Programming for psychologists

Lecture 5: Git & GitHub

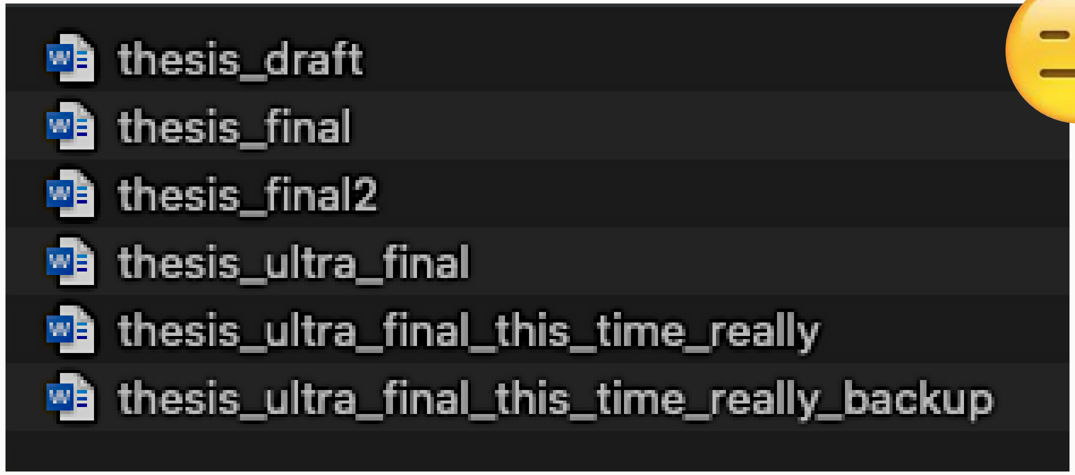
Matthias Nau

Version control

What is **version control** and why do we care?

A system that helps track and manage changes to files, code, and projects over time for you and your collaborators.

Creating copies when updating files



thesis_draft
thesis_final
thesis_final2
thesis_ultra_final
thesis_ultra_final_this_time_really
thesis_ultra_final_this_time_really_backup

Would it not be great to have a time machine instead?

A great time machine would allow you to go back to previous versions of your files, and see who changed what and when.

This time machine exists, and it is called: git

Git

What is Git?

The world's most popular **version control** and **collaboration** system that is **free** and **open source**.

Git allows you to:

- **Revert files** or the whole project to an earlier state
- **Compare changes** over time
- See **who** modified **what** and **when**
- **Control modifications** by collaborators with the permission of admin/owners

There are alternative version control systems but **Git is the most popular one** in the world.

Linus Torvalds
(Creator, also developed Linux)

Git alternatives

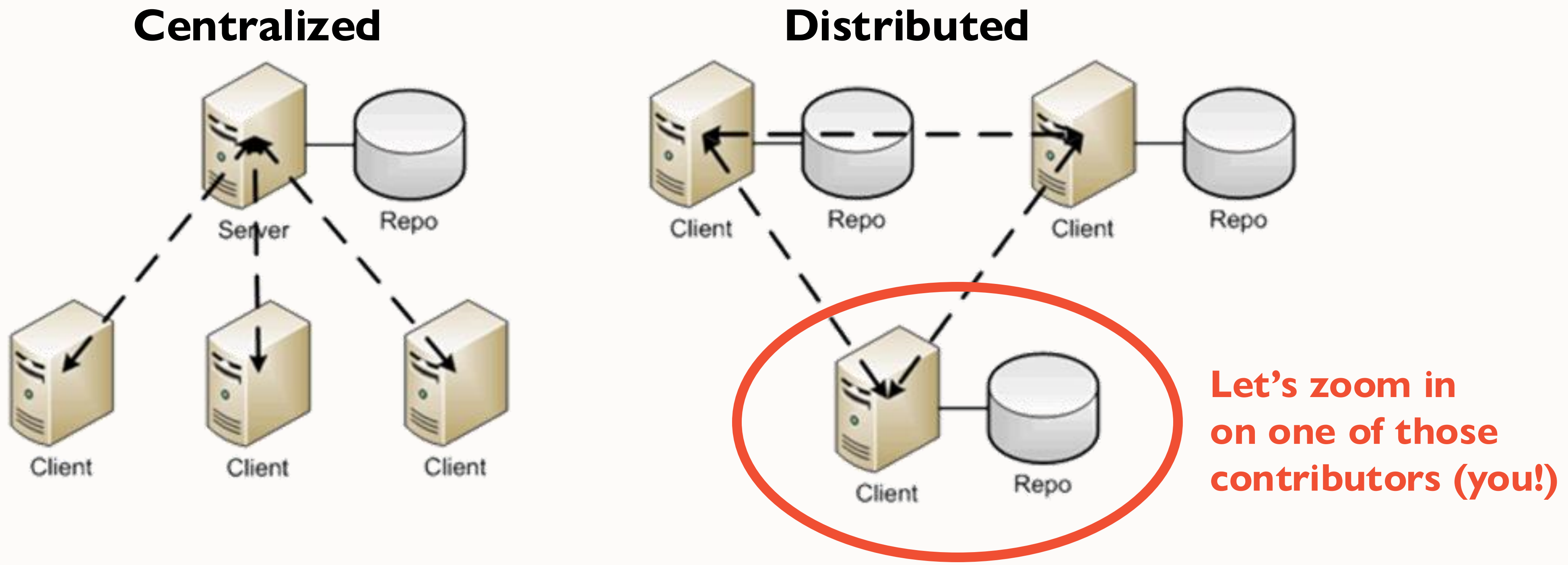
mercurial

SUBVERSION

PERFORCE

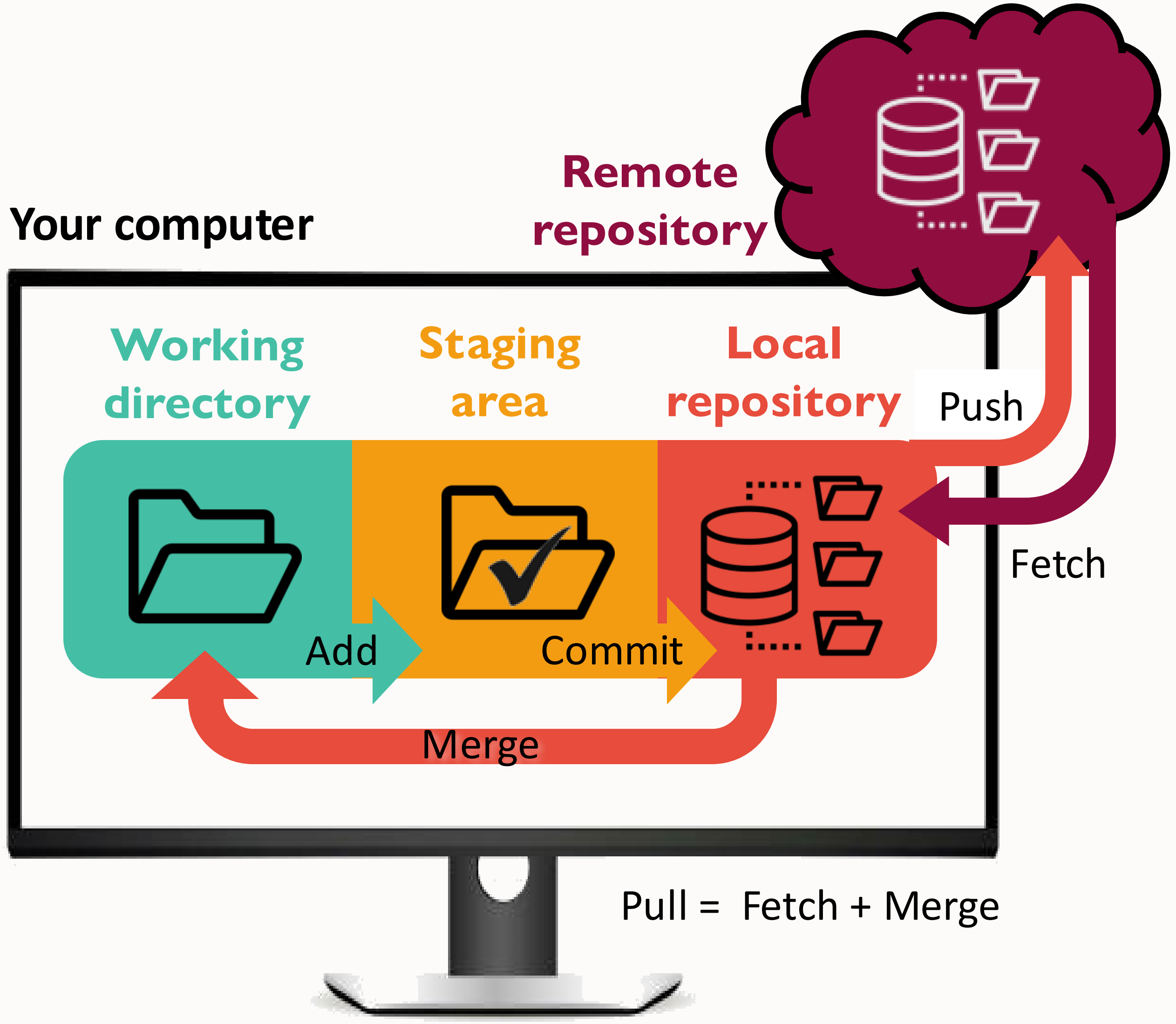
How does Git work?

Git is distributed, meaning that every contributor to the project has a local "snapshot" of all project files and their history. Every contributor has their own **Repository**.



Here is a great, free PDF version of a book on version control with Git:
https://www.fdr.uni-hamburg.de/record/14149

How does Git work?



Remote repository (GitHub)
A shared "picture book" with the central project version accessible to collaborators.

Local repository
Your personal "picture book". Each page contains a snapshot of the project.

Staging area
Your "draft", this is where you gather pieces before comitting to them

Working directory
This is where you create new content, make modifications, etc.

Important commands
Add (to staging area), commit (to local repo), push (to remote repo), fetch (from remote repo), merge (from local repo), pull (from remote repo)

GitHub

What is GitHub?

Extremely popular **online repository hosting service** for Git

Includes **cloud storage** for your code and related files.

While Git is a command line tool (i.e., executed in the Terminal),
GitHub provides a **web-based graphical interface**.

GitHub also has functions of a social network
(e.g., exchange among users, many great collaboration features).

The most popular **code sharing** platform in the world,
and an essential component of the **Open Science** movement.

Owned by Microsoft, like VSCode → Great integration of the two!

Why do we use GitHub?

Version control	Can always go back! No more *thesis*, *thesis_final*, *thesis_final2…*
Collaboration	Integrates edits from multiple people and keeps track who did what & when
Documentation	All scripts are in one place with documentation (Readme file)
Reproducibility	You and others can recreate or modify a script many years later
Visibility	Code sharing increases visibility of your work and strengthens your CV

Happy Scientists!

GitHub: Essentials beyond Git

GitHub repositories provide **online storage** for your **project files** alongside their **version history** and **documentation**.

Repositories can be either **public or private**, in both cases allowing collaboration.
You can have **many repositories** (e.g., typically one per project).

The **first time you download** a repository, you do not pull but you **clone** it.
This creates a full copy of the online repo on your computer.

Often, you may want to **copy someone else's repo** (e.g., to make your own personal changes).
In this case, you **fork the repository**, which creates a copy in your online account.

GitHub: Collaboration

Each collaborator typically works in a separate **Feature branch**,
an online copy of the repository separate from the **Main branch**.

The Main branch is the primary, stable version of the project.

Once you are happy with your edits, you write a **Pull request**,
a proposal of changes that can then be reviewed by others.

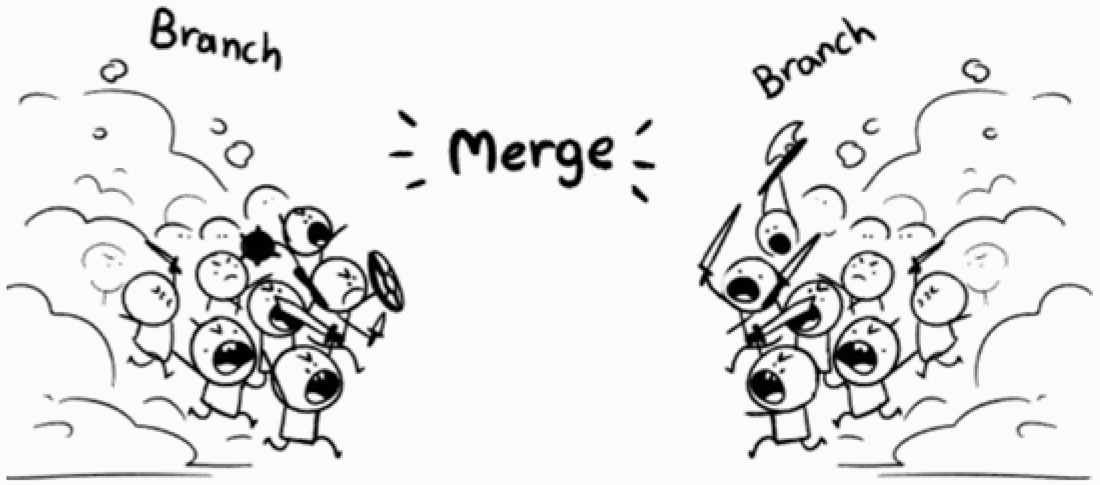
This means that **you rarely push directly to the Main branch** of the GitHub repo,
but **you write a pull request**, which is then reviewed and accepted by someone else.
(We will practice this on Monday).

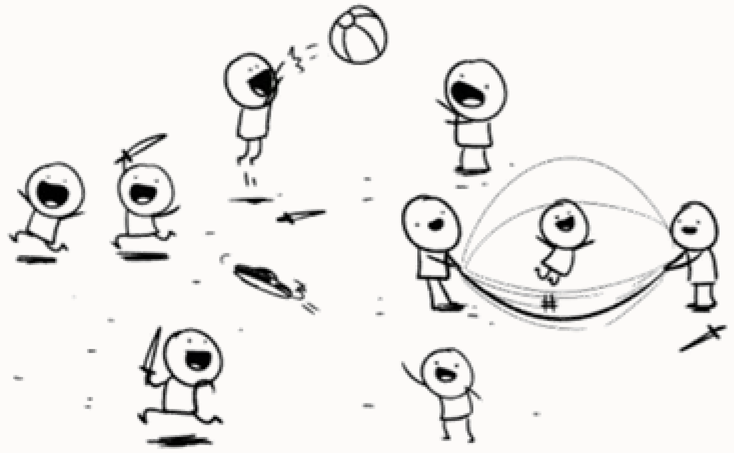
GitHub: Collaboration

Once all collaborators are happy with the changes, the branches are **merged** (i.e., the feature branch is merged into the main branch).

But what if two collaborators made changes to the same line of code? 😱

Merge conflicts → **Resolve by accepting one, by making a new branch, or by adjusting yours**





GitHub: Collaboration

GitHub has a great feature called **GitHub Issues,**
typically used for **organizing tasks and tracking their progress.**

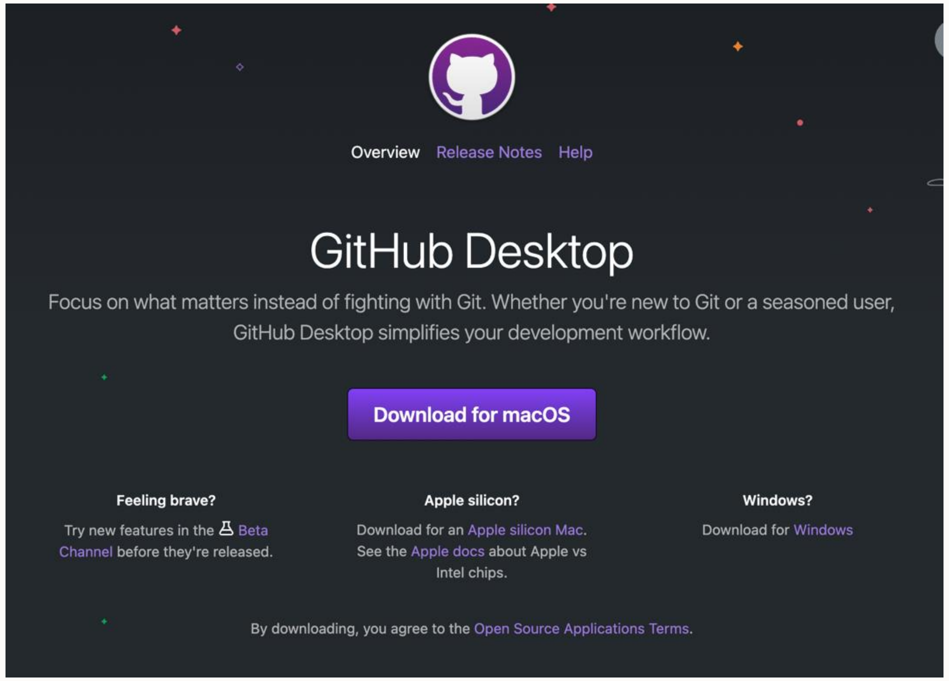
Issues are also used for **reporting problems** with code (e.g., bugs).
When used for bug reports, issues are a bit like a "stack overflow page".

Another essential feature of GitHub is the **Readme file**.
Readme files allow you **add documentation** to a repository
to explain what the project is, how to install it etc.

Readme files use **Markdown** (like Jupyter Notebooks)
and are rendered online in the browser.

GitHub: Let's have a look together!

GitHub Desktop app



Overview Release Notes Help

GitHub Desktop

Focus on what matters instead of fighting with Git. Whether you're new to Git or a seasoned user, GitHub Desktop simplifies your development workflow.

Download for macOS

Feeling brave?
Try new features in the Beta Channel before they're released.

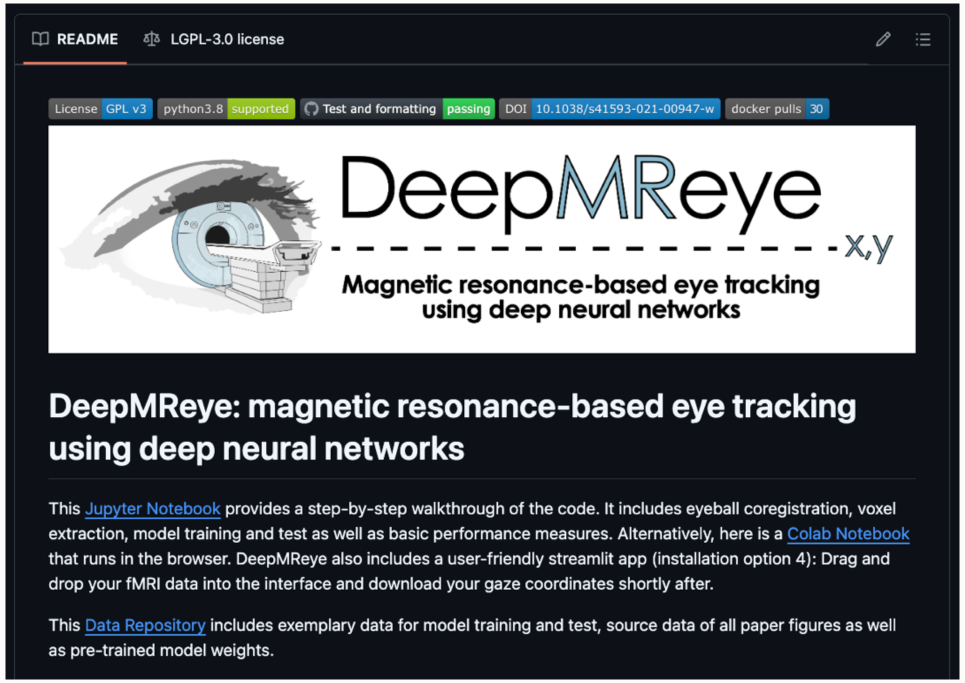
Apple silicon?
Download for an Apple silicon Mac. See the Apple docs about Apple vs Intel chips.

Windows?
Download for Windows

By downloading, you agree to the Open Source Applications Terms.

Most people (that I know) use GitHub from the Terminal/Command Prompt, but there is a Desktop app as well. Feel free to check it out!

Example Repo: DeepMReye



README LGPL-3.0 license

License GPL v3 | python3.8 supported | Test and formatting passing | DOI 10.1038/s41593-021-00947-w | docker pulls 30

DeepMReye
x,y
Magnetic resonance-based eye tracking using deep neural networks

DeepMReye: magnetic resonance-based eye tracking using deep neural networks

This Jupyter Notebook provides a step-by-step walkthrough of the code. It includes eyeball coregistration, voxel extraction, model training and test as well as basic performance measures. Alternatively, here is a Colab Notebook that runs in the browser. DeepMReye also includes a user-friendly streamlit app (installation option 4): Drag and drop your fMRI data into the interface and download your gaze coordinates shortly after.

This Data Repository includes exemplary data for model training and test, source data of all paper figures as well as pre-trained model weights.

Let's click through this repo together: https://github.com/DeepMReye

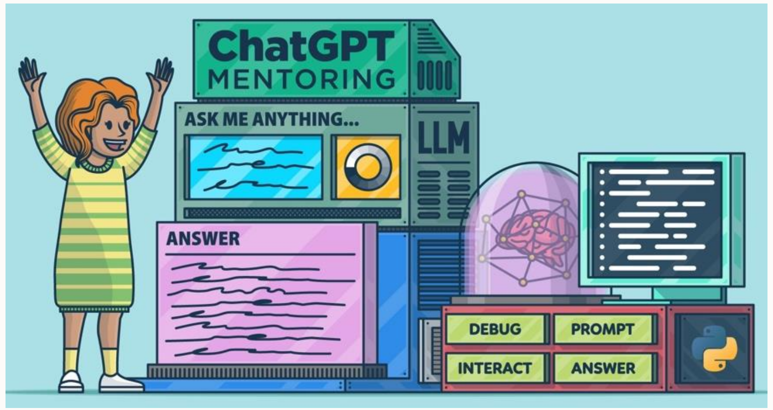
Open Science

Git and **GitHub** are closely aligned with the principles of **Open Science**

- **Transparency**: Version control with Git allows for a transparent history of changes.
- **Reproducibility**: Sharing code and data on GitHub promotes reproducible science.
- **Collaboration**: GitHub fosters open collaboration across disciplines.
- **Open Access**: GitHub simplifies code sharing and long-term maintainance.
- **Documentation**: Helps provide detailed research workflows for replication.
- **Open Review**: GitHub's pull request system facilitates open peer review.
- **Archiving**: GitHub helps with long-term access and citation of research.

GitHub Copilot

GitHub Copilot



Large language models (like ChatGPT) are revolutionizing programming. They can **greatly accelerate your progress** in a project by providing useful code suggestions.

Importantly, to use AI coding assistants responsibly, **you still need to know how to code.** It is **your responsibility** to ensure that the code is doing what you think it does.

Copilot helps you **write code faster** and with **less effort by making suggestions** that you may ignore or accept. Emphasis is on "Co". You are the pilot! It is **integrated into VScode** and **easy to use**.

- **Practical 5.1 (Tuesday):**
  Anna will show you how to use GitHub & Copilot tomorrow.
- **Practical 5.2 (Friday):**
  Guided group discussion on AI-assisted coding.

Before the next practical, go through these slides again!

Do you know what the following terms mean?

- Git & GitHub
- Working directory
- Staging area
- Local repo
- Remote repo
- Add
- Commit
- Push & Pull
- Fetch
- Merge
- Pull request
- Main branch
- Feature branch
- Cloning
- Forking
  Radme file
- GitHub issues
- Copilot

Thanks! Have a good week!

Supplementary material: Using git from the Terminal

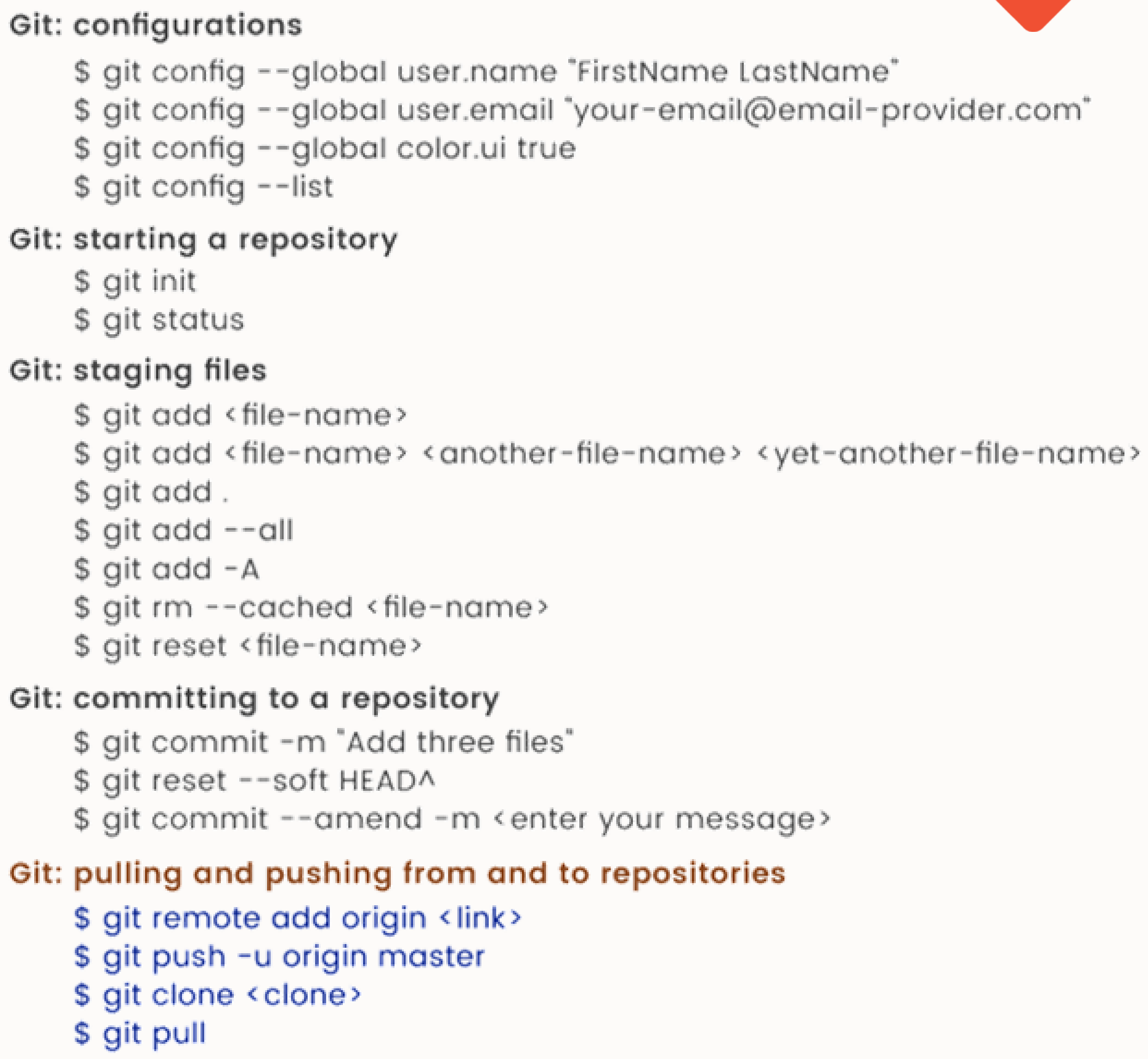
Git commands are typically executed from the **Terminal** or **Command prompt**.

First, your working directory needs to be the repo. In the terminal, navigate there via:
cd /Users/username/your-repo

You can find the **current working directory** using the command **pwd**

You can **list all files and folders** in a directory using the command **ls**

Once you are in the right directory, you can run **git commands** (e.g., git add <fileaname>)



Git: configurations

```
$ git config --global user.name "FirstName LastName"
$ git config --global user.email "your-email@email-provider.com"
$ git config --global color.ui true
$ git config --list
```

Git: starting a repository

```
$ git init
$ git status
```

Git: staging files

```
$ git add <file-name>
$ git add <file-name> <another-file-name> <yet-another-file-name>
$ git add .
$ git add --all
$ git add -A
$ git rm --cached <file-name>
$ git reset <file-name>
```

Git: committing to a repository

```
$ git commit -m "Add three files"
$ git reset --soft HEAD^
$ git commit --amend -m <enter your message>
```

Git: pulling and pushing from and to repositories

```
$ git remote add origin <link>
$ git push -u origin master
$ git clone <clone>
$ git pull
```