Good practices III

Raul P. Pelaez

March 10, 2026

Contents

1 Introduction

In the previous lesson, we learned about Git and version control. Now, we will explore how to share and collaborate on projects using GitHub, a popular platform for hosting Git repositories. We will also cover GitHub Actions, a tool for automating tasks like testing and deployment. Finally, we will discuss best practices for publishing your project on GitHub, including writing a good README and generating documentation with Sphinx.

2 GitHub

GitHub is an online platform that hosts Git repositories and provides collaborative features such as pull requests, issue tracking, and project management tools.

2.1 Setting up your SSH key for GitHub access

In order to be able to push changes to your GitHub repositories, you need to register your laptop with GitHub. This is done by providing GitHub with the SSH key of your laptop.

Info

SSH keys
SSH keys are a secure way to authenticate with remote servers. They consist of a public key (which you share) and a private key (which you keep secret). When you connect to a server, the server checks your public key to verify your identity.
Generate a new SSH key:

```
$ ssh-keygen
# Click enter many times
```

This will create a new SSH key pair in your `~/.ssh` directory. If you inspect this folder, you will see something like:

```
$ ls ~/.ssh
id_rsa  id_rsa.pub
```

The `.pub` file is your public key, which you can share with GitHub. run `cat ~/.ssh/id_*.pub` to see the content of the public key.

After generating and copying your SSH key, enter it in your GitHub account settings. Now, you will be able to use `git push` to send changes to your repositories.

2.2 Sharing and collaborating with GitHub

Info

Forks
A fork is a copy of a repository that you can modify without affecting the original. It's useful for contributing to open-source projects or experimenting with changes. You can fork any public repository on GitHub through its webpage.

Info

Pull requests
A pull request (PR) is a way to propose changes to a repository. You can open a PR from a branch in your fork to the original repository, or even between branches of the same repo. The maintainers can review the changes, ask questions, and suggest improvements before merging them.
See here an example PR that was merged into a project. As you can see, the PR contains a description of the changes, a list of commits, and a discussion thread. After everyone agrees that the changes/new functionality are ready, the PR is merged into the main branch.

Info

Workflow with remote repositories on GitHub

1. Create or fork new repository on GitHub through its webpage.
2. Link your local repo to the remote:

```
git remote add origin git@github.com:username/myproject.git
git push -u origin main
```

   If there is no local copy yet, you can clone the remote repo instead:

```
git clone git@github.com:username/myproject.git
```

3. Collaborate: other team members can clone, create branches, and open pull requests.
4. Review and merge changes: maintainers review pull requests before merging into main.

The syntax for the URL `git@github.com:` tells git to use SSH for authentication, which is required to push changes to GitHub repos.

Advice

Pull requests (PRs) & code reviews

A good PR:

- Has a clear title and description
- Focuses on a single feature or fix
- Includes tests (if relevant)
- Passes automated checks (linting, tests, etc.)

Using code reviews fosters communication and knowledge sharing among contributors.

Advanced

Continuous integration (CI) & continuous deployment (CD)

CI is the practice of automatically testing code changes to catch bugs early. CD is the practice of automatically deploying code changes to production. GitHub Actions can be used for both CI and CD.

For instance, most projects use a CI script that runs at every single commit to the main branch. This script runs tests, checks code style, and so on. If the script fails, developers are notified that something went wrong and they need to fix it.

Advanced

GitHub Actions

GitHub Actions lets you automate tasks such as testing, building, or deploying your software. It runs workflows triggered by events (like pushing to a branch or creating a pull request). Actions are defined in special YAML files stored in the `.github/workflows` directory of your repository. GA are essential for ensuring code quality and reliability in collaborative projects. See a small example.

In GitHub, a red cross or a green tick next to a commit indicates whether the CI checks passed or failed.

2.3 Publishing your project on GitHub

Publishing is as simple as pushing your repository to GitHub and making it public. However, there are expectations from the community when they land on your repo.

When facing a new (and probably large) codebase for the first time, you will often face the same set of challenges:

1. How do I install this?
2. How do I run this, or How do I import this library?
3. How do I test this?
4. What set of features/functions/classes/utilities does this library provide exactly?

When you publish your project, you should address these questions in your README and documentation, trying to put yourself in the shoes of a newcomer to your project. One truly learns this by experience, after facing many projects and trying to understand them.

Info

What people expect to see

1. A clear, concise `README.md`.
2. License information (e.g., `LICENSE` file).
3. Basic instructions on how to install, test, and run your project (often in the README).
4. Some form of examples, either as scripts in an `examples/` folder or as snippets in the README.
5. Organized directory structure (code in `src`, tests in `tests`, etc.).
6. Optionally, continuous integration checks (GitHub Actions) showing build/test passing status.
7. Optionally, documentation (e.g., Sphinx-generated) in a `docs/` folder.

Advice

Tests and CI scripts are also documentation

Often, you will find projects out there have poor or non-existing documentation, outdated examples, poor installation instructions, and so on and so forth. One good way to get familiar with the project is to look at how the tests are written, which will give you some usage examples. On the other hand CI scripts, by construction, must compile/install/build the project in order to run, which requires installing dependencies. You can use this information to understand how to set up and use the project.

2.4 Writing a pretty GitHub README.md

The `README.md` is the first impression of your project. Make it clean and informative. If present in the root of your repo, GitHub will render it on the main page of your project.

Info

Markdown: The README.md format

Markdown is a lightweight markup language with plain text formatting syntax. It is widely used for READMEs, documentation, and other text files. Checkout README.md files around to copy what you need. For instance:

```
# Heading
## Subheading
### Subsubheading
**Bold text**
[A link](https://github.com)
- A list
```

Info

Good README structure

- Project title
- Short description of what your project does
- Installation instructions
- Usage examples/Quick start
- Contributing guidelines, license and citation (if applicable)

Advice

I like READMEs to be concise and to the point. For instance, IMO this is a perfect README, this one is also cool.

3 Automatic documentation with Sphinx

Sphinx is a documentation generator that creates webpages (HTML), PDFs, or other outputs from reStructuredText source files. One of its killer features is that it will automatically extract the docstrings from your Python code and include them in the documentation.

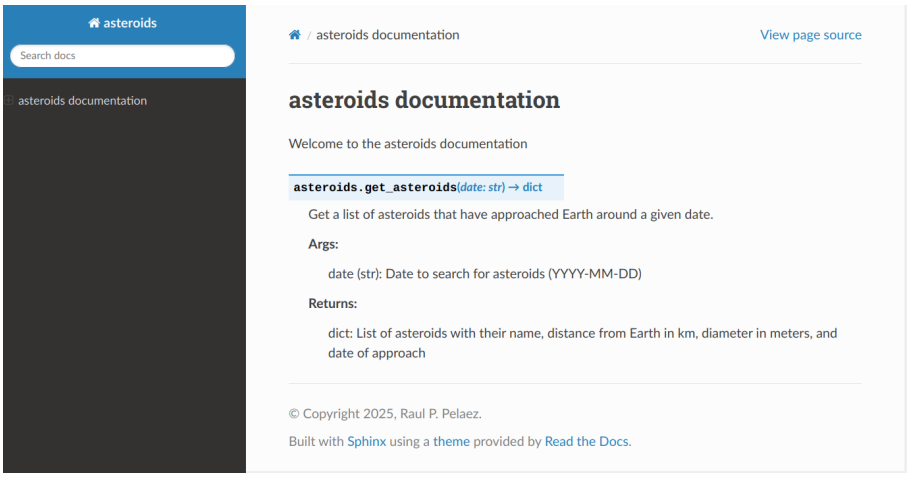
For instance, using the project from last session's exercises, this file, called `index.rst`:

```
  asteroids documentation
=======================

Welcome to the asteroids documentation

.. autofunction:: asteroids.get_asteroids
```

Will be used by Sphinx to generate the following webpage:



Info

Getting started with Sphinx

1. Install Sphinx (you can add this dependency to environment.yml instead):

```
$ pip install sphinx
```

2. Go to a new directory called `docs/` in your project and initialize sphinx:

```
$ sphinx-quickstart
```

   This will ask you some customization questions and then generate a bunch of files.
3. Write docstrings in your Python code.
4. Build the docs:

```
make html
```

You will end up with a `docs/` directory containing your documentation.

Sphinx uses an `extension` system to add new features, some of them being built-in. For instance, the `sphinx.ext.autodoc` extension will automatically generate documentation from your docstrings. You can add this extension to your `conf.py` file like this:

```
extensions = ['sphinx.ext.autodoc']
```

Adding that to `conf.py` will allow you to use the directives like `autofunction` in your `.rst` files, as shown above.

Advanced

Hosting your documentation in ReadTheDocs

RTD is a widespread platform for hosting documentation. It can automatically build your documentation from your GitHub repository and host it online. Once you have setup Sphinx and are able to build the webpage locally, adding your project to RTD is as simple as creating an account, linking your GitHub repo, and configuring the build settings.

4 Exercises

Goal

Lets try to get our repos as close as possible to a professional project. We will also explore the collaborative features of GitHub.

Milestone

Setup your SSH key

Generate a new SSH key and add it to your GitHub account. This will allow you to push changes to your repositories.

Milestone

Create a new repository in GitHub
Create a new empty repository on GitHub. You can name it `myrepo` or something similar. Then, add it as a remote in your local repository and push the main branch to it.

hint

- Use the `git remote add origin git@github:username/myrepo.git` command to add the remote.
- Use the `git push -u origin main` command to push the main branch to the remote.

Milestone

Pull request
In pairs or groups of three, you must fork each others repositories, create a new branch, make a change (e.g., add a new function, fix a bug, etc.), and open a pull request to the original repository. The PR should include a clear description of the changes and a list of commits.
The owner of the original repository should review the PR, ask questions, and suggest improvements before merging it.

Milestone

Give your project a good README.md. Try to put yourself in the shoes of a newcomer to your project and answer the following questions:

1. What does your project do?
2. How do I install it?
3. How do I run it?

Advanced milestone

Add GitHub Actions
Add a basic GitHub Actions workflow to your repository. This workflow should install your project and run the tests.

1. Inside the `myrepo` folder, create a new folder: `.github/workflows`.
2. Add a file `ci.yml` with a basic workflow that installs a Python version and runs a (very simple) test command, e.g. `pytest` or `python hello.py`.
3. Commit and push. Verify on GitHub that Actions run and pass.

hint

You can copy-paste and modify this one to get started. If it does not run, change the branch name here.

Advanced milestone

Generate documentation with Sphinx

1. Add Sphinx to your environment file, recreate it so that it is included.
2. Use `sphinx-quickstart` to initialize the documentation, modify the `conf.py` as needed.
3. Run `make html` to build the documentation.
4. Open `docs/build/html/index.html` in a browser to view your documentation.