ENAC-IT4R Data & Code services

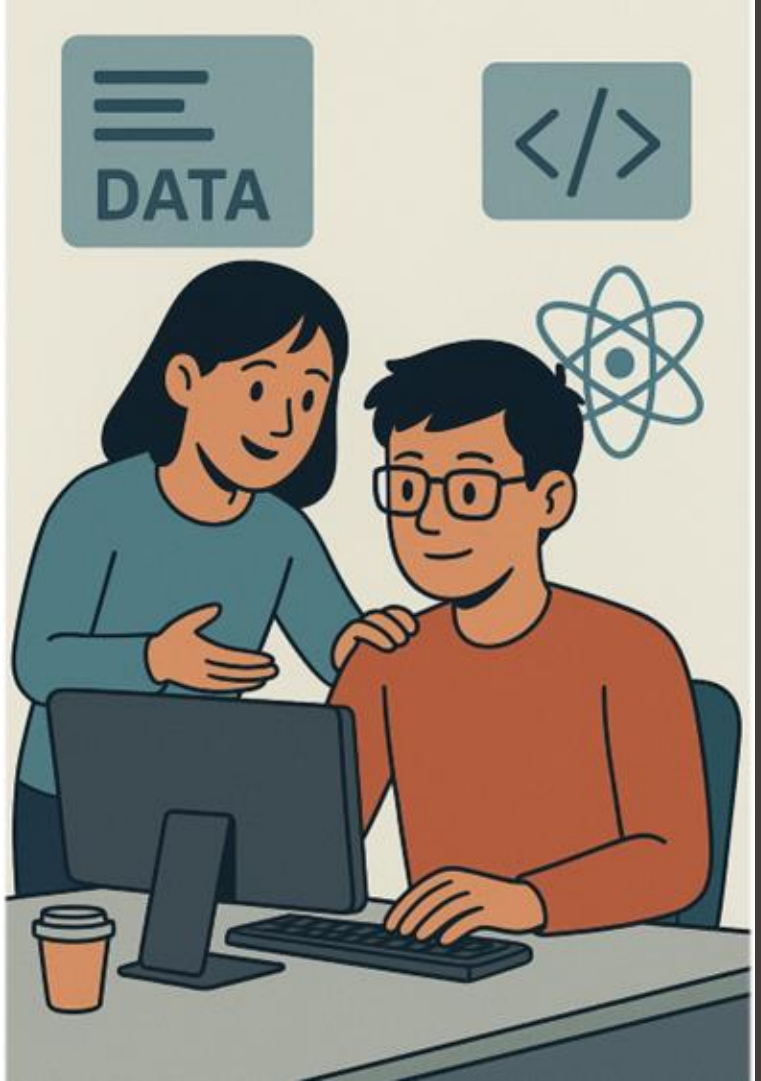
Tech talk x RSE Romandie

02.06.2026

École polytechnique fédérale de Lausanne

Tuesday Tech talks?
Tuesday Tech talks!

- Informal lunchtime gatherings (~~*bring your own lunch!*~~) designed to foster interdisciplinary knowledge sharing within the ENAC community (and beyond!) on **data and coding topics**, across academic domains and career levels.



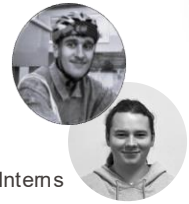
Agenda

- Welcome
- Literate programming
  - Pierre-Olivier Valès, CEDE, EPFL
  - Simon Dürr, HES-SO Valais-Wallis
- From docstrings to live docs
  - Hugo Solleder, ENAC-IT4R (10 min)
- Static site generation for project & code documentation
  - Alexander Lepper, LASUR (10 min)
- Open discussion

ENAC-IT4R Data & software engineers

Data & Code support services enac-it4r.epfl.ch

Open Research Data National Prize 2024



Interns

Ask us anything about your Data & Code !

enacit4research@epfl.ch

Supporting researchers with their data and code

Fostering Collaborative, FAIR, Open Research Data & Code for Scientific Valorization

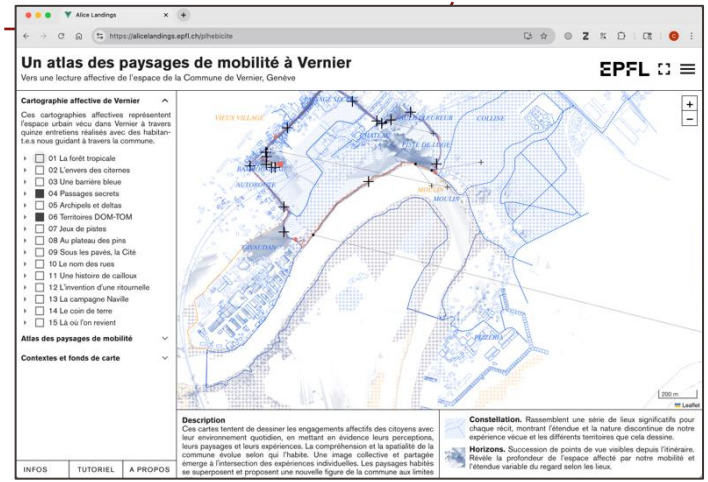
Research Data Lifecycle

Software dev & packaging

- Code packaging, refactoring, migration, deployment
- Collaborative repository mgt

Data valorization

- Web application development
- Database set-up
- Data pipelines automation



Pilot projects

- Open-source software or research scripts for packaging and distribution
- Dataset valorization through tailored visualization

Literate programming & computing

Literate programming
What is it

- Introduced 1984 by Donal Knuth
- Code should be written for humans, not machines

Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do – Literate Programming (1984)

- Originally TeX as document formatting language, PASCAL as programming language
- Two main operations:
  - Tangle: produces compilable source command
  - Weave: generates documentation files from the literate file

Literate programming

R - Sweave, knitr, Quarto

- *The traditional way of writing a report as part of a statistical data analysis project uses two separate steps: running the analysis using some software, and then copy and pasting the results into a word processing tool* - Leisch (2002)
- Literate programming as a way to avoid copy-pasting
- Vignettes as long-form guide to packages, framed around a problem that the package is designed to solve.

Literate computing

IPython, Jupyter

A literate computing environment is one that allows users not only to execute commands interactively but also to store in a literate document format the results of these commands along with figures and free-form text that can include formatted mathematical expressions.

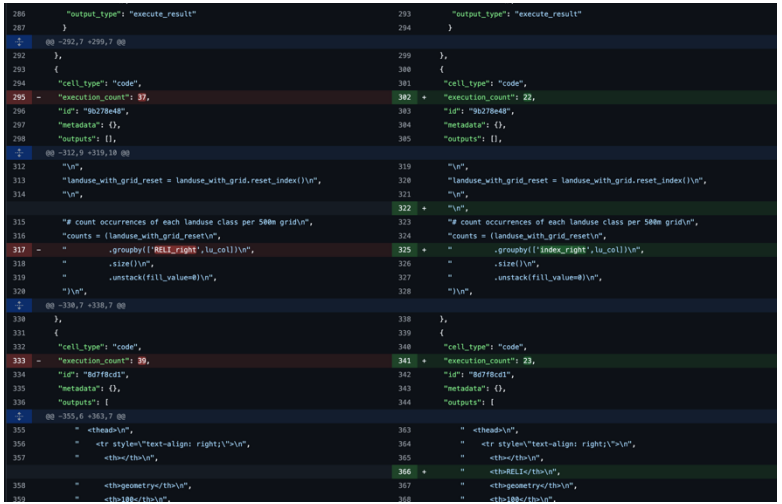
- Interactive computing across multiple programming languages
  - **Ju**lia, **Pyt**hon, **R**
- Focus on **computing** rather than **writing**
- Execute commands interactively and store the results, figures and free-form text in literate document

Literate computing in practice

- (G)noto at EPFL
- Quarto at HES-SO Valais Wallis

Limitations

- Execution order:
  - Can execute cells out of order, making the analysis not reproducible
  - Safest approach: restart & run
- Version-control
  - JSON behind the scenes
  - Very verbose diffs
  - Can use external tools (nbstripout, nbdime, …)
- Dependency issues



Alternatives

- Marimo – Python
- Pluto.jl – Julia

- Perks
  - Reactive notebooks: input change ➔ dependent cells re-run automatically
  - Version-control friendly: pure Python/Julia!
  - Built-in dependency management
  - Can deploy as app

- DEMO TIME

Documentation generation

Documentation

- In addition to structured code and clear variable names, adding documentation to functions/classes/methods helps other understand (how to use) your code
  - (Future you will also be grateful)

- Each language has its own approach/styles
  - C++: Javadoc/QT
  - Python: Google, Numpydoc, Javadoc-like

Source-based documentation

Two approaches

- Docstrings
  - String literal, part of the the runtime object and can be inspected programmatically
  - Julia, Python,…
- Documentation comments
  - Serves a similar purpose
  - Processed by external documentation tools ➔ comment extraction systems
  - R, C/C++, …

```
julia> """
       Return a greeting for the given name.
       """
       function greet(name)
           return "Hello, $name"
       end
greet

julia> Docs.doc(greet)
  Return a greeting for the given name.

help?> greet
search: greet get reset parent Base.rest repeat

  Return a greeting for the given name.
```

Demo-time

Beyond code documentation

MkDocs

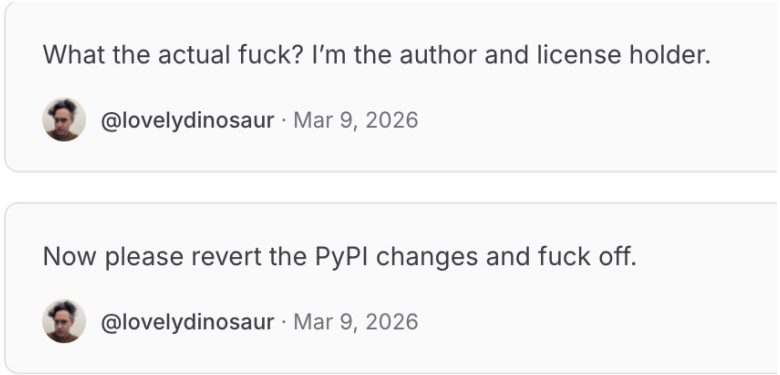
- MkDocs at the Urban Sociology Laboratory (LASUR)

MkDocs

Open source horror story?

- Original maintainer AWOL, then works on v2.0, completely breaking plugin compatibility
- Internal infighting between *de facto* maintainers

- Community split
  - ProperDocs – drop-in replacement
  - Zensical –
  - MkDocs 2.0 - ?



https://fpgmaas.com/blog/collapse-of-mkdocs/