



Enhancing Data Support: Practical Reproducibility

Samantha Wittke

CSC - IT Center for Science

Outline

Intro and practicalities

GitHub

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

Break

Jupyter

- Computational notebooks
- Basic features

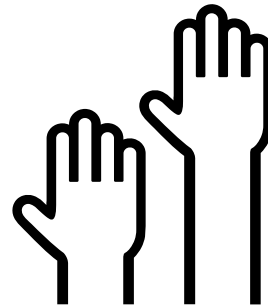
Where to go from here



[The Turing Way project illustration by Scriberia. Used under a CC-BY 4.0 licence. DOI: <https://zenodo.org/records/13882307>]

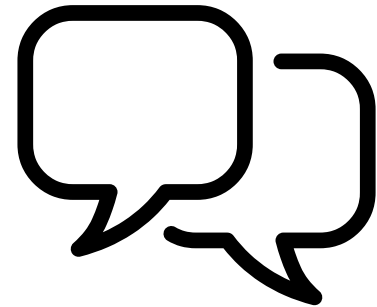
Questions

Ask at any time -> Zoom chat or
raise hand



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!



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

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

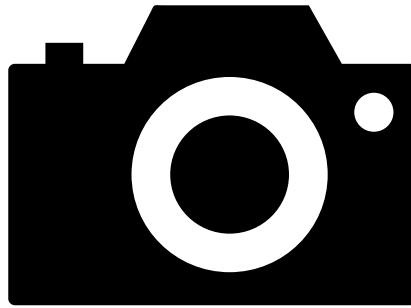
What reproducibility means in practice

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

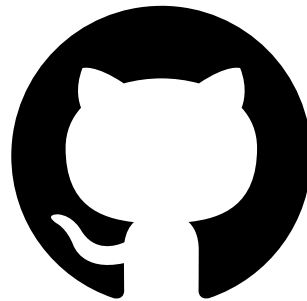
*Code that works and is shared is not the same as
reproducible code*

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.

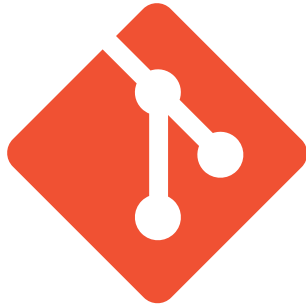
GitHub



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

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

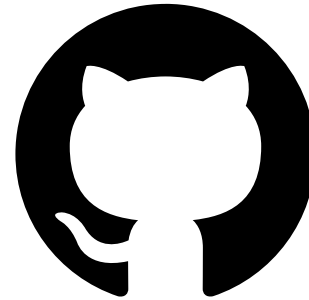
Git



Tool/format for version control

Others: Subversion, Mercurial, ...

GitHub



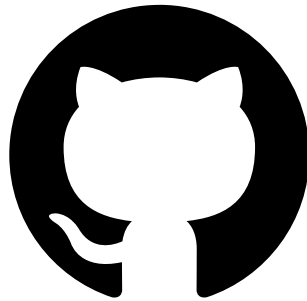
Hosting service for Git repositories with web interface -> Share and collaborate

Others: GitLab, Codeberg, ...

Repositories - a place to store

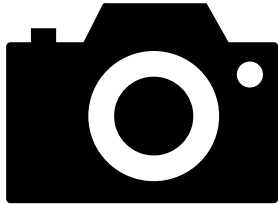
*A repository is the most basic element of GitHub. It's a place where you can **store your code**, your files, and each file's **revision history**. Repositories can be **owned by persons or organisations**, have **multiple collaborators** and can be either **public or private***

Clone - download



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

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 locally

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

Pull: Get latest version from GitHub

... continue work on GitHub

1. Work on it, make updates, ...
2. Commit: take snapshots of units of work (one)

In case of fire



 1. `git commit`

 2. `git push`

 3. `leave building`

Demo: Starting new

<https://github.com/>

Create a new repository

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

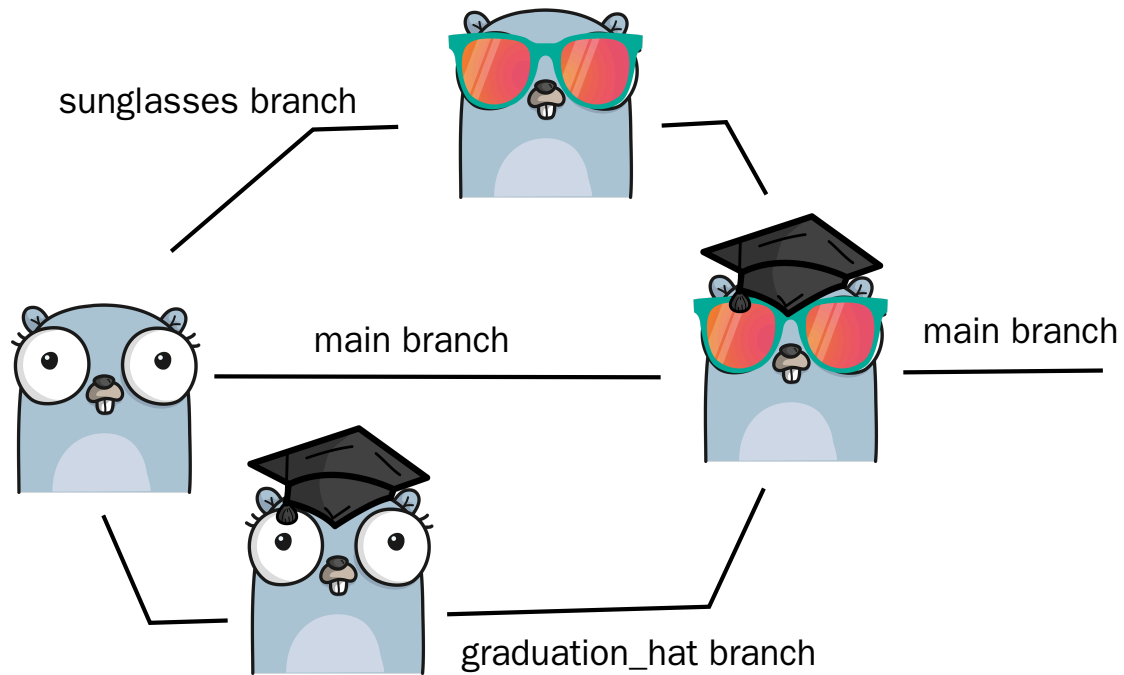
Add new file

- Edit
- Commit
- **main**

History

Annotate

Branches and merge

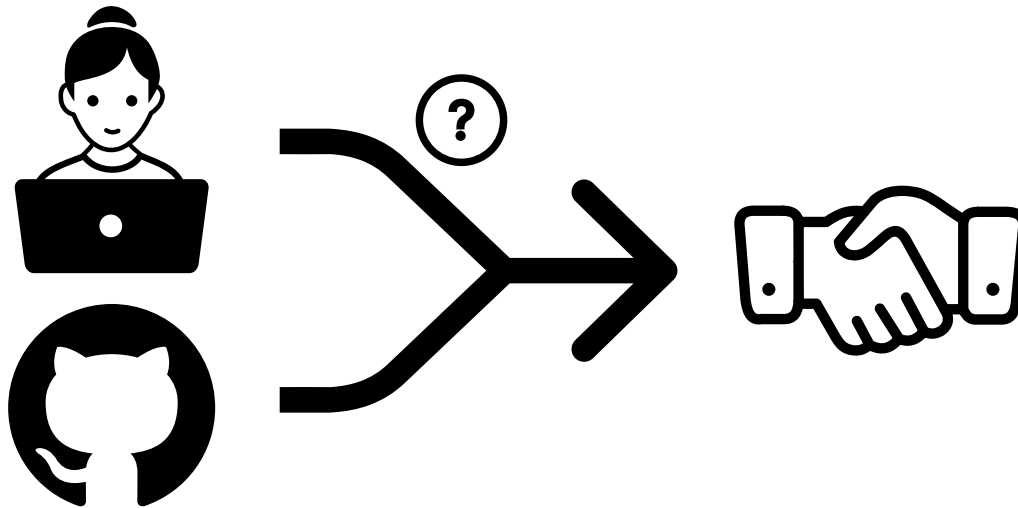


[

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

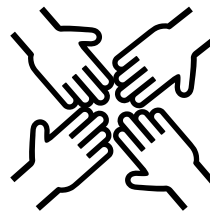
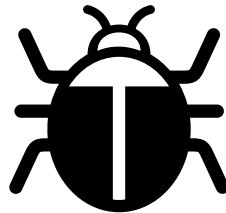
]

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 else's 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!

Making a suggestion

Full workflow **local**:

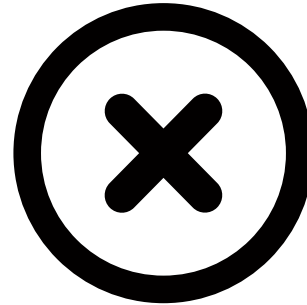
1. Suggest idea: issue
2. Discussion -> OK
3. Get the work: (fork) - clone - pull
4. Work: work - add - commit (one or more)
5. Put it on GitHub: push
6. Suggest work: pull request
7. Accept: merge

➡ You made it to history!

What to track using Git(Hub)?



- Software
- Scripts
- Documents
- Manuscripts
- Configuration files
- Website sources
- Data*



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

Demo - exploring an existing repo

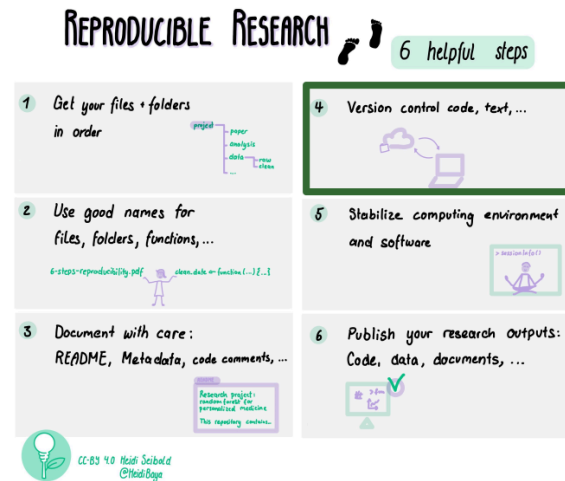
- History
- Branches
- Forks
- Issues
- Pull requests

Demo - contribute

- Issue
- Fork / Branch
- Work
- Pull request

New file vs changing file

GitHub and reproducibility



Is sharing your work on GitHub making it FAIR?

➡ Support yes, but **GitHub link is not persistent!** ➡ 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

Collaborate and share with others and yourself

Summary - words

Repository

Issue

Pull request

Clone

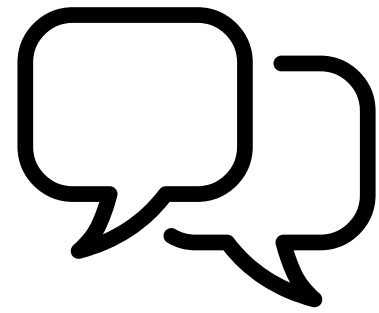
Commit

Fork

Branch

Chatter

Which of the concepts would need more explanation?



Break



A tool for people who write code

Executable notebooks



Pluto.jl 

...

Researcher perspective

Exploration

code, notes/explanation, visualization

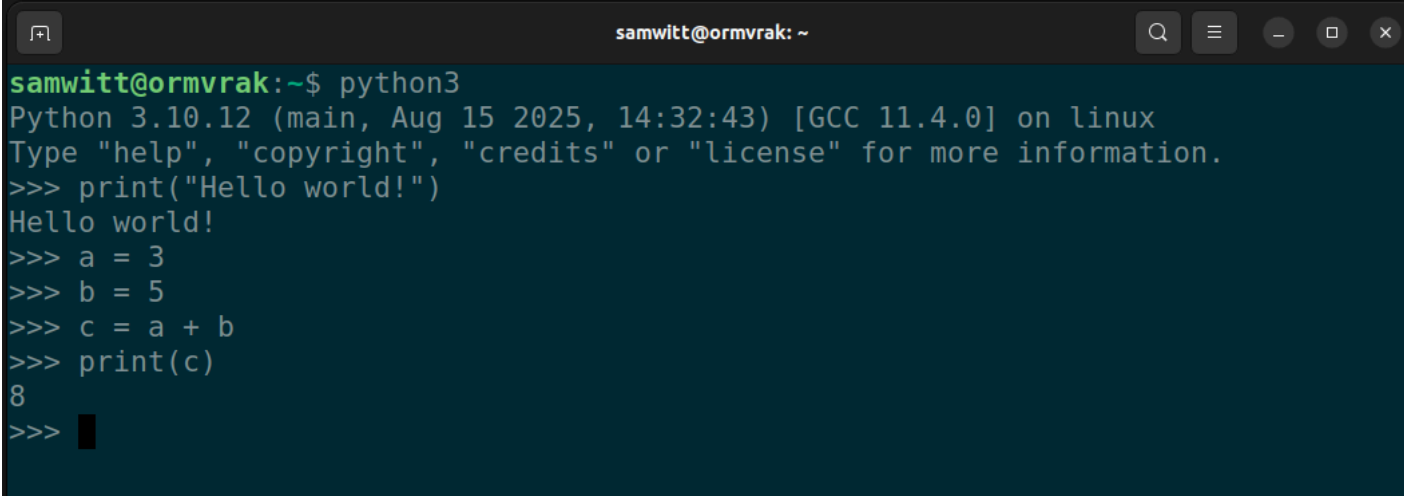
Publish a paper

Sharing narrative

Share to test and adapt

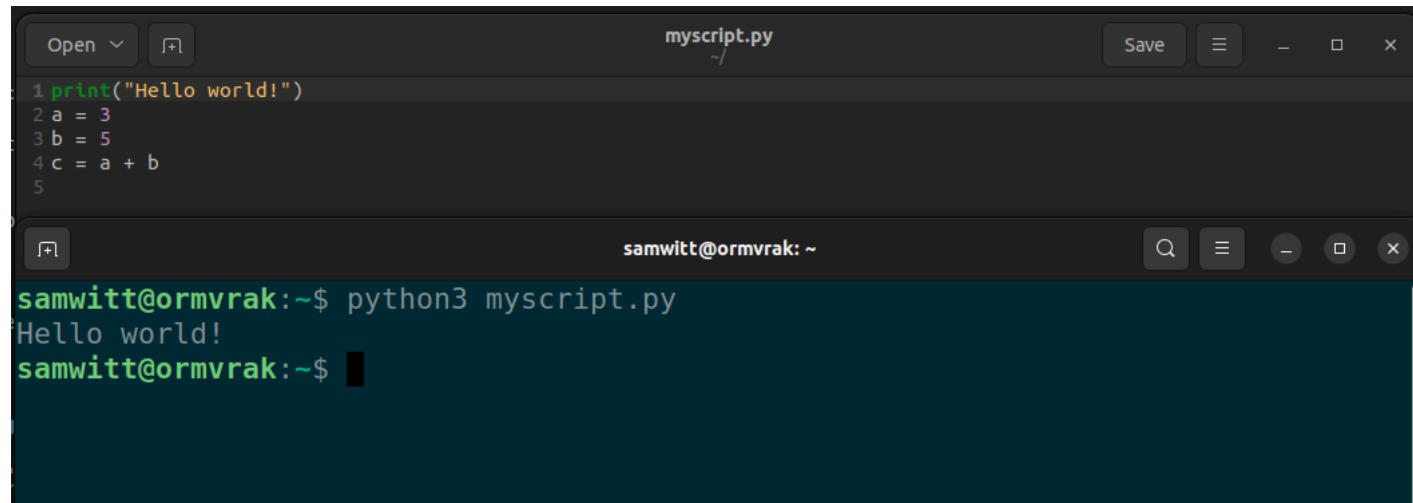
executable for others

Coding in the terminal

A terminal window with a dark background and light text. The window title bar shows 'samwitt@ormvrak: ~' and standard window controls. The terminal content shows a Python 3.10.12 prompt where the user runs 'python3'. The prompt shows the version and GCC information. The user then enters a series of Python commands: 'print("Hello world!")', 'a = 3', 'b = 5', 'c = a + b', and 'print(c)'. The output shows 'Hello world!' and '8'. The prompt is currently at '>>>' with a cursor.

```
samwitt@ormvrak:~$ python3
Python 3.10.12 (main, Aug 15 2025, 14:32:43) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello world!")
Hello world!
>>> a = 3
>>> b = 5
>>> c = a + b
>>> print(c)
8
>>> █
```

Script: summary

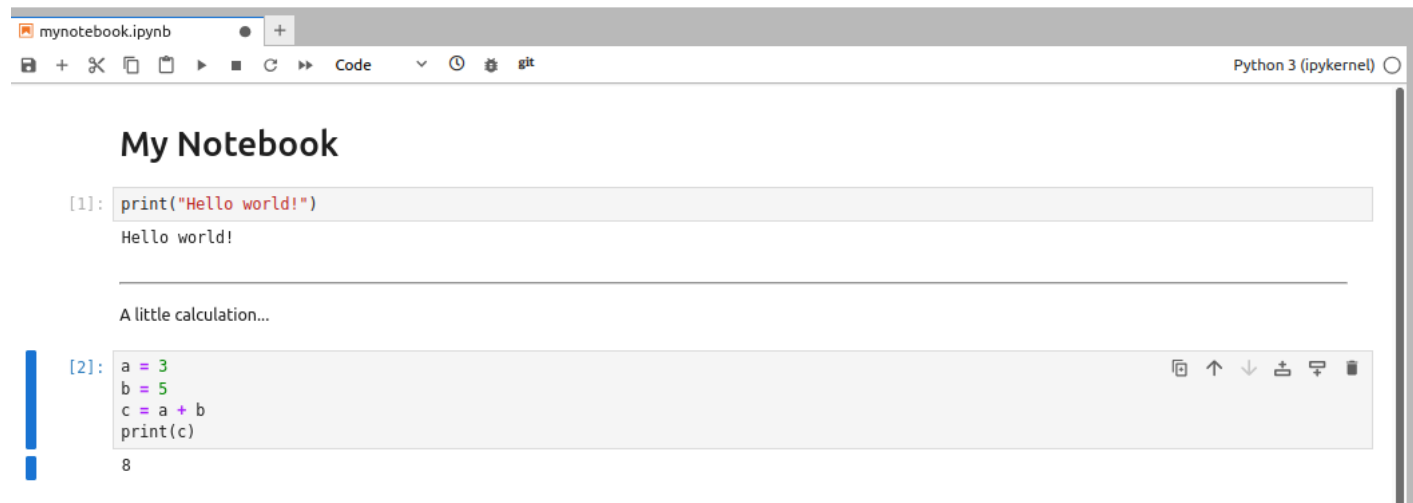


The image shows a screenshot of a code editor and a terminal window. The code editor at the top displays a Python script named `myscript.py` with the following content:

```
1 print("Hello world!")
2 a = 3
3 b = 5
4 c = a + b
5
```

Below the code editor is a terminal window with the prompt `samwitt@ormvrak: ~`. The terminal shows the command `python3 myscript.py` being executed, which outputs `Hello world!`. The prompt then returns to `samwitt@ormvrak:~$` with a cursor.

Notebooks: interactive



Demo usecase: Prototyping / Exploration

- Create notebook - naming
- Create cells - code / markdown
- Execute cells
- Restart and run all

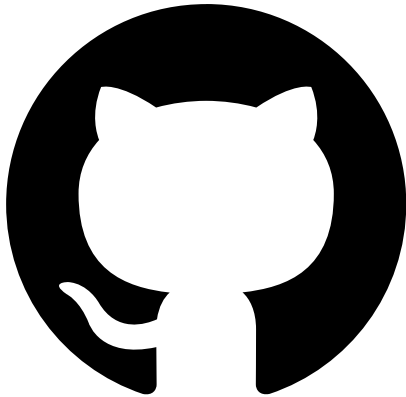
Demo usecase: Teaching

- Prefilled
- Exercises as rendered text
- Automatic checks

Demo usecase: Sharing

Tutorial / Walkthrough ➡ Let others explore

Sharing

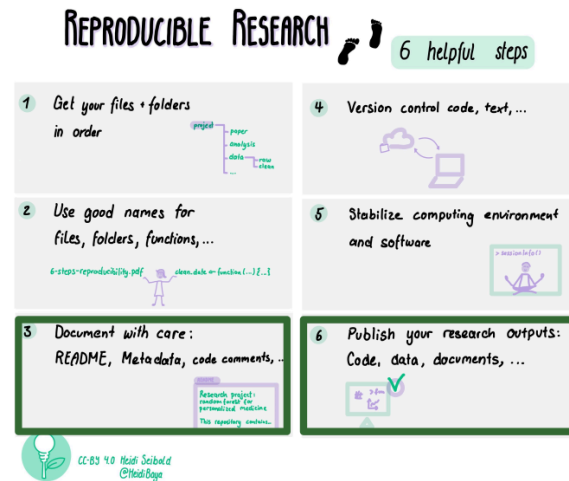


Share to view



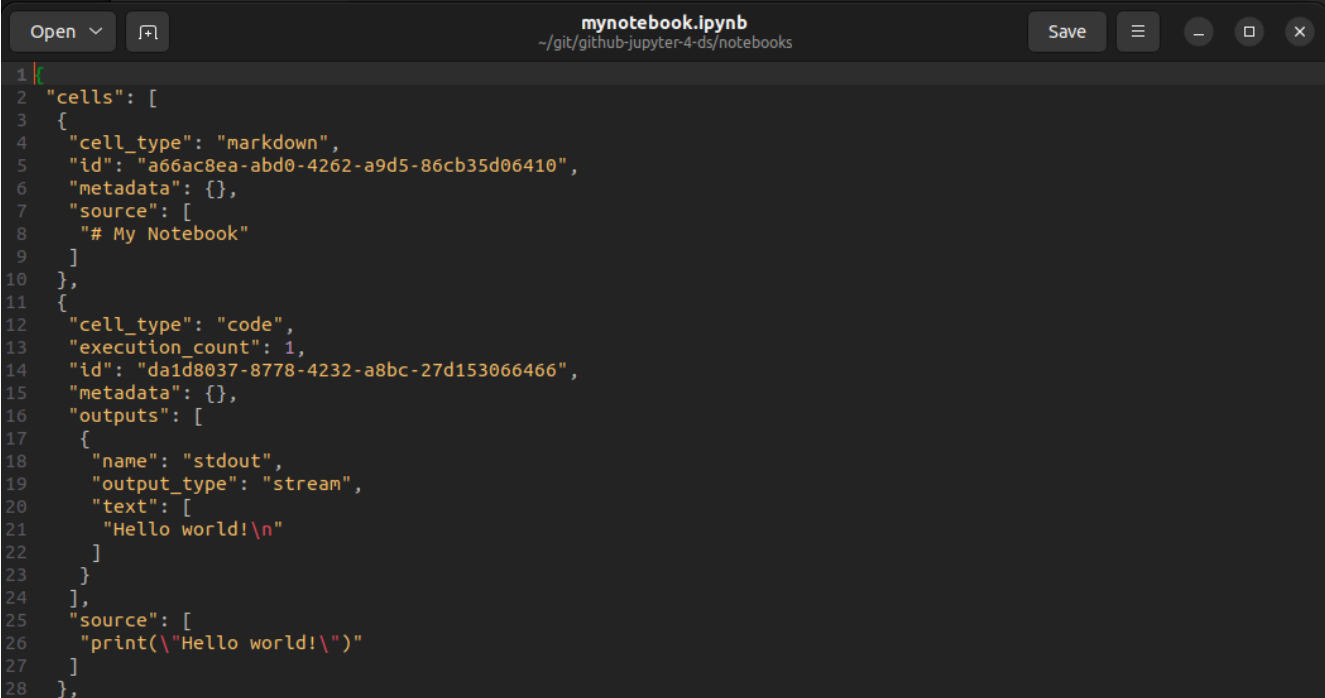
Share to execute and change in
the cloud

Jupyter and reproducibility



Jupyter supports code modularity + documentation and is a good format to share usage example

Under the hood



The screenshot shows a Jupyter Notebook window titled "mynotebook.ipynb" with the file path "~/git/github-jupyter-4-ds/notebooks". The window contains a single code cell. The JSON structure of the notebook is displayed in the editor, showing a list of cells. The first cell is a markdown cell with the text "# My Notebook". The second cell is a code cell that has been executed once, containing the code "print('Hello world!\\n')". The output of this cell is "Hello world!\\n".

```
1 |
2 | "cells": [
3 |   {
4 |     "cell_type": "markdown",
5 |     "id": "a66ac8ea-abd0-4262-a9d5-86cb35d06410",
6 |     "metadata": {},
7 |     "source": [
8 |       "# My Notebook"
9 |     ]
10 |   },
11 |   {
12 |     "cell_type": "code",
13 |     "execution_count": 1,
14 |     "id": "da1d8037-8778-4232-a8bc-27d153066466",
15 |     "metadata": {},
16 |     "outputs": [
17 |       {
18 |         "name": "stdout",
19 |         "output_type": "stream",
20 |         "text": [
21 |           "Hello world!\\n"
22 |         ]
23 |       }
24 |     ],
25 |     "source": [
26 |       "print('Hello world!\\n')"
27 |     ]
28 |   },
```

IPYNB ➡ JSON

Jupyter diff

```
11 19 {
8 8     "# My Notebook"
9 9     ]
10 10    },
11 +   {
12 +     "cell_type": "markdown",
13 +     "id": "e1a6ed08",
14 +     "metadata": {},
15 +     "source": [
16 +       "Change something to show JSON diff."
17 +     ]
18 +   },
11 19   {
```

Version control possible, but limited benefits

Moving away from Jupyter?

A Jupyter notebook ...

- is super useful in prototyping.
- 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.

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

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](#)

Acknowledgement

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