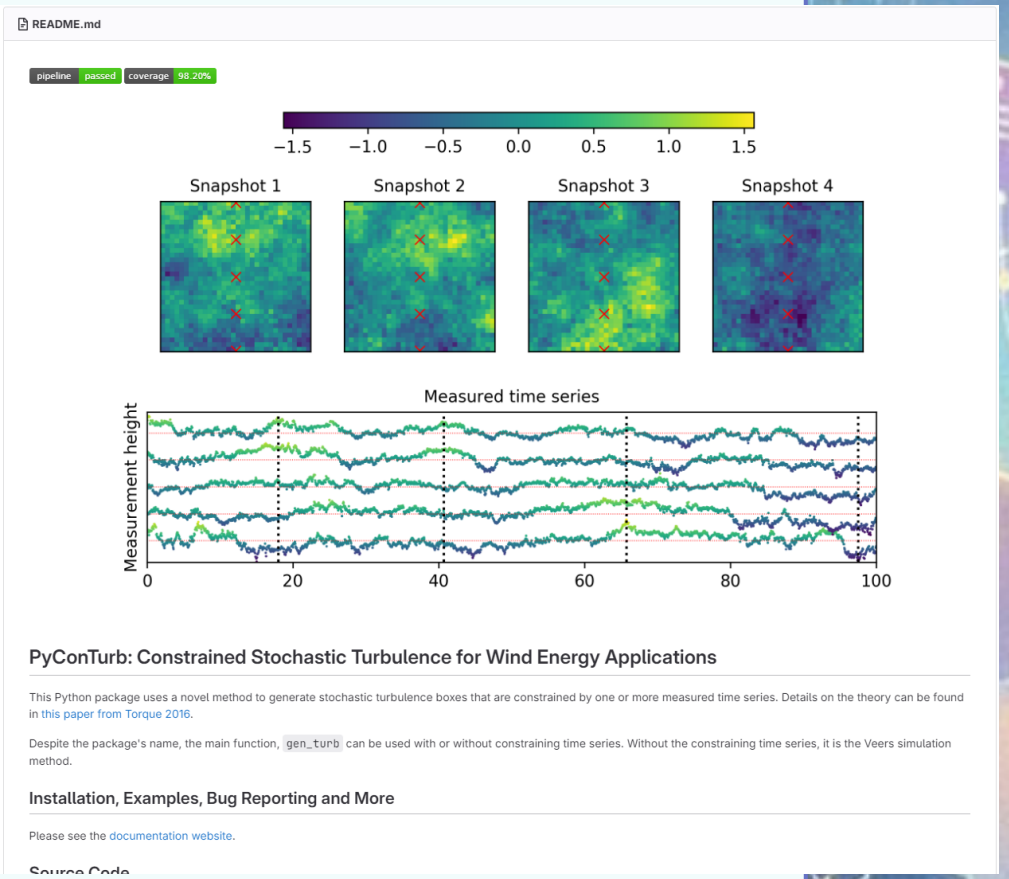
- Update the cloud from your local copy (“committing and pushing”):

```
git add <file1> <file2> <folder1>
git commit -m "<insert commit message>"
git push [origin main]
```

Push to cloud

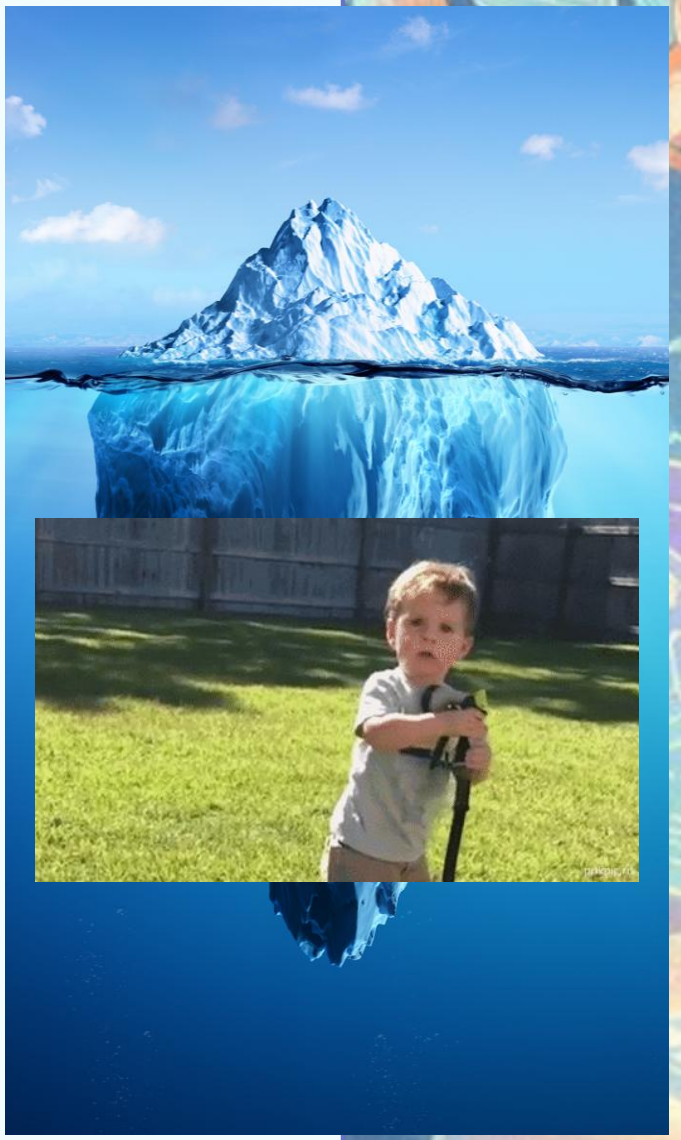
Example of GitLab/GitHub repositories in the wild.

- TOPFARM.
  - A Python package for wind-farm optimization.
- PyConTurb and Hipersim.
  - Python packages for creating turbulence fields from measurements.
- IEA Reference wind turbines on GitHub:
  - 15 MW.
  - 22 MW.
- Open-science example of a repo containing code for a paper.

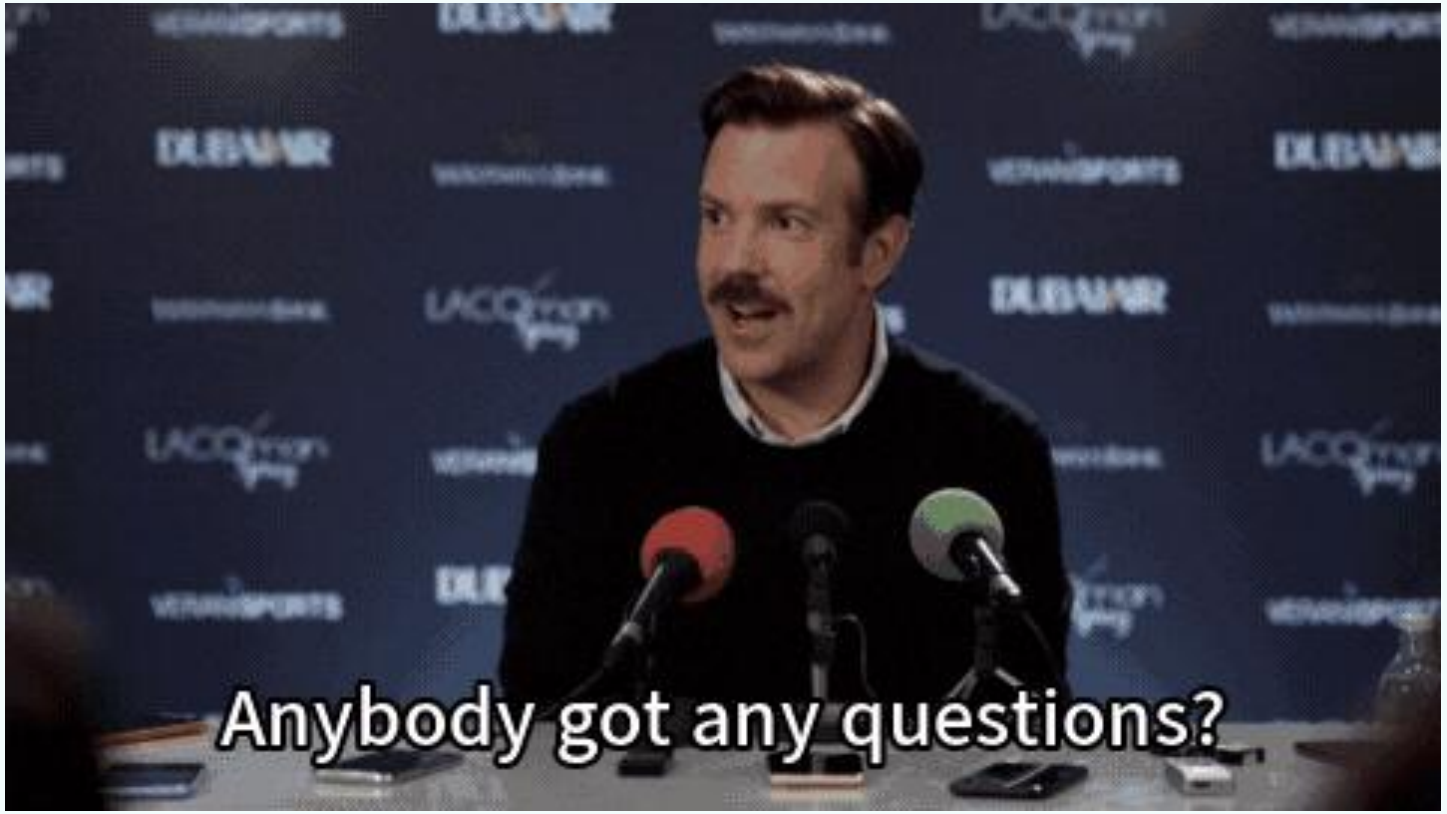


Final remarks.

- **Git has a lot of features/complexity**.
  - Only basics here. And multiple ways to do same thing.
  - Be aware when searching the internet and using GenAI.
- **Be very, very patient with yourself**.
  - A lot of new things: command line, git, GitHub, etc.
  - This week may feel overwhelming. It slows down, promise.
- **Seek support when needed**.
  - Try teammates, Google or GenAI to solve issues first, but of course utilize Slack, office hours and/or TAs.
- **NEVER clone a repo into an existing local repo**.
  - All repos should be in separate folders on your computer.



Questions?



Homework for this week

Time to get your hands dirty!

NB: Detailed instructions on 46120 GitHub.

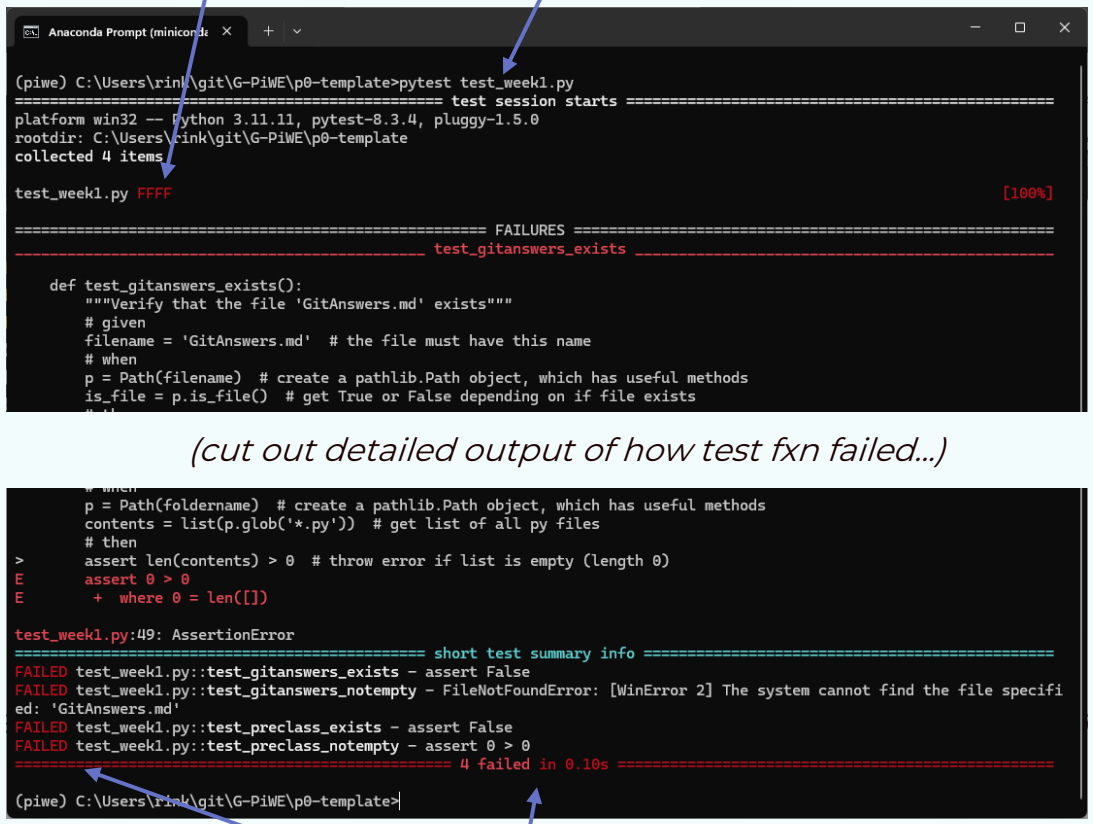
Overall objective.

In a team of 2 to 3*:
Create, review, and merge two “feature branches” such that all tests in test_week1.py pass.

** These teams are only for weeks 1 and 2, so don’t panic about group formation.*

call pytest: pytest [name of file with tests]

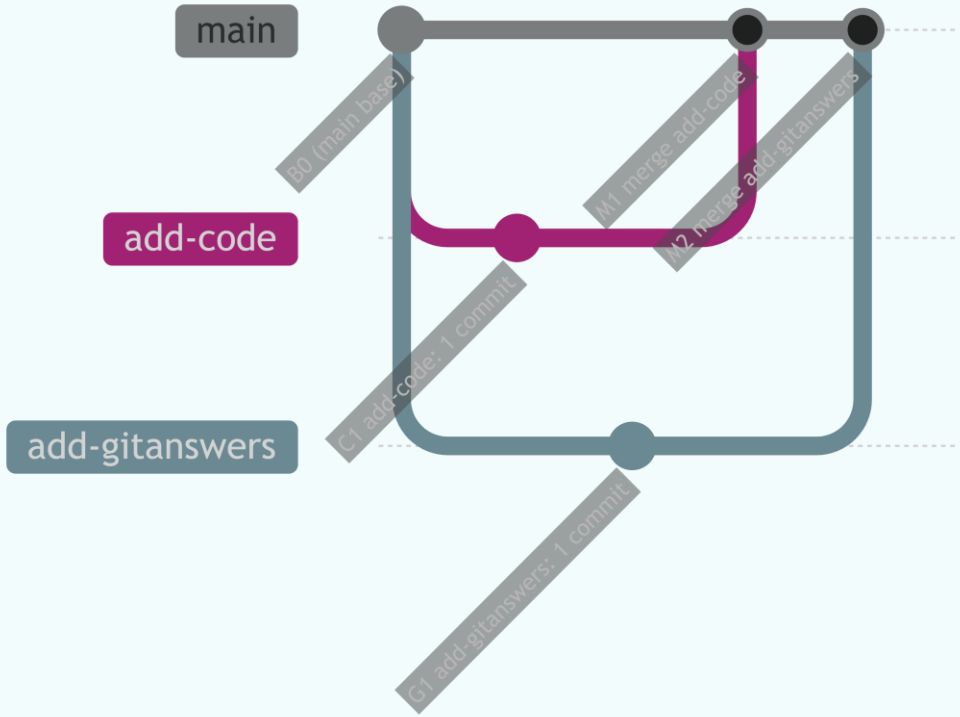
“F” indicates a failing test



(cut out detailed output of how test fxn failed…)

short summary of which tests failed with which error messages

Overview of steps.



NB: Detailed instructions on 46120 GitHub.

In class

1. Form teams, join the GitHub assignment, plan work. *check tests*

Individually or as team

2. Watch videos on introduction to git/GitHub.
3. Make a feature branch: add-code. *check tests*
4. Make a feature branch: add-gitanswers. *check tests*

As team

5. Review & merge feature branches. *check tests*

Well, what are we waiting for?

- Week 1 folder on 46120 GitHub has detailed instructions.
  - Complete at least **Part 1** before you leave.
  - NB: You're expected to work **about 6 hours outside of class**. Schedule time appropriately.
- If there are issues joining your team repo on GitHub, contact a TA/instructor today.
- Online students:
  - We will open self-navigable BORs in a minute.
  - Use the Zoom chat and/or microphone to arrange groups and claim BORs.

Any questions?