Enhancing Data Support: Practical Reproducibility

Samantha Wittke

samantha.wittke@csc.fi

CSC - IT Center for Science

Outline

Intro and practicalities (10min)

GitHub (45min)

- Version control
- Creating a repository
- Contributing to a repository

Break (10min)

Jupyter (45min)

- Computational notebooks
- Basic features

Where to go from here (10min)

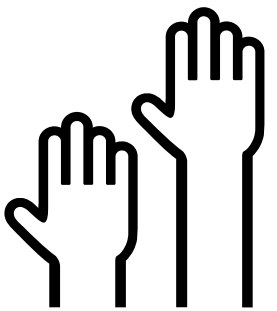
*This session will only provide a **glimpse** into the world of GitHub and Jupyter.*

Questions

Please ask at any time -> Zoom chat or raise hand

Please be nice and patient with one another. Everyone is on different level and that is OK!

No breakoutrooms today.



Chatter

1. Type your response into the chat, but WAIT to hit enter
2. Listen for the countdown (three, two, one, CHAT!)
3. Hit enter and watch the responses!

Chatter - Let's practice!

One word to describe your morning today?

Type your answer in chat, but wait to send

Prerequisites for following along

- GitHub account

- Access to Noppe

You can also just watch and ask questions :) But do try it out by yourself later, it is the best way to learn! Recording will be made available.

Today: Two perspectives

Researcher who codes

Research support

What reproducibility means in practice

Clear, accurate, and complete record-keeping of:

- **Research methods**, including data collection, processing, analysis, visualization
- **Research code and computational workflows**, including models, data processing scripts, and software notebooks
- **Computational environment**, including system software and hardware requirements, including the version number of each software used
- **Data files** (and other outputs) properly formatted and accompanied by rich metadata
- **Project history and narrative**, from the project planning and development, through project activities during execution, to the project completion

[Josefine Nordlings slide from part 1 of this training]

What reproducibility means in practice

Clear, accurate, and complete record-keeping of:

- **Research methods**, including data collection, processing, analysis, visualization
- **Research code and computational workflows**, including models, data processing scripts, and software notebooks
- **Computational environment**, including system software and hardware requirements, including the version number of each software used
- **Data files** (and other outputs) properly formatted and accompanied by rich metadata
- **Project history and narrative**, from the project planning and development, through project activities during execution, to the project completion

[Josefine Nordlings slide from part 1 of this training]

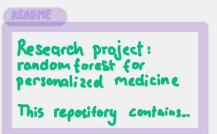
Reproducible Research

6 helpful steps

1 Get your files + folders in order

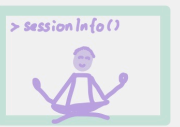
2 Use good names for files, folders, functions, ...

3 Document with care: README, Metadata, code comments, ...



4 Version control code, text, ...

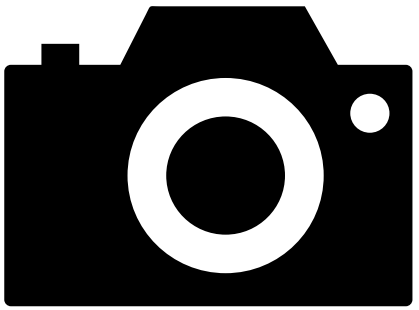
5 Stabilize computing environment and software



6 Publish your research outputs: Code, data, documents, ...

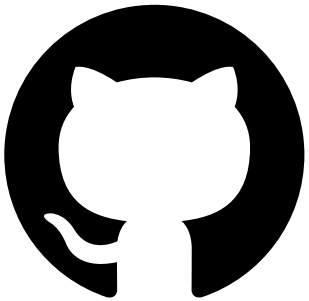
Code that works (for you) and is shared is not the same as reproducible code!

Version control



- Version control is the practice of **tracking and managing changes over time**.
- You can think of version control like regularly taking a photo ("snapshot") of your work.

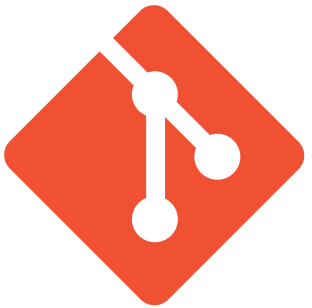
GitHub



-> one place to **find the source** of software, webpages, presentations, books, games, ...

... and a **place to collaborate** and share

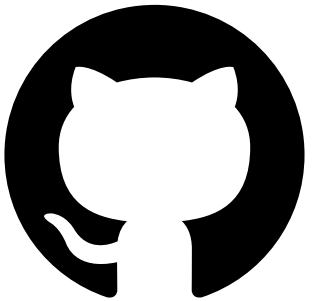
Git



Tool/format for version control

Others: Subversion, Mercurial, ...

GitHub



Hosting service for Git repositories with web interface -> Share and collaborate

Others: GitLab, Codeberg, ...

Did you know?

In-house GitLab: Host your own repositories safely within the walls of your organisation.

Collaboration in the Nordics: Nordic GitLab hosted by DeIC

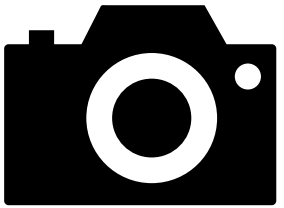
Why do we teach GitHub? ➡ Most used, beyond borders

Repositories - a place to store

*A repository is the most basic element of GitHub. It's a place where you can **store your code**, your files, and each file's **revision history**. Repositories can be **owned by persons or organisations**, have **multiple collaborators** and can be either **public or private***

[Adapted from https://docs.github.com/en/repositories/creating-and-managing-repositories/about-repositories]

Commit - a snapshot



Snapshot of current state of your repository ... like taking a picture with metadata

- Who?
- What?
- Why? -> Commit message!
- When?

My own GitHub repository

... continue work on GitHub

1. Work on it, make updates on GitHub, ...
2. Commit when done (per file): take snapshots of units of work (one)

Quick demo (please just watch):

All materials for this session are on GitHub too!

https://github.com/samumantha/github-jupyter-4-ds

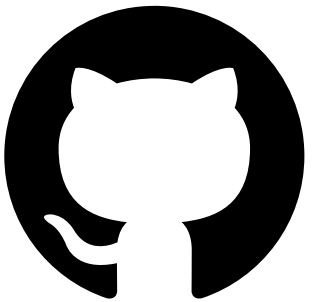
Finding a typo in my own repository...

Detour: Markdown

Machine and human readable plain text format. Useful for both GitHub and Jupyter.

```
# This is a section in Markdown

## This is a subsection

Nothing special needed for
a normal paragraph.

**Bold** and *emphasized*.

A list:
- this is an item
- another item

![](images/img/slides/CSC.png)


[This is a link to CSC](https://research.csc.fi/)
```

(Optional) Clone - download



...get the latest (working) version on your computer

(Optional) My own GitHub repository

... continue work locally

1. Clone: get a copy to my computer
2. Work on it, make updates, ...
3. Add, Commit: take snapshots of units of work (one or many)
4. Push: submit snapshots to GitHub

Pull: Get latest version from GitHub

[from https://raw.githubusercontent.com/hendrixroa/in-case-of-fire-1/master/in_case_of_fire.png]

Demo: Starting new

Just watch for now.

https://github.com/

[See also: https://samumantha.github.io/github-jupyter-4-ds/creating-repo-using-web/]

Create a new repository

- Namespace
- Name
- Description
- README
- LICENSE
- `.gitignore`

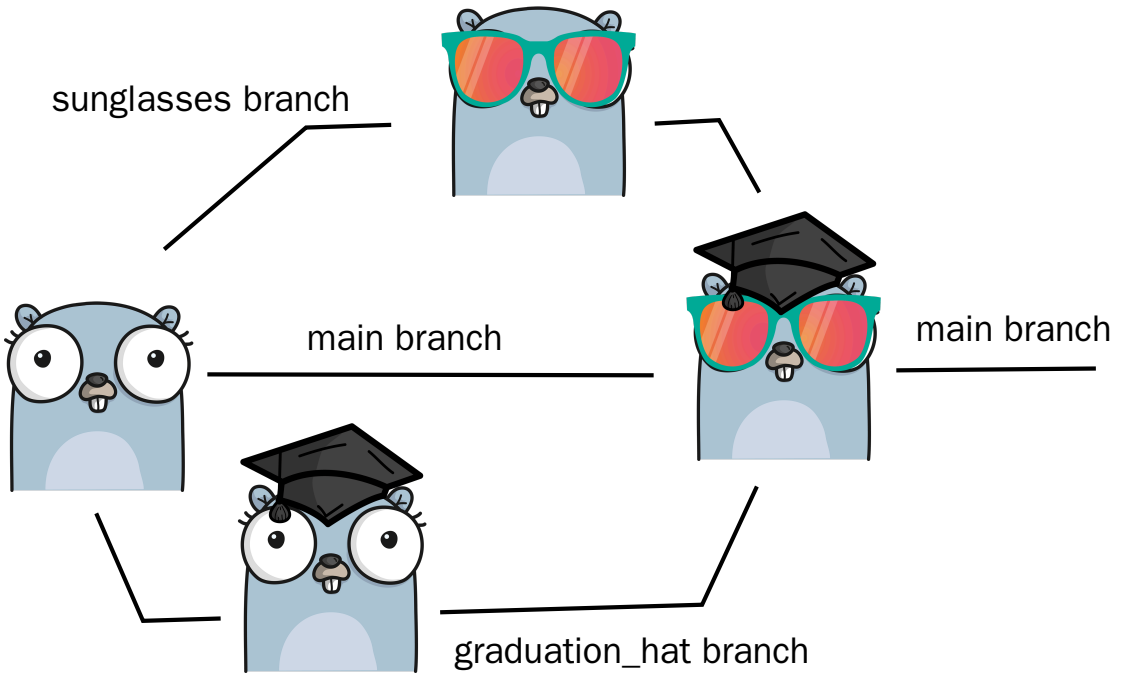
Add new file

- Edit
- (Special: license/citation.cff)
- Commit
- `main`

History

Annotate

Branches and merge

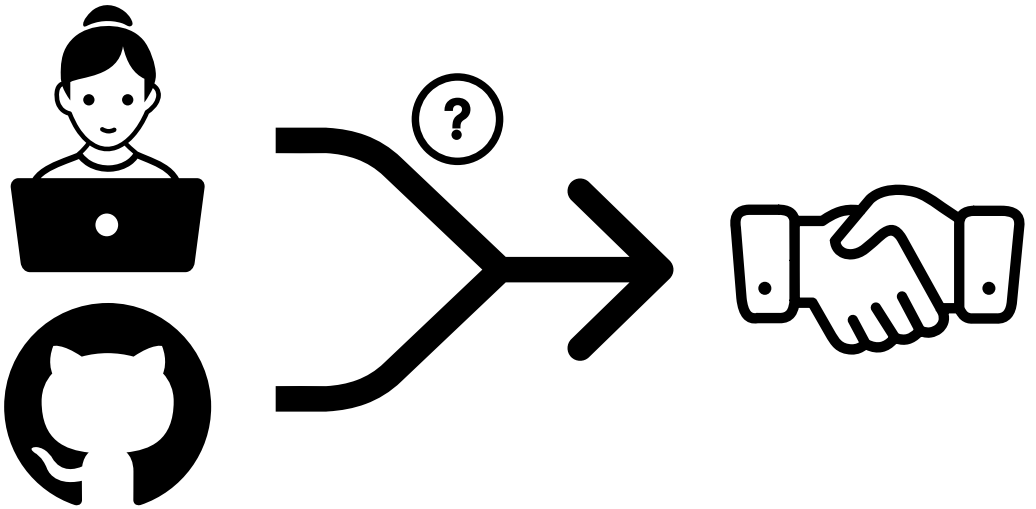


[

Image created using https://gopherize.me/ (inspiration)

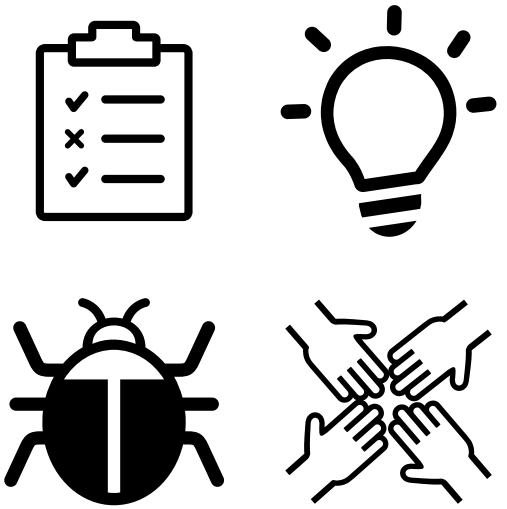
]

GitHub pull request



Making a contribution: A request to merge

GitHub issues



Inform, ask and collaborate

GitHub fork

github.com/**myusername**/myrepo

➡ github.com/**yourusername**/myrepo

- Propose changes
- Use someone elses work as starting point

Useful when you cannot edit directly

Making a suggestion

Full workflow **GitHub**:

1. Suggest idea: issue
2. Discussion -> OK
3. Separate your work: branch / fork
4. Work: work - commit (one or more)
5. Suggest work: pull request
6. Accept: merge

➡ You made it to history!

(Optional) Making a suggestion

Full workflow **local**:

1. Suggest idea: issue
2. Discussion -> OK
3. Get the work: (fork) - clone - pull
4. Work: work - add - commit (one or more)
5. Put it on GitHub: push
6. Suggest work: pull request
7. Accept: merge

➡ You made it to history!

(Optional) Demo - exploring an existing repo

Just watch :)

- History
- Branches
- Forks
- Issues
- Pull requests

➡ https://github.com/the-turing-way/the-turing-way/

Demo - contribute

You may suggest your own recipes!

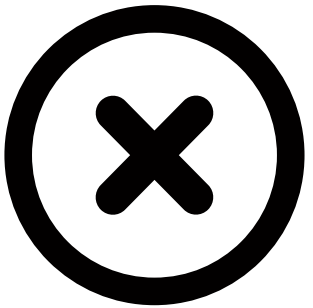
- History
- Issue
- Fork / Branch
- Work
- Pull request

New file vs changing file

➡ https://github.com/samumantha/data_support_recipe_book

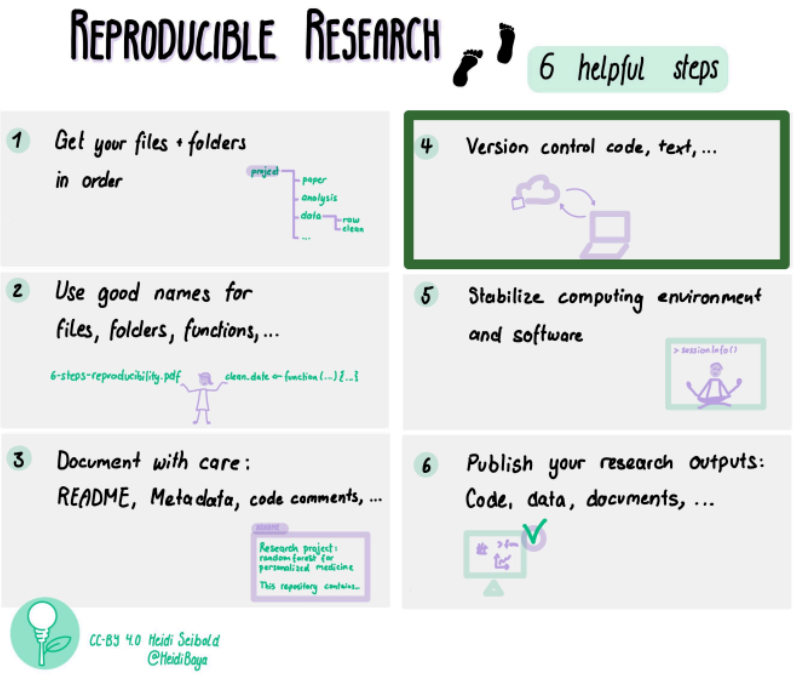
What to track using Git(Hub)?

- Software
- Scripts
- Documents
- Manuscripts
- Configuration files
- Website sources
- Data (better options available!)



- Secrets
- Passwords
- Binaries; files that are difficult to diff
- Files generated from builds

Is sharing your work on GitHub making it reproducible?



➡ Only in combination with other steps! **Computing environment**: Conda, Pip, Poetry, ...; **Persistent identifier**: Zenodo, ...

[Barker, M., Chue Hong, N.P., Katz, D.S. et al. Introducing the FAIR Principles for research software. Sci Data 9, 622 (2022). https://doi.org/10.1038/s41597-022-01710-x]

Motivation to use Git(Hub)

“It broke... hopefully I have a working version somewhere?”

"Where is the latest version, and which one should I trust?"

"I am sure it used to work. What changed, when, and why?"

"When did this problem appear?"

“Something looks different - what was updated, and who accepted it?”

Summary - GitHub

Track versions and collaborate and share with others and yourself

Summary - words

Repository

Commit

Fork

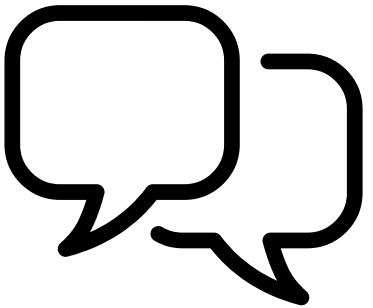
Branch

Issue

Pull request

Chatter

What usecases for GitHub can you see in your work?

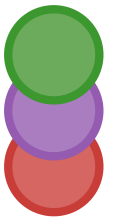


Type your answer in chat, but wait to send

Break

A tool for people who write code in Python, R or Julia

Executable notebooks

Pluto.jl 

...

Researcher perspective

Exploration

code, notes/explanation, visualization

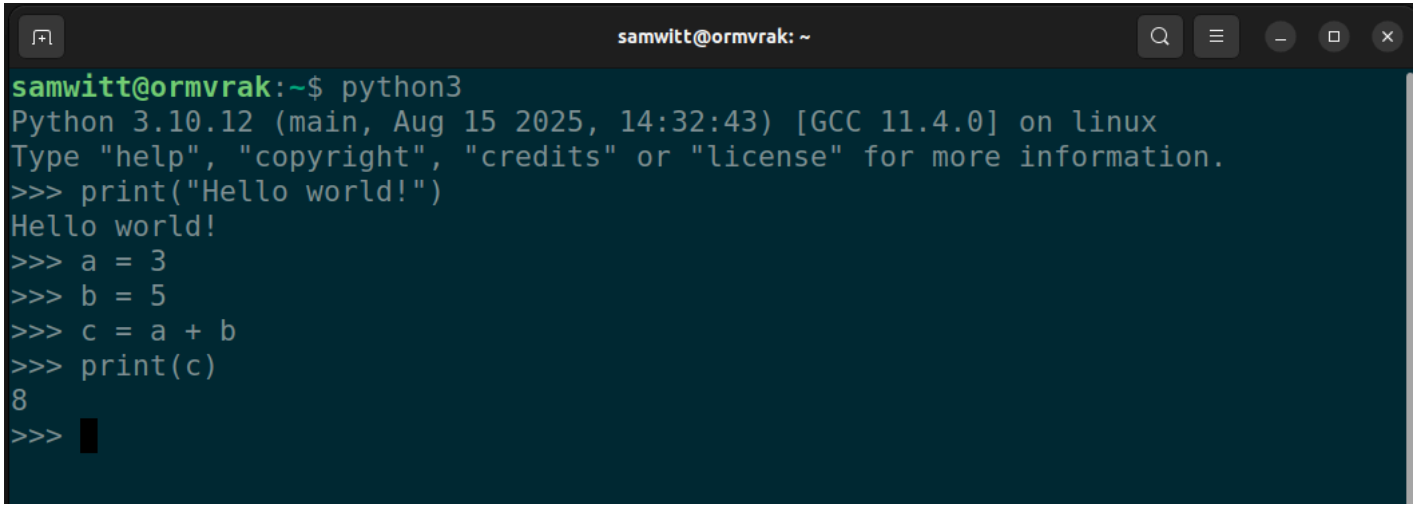
Publish a paper

Sharing narrative

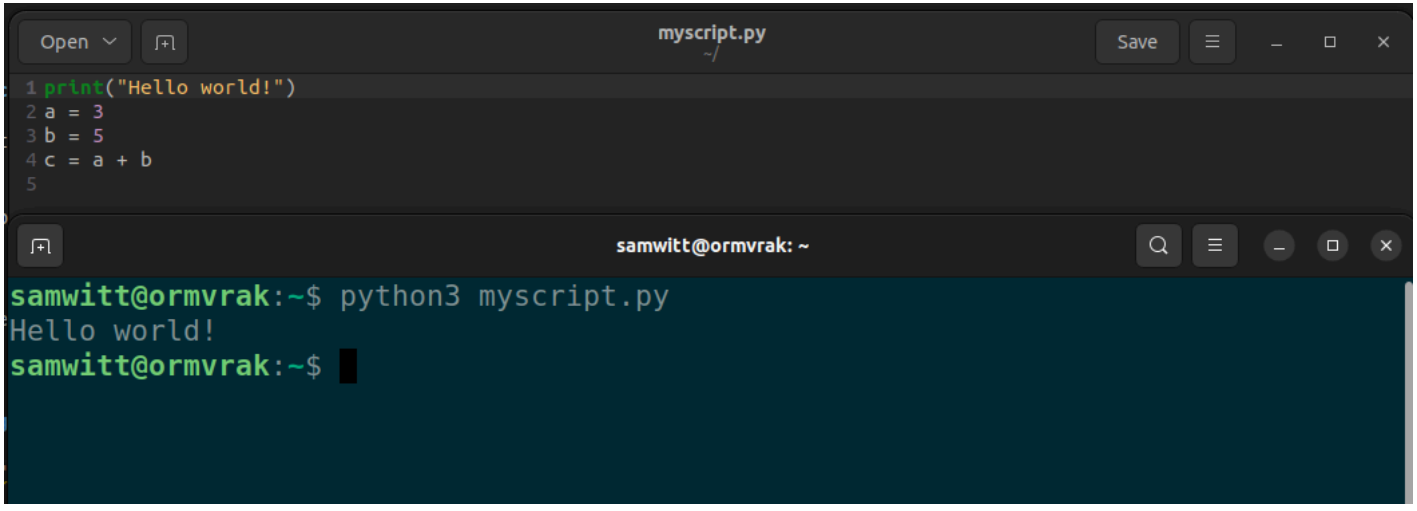
Share to test and adapt

executable for others

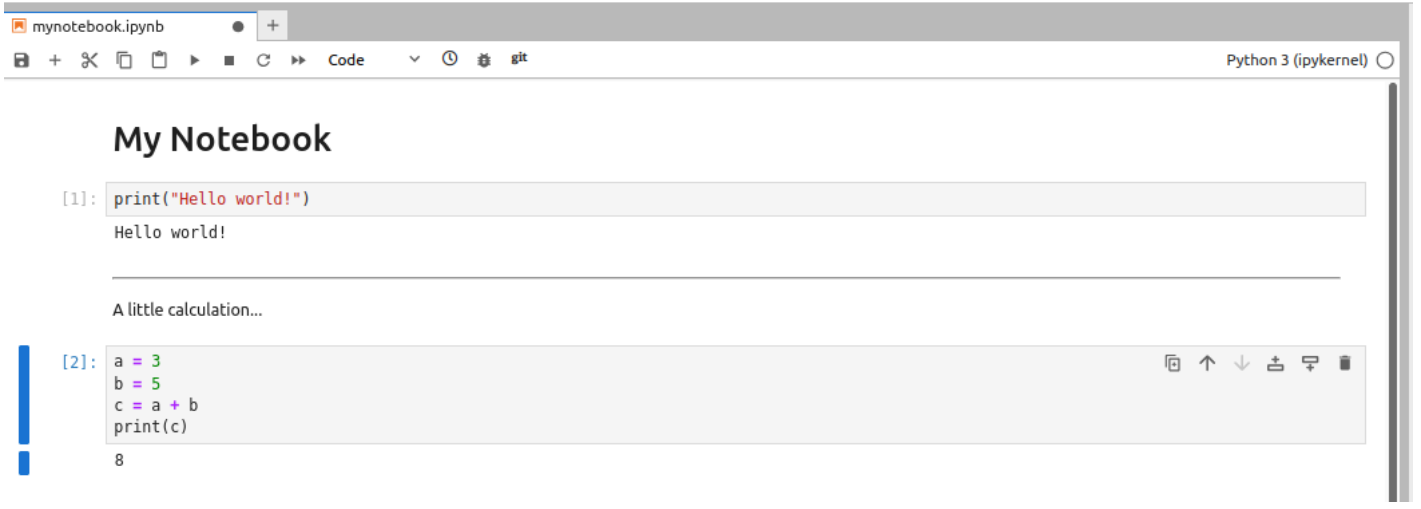
Coding in the terminal



Script: A step by step recipe



Notebooks: Interactive exploration



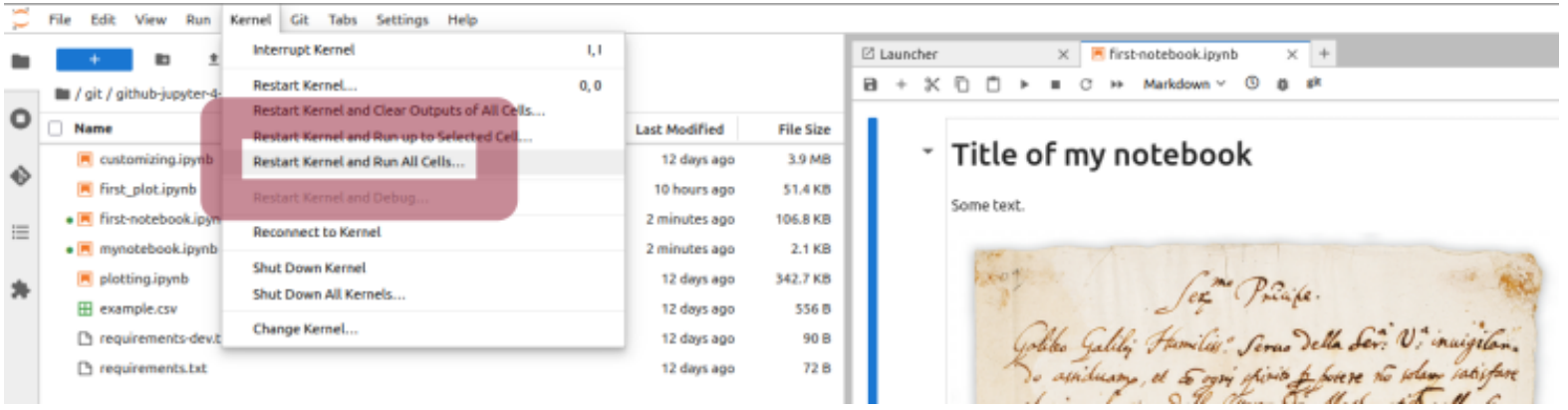
Demo usecase: Protopyping / Exploration

You may follow along: https://noppe.csc.fi/

- Create notebook - naming
- Create cells - code / markdown
- Open an existing notebook (`first_plot.ipynb`)
- Execute cells
- Restart and run all

Good practice

Kernel > Restart kernel and run all cells



Usecase: Teaching

- Prefilled
- Exercises as rendered text
- Automatic checks possible

➡ https://github.com/csc-training/python-introduction/blob/gh-pages/notebooks/examples/1%20-%20Introduction.ipynb ➡ Also available in our Noppe workspace (introduction_to_python_csc.ipynb).

➡ https://github.com/csc-training/PythonGIS_CSC/blob/master/Raster/Seurasaari_trees.ipynb
➡ Available in Noppe application "Introduction to geospatial Python".

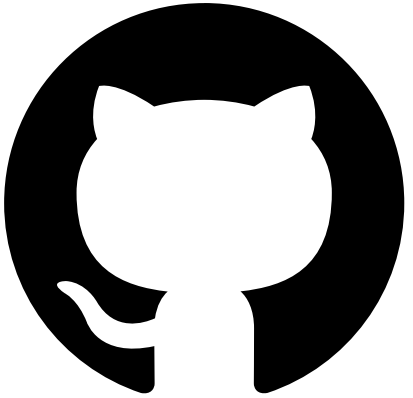
Usecase: Sharing

Tutorial / Walkthrough ➡ Let others explore

➡ https://documentation.dataspace.copernicus.eu/APIs/openEO/openeo-community-examples/python/ParcelDelineation/Parcel%20delineation.html

➡ https://github.com/eu-cdse/notebook-samples/blob/main/geo/stac_ndvi.ipynb

Sharing



GitHub, Websites: **Share to view**

Binder, Noppe, Google Colab: **Share to execute and change in the cloud**

Did you know?

Noppe: CSC service for running **Jupyter or RStudio** in the cloud, free of charge for researchers in Finland

➡ **Readymade applications** (e.g. introduction to Python)

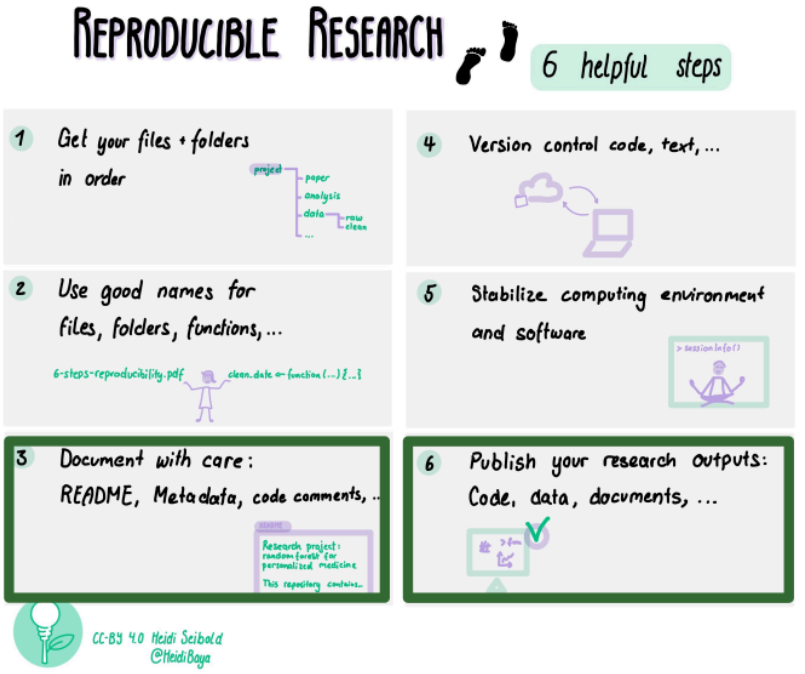
➡ **Teaching** in custom environment, same for all students

➡ **Collaboration**

Brought to you by ministry of education and culture!

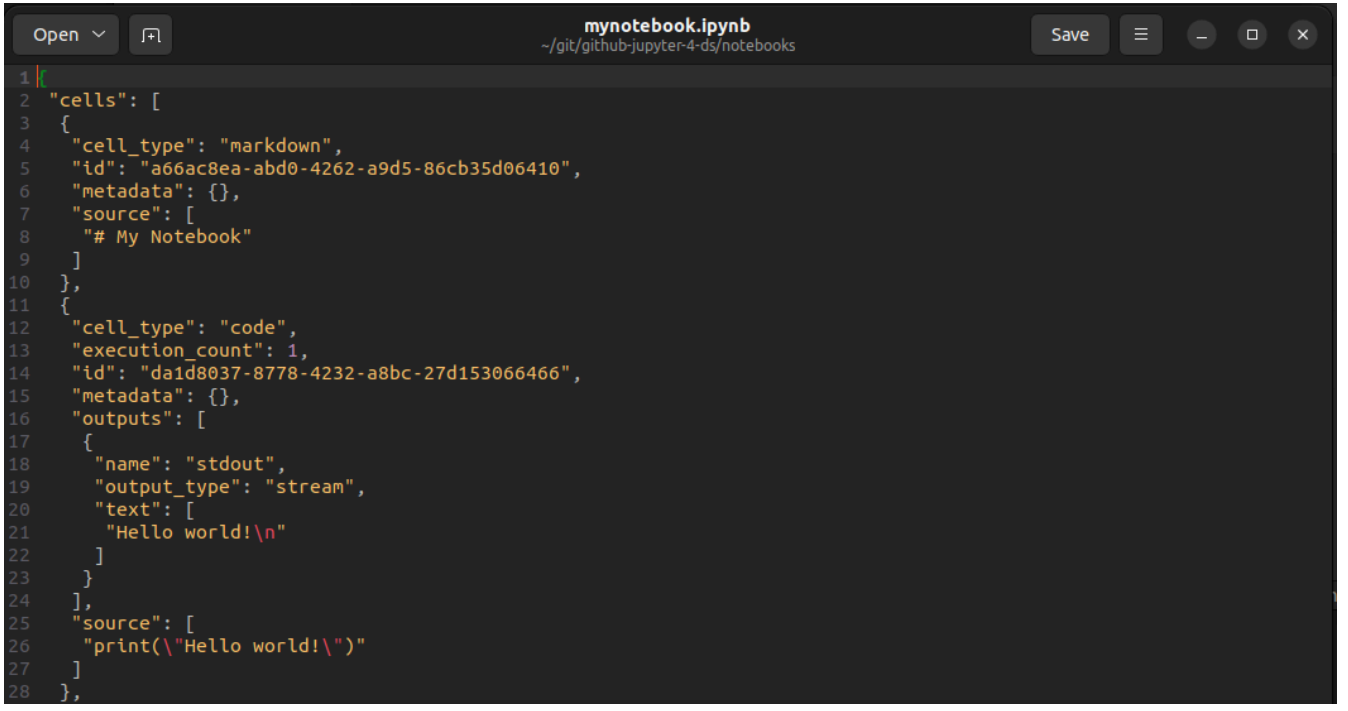
Support available via **`servicedesk@csc.fi`** or in our weekly user support sessions.

Jupyter and reproducibility



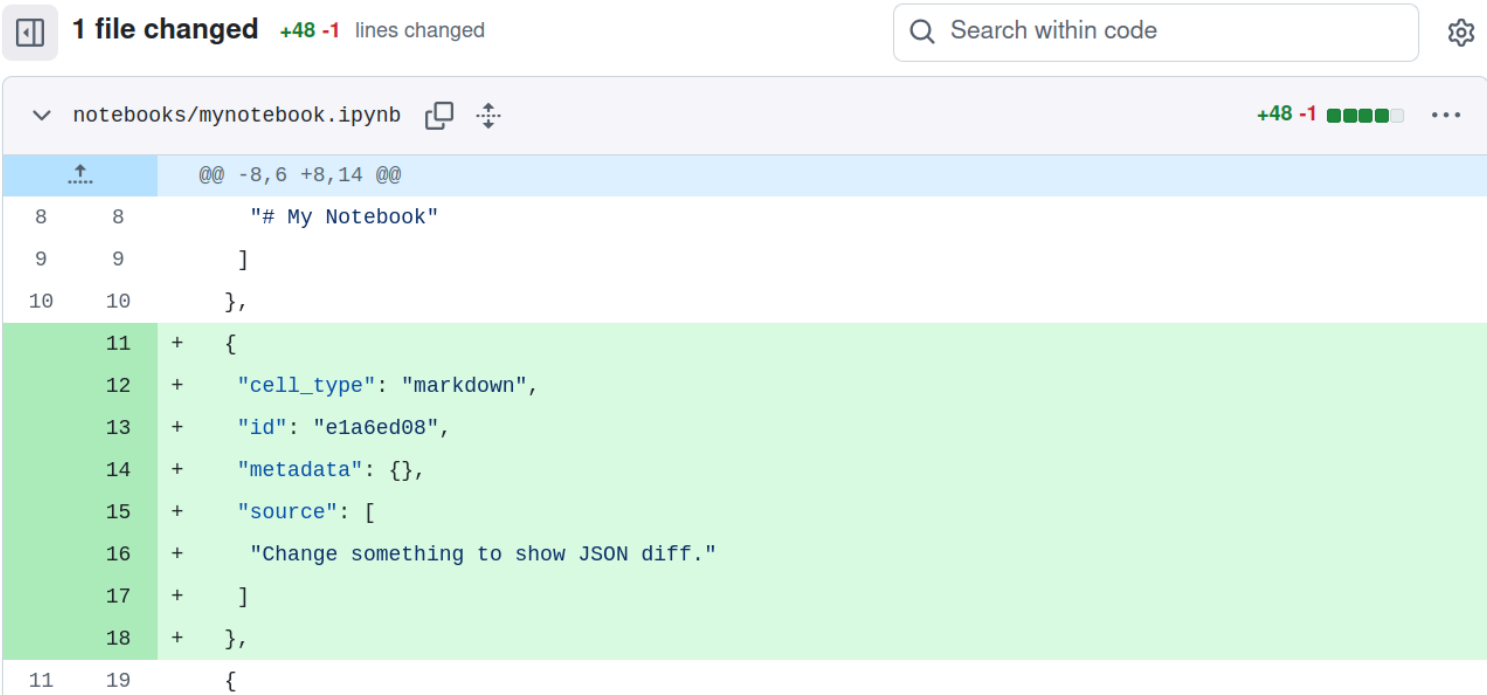
Supports code modularity + documentation and is a good format to share. Needs also info on computing environment.

Under the hood



IPYNB ➡ JSON

Tracking changes in ipynb files



Version control possible, but limited benefits. Solutions exist, e.g. `nbdime`

Moving away from Jupyter?

A Jupyter notebook ...

- is super useful in protoyping.
- can even be the endpoint.
- can be used in high performance computing environments.

You may want to switch to scripts when ...

- building a (command line/graphical) tool.
- you need to run it with multiple datasets/parameters.
- efficiency is the goal.

Jupyter ecosystem

Notebook : Code + markdown cells ➡ `.ipynb`

Lab : Interface to view `.ipynb` files, layout, extensions

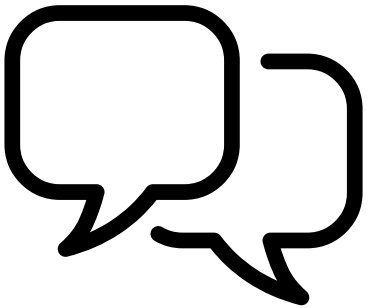
Hub : Jupyter for servers, multiple users

Summary

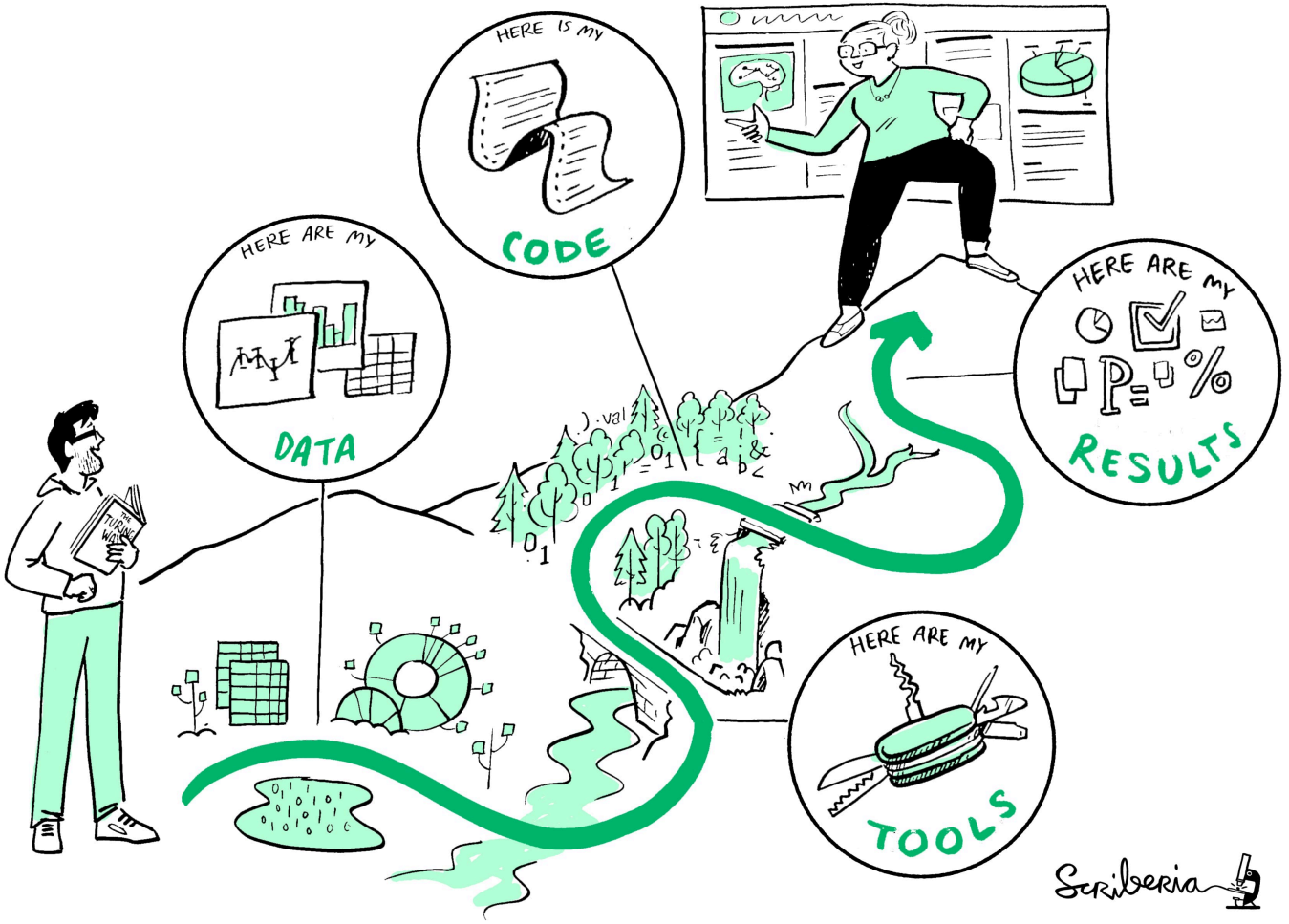
Jupyter is a helpful tool in the beginning and end of the research process, and can also be used throughout.

Chatter

How might Jupyter benefit your work in the future?



Type your answer in chat, but wait to send



Where to go from here ...

Play around with **GitHub**

- Contribute to our recipe book
- Create your own repo
- Try things out with colleagues

Play around with **Jupyter**

- Noppe workspace available for a while
- Check out other Noppe applications
- Install on your own computer (advanced!)

Learn more...

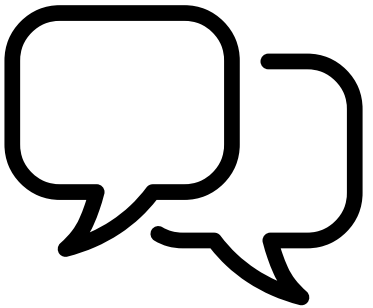
Tools and techniques for researchers who code...

3 half days of **Git** (-Hub, VSCode, command line) + 3 half days of **reproducible research** (computing environments and workflows), **documentation**, **social coding** (sharing and licensing), **modular code development**, **jupyter** (widgets and other tricks) and **automated testing**

- Materials
- Recordings
- Next workshop in March '26: Sign up for newsletter
- **Bring your own classroom**, contact `support@coderefinery.org`

Chatter

The most exciting thing you learned today?



Type your answer in chat, but wait to send

Acknowledgements

Reuse and inspiration was drawn from CodeRefinery and Skills4EOSC.

CodeRefinery lessons:

- Introduction to and collaborative git
- Git without the command line
- Jupyter
- Programming for Data Stewards

Logos are belong to the companies they represent

Icons used are from UXWing

CSC slide by Josefine Nordling from part 1 of this training.