WI-Project: Open source project

Best practice session

Prof. Dr. Gerit Wagner
Faculty of Information Systems and Applied Computer Sciences
Otto-Friedrich-Universität Bamberg

Learning objectives

Our objectives for today are to discuss and overcome current challenges related to:

- Technical setup: Git, Codespaces, or local setup
- Programming: Python, Python packages, and CoLRev
- Teamwork: Task distribution, forks, branches, and roles in the team

The focus is on **helping teams organize their work effectively**. To this end, we

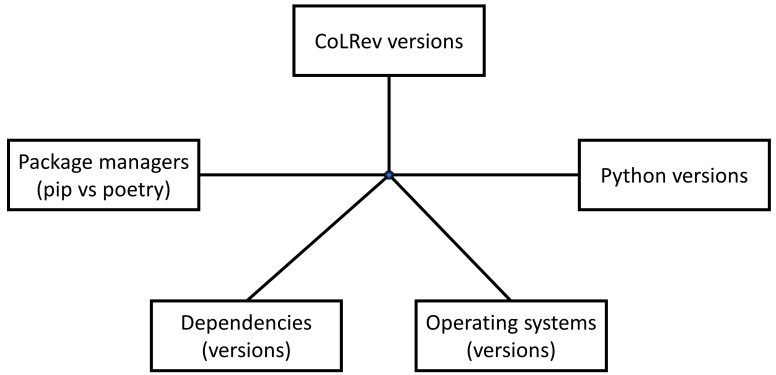
- Encourage you to share challenges, errors, or lessons learned
- Do not introduce new commands, except when you ask for them or when they are useful for your work

Technical setup

- Does everyone have a working setup?
- Are you working on Codespaces or in a local environment?
- What errors and challenges have come up?

Technical setup: Sources of errors

- Errors may arise only in specific settings, i.e., versions of operating systems, Python, package managers, dependencies, and CoLRev.
- What we do to identify and fix errors:
  - Run matrix tests covering 16 different environments
  - Reduce dependencies
  - Fix reported errors
- What you can do to avoid errors:
  - Use supported environments, such as GitHub Codespaces
  - Avoid outdated and cutting-edge versions (of operating systems and Python)
  - Report errors



Technical setup: Best practices

- Use Codespaces with the pre-installed setup
- Carefully select changes (`git add -p`) and review changes before creating a commit
- Commit often
- Synchronize regularly in your fork and with `remotes/upstream`

Technical setup: Open questions

Raise them directly or add them below (anonymously):

<iframe src="https://rustpad.io/#5YH3Wq" width="100%" height="400px" style="border:none;"></iframe>

Programming

- Are you confident with your programming environment, i.e., do you know how to modify and run your code?
- Do you know how to execute pre-commit hooks to evaluate and improve code quality?
- Do you know where and how to contribute your code, i.e., the modules and classes to use or create?
- Are there any questions related to Python packages or CoLRev?

Programming: Best practices

- Search, read, and understand your code and the code in related projects
- Check and fix code quality regularly (at least before creating a commit):

```
pre-commit run --all
```

- Understanding of CoLRev:
  - **User workflow**: Start with the video and check `gitk` after each step
  - **Architecture**: Starting point: API chart and reference and modules
  - **Package development and SearchSources**: Starting point: package development resources, CEP 003 - SearchSources

Programming: Open questions

Raise them directly or add them below (anonymously):

<iframe src="https://rustpad.io/#2Nra6z" width="100%" height="400px" style="border:none;"></iframe>

Teamwork

- How can we split and distribute tasks?
- Which branches should we set up?
- Does everyone contribute fairly? (*)

* Normally, we give **one grade per group**. We will check individual Git contributions. We can assign different grades for group members, with a **bonus** for those who contribute more.

Teamwork: Best practices (branch setup)

Recommended branch setup in your fork:

1. Work on a shared **feature branch**, such as `unpaywall_search`. This is where your latest, working version is developed.
2. Do not commit directly to `remotes/fork/main`. This branch should be kept in sync with `remotes/origin/main`.
3. Regularly merge `remotes/origin/main` into `remotes/fork/main` and `remotes/fork/main` into your feature branch using merge commits (i.e., sync, which will fast-forward, `git fetch`, `git switch feature_branch`, and `git merge main`).

Optional: Merging into a target branch, i.e., your shared feature branch:

- Squash if you have worked on a single coherent task, which should be combined in a single commit
- Rebase if you would like to preserve a simple linear history
- Merge commit otherwise

Optional: When tasks are distributed, and you work alone, work in local non-shared branches (e.g., `api_retrieval`):

- Rebase on (parent) feature branch to keep your branch "up-to-date" (`git rebase unpaywall_search`)
- Once the branch is online, use merge commits

Teamwork: Open questions

Raise them directly or add them below (anonymously):

<iframe src="https://rustpad.io/#WjqD80" width="100%" height="400px" style="border:none;"></iframe>

Group work phase

After the introductory sessions, the stage is yours.

- You have six weeks to complete the project.
- Remember: We are here to support you!
- To discuss your plans, current challenges, and next steps, schedule a hacking session:

  Schedule a meeting

> In the last semesters, each group scheduled 3-5 hacking sessions.