Introduction to GitHub.

Eirini Zormpa, 28 June 2023

AI for Multiple Long-term Conditions
Research Support Facility

Summary of last week

- ✅ Version control is an approach to recording changes in file(s) over time so that you can track their history, review changes, and go back to earlier versions.
- ✅ Git is a version control system (arguably the most popular)
- ✅ You can use git locally on your computer through the command line, specialised GUIs, or plug-ins for IDEs, but you can also use git online
- ✅ You learned how to start tracking files, save the edits that you made to them, and revert to earlier versions

Questions from last time?

Learning objectives

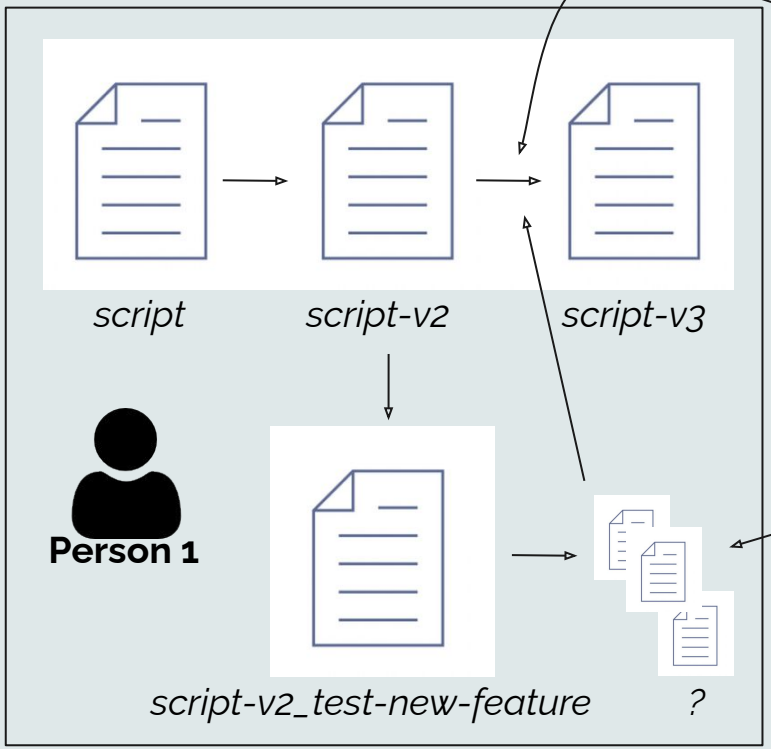
By the end of this session, you'll be able to:

- Explain what GitHub is and why it's useful
- Connect local and GitHub repositories
- Understand what branches are and when to use them
- Create and merge branches
- Open and merge pull requests

Materials partly based on Code Refinery, esp. the lesson Introduction to version control with Git

Working on GitHub.

My git & GitHub journey



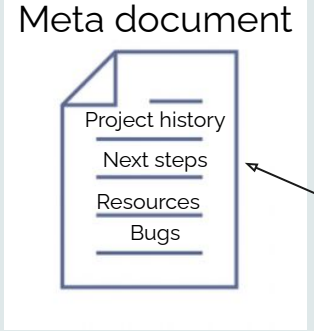
Person 3

Suggests a change but doesn't edit the files

Person 2

Helps with testing new feature

Person 1, 2 & 3



script-workflow

Person 4

Slide is intentionally overwhelming & horrible to look at

GitHub

Git is super useful for working locally on your own computer, but it falls short if you want to collaborate on a project with other people or even if you want to work on your own from two devices.

GitHub is a popular website for hosting and sharing online projects you have been tracking with git ✨

Benefits of GitHub

- Much easier to collaborate when everyone has access to the same repository

Benefits of GitHub

- Much easier to collaborate when everyone has access to the same repository
- Robust annotation of contributions

Benefits of GitHub

- Much easier to collaborate when everyone has access to the same repository
- Robust annotation of contributions
- Advanced permission options in terms of setting who can see, edit, and manage repositories

Benefits of GitHub

- Much easier to collaborate when everyone has access to the same repository
- Robust annotation of contributions
- Advanced permission options in terms of setting who can see, edit, and manage repositories
- Functionality to record issues and bugs, as well as to have discussions with the community that uses the repository

Benefits of GitHub

- Much easier to collaborate when everyone has access to the same repository
- Robust annotation of contributions
- Advanced permission options in terms of setting who can see, edit, and manage repositories
- Functionality to record issues and bugs, as well as to have discussions with the community that uses the repository
- Functionality for project management

Benefits of GitHub

- Much easier to collaborate when everyone has access to the same repository
- Robust annotation of contributions
- Advanced permission options in terms of setting who can see, edit, and manage repositories
- Functionality to record issues and bugs, as well as to have discussions with the community that uses the repository
- Functionality for project management
- Continuous integration and continuous development (CI/CD) functionality

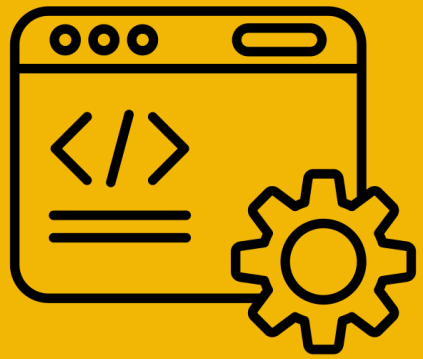
Benefits of GitHub

- Much easier to collaborate when everyone has access to the same repository
- Robust annotation of contributions
- Advanced permission options in terms of setting who can see, edit, and manage repositories
- Functionality to record issues and bugs, as well as to have discussions with the community that uses the repository
- Functionality for project management
- Continuous integration and continuous development (CI/CD) functionality

Working on GitHub*.

*with yourself

Follow along with me!



Keeping things private 🙊

.gitignore

This is a hidden file that keeps **`git`** from tracking files altogether.

Good files (or folders) to add there: data, files about the workings of your computer (e.g. DS_Store on Macs)

Private repos

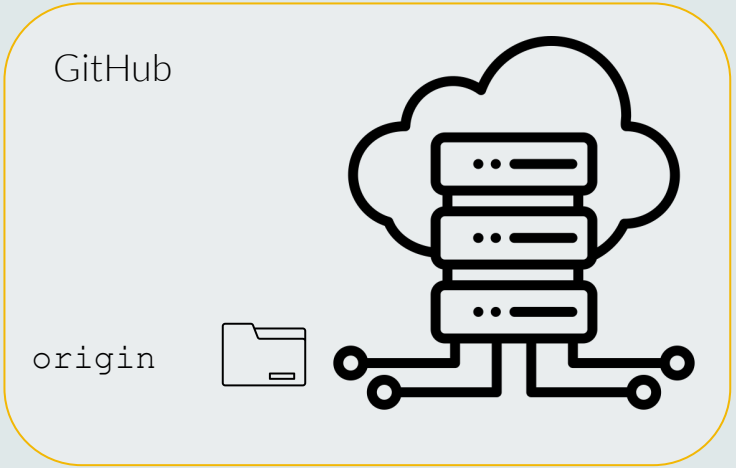
An option offered by GitHub to give you control over who can view your projects.

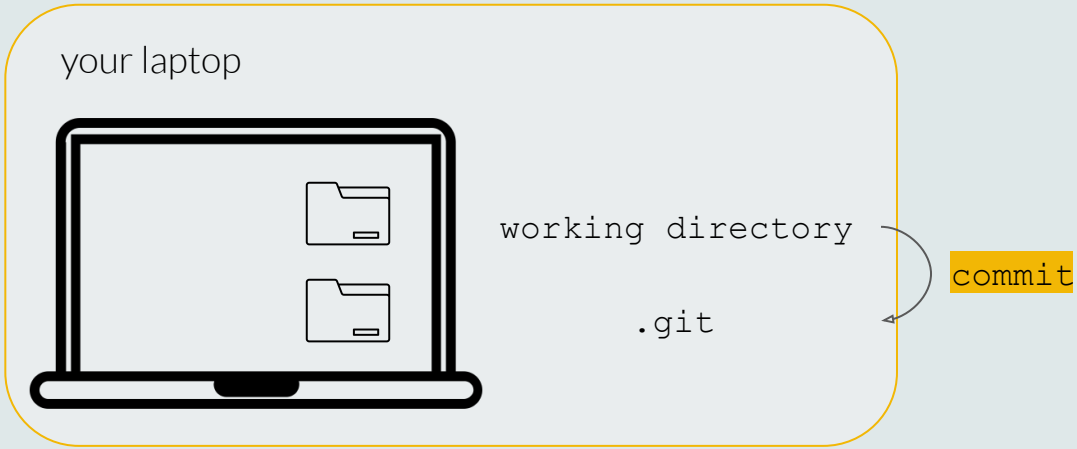
Often a temporary stage before publishing projects

*Note that even in public repos, you control who can **contribute** to your repo.*

Much simplification!

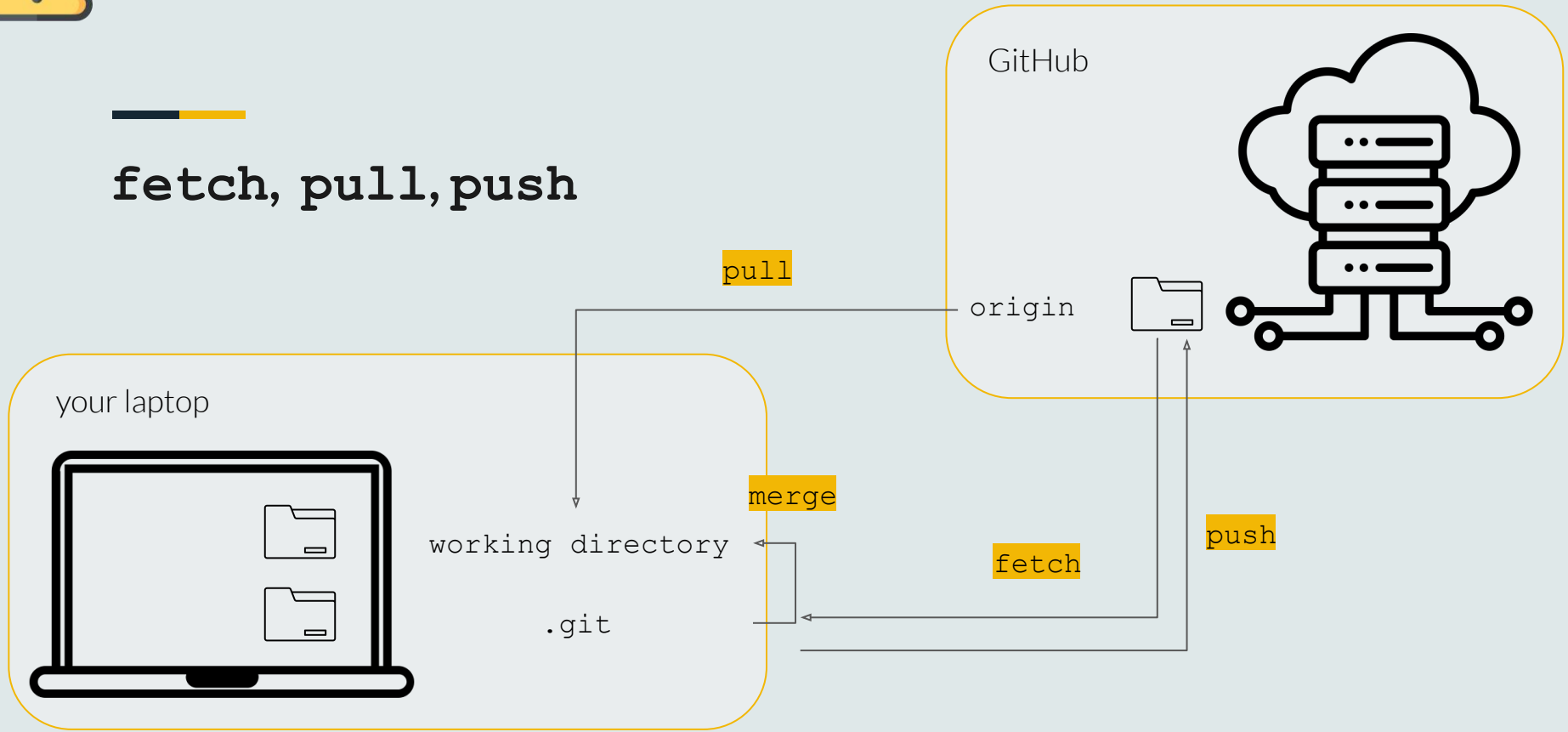
`fetch, pull, push`





Much simplification!

`fetch, pull, push`



Questions?

Break.

Working with `branches`.

Linear workflow

Image from Chapter 3 of the *Mozilla Science Lab's Study Group Orientation handbook*, used under a Mozilla Public License Version 2.0.

Linear workflow with GitHub

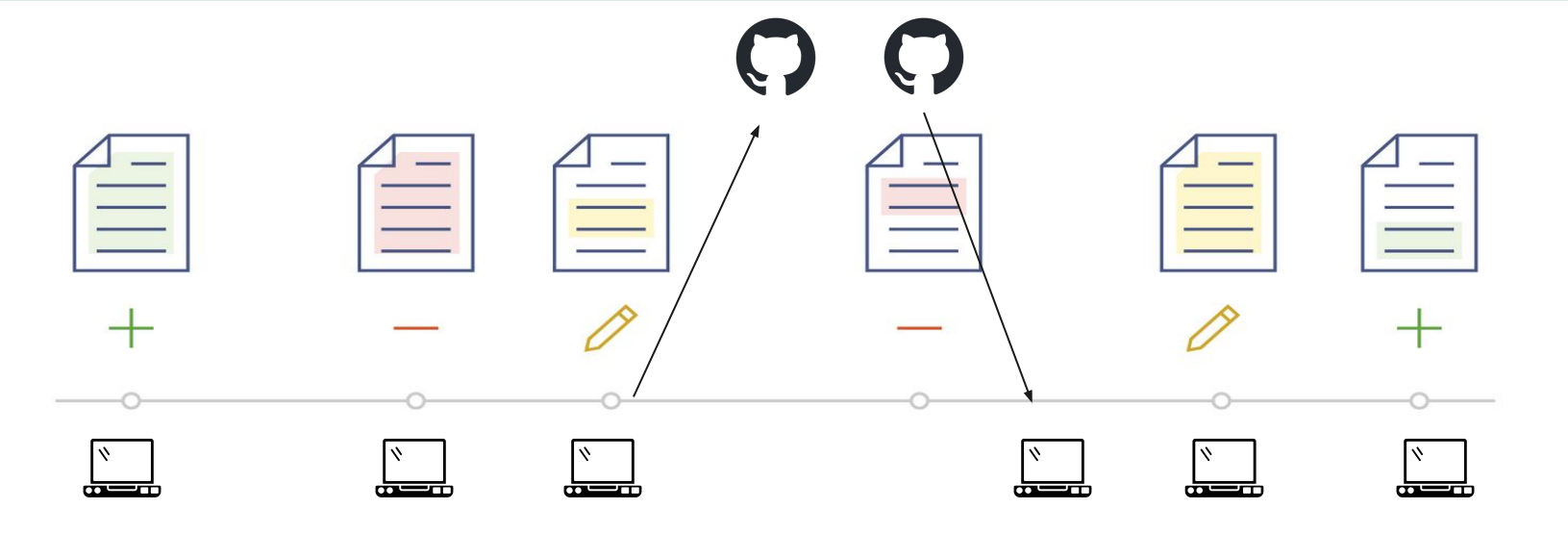
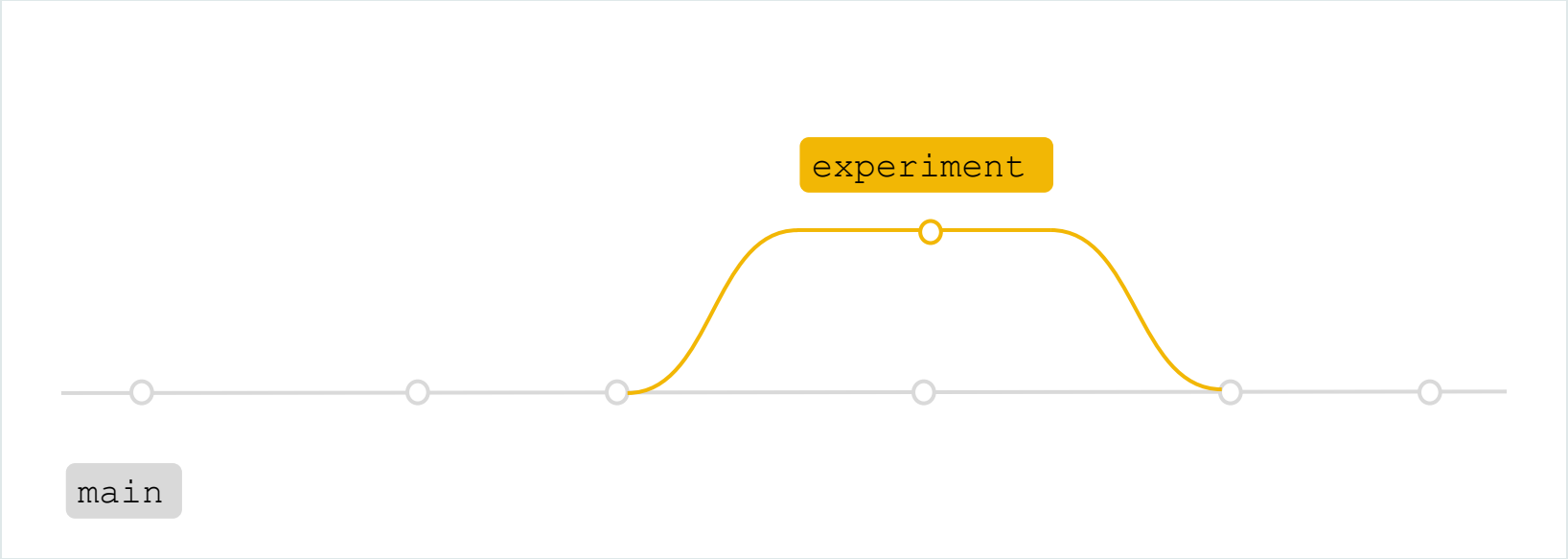


Image from Chapter 3 of the *Mozilla Science Lab's Study Group Orientation handbook*, used under a Mozilla Public License Version 2.0.

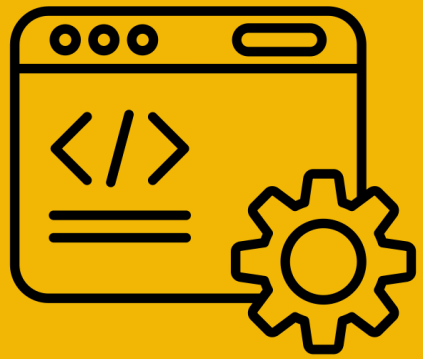
Branching

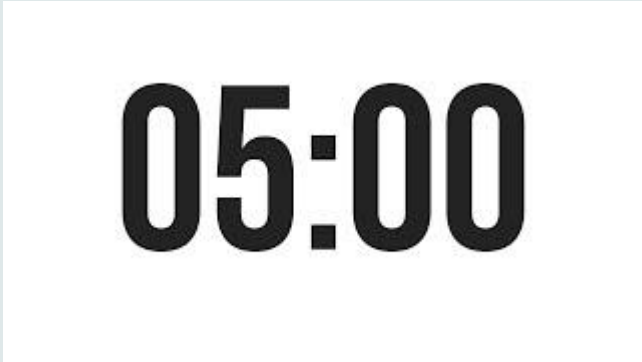


Branches help retain a working version of the code, while allowing us to experiment.

Follow along with me!

… and note that the example I'm using is silly





Exercise 1

⏰ **5 minutes**

We have already published the `recipe-experiment` branch to GitHub.

Following the same steps as before:

1. Make the BT vegetarian and commit your changes to the `recipe-experiment` branch
2. Merge the `recipe-experiment` branch to `main` and
3. Push the changes to GitHub

Beware of bugs ❗ 🐛

Deleting a branch on GitHub doesn’t delete the branch on GitHub Desktop (but the opposite works).

Exercise 2

⏰ **10 minutes**

1. Create a new branch called `my-sandwich`
2. Publish the branch on GitHub
3. Edit the sandwich-recipes.md to include a new recipe
4. Commit the changes
5. Push the changes to origin
6. Open a pull request
7. Merge the pull request
8. Delete the branch

Branch usage

One-person project 🧑‍💻

- Start working with `main` only
- Use branches to test out new ideas and things you're not sure about
- You can merge branches directly to `main`

Multiple-people projects 🧑‍💻🧑‍💻

- Keep `main` protected
- Create branches for changes
- Request a review from a collaborator before merging

Platforms beyond GitHub

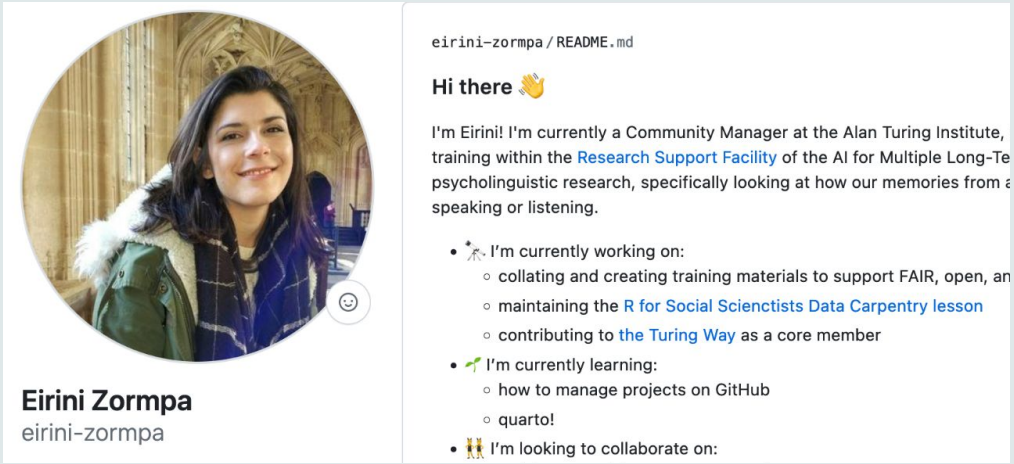
- GitHub isn't the only option for working collaboratively with git, though it is the most popular
- Other options:
    - GitLab
    - BitBucket

Terminology

- **`Remote`**: a version of your project that is hosted on the Internet (e.g. GitHub) or a network, for the purpose of collaboration
  - **`origin`**: a common name for a remote, typically the one that the project was cloned from
- **`Branch`**: an independent line of development. Technically, it is a pointer to a specific commit.
  - **`main`**: the default branch name. The default name used to be **`master`**.
- **`Merge`**: an action to incorporate changes from another repository or branch
- **`Pull request`**: a mechanism to inform others you've pushed changes to a repo

Questions?

Optional exercise(s)/homework



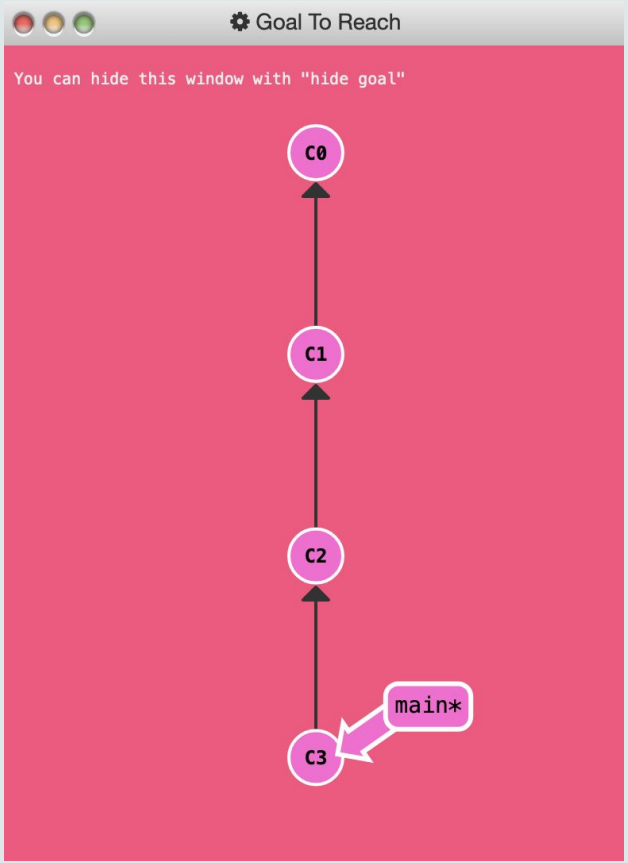
Eirini Zormpa
eirini-zormpa

eirini-zormpa / README.md

Hi there 👋

I'm Eirini! I'm currently a Community Manager at the Alan Turing Institute, training within the Research Support Facility of the AI for Multiple Long-Te psycholinguistic research, specifically looking at how our memories from a speaking or listening.

- 🔭 I'm currently working on:
  - collating and creating training materials to support FAIR, open, an
  - maintaining the R for Social Scienctists Data Carpentry lesson
  - contributing to the Turing Way as a core member
- 🌱 I'm currently learning:
  - how to manage projects on GitHub
  - quarto!
- 👯 I'm looking to collaborate on:



1. [create a personal README]
2. [practise git branching]


THANK YOU!!
Scribbria