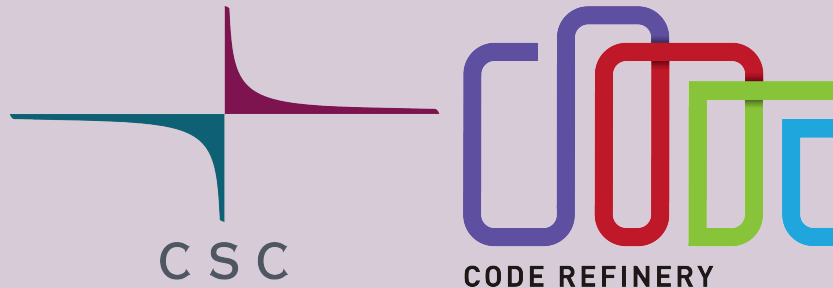


Research Software Engineering for you

Samantha Wittke (CSC - IT Center for Science)



Text: CC-BY 4.0 (Big thanks to previous versions by Radovan Bast and the CodeRefinery team)

Link to these slides: <https://doi.org/10.5281/zenodo.16891176>

Part 1/6 - Welcome - 15 min

- About us
- Why are we talking about this?
- CodeRefinery
- Topic overview
- Example project

About us

Johan you already know :)

Samantha:

- Geocology and Geoinformatics background
- Researcher who writes code -> Research Software Engineer
- Training, outreach and collaboration coordination
- 2025 fellow of the [Software Sustainability Institute](#) - community building around research software in Nordics (see also [Nordic-RSE](#))
- Leading the [CodeRefinery.project](#).

What is Research Software Engineering?

Here it is about all the essential tools and techniques so everyone can make full use of software, computing, and data with focus on reusability, reproducibility, and openness.

-> Everything around programming, that helps you be more efficient in your work and supports the reusability of your code.

Why are we talking about Research Software Engineering?

- RSE is part of computational research, but not often taught
- Useful when working alone, working with others, sharing your work
- "You don't know what you don't know"
- Hands-on experience -> integrate into your own workflow

You don't need to be a "proper software engineer/developer" to produce research software

We consider **any code, script, notebook, or file, regardless of size**, as "research software" if it is needed to generate, visualize, or reproduce data/results as part of a publication.

Do you write research software?

CodeRefinery workshop

Typical format: 3 half-days + 6 topical sessions, lessons + exercises [twice per year](#), online, free, live-streamed, recorded, archived asynchronous Q&A in collaborative document

- Version control
- Collaboration using Git
- Testing
- Documentation
- Notebooks
- Modular code development
- Reproducible research
- Software licensing
- How to share and publish code
- How to organize a code project
- ...

Next workshop September 9-11 and 6 following Wednesdays, register here:

<https://coderefinery.github.io/2025-09-09-workshop/>

Lesson materials:

<https://coderefinery.org/lessons/>

Topics of today

Lectures and exercises:

- [Version control](#) + Exercises
- [Reproducibility](#)
- [Documentation](#) + Exercises
- [Sharing and reusing](#) + Exercises

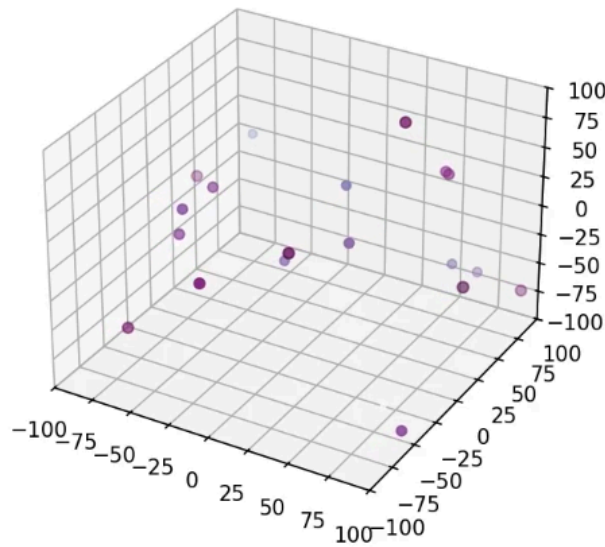
Longer version of all topics including more exercises can be found in [CodeRefinery lesson materials](#).

Exercises - hands on

- 3 x 20 min sessions
- There are likely more exercises than time, don't stress to do them all
- Please start with the ones that are most relevant to you and your work
- Some tasks provide a possibility to work with your neighbors
- Please ask if something is unclear or you need help (sticky notes)

Link to exercises: <https://samumantha.github.io/research-software-engineering/>

Example project: Simulating the motion of a number of planets



- N-body simulation
- Written in Python
- No need to understand the code in any detail
- Some exercises link to it

Link to example project:

<https://github.com/coderefinery/planets>

Part 2/6 Version Control - 20 min

- Find more info
- Motivation
- Commits
- Branching
- Talking about code
- Reproducibility
- Collaboration
- Code Review

Version control with git



Inspiration and where to find more:

- [Introduction to version control with Git](#)
- [Collaborative distributed version control](#)
- [Collaborating and sharing using GitHub without command line](#)

Have you worked with version control before?

Git and GitHub



Git - version control system (track changes)

GitHub, GitLab, Codeberg, ... - hosting, collaboration and sharing

Motivation: Version control is an answer to these questions:

"It broke ... hopefully I have a working version somewhere?"

"Can you please send me the latest version?"

"Where is the latest version?"

"Which version are you using?"

"Which version have the authors used in the paper I am trying to reproduce?"

"Found a bug! Since when was it there?"

"I am sure it used to work. When did it change?"

Commits: keeping track of changes (example repository)

```
$ git log
commit 42fdf8d954c27fb1505685f66a1ac5132935fa53 (HEAD -> main,
Author: Richard Darst <richard.darst@aalto.fi>
Date: Thu Jul 6 16:03:08 2023 +0300

    content/conf: exclude prompts from being copied

commit 4dc7507a885fc9291dea9e1101246f1f5d1d9742
Author: Richard Darst <richard.darst@aalto.fi>
Date: Fri Mar 24 10:17:00 2023 +0200

    content/reference: fix link

commit d6972daf51ce5964cd73080a2f7b519408c824a1
Author: Diana Iușan <diana.iusan@uppmx.uu.se>
Date: Wed Mar 22 09:30:47 2023 +0100

    changed from ssh to https in clone

commit b3d94e50eb8b83a34853d6390294d4f91158ca8d
Author: Diana Iușan <diana.iusan@uppmx.uu.se>
Date: Tue Mar 21 16:07:16 2023 +0100

    small style change

commit bf09389956e0656975dee7606281c2a8ecbe9219
Author: Diana Iușan <diana.iusan@uppmx.uu.se>
Date: Tue Mar 21 16:02:17 2023 +0100

    how do you use git

commit 1cc601e1d6f4033784396f5e5e639714ee4a3273
Author: Diana Iușan <diana.iusan@uppmx.uu.se>
Date: Tue Mar 21 15:00:00 2023 +0100

    added exercise

commit e0b19f16de31565a2be9c77e3a0d1ff798126991
Author: Diana Iușan <diana.iusan@uppmx.uu.se>
Date: Tue Mar 21 14:46:24 2023 +0100

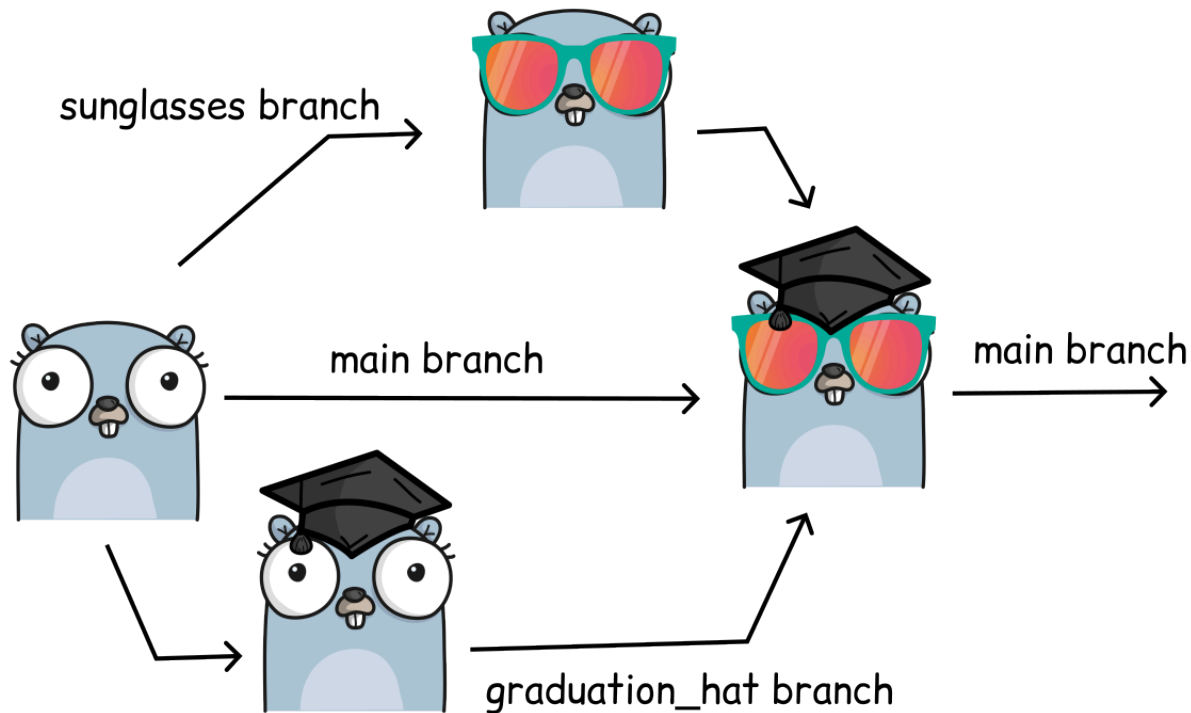
    changed https to git and exercise title

commit d16a0f3e2ba23fc622174fc40a3366dad5883b8b
```

The screenshot shows the GitHub interface for the repository 'coderefinery / git-intro'. The 'Commits' tab is selected, showing a list of commits grouped by date. The commits are as follows:

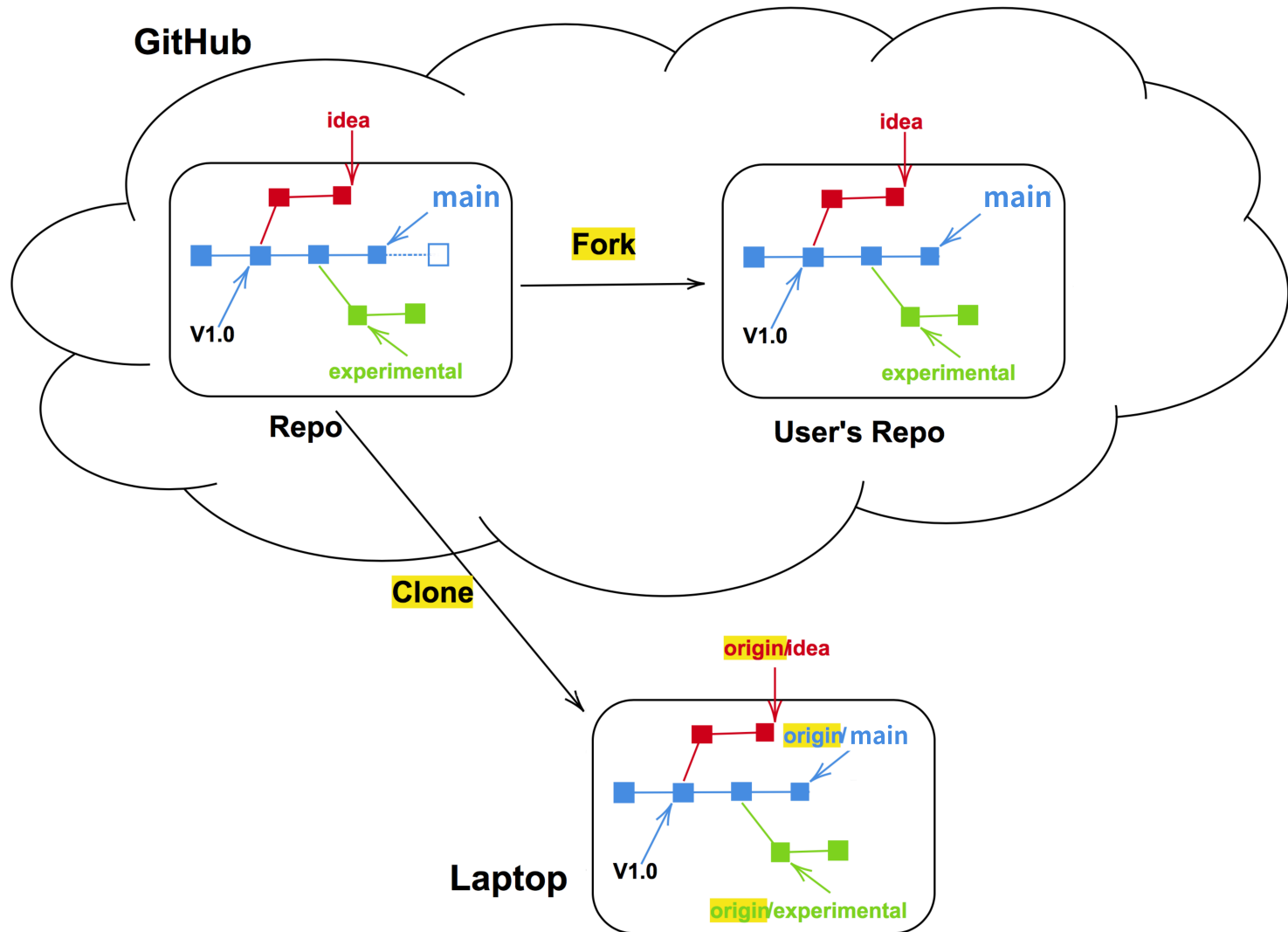
- Commits on Jul 6, 2023**
 - commit 42fdf8d: content/conf: exclude prompts from being copied (by rkdarst)
- Commits on Mar 24, 2023**
 - commit 4dc7507: content/reference: fix link (by rkdarst)
- Commits on Mar 22, 2023**
 - commit d6972da: changed from ssh to https in clone (by dianaiusan, Verified)
- Commits on Mar 21, 2023**
 - commit b3d94e5: small style change (by dianaiusan, Verified)
 - commit bf09389: how do you use git (by dianaiusan, Verified)
 - commit 1cc601e: added exercise (by dianaiusan, Verified)
 - commit e0b19f1: changed https to git and exercise title (by dianaiusan, Verified)
 - Merge pull request #391 from coderefinery/radovan/fix-recovery-steps (Verified)

Working on the same basis - on different tasks




What if two people, at the same time, make two different changes? Git can support merging them together. Image created using <https://gopherize.me/> ([inspiration](#)).


Collaboration through branches or forks



Traceability ([browse this example online](#))

 main networkx / networkx / algorithms / boundary.py Go to file t ...










i Ignoring revisions in .git-blame-ignore-revs. x

 **eriknw** Add @nx._dispatch decorator to most algorithms (#6688) ... ✓

fae8af6 · 2 weeks ago History

Code Blame 167 lines (129 loc) · 5.21 KB Raw Copy Download Edit ...

Older ■■■■■■■■■■ Newer Contributors 12

8 years ago		1	"""Routines to find the boundary of a set of nodes.
		2	
		3	An edge boundary is a set of edges, each of which has exactly one
		4	endpoint in a given set of nodes (or, in the case of directed graphs,
		5	the set of edges whose source node is in the set).
15 years ago	Merged revisions 741-766,769-770...	6	
8 years ago		7	A node boundary of a set *S* of nodes is the set of (out-)neighbors of
		8	nodes in *S* that are outside *S*.
15 years ago	Merged revisions 741-766,769-770...	9	
8 years ago		10	"""
		11	from itertools import chain
15 years ago	Merged revisions 741-766,769-770...	12	
9 months ago		13	import networkx as nx
		14	
8 years ago		15	__all__ = ["edge_boundary", "node_boundary"]
		16	
		17	
2 weeks ago		18	@nx._dispatch(edge_attrs={"data": "default"}, preserve_edge_attrs="data")
8 years ago		19	def edge_boundary(G, nbunch1, nbunch2=None, data=False, keys=False, default=None):
8 years ago		20	"""Returns the edge boundary of `nbunch1`.
8 years ago		21	
		22	The *edge boundary* of a set *S* with respect to a set *T* is the
		23	set of edges (*u*, *v*) such that *u* is in *S* and *v* is in *T*.
		24	If *T* is not specified, it is assumed to be the set of all nodes
		25	not in *S*.

Features

- Through commits you know what changed when and by whom
- Roll-back: You can always go back to a previous version and compare
- Branching and merging: Work on different ideas at the same time
- Collaboration: Review, compare, share, discuss

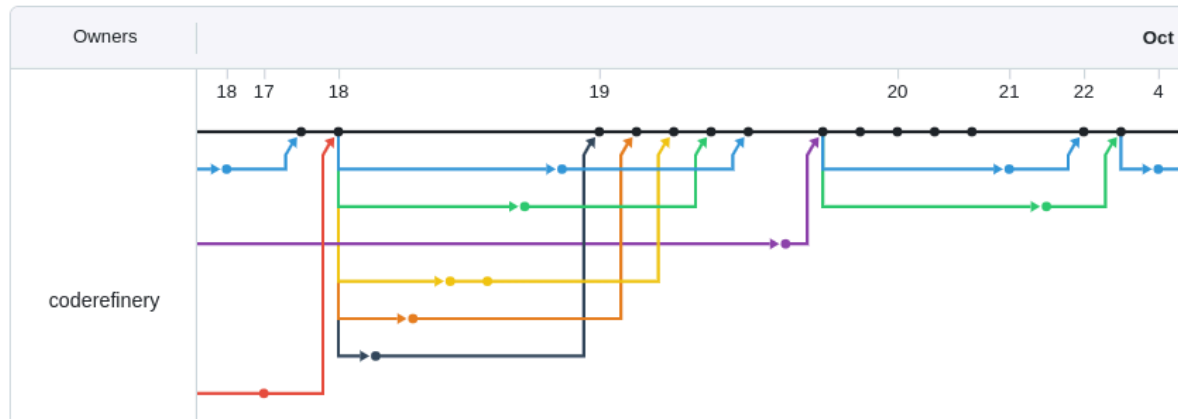
Talking about code

Download the code, go to the file "src/util.rs", and search for "time_iso8601". Oh! But make sure you use the version from August 2023.

Or I can send you a [permlink](#)

```
37     #[cfg(test)]
38     pub(crate) use set;
39
... 40     // Get current time as an ISO time stamp.
41     pub fn time_iso8601() -> String {
42         let local_time = Local::now();
43         format!("{}", local_time.format("%Y-%m-%dT%H:%M:%S%Z"))
44     }
45
46     // Carve up a line of text into space-separated chunks + the start indices of the chunks.
47     ✓ pub fn chunks(input: &str) -> (Vec<usize>, Vec<&str>) {
48         let mut start_indices: Vec<usize> = Vec::new();
```

Code review



- Ask for review via pull request
- Changes are reviewed before they are merged
- Main motivation for code review is the collaborative learning
- Also: better code quality

Demonstration

<https://github.com/coderefinery/planets>

- Commits = snapshots with author, date, message, and ID
- Branches = parallel universes for safe experimentation
- Annotate code to track changes line by line
- Collaboration: fork, clone, review, share, discuss
- Code review via pull or merge requests

Note: Git can support more than code!

- **Software** (this is how it started but Git/GitHub can track a lot more)
- **Scripts**
- **Documents** (plain text files much better suitable than Word documents, this material and slides are tracked using Git)
- **Manuscripts** (Git is great for collaborating/sharing LaTeX or [Quarto](#) manuscripts)
- **Configuration files**
- Website **sources** : [Source for CodeRefinery website](#)
- **Data** (see also [git-annex](#), [git LFS](#))

Exercises I - 20 min

Version control exercises

- Git-1: Create your own GitHub repository
- Git-2: Archaeology with git
- Additional-vc: What do you do with a code you find online

Part 3/6 - 20 min

- Reproducibility
- Directory structure
- Recording dependencies
- Recording computational steps

Reproducibility



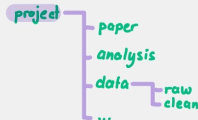
Inspiration and where to find more:

- [CodeRefinery reproducible research lesson](#)
- [The Turing Way: Guide for Reproducible Research](#)
- [Ten simple rules for writing Dockerfiles for reproducible data science](#)
- [Computing environment reproducibility](#)


REPRODUCIBLE RESEARCH

6 helpful steps

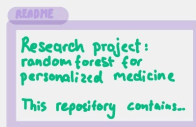
- 1 Get your files + folders in order



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

`6-steps-reproducibility.pdf`  `clean.data <- function(...) { ... }`

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



CC-BY 4.0 Heidi Seibold
@HeidiBaya

- 4 Version control code, text, ...



- 5 Stabilize computing environment and software



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



It all starts with a good directory structure ...

```
project_name/
├── README.md           # overview of the project
├── data/               # data files used in the project
│   ├── README.md      # describes where data came from
│   └── sub-folder/     # may contain subdirectories
├── processed_data/    # intermediate files from the analysis
├── manuscript/        # manuscript describing the results
├── results/           # results of the analysis (data, tables, figures)
├── src/               # contains all code in the project
│   ├── LICENSE        # license for your code
│   ├── requirements.txt # software requirements and dependencies
│   └── ...
└── doc/               # documentation for your project
    ├── index.rst
    └── ...
```

Lottery factor: If you win the lottery and leave research today, will others be able to continue your work?

Have you ever been sure your code works... until your
colleague runs it?

"it works on my machine 🙄"

We usually do not start from scratch ...

... but build on other tools and packages

-> *We need to communicate what is needed to run our code!*

Recording dependencies

Conda, Anaconda, pip, virtualenv, Pipenv, pyenv, Poetry, rye, requirements.txt, environment.yml, renv, makefile, CMakeLists.txt, easyconfig, spack.yml, are related to package and environment management.

They (alone or in combination) help to...

- Define dependencies
- Communicate dependencies
- Install these dependencies
- Record the versions
- Isolate environments
- Provide tools and services to share packages

Isolated environments help you make sure that you know your dependencies!

Demonstration

<https://github.com/coderefinery/planets>

- README
- environment.yml

Small exercise - discuss with neighbor:

Reproducibility exercise

Recording computational steps

Apart from the environment we also need a way to record and communicate computational steps:

- **README** (steps written out "in words")
- **Scripts** (typically shell scripts)
- **Notebooks** (Jupyter, R Markdown, Pluto)
- **Workflows** (Snakemake, Nextflow, doit, ...)

Workflows become interesting when you want to build scalable, reliable and reproducible pipelines.

Break - 20 min

Part 4/6 - Documentation - 20 min

- Where to find more
- Why?
- Checklist
- Comments
- In-code
- README
- Growing out of README

Documentation

Inspiration and where to find more:

- [Documentation - CodeRefinery lesson material](#) by [CodeRefinery](#).
- [My short talk on "Documenting code"](#)

Have you written any kind of code documentation?

Why? - to your future self

- You will probably use your code in the future and may forget details
- You may want others to use your code (almost impossible without documentation)
- You may want others to contribute to the code
- Time is limited - let the documentation answer FAQs

In-code documentation

```
# now we check if temperature is larger than -50
if temperature > -50:
    print("ERROR: temperature is too low")
```

-> Not very useful (more commentary than comment)

```
# do not run this code!
# if temperature > 0:
#     print("It is warm")
```

-> Keeping zombie code "just in case" (rather use version control)

```
# somebody: threshold changed from 0 to 15 on August 5, 2013
if temperature > 15:
    print("It is warm")
```

-> Emulating version control

In-code documentation

```
# we regard temperatures below -50 degrees as measurement errors
if temperature > -50:
    print("ERROR: temperature is too low")
```

-> More useful : Explaining why

In-code documentation - docstrings

- Useful for those who want/need to understand and modify the code
- Docstrings can be useful both for developers and users of a function

```
def kelvin_to_celsius(temp_k: float) -> float:
    """
    Converts temperature in Kelvin to Celsius.

    Parameters
    -----
    temp_k : float
        temperature in Kelvin

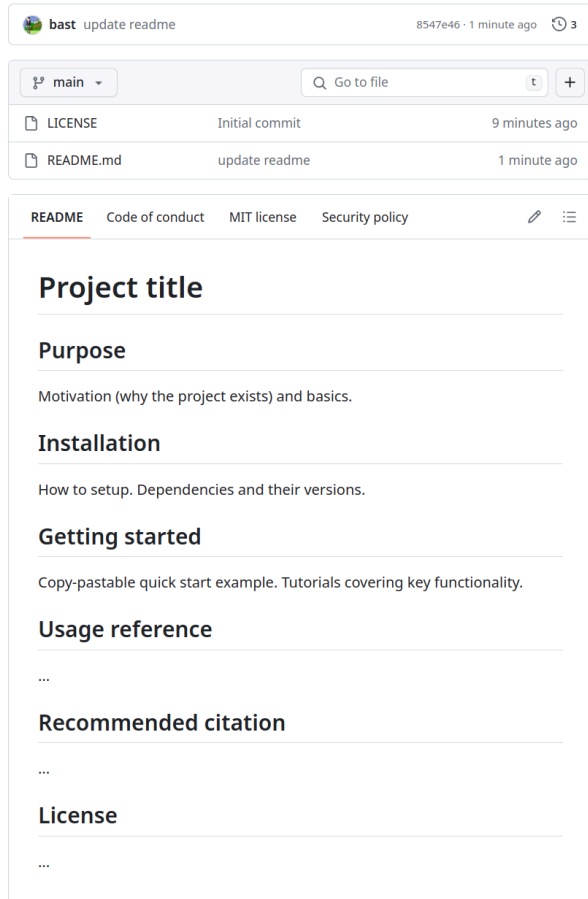
    Returns
    -----
    temp_c : float
        temperature in Celsius
    """
    assert temp_k >= 0.0, "ERROR: negative T_K"

    temp_c = temp_k - 273.15

    return temp_c

print(kelvin_to_celsius.__doc__)
```

README provides the first impression



About

Example project.

- Readme
- MIT license
- Branches
- Tags
- Activity
- 0 stars
- 1 watching
- 0 forks

Releases

No releases published
[Create a new release](#)

Packages

No packages published
[Publish your first package](#)

Project title

Purpose

Motivation (why the project exists) and basics.

Installation

How to setup. Dependencies and their versions.

Getting started

Copy-pastable quick start example. Tutorials covering key functionality.

Usage reference

...

Recommended citation

...

License

...

README - Checklist

- Purpose
- Installation instructions
- Dependencies and their versions or version ranges
- Copy-paste-able example to get started
- Tutorials covering key functionality
- Reference documentation (e.g. API) covering all functionality
- How do you want to be asked questions (mailing list or forum or chat or issue tracker)
- Possibly a FAQ section
- Authors
- Recommended citation
- License
- Contribution guide

See also [JOSS review checklist](#)

When projects grow out of a README

- Write documentation in [Markdown \(.md\)](#) or [reStructuredText \(.rst\)](#) or [R Markdown \(.Rmd\)](#).
- In the **same repository** as the code -> reminder to update both, version control and **reproducibility**
- Use one of many tools to build HTML out of md/rst/Rmd: [Sphinx](#), [Zola](#), [Jekyll](#), [Hugo](#), RStudio, [knitr](#), [bookdown](#), [blogdown](#), ...
- Deploy the generated HTML to [GitHub Pages](#) or [GitLab Pages](#)

Examples

- [All CodeRefinery lessons](#)
- <https://github.com/networkx/networkx>
- [EODIE - my own project, now continued by others](#)

Exercises II - 20 min

Documentation Exercises

- Documentation-1 a+b : Build Documentation webpage from Markdown text with Sphinx
- Additional-doc: Where to find information

Part 5/6 - Sharing and reusing - 10 min

- Where to learn more
- Why it matters
- Beginning and later
- Derivative
- In practice
- Citation
- Reusing

Sharing and reusing





Inspiration and where to find more:

- [Social coding lesson material](#) by [CodeRefinery](#).
- [Practical software licensing slides](#) by [Radovan Bast](#)
- [UiT research software licensing guide \(draft\)](#).

Why software licenses matter

You find some great code that you want to reuse for your own publication.

- This is good for the original author - you will cite them. Maybe other people who cite you will cite them.
- You modify and remix the code.
-  Two years later ...
- Time to publish: You realize **there is no license to the original work** 

Now we have a problem

 "Best" case: You manage to publish the paper without the software/data

Others cannot build on your software and data.

 Worst case: You cannot publish it at all

Journal requires that papers should come with data and software so that they are reproducible.

Beginning of a project



[Midjourney, CC-BY-NC 4.0]

- License does not seem important
- Easy to change (*)
- Work as if the code is public even though it still may be private
- "Open core" approach: Core can be open and on a public branch, unpublished code can be on a private repository

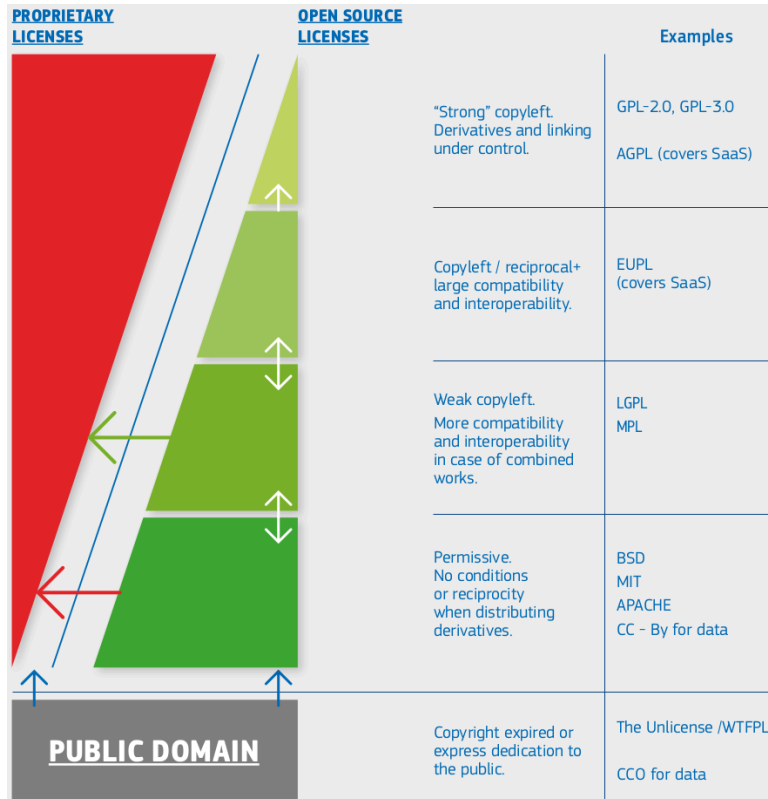
Later in the project



[C.Stadler/Bwag, CC-BY-SA 4.0]

- Can be important
- Especially when combining codes or organizations
- Difficult to change
- Difficult to remove code that should not be published
- Authors change affiliation

Is your work derivative work or not?



- **Derivative work:** You have started from an existing code and made changes to it or if you incorporated an existing code into your code, beware the "sticky licenses" (arrows in the image represent compatibility: A -> B: B can reuse A)
- You have started from scratch: **not derivative work**

[European Union Public Licence (EUPL): guidelines July 2021,

<https://data.europa.eu/doi/10.2799/77160>]

Citability

Provide all information about the code, including its name, current version and authors with the code:

E.g. via [CITATION.cff](#) file:

```
cff-version: 1.2.0
message: "If you use this software, please cite it as below."
authors:
  - family-names: Doe
    given-names: Jane
    orcid: https://orcid.org/1234-5678-9101-1121
title: "My Research Software"
version: 2.0.4
doi: 10.5281/zenodo.1234
date-released: 2021-08-11
```

Many tools understand CITATION.cff

The screenshot shows a GitHub repository named 'bast' with the description 'generate .zenodo.json from CITATION.cff'. The repository has 3b210d2 commit, updated 2 months ago, and 314 stars. A red arrow points from the 'CITATION.cff' file in the file list to the 'Cite this repository' modal.

File List:

File	Description	Updated
.github/workflows	test python 3.8 and up	2 months ago
LICENSES	mv LICENSE file to LICENSES/MPL-2.0.txt	2 months ago
doc	use current year	7 months ago
img	add image	
runtest	add copyright and licensing information to ea...	
.gitignore	adapt .gitignore	
.mailmap	add .mailmap	
.zenodo.json	generate .zenodo.json from CITATION.cff	
CITATION.cff	add CITATION.cff	
README.md	mv LICENSE file to LICENSES/MPL-2.0.txt	
pyproject.toml	use markdown for the readme file	6 months ago
requirements.txt	generate .zenodo.json from CITATION.cff	2 months ago

About

Numerically tolerant end-to-end test library for research software.

[runtest.readthedocs.io](#)

python integration-testing

Readme

Cite this repository

Cite this repository

If you use this software in your work, please cite it using the following metadata. [Learn more about CITATION files.](#)

APA **BibTeX**

Bast, R. (2023), runtest: Numerically tolera

[View citation file](#)

6 months ago + 9 releases

Contributors 6

Make your code persistent

Making a code public on GitHub is **not enough**. It may not exist anymore in X years. Links may change...

Better:

Get a [digital object identifier \(DOI\)](#) for your code from [Zenodo](#) or similar.

[Software Heritage](#) and [CodeMeta](#) exist as an alternative ecosystem that is currently receiving some attention on a European level. Comparison and links to converters can be found in <https://zenodo.org/record/8086413>.

Sharing and reusing - Great resources

- [Joinup Licensing Assistant - Find and compare software licenses](#)
- [Joinup Licensing Assistant - Compatibility Checker](#)
- [Social coding lesson material](#) by [CodeRefinery](#).
- [Citation File Format \(CFF\)](#).
- [License Selector](#)
- [Research institution policies to support research software](#)
([compiled by the Research Software Alliance](#)).

Exercises III - 20 min

Sharing and reusing exercises

Choose what seems most relevant:

- Sharing-1: Check your organizations licensing recommendations
- Sharing-2: Create your own CITATION.cff file
- Sharing-3: Discuss derivative work
- Sharing-4: Discuss common licensing situations
- Sharing-5: Get a "dummy DOI" for your repository
- Additional-sharing: What can you do with the planets code?

Part 6/6 - Where to go from here - 10 min

- Practical info
- Where to go from here
- Recommendations
- Thanks

Practical version control

- [Install and configure Git](#)
- In 3 commands from nothing to first commit:

```
$ git init  
$ git add myscript.py  
$ git commit
```

- Go through [CodeRefinery](#) lessons [Git intro](#) and [Collaborative Git](#)

Practical version control

Simple personal projects

- Start with just the **main** branch
- Later use branches for unfinished/untested ideas
- Use tags to mark important milestones (**phd-thesis-submitted**, **published-manuscript**)
- Better too many commits than too few
- Better imperfect commits than no commits

Projects with few persons

- Write-protect the **main** branch
- New idea/feature: new branch
- Use code review: changes are reviewed and discussed before they are merged

Practical reproducibility

- Use file- and variable names that still make sense to you in a few weeks
- Use an **intuitive project structure**
- Write down your dependencies in a file
- Write down how to run your codes step by step -> If it gets complicated, look into workflow tools

Practical documentation

- With **good naming** you might need less in-code documentation
- Where things are not obvious, write a comment
- Provide a README with all headers from the checklist
- If the README gets hard to navigate, look into static site generators
- Extensions may help integrating docstrings automatically

Practical licensing

- Create a **LICENSE** file or **LICENSES/** folder in your project which will hold [license texts](#).
- On top of each file add and adapt the following header ([more examples](#)):

```
# SPDX-FileCopyrightText: 2023 Jane Doe <jane@example.com>  
#  
# SPDX-License-Identifier: MIT
```

Practical steps for making **changes to an existing project** (with a license that allows you to do so):

- Fork (copy) the project.
- Summarize your changes in file headers and bigger-picture changes in the README.
- Some licenses are more permissive (you can keep your changes private) but some licenses require you to publish the changes (share-alike).

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>]

Recommendations - It's about communicating!

- Track your code with Git
- Help each other with reviewing code: great learning
- Documentation: start with a README in the same Git repo
- Document your dependencies and computational steps
- Make your code/script/notebook citable and give it a license

Embrace "Good enough"!

Spending a thought on these topics now may make your life easier in the future.

What we did not talk about but is important

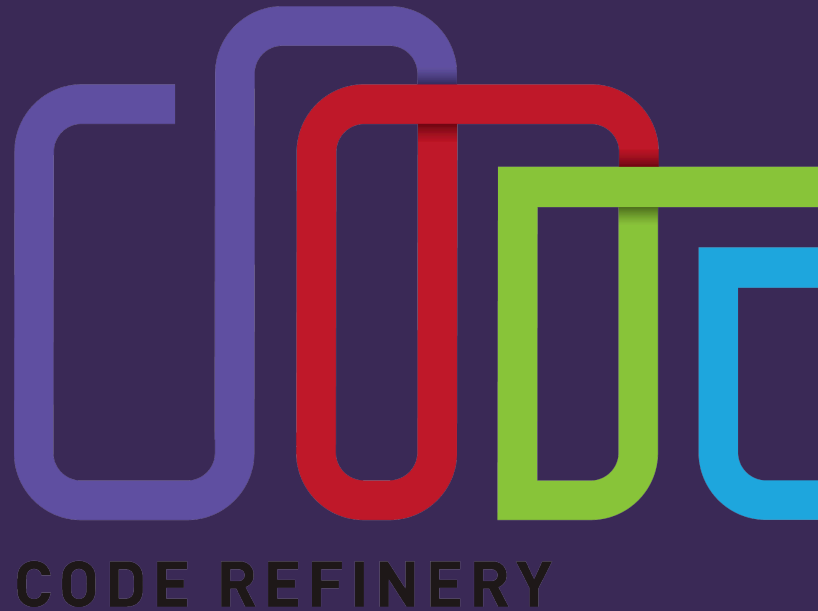
- Containers (operating system and tools within one file)
- Automated testing
- Modular code development

-> Come to [CodeRefinery workshop in September/October](#) or [read materials!](#)

-> Discuss these and other topics with other enthusiasts at [Nordic-RSE](#).

You can also find other courses on software design, performance optimization, debugging etc all around.

Big Thanks to the CodeRefinery team and partners!



<https://coderefinery.org/>