Code Publishing Cheat Sheet

Project

Must: Share the source code

Storage & version control

Source code should be stored & published on a version control system, such as Git. Most platforms offer other functionalities (collaborative tools, bugs tracking, wiki, CI/CD..).

WHY? Allow collaboration in an async way at any time. Allow you to go back in time in your project.

TOOLS:
- ❑ GitHub
- ❑ GitLab (self-hosted version)
- ❑ c4science (deprecated platform of EPFL).

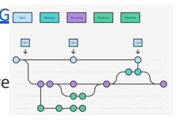
Interactive scripts in Jupyter notebooks may be shared via nbviewer (open to the world), Noto EPFL (for EPFL community).

Git workflow

WHY? Be consistent to facilitate work between contributors

A Git workflow will help you to structure how you use Git (branches, pull/merge requests, …), you can adapt it for your use case, the point is to be consistent between all contributors.

TOOLS: Git Workflows: Gitflow / Gitlab flow / G



Typical branch structure:
- ➔ main: always points to the latest release / ve
- ➔ develop: integration/working branch
- ➔ feature/*: work on separate feature branches

Git commit message

WHY? Having a consistent message format enable a better collaboration

- ❑ Separate subject from body with a blank line
- ❑ Limit the subject line to 50 characters
- ❑ Capitalize the subject line
- ❑ Do not end the subject line with a period
- ❑ Use the imperative mood in the subject line
- ❑ Wrap the body at 72 characters
- ❑ Use the body to explain *what* and *why* vs. *how*

TOOLS:
- ➔ gitmoji
- ➔ conventional-commit
- ➔ commit-lint

Project structure

Following the language and/or framework scaffolder by using a project initializer:

WHY? Allow the app to scale: grow, stay maintainable, extendable; Easier for new contributors since it will follow existing patterns.

TOOLS:
- JavaScript/TypeScript
  - Vue CLI (have a look to existing project like vuetify/quasar/element-ui)
  - Angular CLI
- Python: there is no "official" project layout, but a general one would look like the following on the right

```
my_app/
│
├── my_app/
│   ├── __init__.py
│   ├── core.py
│   └── helpers.py
│
├── tests/
│   ├── test_basic.py
├── docs/
│   ├── conf.py
│   ├── index.rst
└── setup.py
├── .gitignore
├── LICENSE
├── README.md
├── requirements.txt
```

Code

General principles
- Avoid code smell & design smell (cf wikipedia)
- Don't Repeat Yourself (DRY/AHA)
- Separation of concerns
  - Separate Code and Data
  - Follow design patterns like mvc or mvvm.
  - css/js/html
- REST Principles (for APIs)
  - Statelessness
  - Cacheability
  - Uniform interface

General advices

- Meaningful names for variables, functions et c
- Avoid acronyms and abbreviations: prefer readability
- Avoid the obvious comment: best practices for writing code comments
- Better no comment than a faulty comment!
- Be consistent
- Refactor Early and Often
- No dead code: **WHY?** It's unnecessary to keep code that is unused or commented. That's what the versioning is for.
- No magic numbers: **WHY?** More readable to name constant, more maintainable (one place)
- Limit lines per files : If it's too big you'll spent time looking for the part of the code, use modules
- limit characters per line : **WHY?** you don't want to scroll horizontally, it's more readable for the brain (usually max 120/160 char, 80 char recommended). E.g Books

Coding style

Select a code style for each language and enforce them in the whole project

WHY? consistency is key for maintainability/readability/contributors

TOOLS:

Language	Style Guide	Enforcement tool
Python	PEP 8	Flake8
JavaScript/TypeScript	Prettier	ESLint
C++	Google C++ Style Guide	Cpplint
VueJs	eslint vue	eslint-plugin-vue

12 Factor (link to an illustrated version)

1. One codebase tracked in revision control, many deploys/environemnent(prod/test/dev)
2. Explicitly declare and isolate dependencies (requirements.txt/package.json/lock)
3. Store config in the environement (env variable/password/port/host_url)
4. Backing services as resources (meaning separate : use docker-compose)
5. Strictly separate build and run stages
6. Execute the app as one or more stateless processes (OS meaning)
7. Export services via port binding (identity of service is via a port not a host)
8. Scale-out via the process mode (separate process by concerns/purpose)
9. Maximize robustness with fast startup and graceful shutdown Keep development, staging, and production as similar as possible
10. Treat logs as event streams (the app does not handle the logs just write to stdout)
11. Run admin/management tasks as one-off processes (part of the release cycle; use code/env same as the app)

Documentation

Must: Document the project/code

Code documentation:

Code should also be documented:
- Python Docstring (pep 257)
- JavaScript/TypeScript: JSDoc

Documentation Generator : you can automate with:
- Python: Sphinx, Read the Docs, Doxygen
- Other languages: Doxygen, Others

Use may use standard-version/conventional changelog to generate changelog and documentation related to releases

Must: Explain how to run the code

Project documentation

Readme should :
- contain info on how to install, run, deploy the code
- credits authors and share license
- describe input and output data
- Link to open data if relevant - include sample/dummy data in expected formats to test running the code.

Contributing info:
- Add a CONTRIBUTING.md at root (to help contributors onboarding)
- More infos on open source projects (guide on how to do a good open source project)
- Examples of github repository having a good contributing file: Atom editor

Packaging

Must: List dependencies

Dependency Management

WHY? Using a dependency management tool is crucial to ease project setup/installation. It follows the 12 factor guidelines -> reproducible build step

Tools:
- Python
  - Pipenv: Pipfile
  - conda: environment.yml
  - pip: requirements.txt
- JavaScript/TypeScript
  - npm: package.json

An abstraction layer such as a Docker image can remove the hassle of dependencies (see **Distribution** below)

Distribution

To distribute your application on a public package repository (such as PyPI) and make it easily installable by everyone, you will need to build and pack it in a distribution package.
- Python: Packaging Python Projects to PyPI
- JavaScript/TypeScript: Contributing packages to npm

Docker
Containerization with Docker helps reproducibility of the project. A container is like a virtual machine, but lighter and faster: https://www.docker.com/resources/what-container
Once you define the image, it can be run on any machine in the same way.

Docker Compose
With Docker Compose, you can define how a set of Docker containers starts.

Versioning
Publishing your application also means creating some releases. This helps the user to know which version it uses and what are the existing ones.
Each release should have a version number and it should follow the well known semantic versioning.

Dev Tools

Testing

Testing is key at each release of the software, to avoid breaking the code. (Read more about tests in Python).

There are many kinds of tests and the most commons are:
- **Functional Tests**: verify the output given some inputs
  - **Unit Tests**: test individual methods
  - **Integration Tests**: check behavior on a running application with other components (database, service, …)
- **Non-Functional Tests**: to determine breaking points
  - **Performance Tests**: check that the application responds in the expected time

It is a good practice to at least create some unit tests during the development instead of testing manually your application.

Language	Unit Tests	Integration Tests
Python	unittest	
JavaScript/TypeScript	Mocha JEST	Cypress Nightwatch.js
Vue3	vitest	Cypress

Continuous Integration / Continuous Delivery

WHY? Instead of running the tests, code analysis and/or generating the documentation manually, this can be done automatically by a continuous integration pipeline. Whenever you push some commits on the repository, a process can check if there is no regression in your application.

TOOLS:	CI	Static Code Analysis
GitHub	GitHub Actions	Code Scanning
GitLab	GitLab CI	Code Quality Code Security
Any Platforms	Jenkins Travis CI	SonarCloud

Publish

User Support

User support is very important to building a community of users around your software. Choose an outlet to keep track of user's comments (and/or bug tracking), and point your users to it. General forums also work (StackOverflow…)
- Google groups
- GitHub Issues
- GitLab Issues
- Discourse

Communication & dissemination
Dissemination time/effort is not to be underestimated to achieve best impact of open software

- Website
- Communication strategy (community of users, who's the audience?)
- Publications: Open software journals such as JOSS.

Why Open Software (Open Science) ? Don't forget license file
Societal impact : scientific vulgarisation, research valorisation
Academic career recognition: peers usage, citations…
Innovation/private sector : a start-up out of your research software!

Must: Choose a license

License

Licenses are a legal contract between authors and users of a creative work; the authors can grant different levels of permission and usability under specified conditions. The work is only open-source if it is subject to an open source license.

A license file, containing the full license, can be placed in the root folder. For common licenses, a license notice (short version of the license) is enough.

At EPFL, EPFL owns the original data & code, but the authors can use it for research and IP.

Choose a license: choosealicense.com/, Library RDM Licensing Guide, tldrlegal.com, choose-the-right-license by EPFL TTO…
Examples :
- GPL3.0 : strong open source license enforcing that copies must be open-source.
- MIT : permissive open source license allowing any re-use-

Resources

Funding:

You can find below potentially interesting funding opportunities, at ENAC, EPFL, Swiss or International level, selected by the Dean's Office due to their connection with ENAC's sustainability challenges.

For an exhaustive list of research funding opportunities, you can consult following webpages and tools of the EPFL Research office and the Swiss Federal Office for the Environment FOEN:

Research Office selection
Research Office memento (ordered by deadline)
Research Office Foundations Compendium
Research Office Collaborations Compendium
Research Professional
Overview national and international funding instruments FOEN

References:

- Guide for Reproducible Research
- Follow FAIR principles
- Code & Data mgt workshop, EPFL Library
- Awesome list of references
  List of dev resources related to every topics concerning computer science: hosted on github/open-source (code/blog/article/documentation/books..)

ENAC-IT4R

pierre.guilbert@epfl.ch /
21.06.2022 / enac-it4r