CSC

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

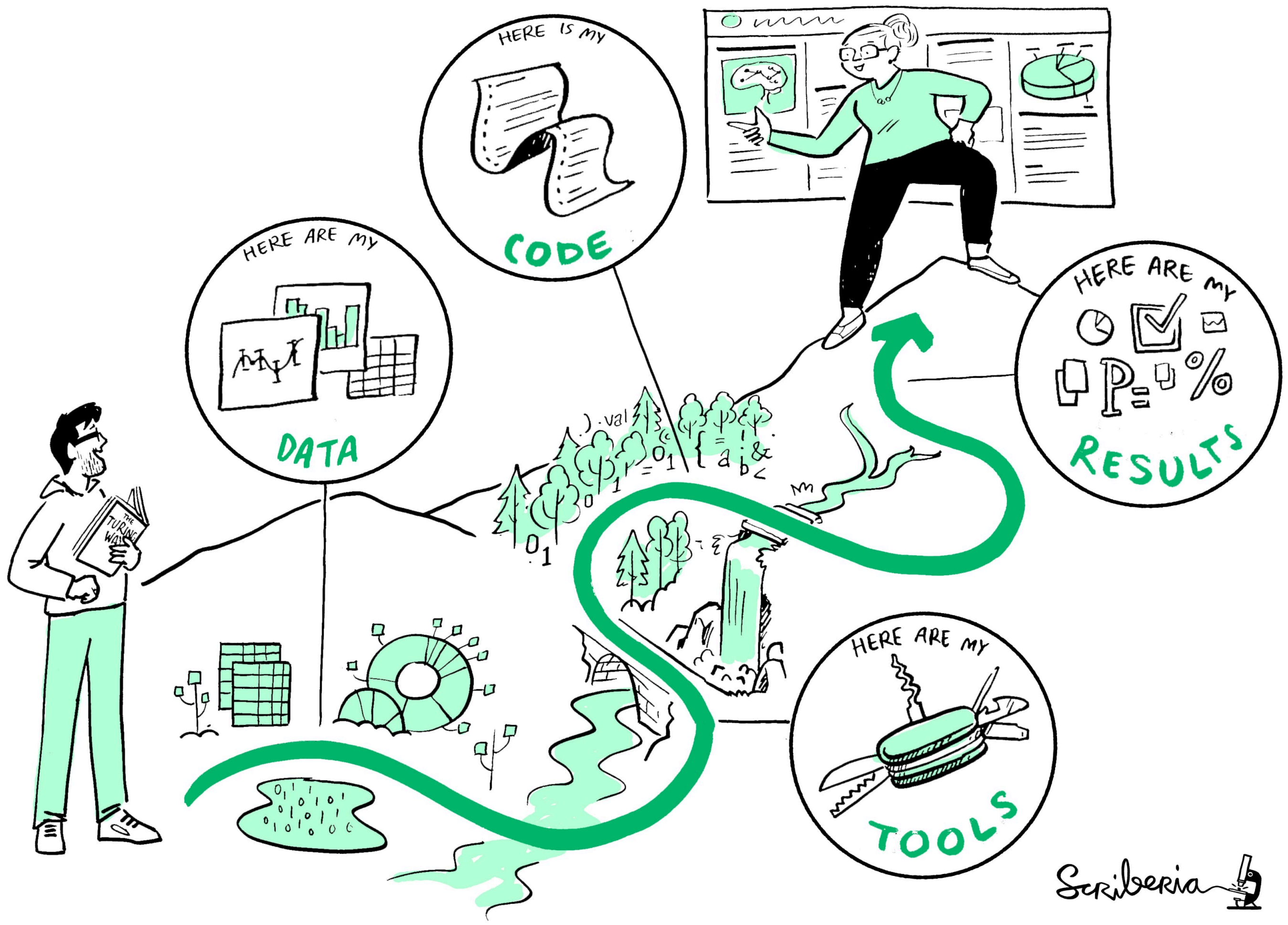
- Computational notebooks
- Basic features

Where to go from here

(Optional) Prerequisites for following along

- GitHub account

- Access to Noppe



[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!

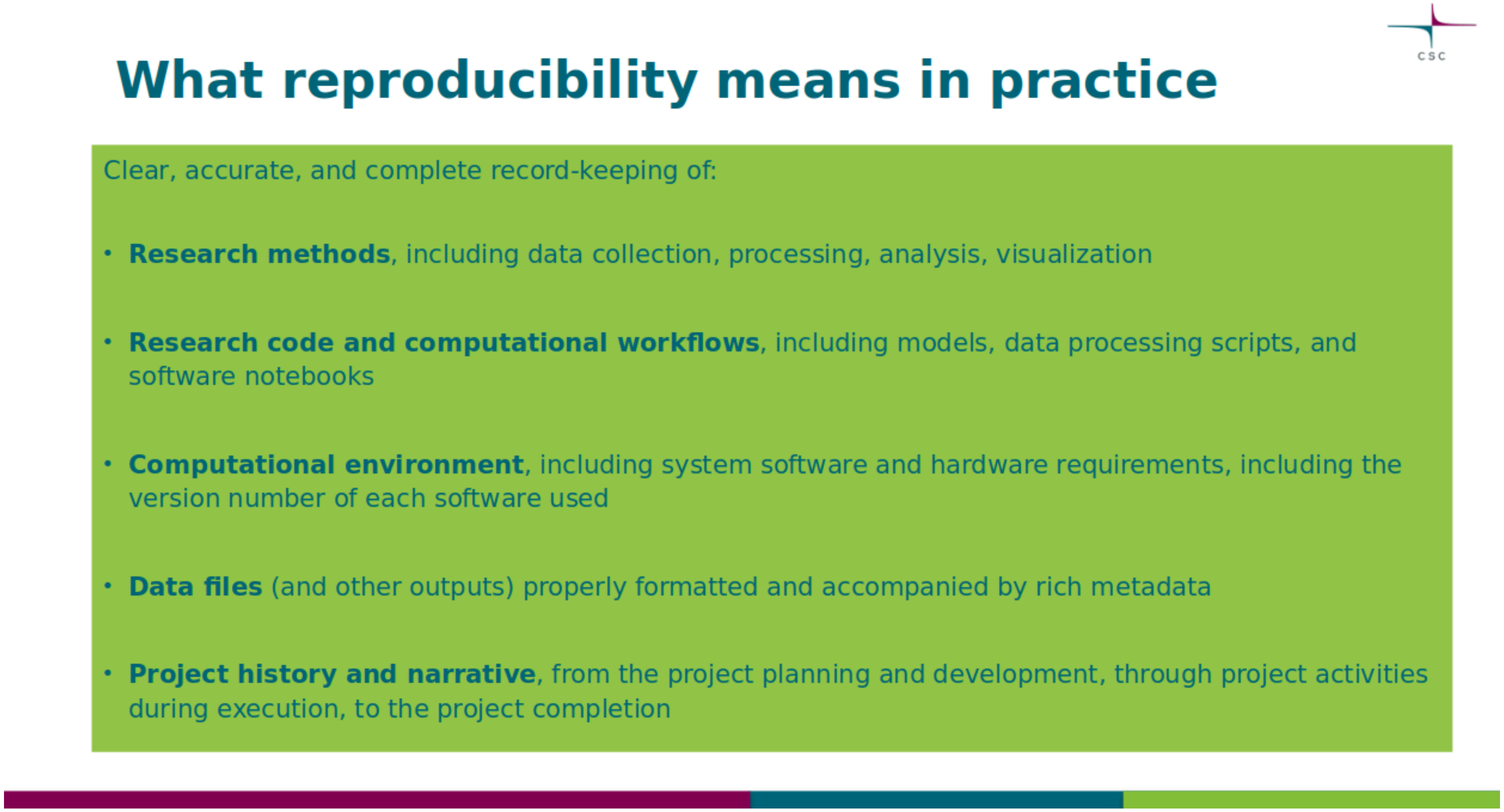
Chatter - Let's practice!

One word to describe your morning today?

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

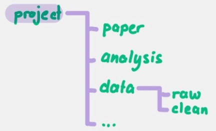
Clear, accurate, and complete record-keeping of:

- **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

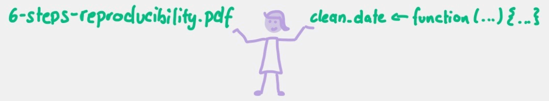
Reproducible Research

6 helpful steps

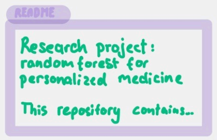
1 Get your files + folders in order



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


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

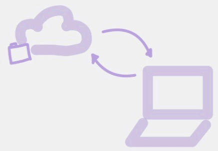
3 Document with care:
README, Metadata, code comments, ...


README

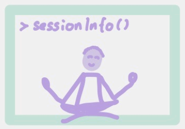
Research project:
random forest for
personalized medicine

This repository contains...

4 Version control code, text, ...

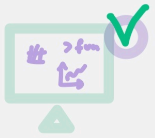


5 Stabilize computing environment and software



```
> sessionInfo()
```

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



CC-BY 4.0 Heidi Seibold
@HeidiBaya

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, ...

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**

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. Add, 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/

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

Create a new repository

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

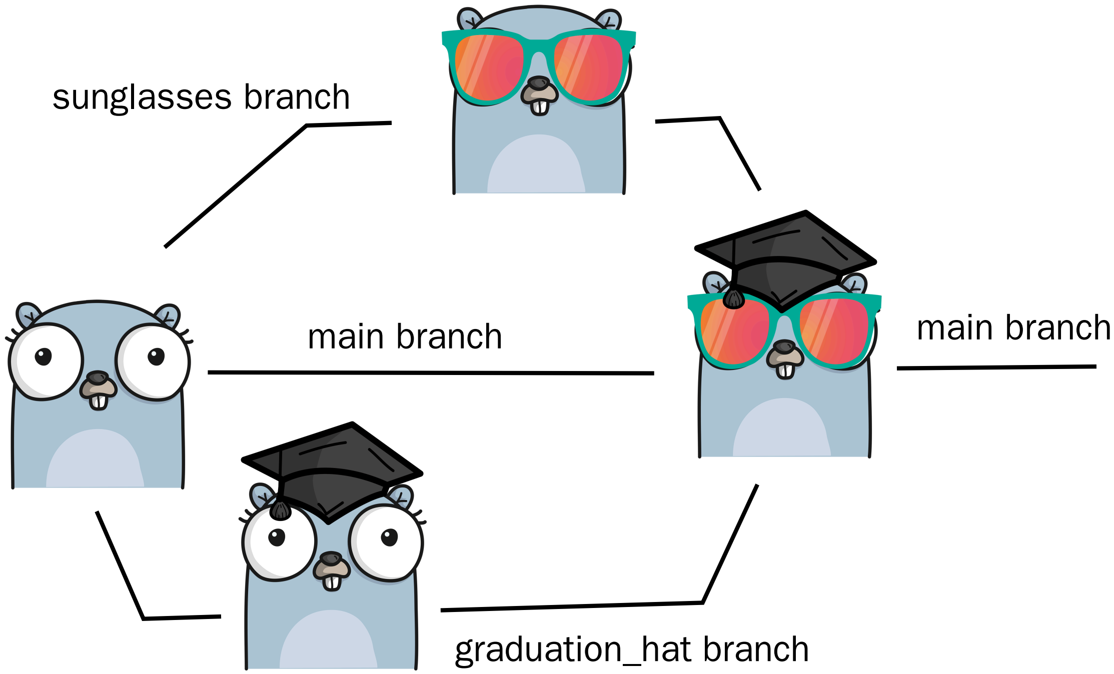
Add new file

- Edit
- Commit
- `main`

History

Annotate

Branches and merge

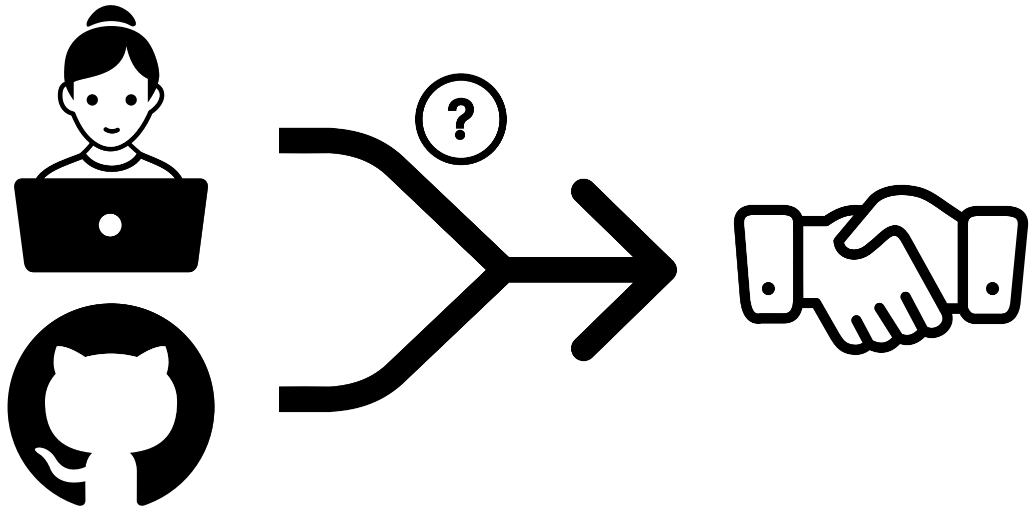


[

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!

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!

Demo - exploring an existing repo

- History
- Branches
- Forks
- Issues
- Pull requests

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

Demo - contribute

- 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)?

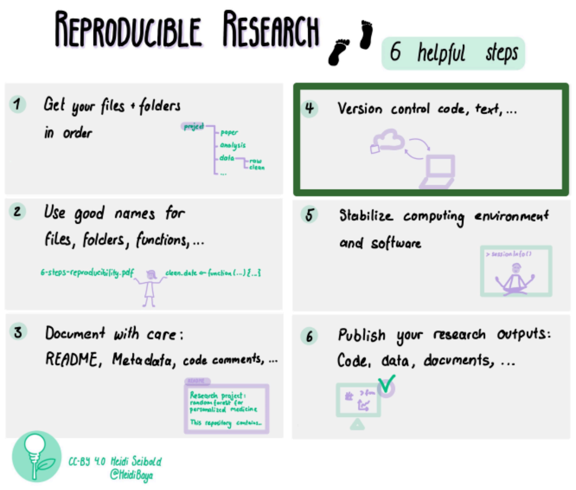
✓

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

✗

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

GitHub and reproducibility


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, ...
CC-BY 4.0 Heidi Seibold @HeidiBaya

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

Do you see any usecases for your work and GitHub in the future?

Break

jupyter

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

Executable notebooks

quarto

jupyter

Pluto.jl

...

Researcher perspective

Exploration

code, notes/explanation, visualization

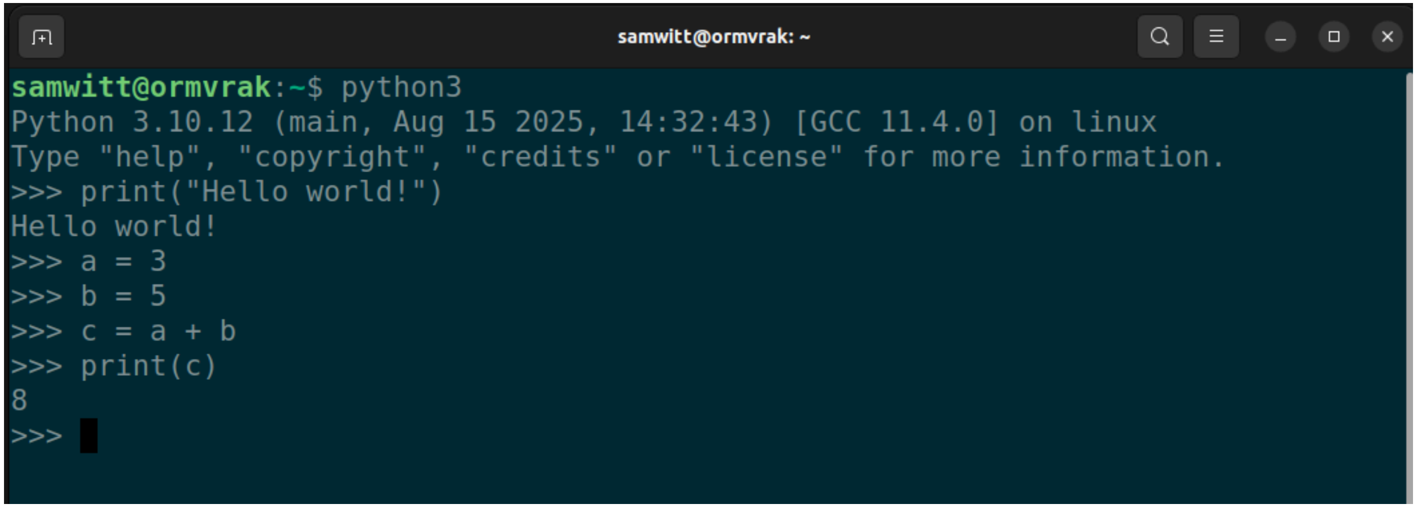
Publish a paper

Sharing narrative

Share to test and adapt

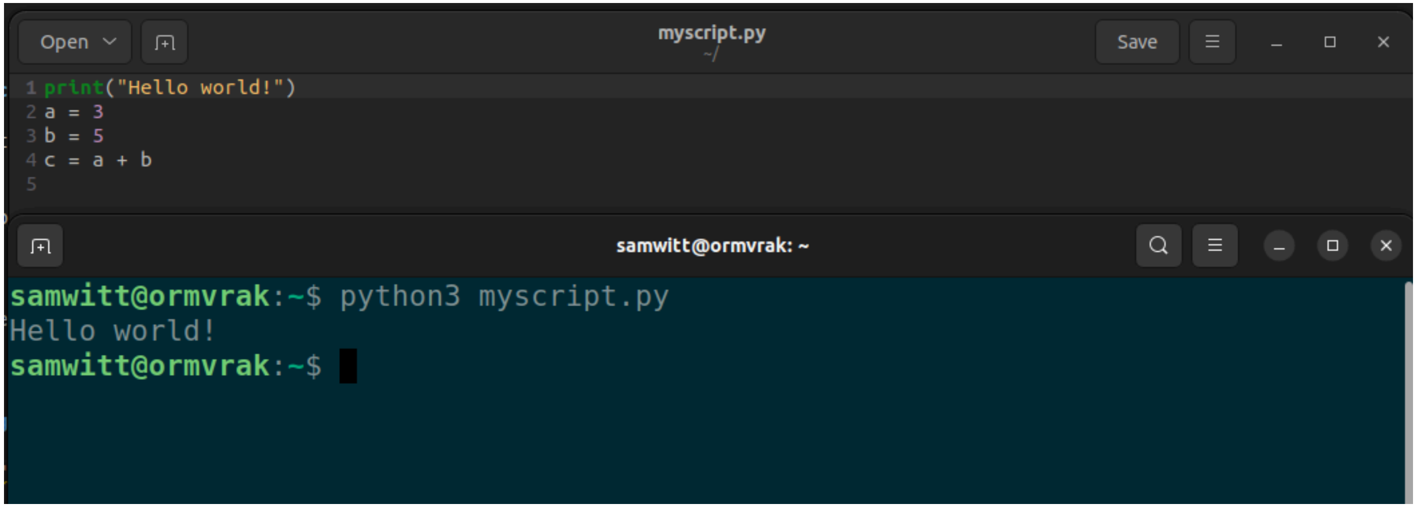
executable for others

Coding in the terminal



```
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

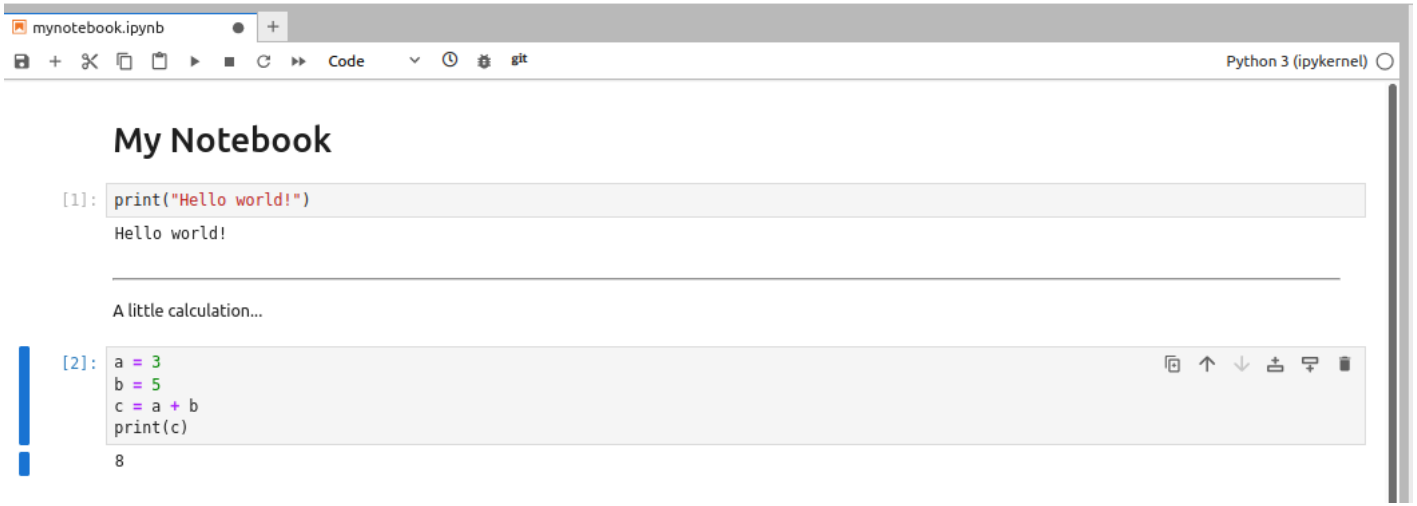


```
print("Hello world!")
a = 3
b = 5
c = a + b

```

```
samwitt@ormvrak:~$ python3 myscript.py
Hello world!
samwitt@ormvrak:~$ 
```

Notebooks: interactive



My Notebook

```
[1]: print("Hello world!")
Hello world!
```

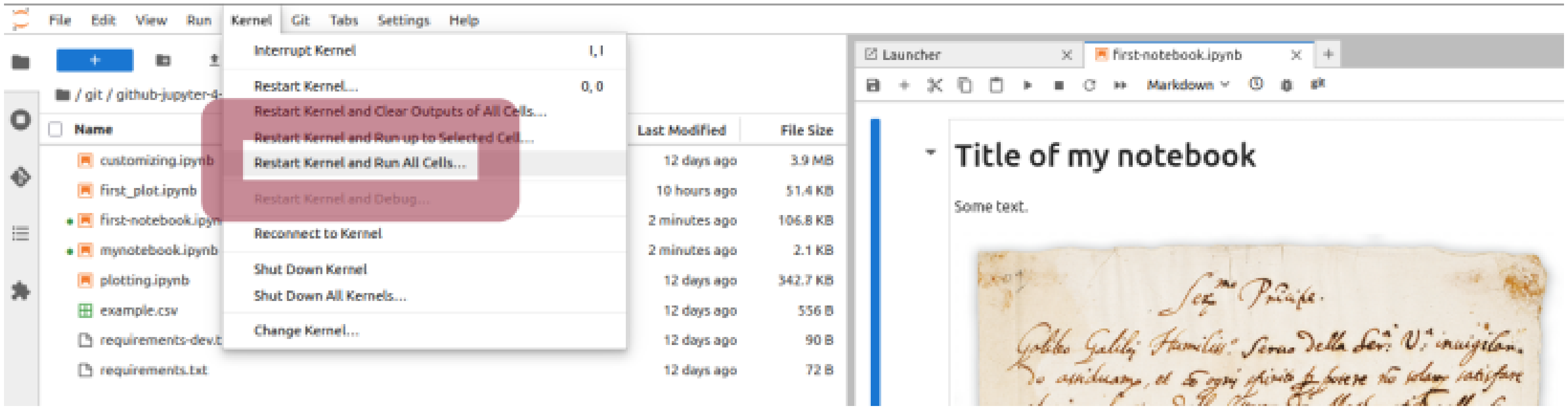
A little calculation...

```
[2]: a = 3
     b = 5
     c = a + b
     print(c)
8
```

Demo usecase: Protopyping / Exploration

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

Good practice



/ git / github-jupyter-4-

Name	Last Modified	File Size
customizing.ipynb	12 days ago	3.9 MB
first_plot.ipynb	10 hours ago	51.4 KB
first-notebook.ipyn	2 minutes ago	106.8 KB
mynotebook.ipynb	2 minutes ago	2.1 KB
plotting.ipynb	12 days ago	342.7 KB
example.csv	12 days ago	556 B
requirements-dev.t	12 days ago	90 B
requirements.txt	12 days ago	72 B

Title of my notebook

Some text.

Demo usecase: Teaching

- Prefilled
- Exercises as rendered text
- Automatic checks

➡ https://github.com/csc-training/python-introduction/blob/gh-pages/notebooks/examples/1%20-%20Introduction.ipynb

➡ https://github.com/csc-training/PythonGIS_CSC/blob/master/Raster/Seurasaari_trees.ipynb

Demo 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**

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

Did you know?

Noppe: CSC service free of charge for Finnish researchers

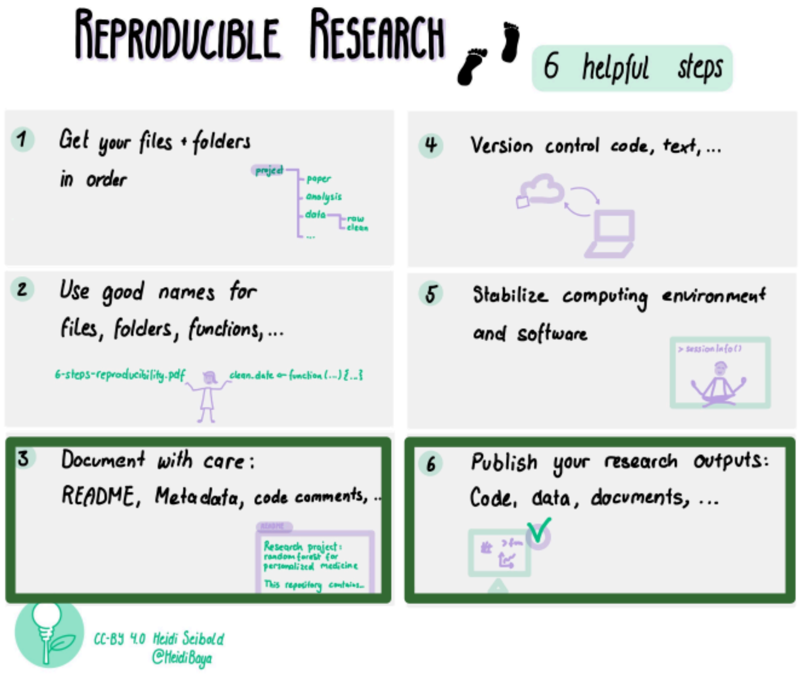
➡ Readymade notebooks

➡ Teaching

➡ Collaboration

Brought to you by ministry of education and culture!

Jupyter and reproducibility



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

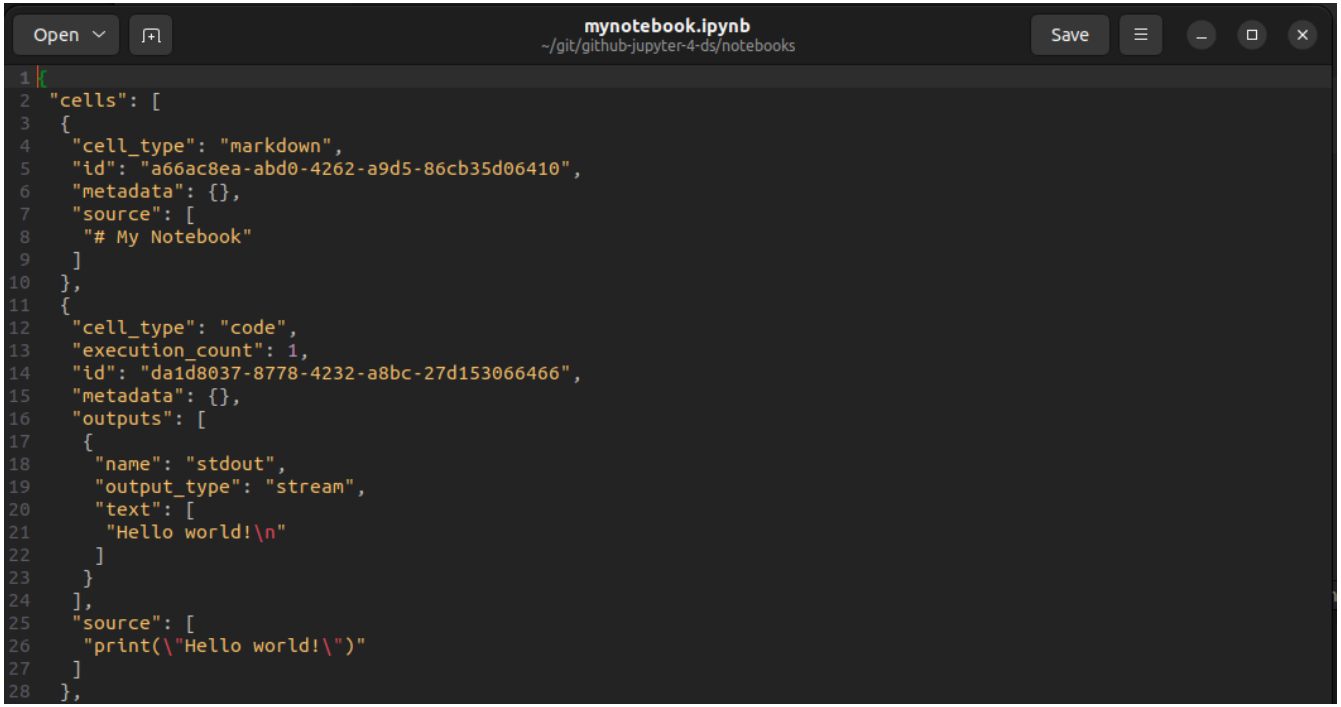
6 Publish your research outputs: Code, data, documents, ...

project
paper
analysis
data
raw
clean
6-steps-reproducibility.pdf
clean_data <- function(...){...}
> sessionInfo()
README
Research project: random forest for personalized medicine
This repository contains...

CC-BY 4.0 Heidi Seibold @HeidiBaya

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

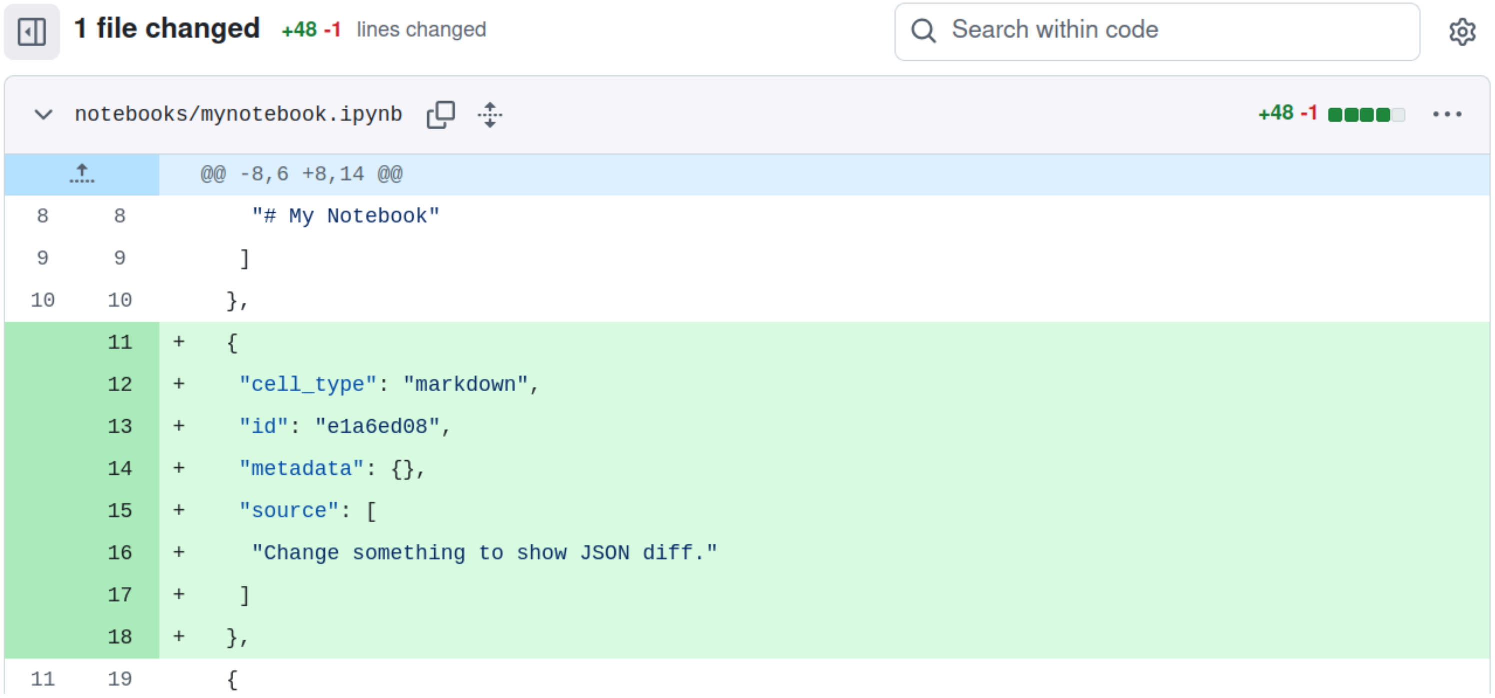
Under the hood



```
{
 "cells": [
  {
   "cell_type": "markdown",
   "id": "a66ac8ea-abd0-4262-a9d5-86cb35d06410",
   "metadata": {},
   "source": [
    "# My Notebook"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 1,
   "id": "da1d8037-8778-4232-a8bc-27d153066466",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Hello world!\n"
     ]
    }
   ],
   "source": [
    "print(\"Hello world!\")"
   ]
  },
```

IPYNB ➡ JSON

Jupyter diff



1 file changed +48 -1 lines changed

Search within code

notebooks/mynotebook.ipynb +48 -1

```
@@ -8,6 +8,14 @@
     "# My Notebook"
    ]
   },
+  {
+   "cell_type": "markdown",
+   "id": "e1a6ed08",
+   "metadata": {},
+   "source": [
+    "Change something to show JSON diff."
+   ]
+  },
   {
```

Version control possible, but limited benefits

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 understanding Jupyter support your work in the future?

Where to go from here ...

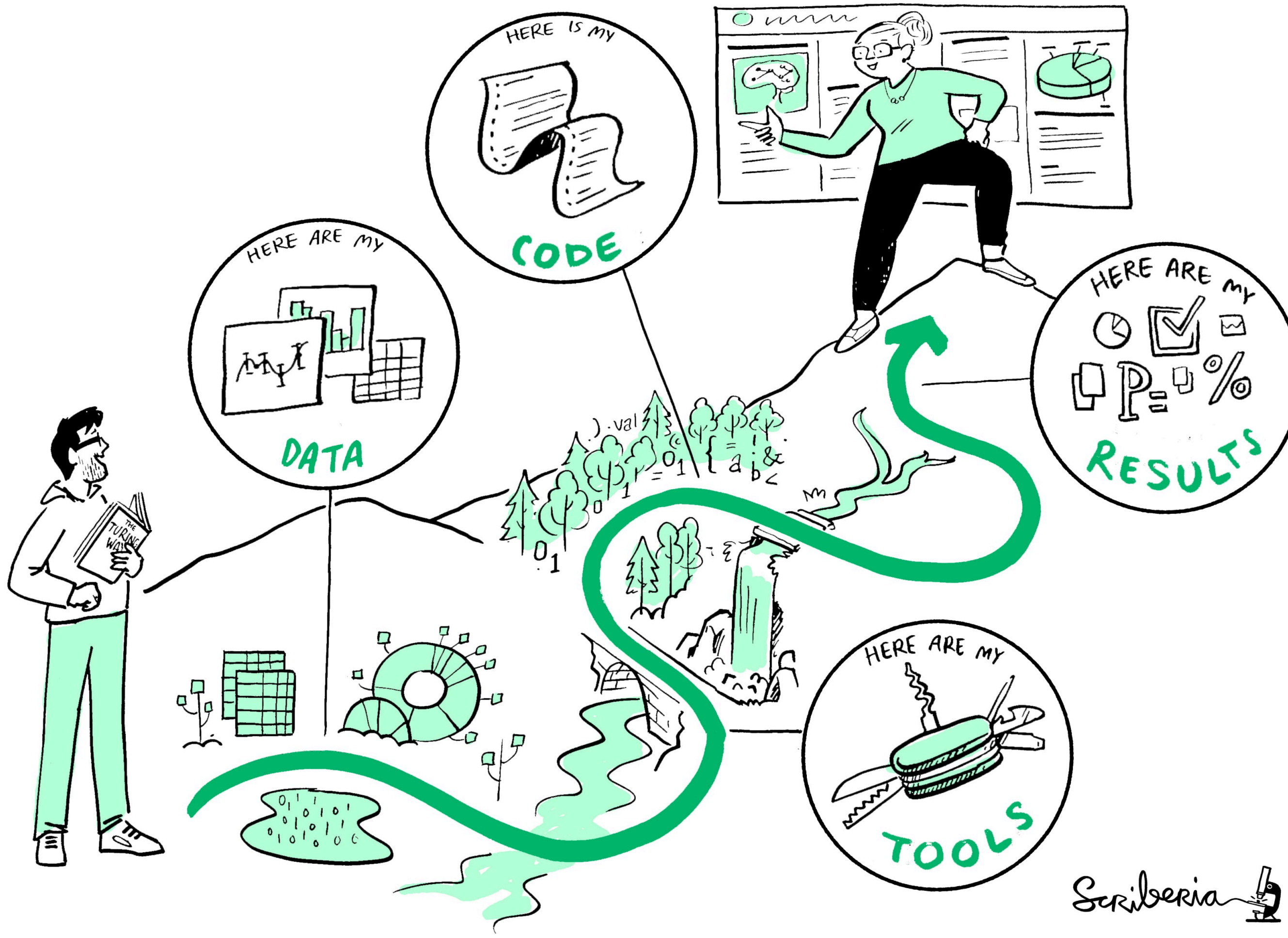
Play around with **GitHub**

- 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...



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

CODE REFINERY

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

One new thing you learned today?

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.