DSL

Python packaging, Part 1

Guillaume Witz
Data Science Lab, University of Bern

Program

Morning 9.00 - 12.00

- Creating a package
- Testing code

Lunch 12.00 - 13.00

Afternoon: 13.00 - 17.00 (or whenever I run out of material)

- Release on PyPi
- Automated testing and release

The Python scientific ecosystem

... A **programming language** and a way to execute it ...

... **Specific tools** for our scientific task (packages) ...

... A solution to **interact** with code and data ...

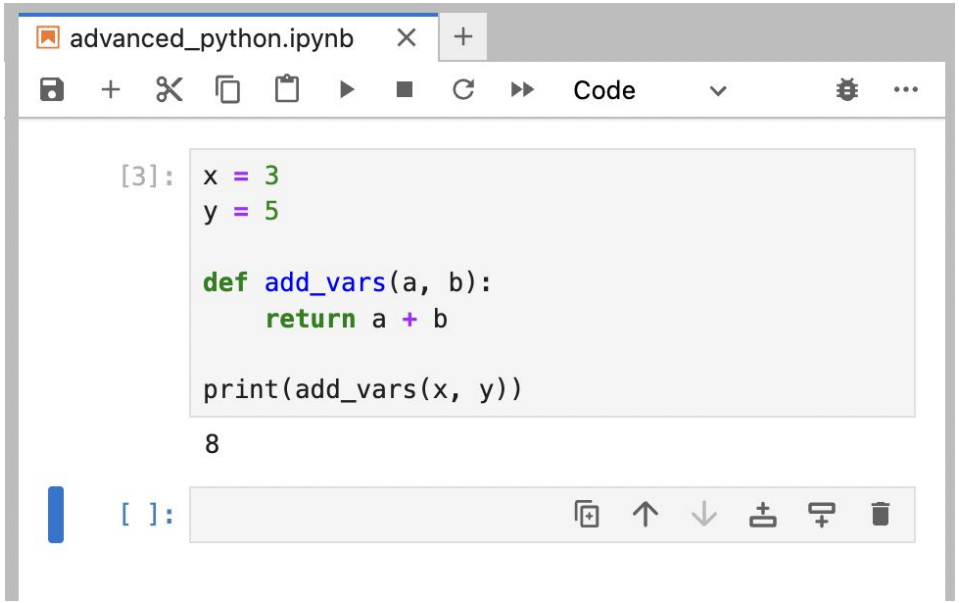
... **Infrastructure** to install and run the code

DSL

Executing scripts / modules

Executing Python

Interactively e.g. Ipython, notebook, command line



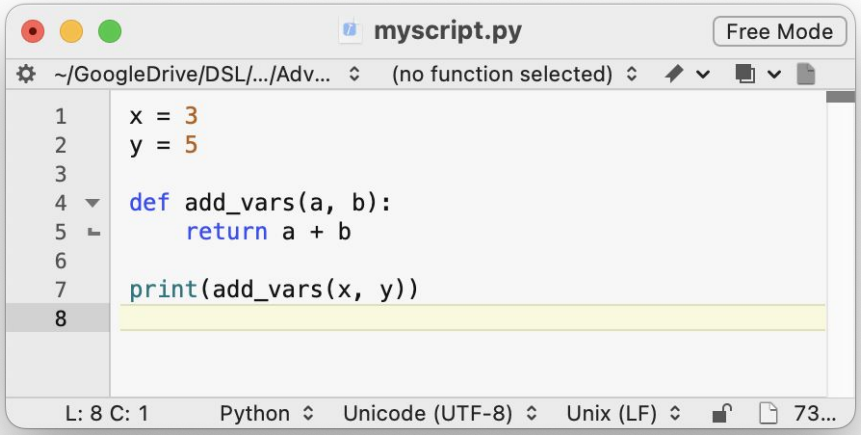
Executing Python

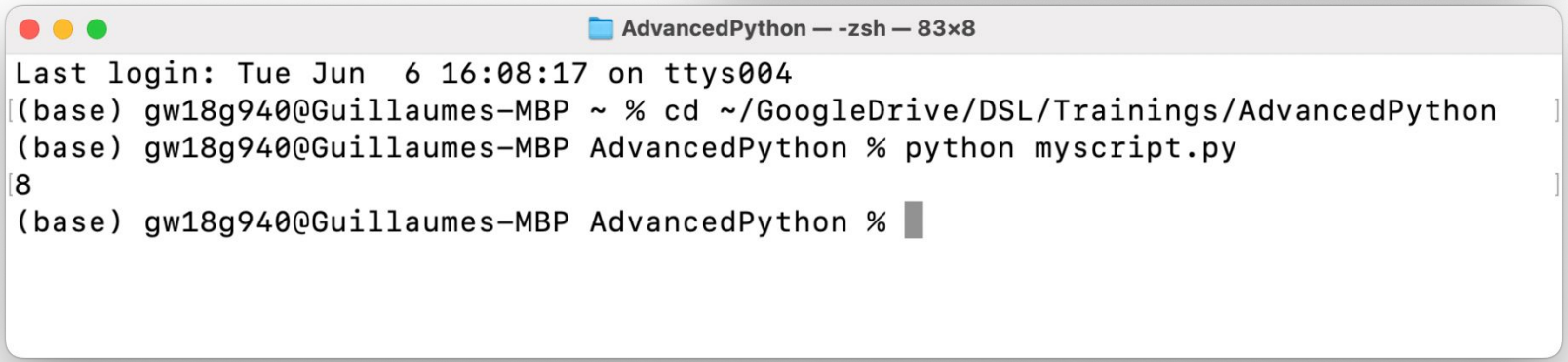
Interactively e.g. Ipython, notebook, command line

Executing Python

Interactively e.g. Ipython, notebook, command line

Write a script, execute in terminal



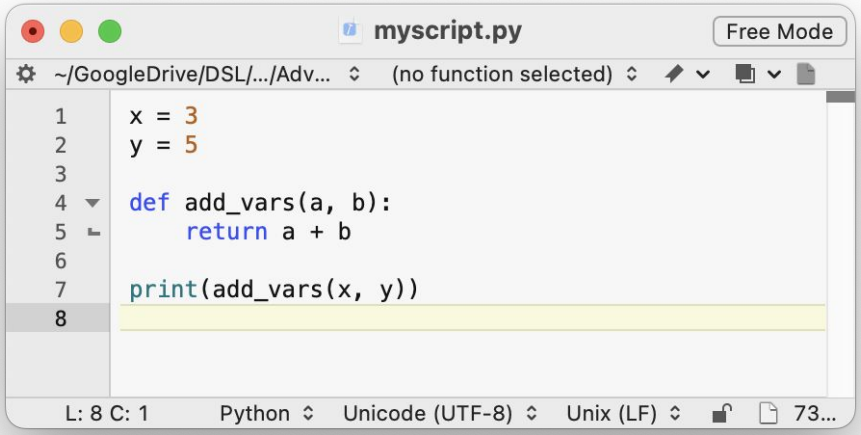


Executing Python

Interactively e.g. Ipython, notebook, command line

Write a script, execute in terminal

Import the script as a module: problem: everything gets executed at import time




```
[4]: import myscript
     8

[5]: myscript.add_vars(3,4)

[5]: 7
```

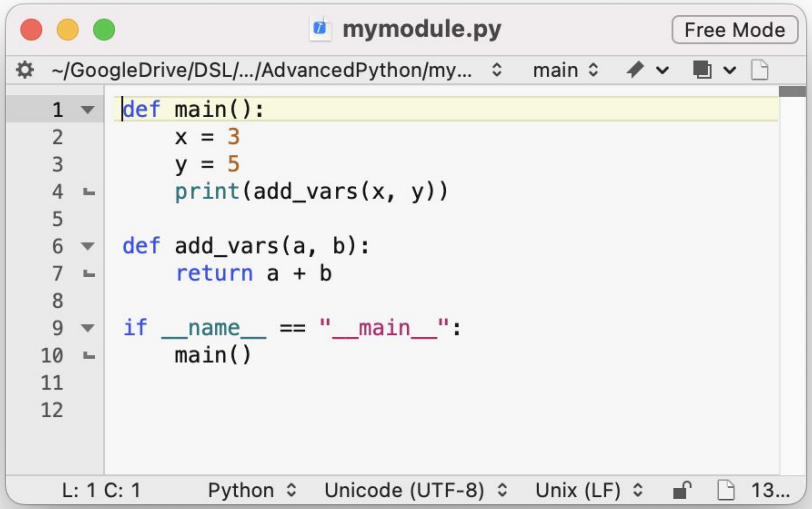
Executing Python

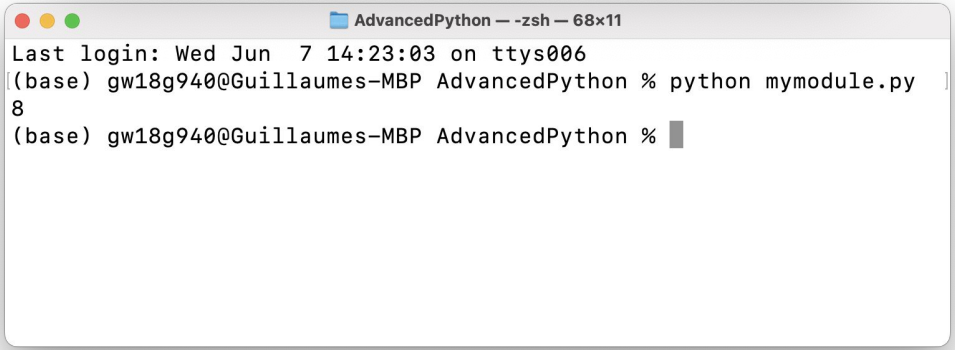
Interactively e.g. Ipython, notebook, command line

Write a script, execute in terminal

Import the script as a module: problem: everything gets executed at import time

Adding a main() file and checking the context of execution (__name__) executes main() only when used as script





Executing Python

Interactively e.g. Ipython, notebook, command line

Write a script, execute in terminal

Import the script as a module: problem: everything gets executed at import time

Adding a main() file and checking the context of execution (__name__) executes main() only when used as script

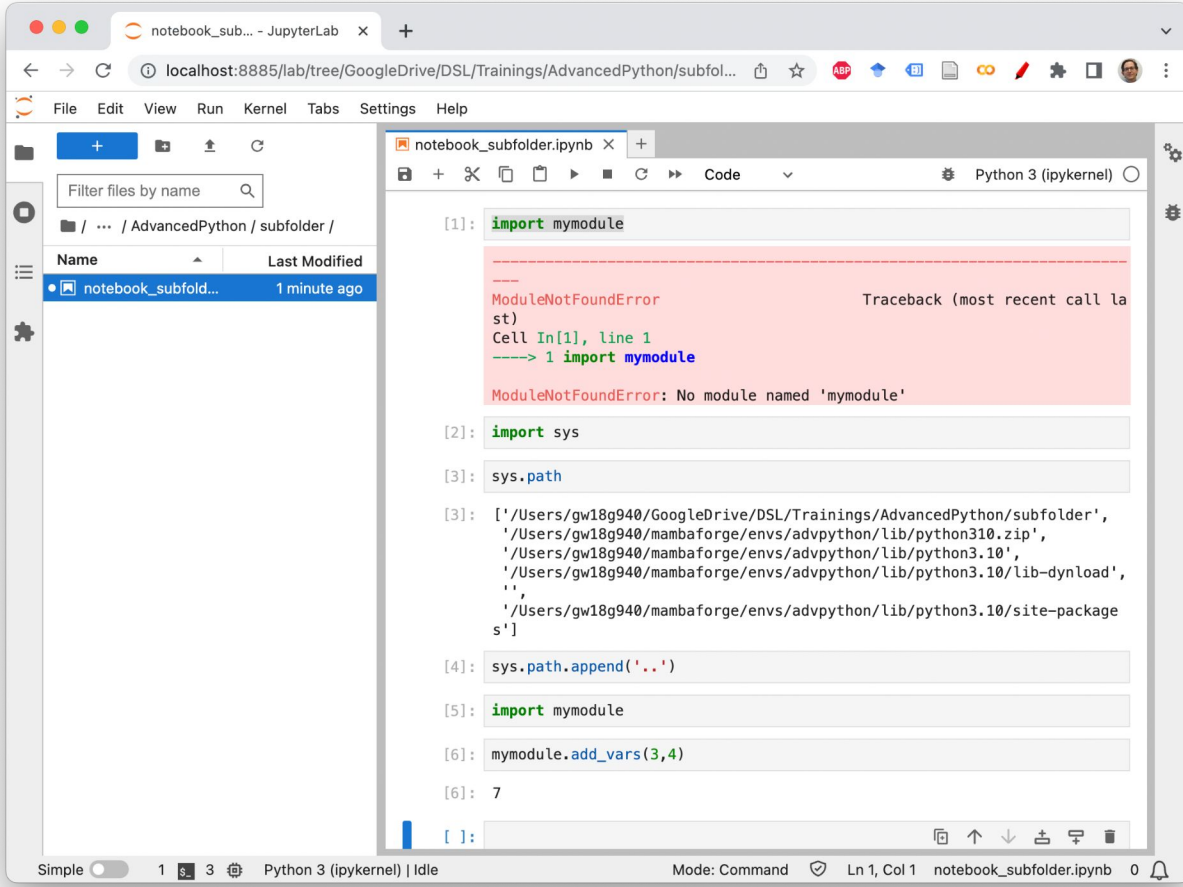
```
[4]: import myscript

     8

[5]: myscript.add_vars(3,4)

[5]: 7

[6]: import mymodule

[7]: mymodule.add_vars(3,4)

[7]: 7
```

First problem

Your module is not in the same folder as your current working directory: unfindable

Possible to add to path

Better solution…



Beyond scripts: packaging

More than one module?

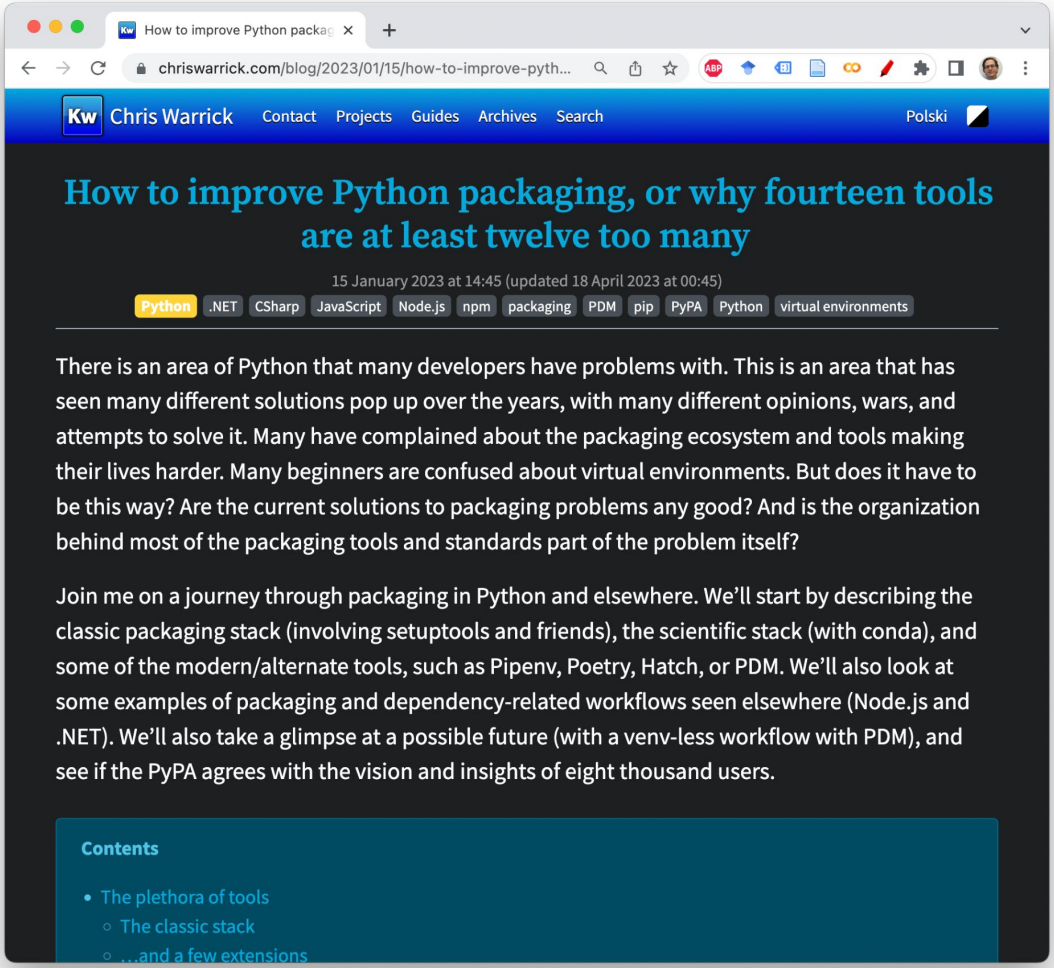
Re-use in other folder? Re-use by other people?

Sure that someone using the module has all required dependencies?

Want to test your code on multiple platforms, with multiple Python versions?

DSL

Packaging and distributing (it's a mess 😱)



https://chriswarrick.com/blog/2023/01/15/how-to-improve-python-packaging/

Packaging and distributing

```
pyadvcourse
├── pyproject.toml
└── src
    └── pyadvcourse
        ├── algos.py
        ├── __init__.py
        └── utils
            ├── compute.py
            ├── ui.py
            └── __init__.py
```

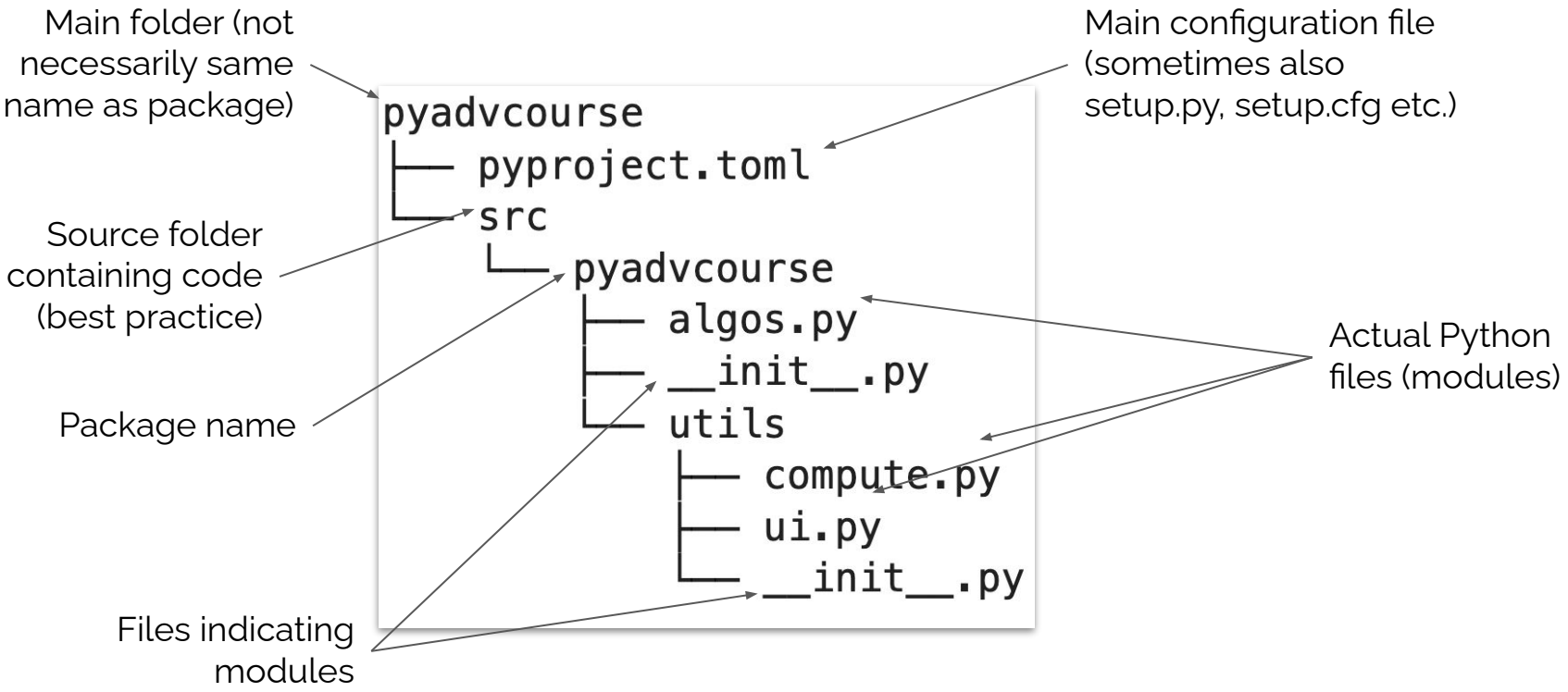
Folder with modules, infos about package and dependencies

Building process

Built package to be:

- used by others
- uploaded to PyPi

Package structure

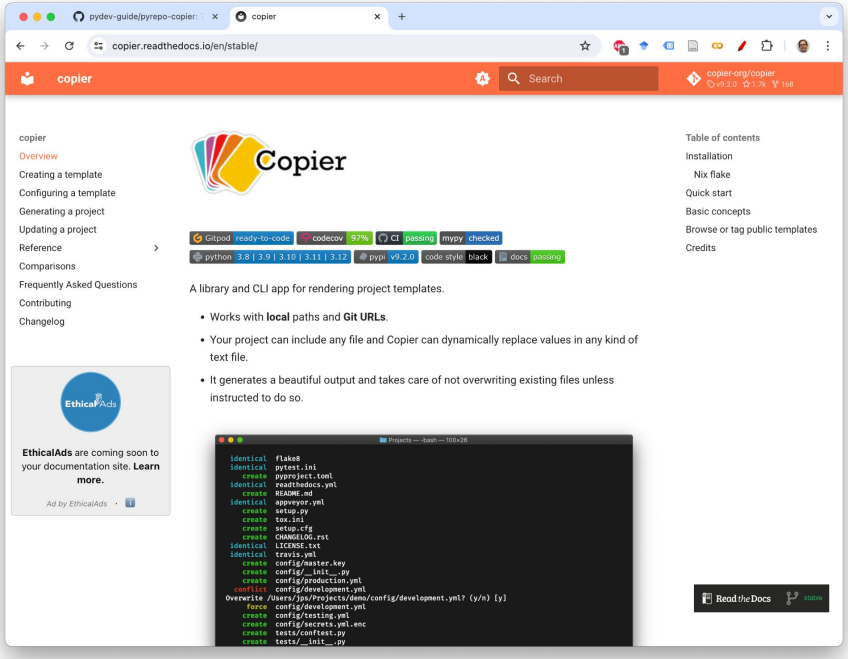


DSL

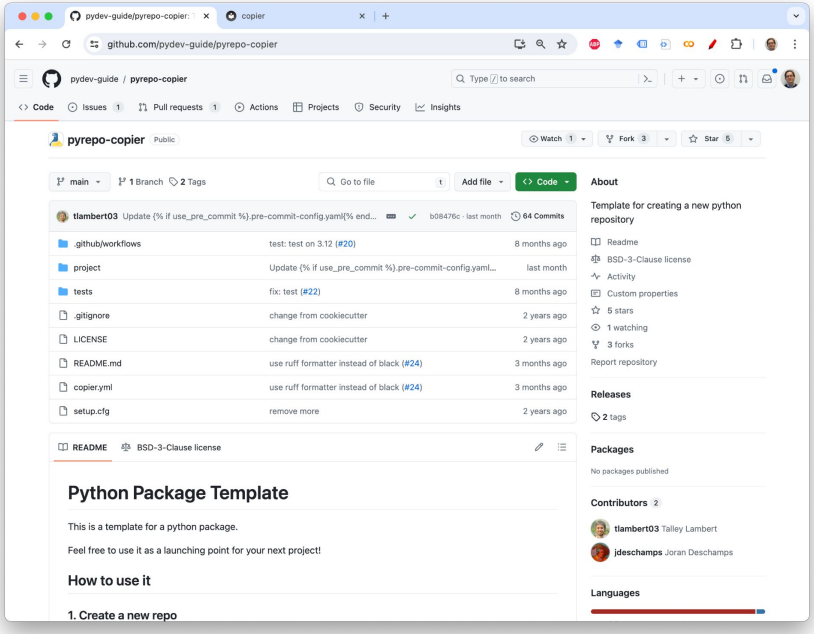
Creating a package

Instead of doing this by hand: cookiecutter

Tools like cookiecutter and copier allow to create entire projects populated with relevant files with good defaults.



https://copier.readthedocs.io/en/stable/



https://github.com/pydev-guide/pyrepo-copier

Create a package

Create an environment and install the cookiecutter package:

```
conda create -n mypackaging python=3.10 jupyterlab copier -c conda-forge
conda activate mypackaging
```

Then move to where you want to create the folder containing your package

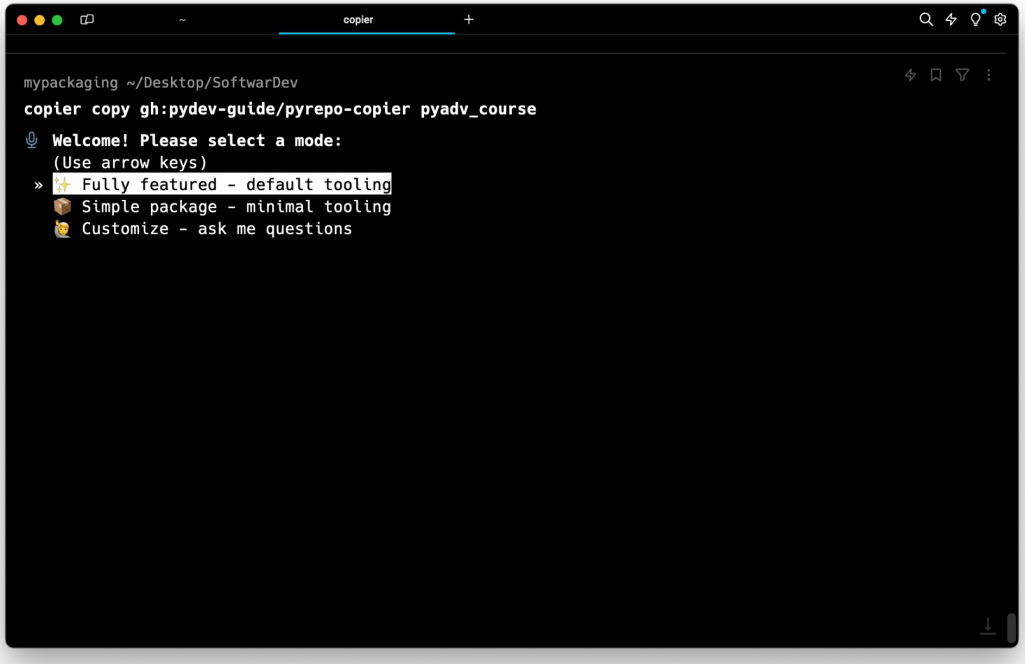
```
cd /to/my/favorite/place
```

Use copier with pyrepo-copier:

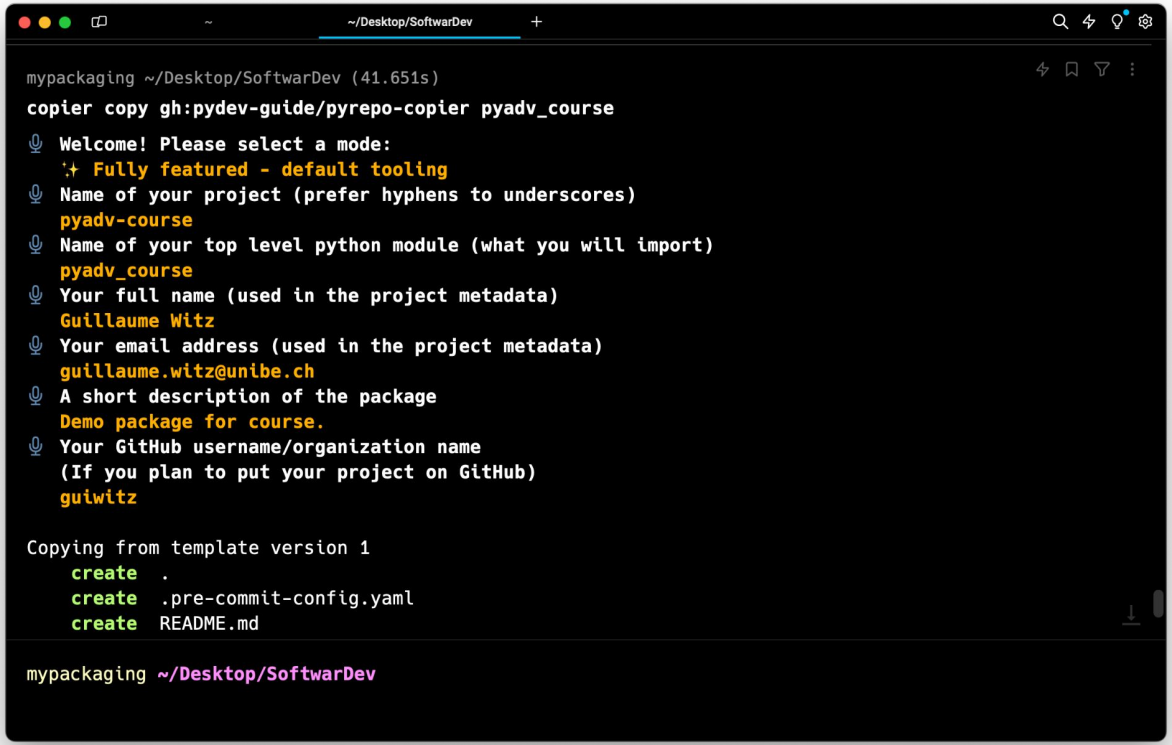
```
copier copy gh:pydev-guide/pyrepo-copier your-package-name
```

copier package

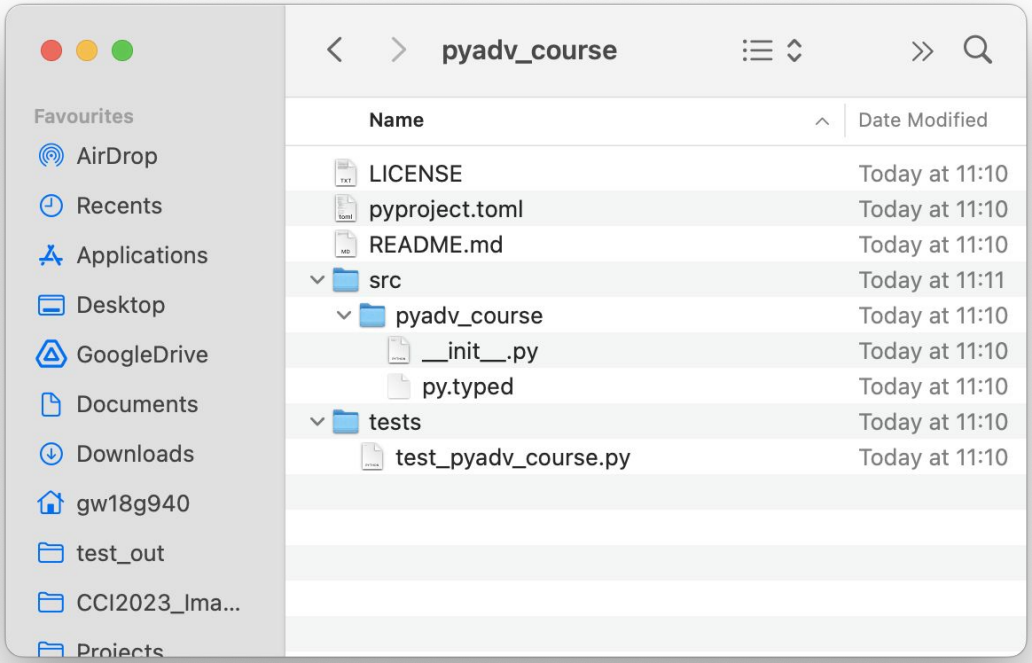
copier walks you through package creation by asking you a set of questions.
Let's use "Fully featured" even if we won't be using everything.



copier package



copier result



Tour of contents

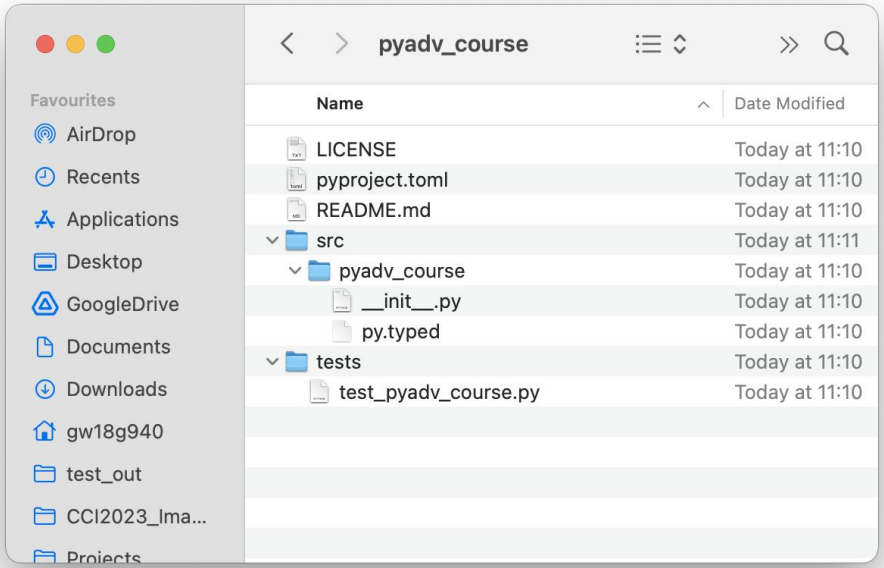
LICENSE: Markdown file to edit

README.md Markdown file to edit as you want

src: folder with your code. Empty for now

tests: single python file with pass function

pyproject.toml: package configuration



More infos on building

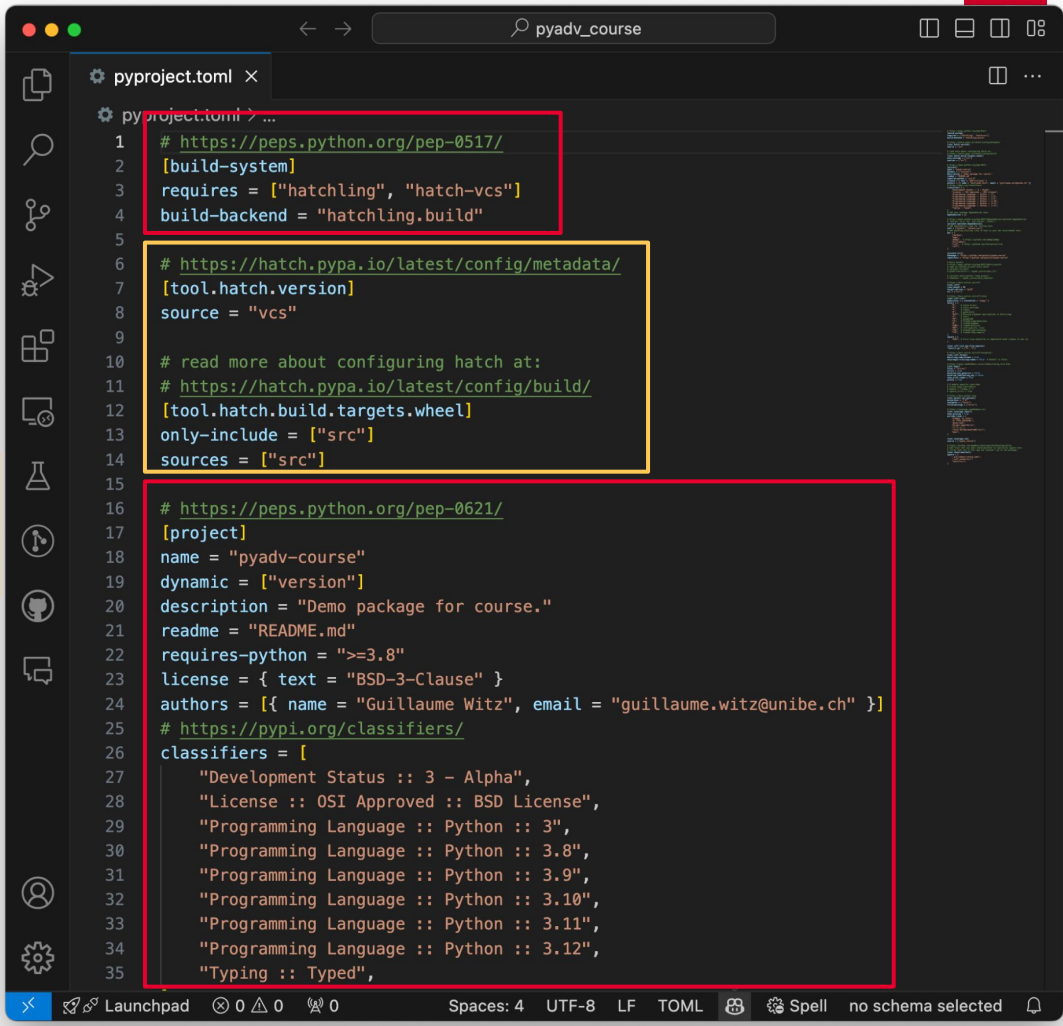
https://www.pyopensci.org/python-package-guide/package-structure-code/python-package-build-tools.html

pyproject.toml

Sections of the file

Some sections are necessary (build-system, project)

Some depend on the tooling used (tool.hatch.version)



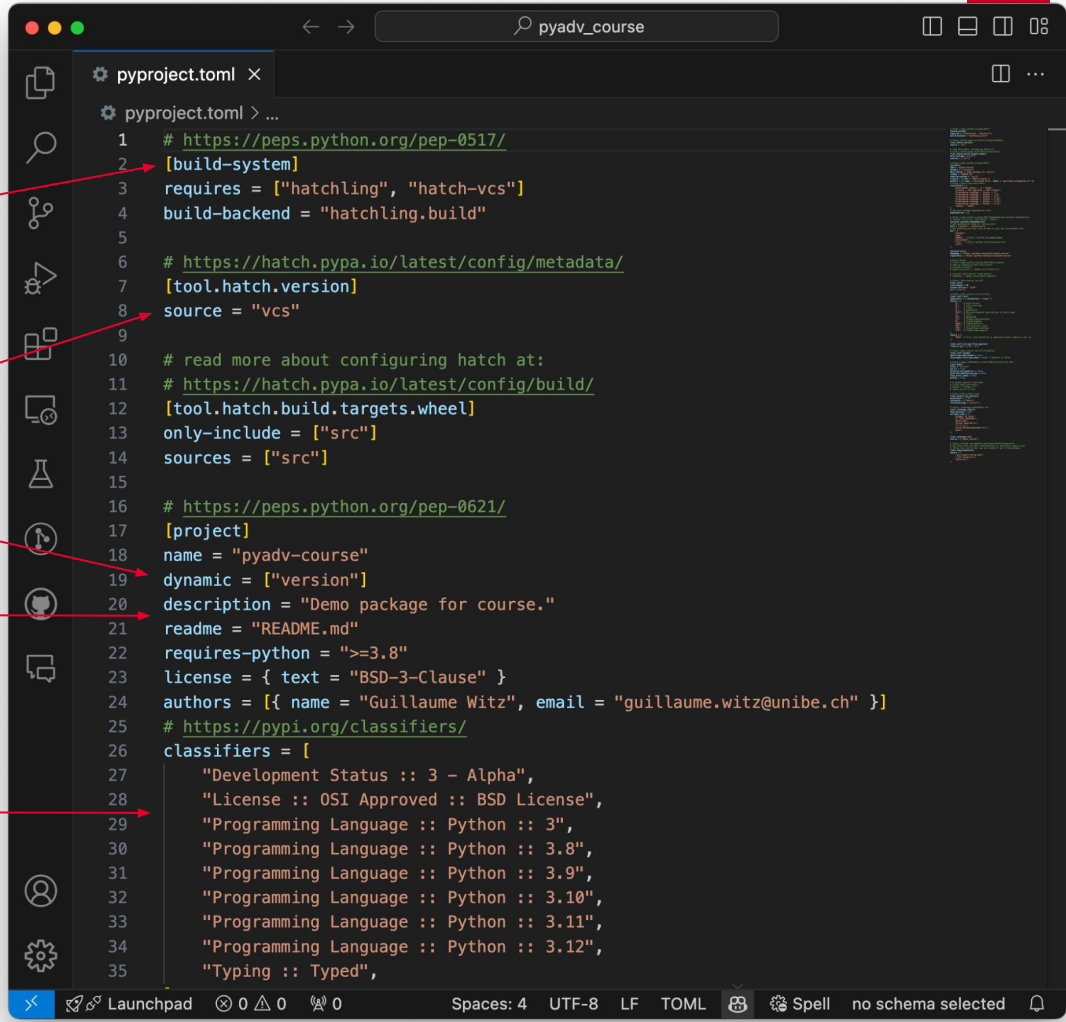
pyproject.toml

Specifies the software that builds to package. Hatch should be the future…

Versioning done automatically via git tags (more later)

Common metadata

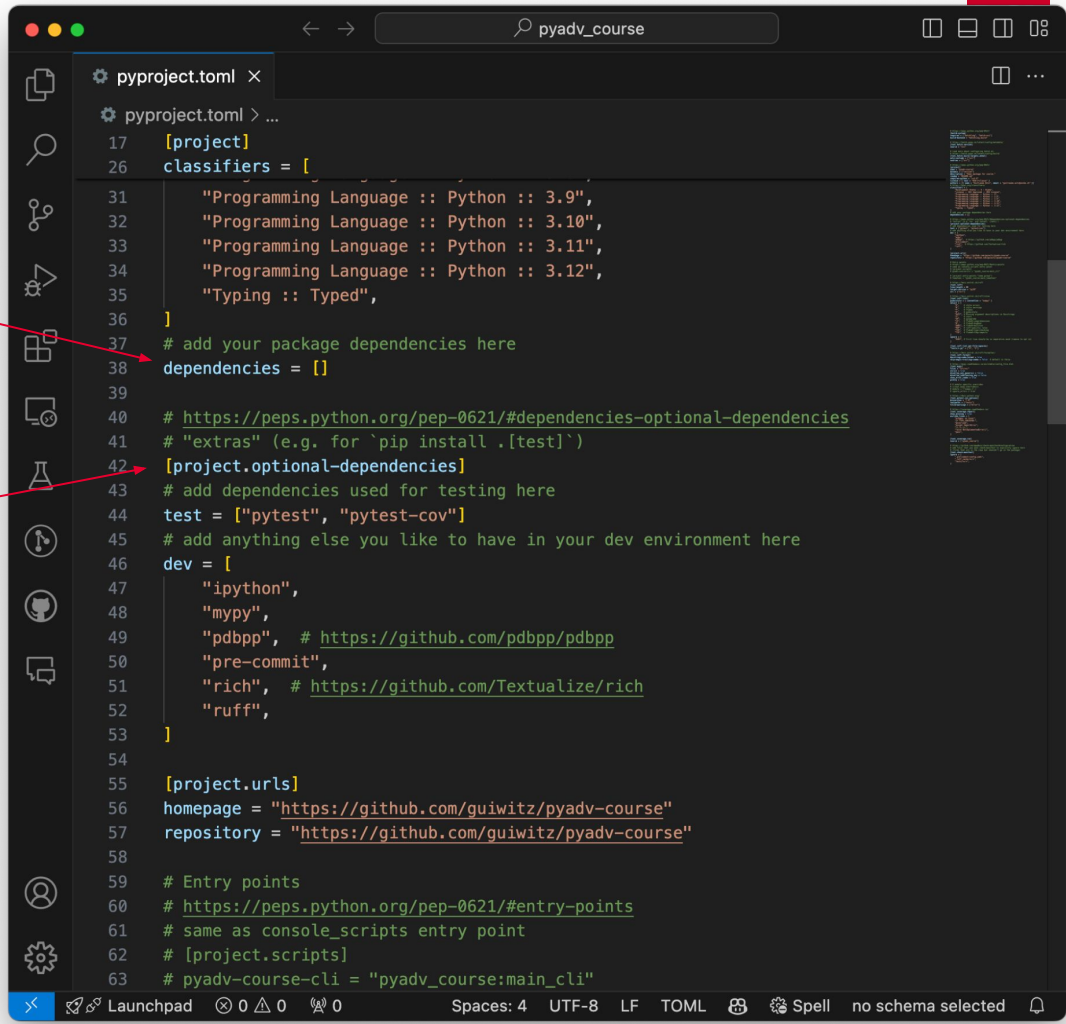
Making plugin findable on PyPi (e.g. napari plugin)



pyproject.toml

Dependencies

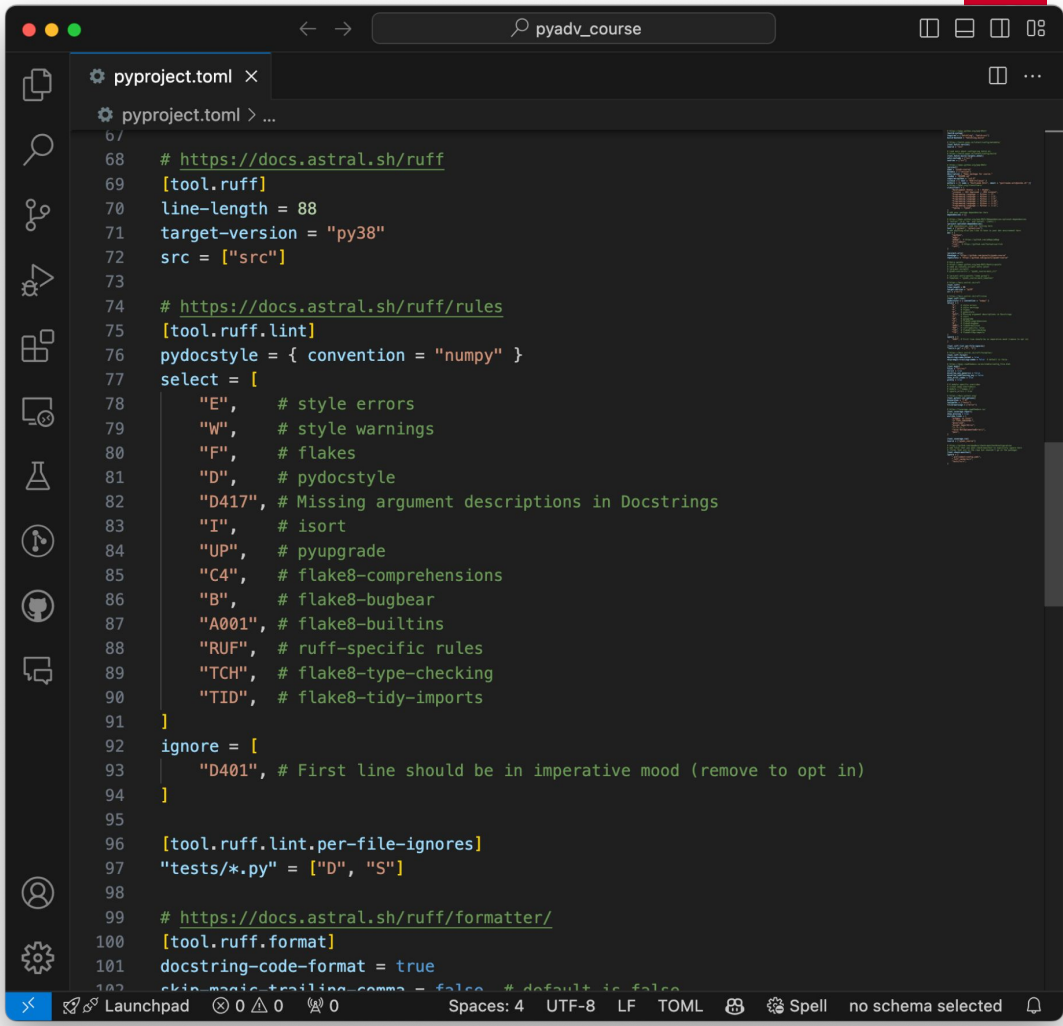
Optional dependencies, e.g.
`pip install aicsimageio[nd2]`



pyproject.toml

Additional sections for:

- formatting (ruff), absent for simpler package mode
- testing (pytest, coverage)
- code typing (mypy)



DSL

Push to GitHub

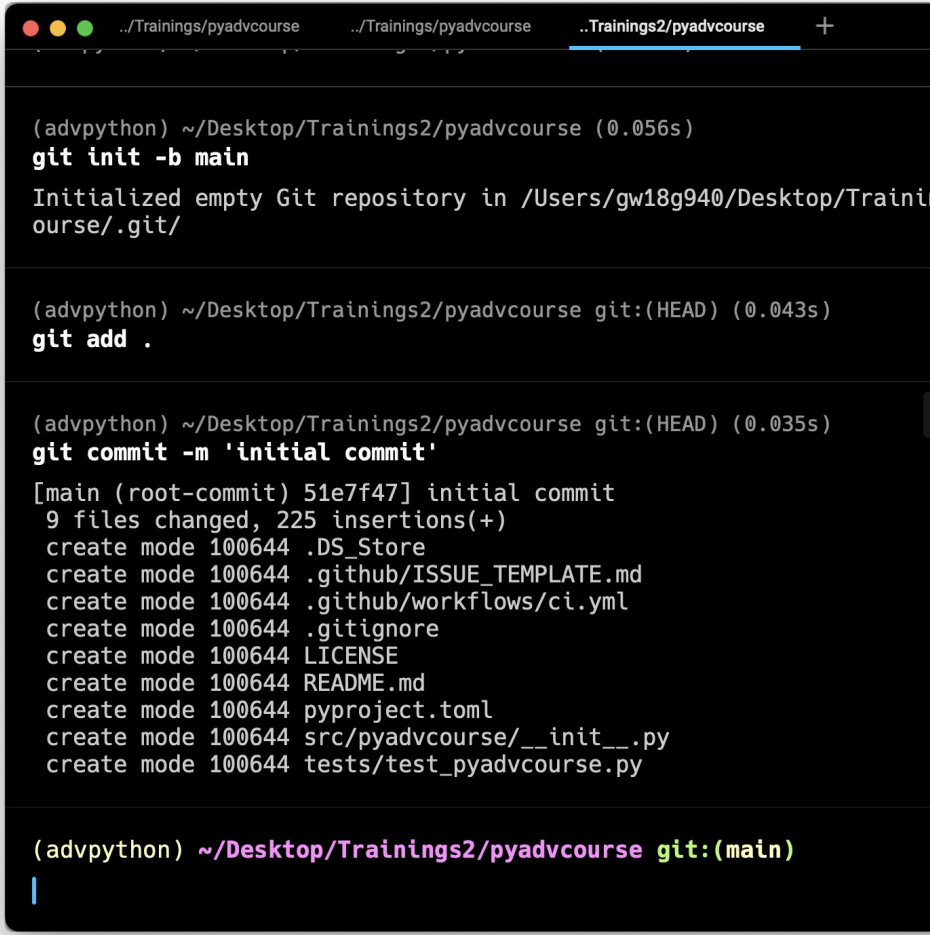
Initialize git

Eventually we want to push to GitHub and use GitHub services

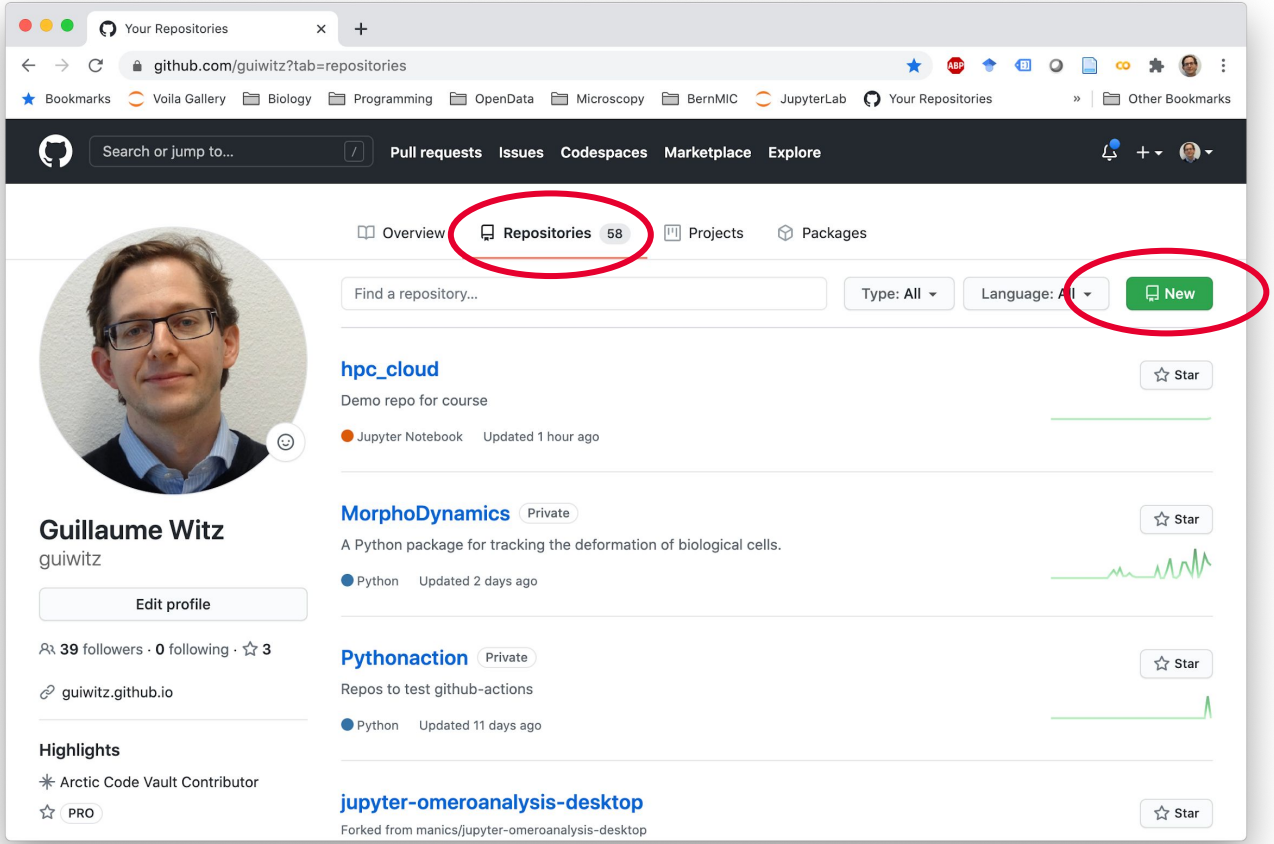
So we need to 1) initialize git, 2) connect to GitHub

Inside your project folder use:

```
git init -b main
git add .
git commit -m 'initial commit'
```



Create a repository on Github



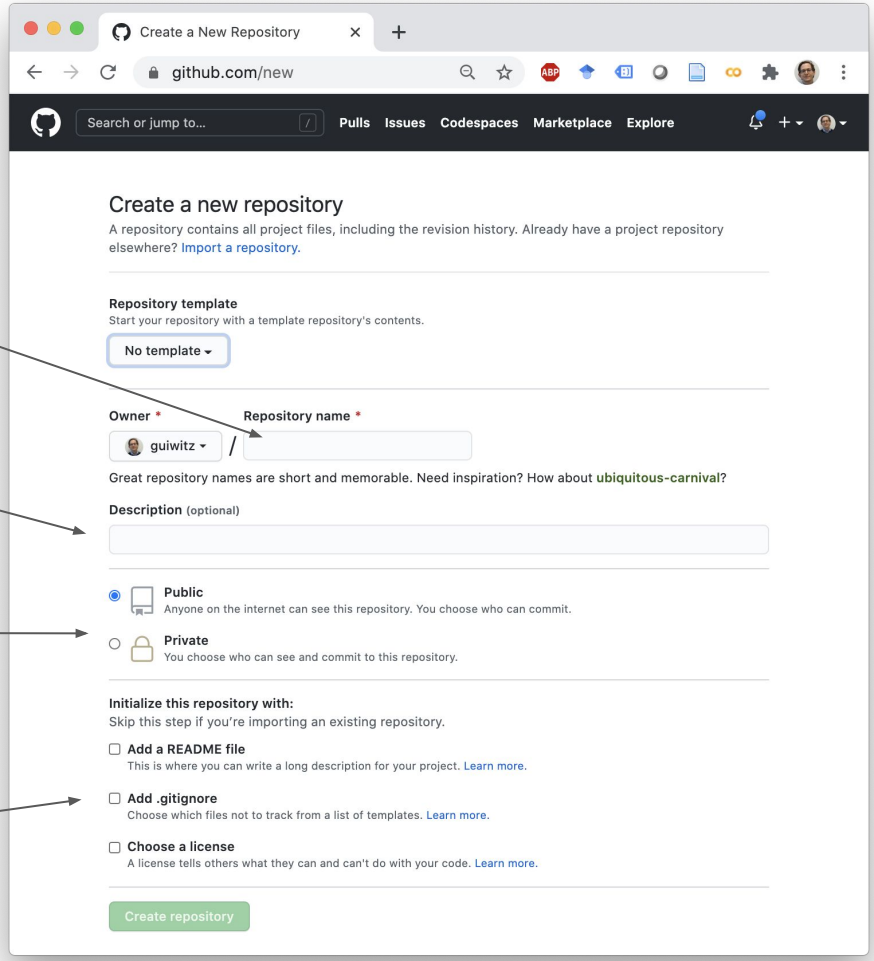
Create a repository

Choose the same name as your project: pyadv-course

Add a short description

Make your repository public. You can keep not-yet released projects (e.g. article) private

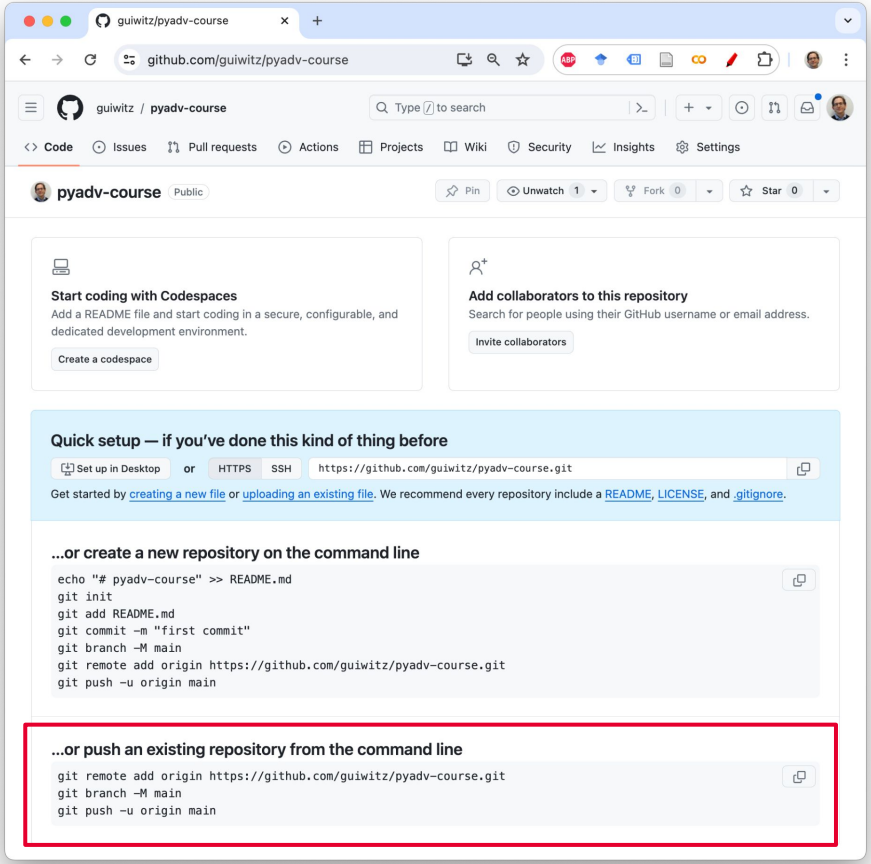
Add NO files



Create a repository

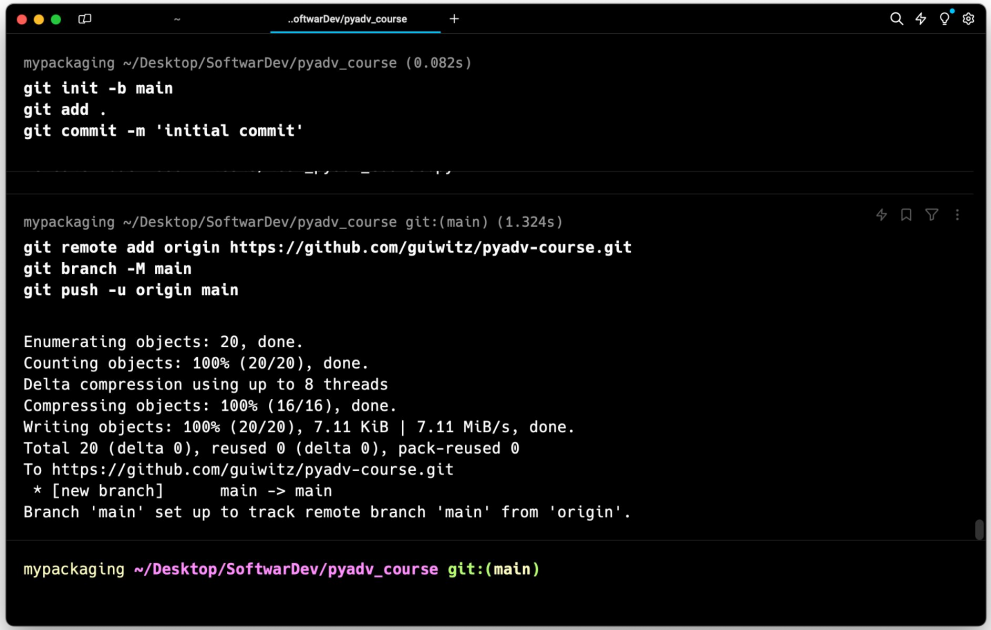
Connect your local repository to the GitHub repository

You will be prompted to login to GitHub in one way or another…



Connect your local project to GitHub

```
git remote add origin https://github.com/guiwitz/pyadv-course.git
git branch -M main
git push -u origin main
```



DSL

Install project

Editable install

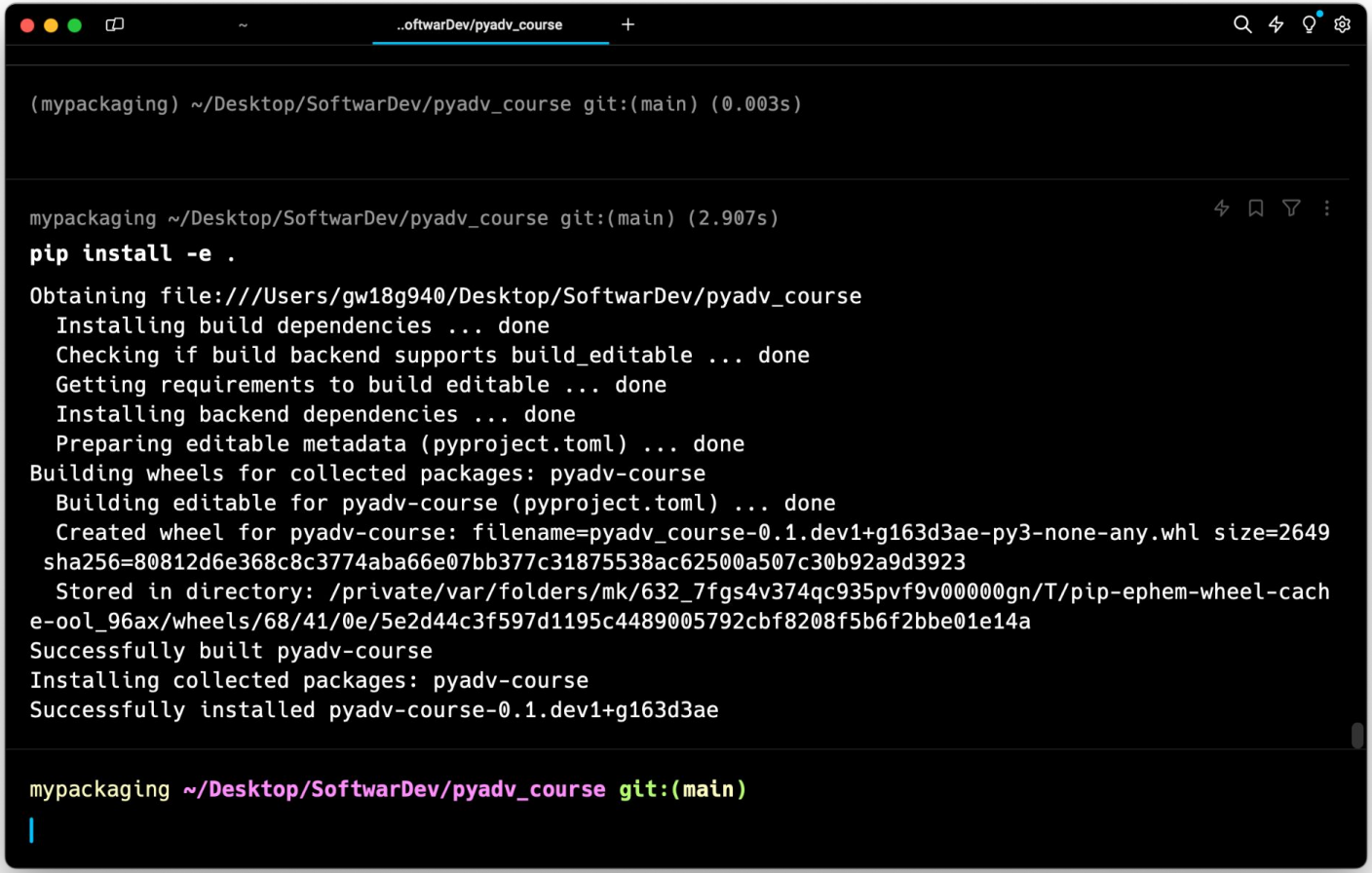
When you install a package with `pip install mypackage`, the code is put in a place where it should not be edited.

When you work on a project you want to be able to edit it.

You can install in editable mode i.e. the package you use (e.g. `import mypackage`) is the one that you can edit.

For this, cd to your package (level of `pyproject.toml`) and use (don't forget the point): **`pip install -e .`**

Editable install



Use your package

Start Jupyter:

jupyter lab

Create a notebook and try to import your package.

You should NOT get an error but… the package is empty

It has a version though!

```
[1]: import pyadv_course

[2]: pyadv_course.__version__

[2]: '0.1.dev1+g163d3ae'
```

Edit your package

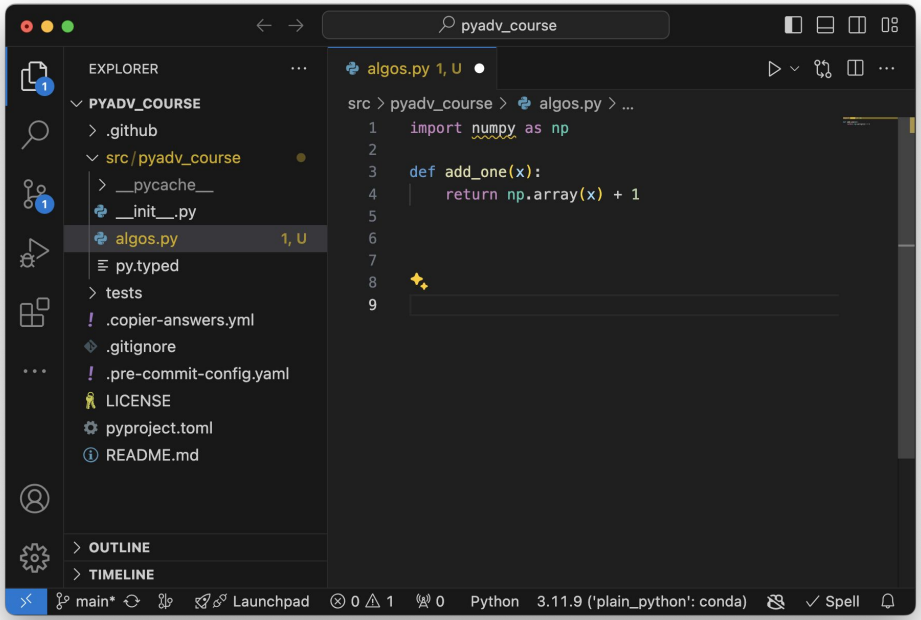
Open the package in VSCode

Open Folder -> pyadv_course

Let's add an `algos.py` file inside `src`

And let's add some dummy code:

```
import numpy as np


def add_one(x):
    return np.array(x) + 1
```

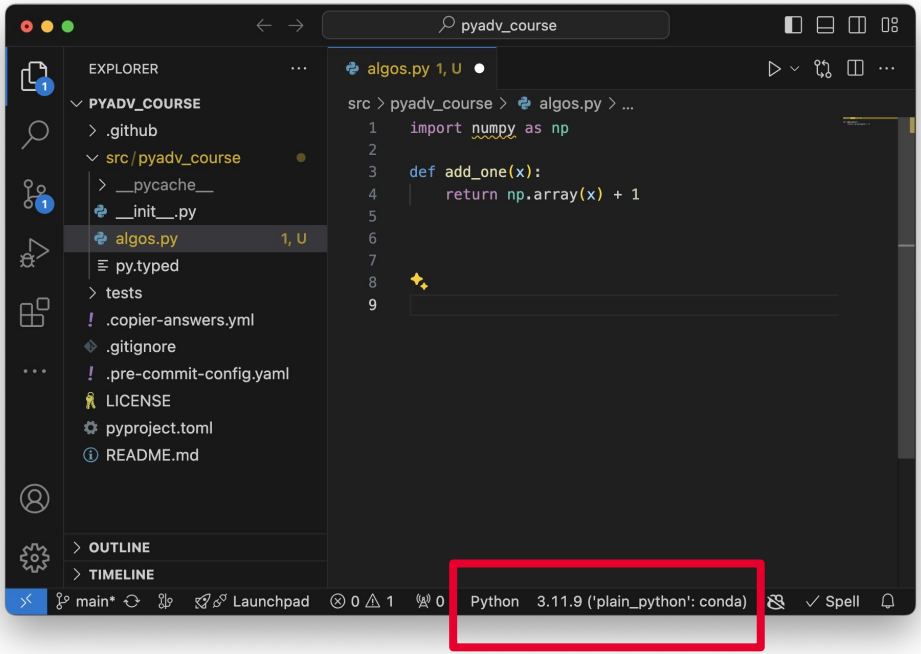


Open the package in VSCode

Let's fix the environment

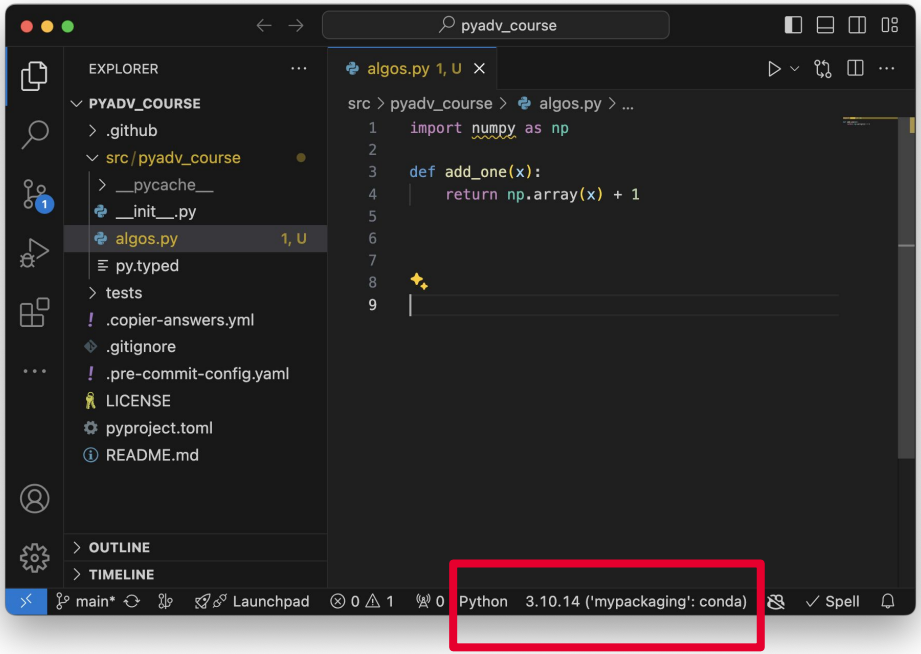
And commit the file:

```
git add src/algos.py

git commit -m 'adding algos'
```



Open the package in VSCode

Let's fix the environment



Use your package

Try now to import the function

Module exists but dependency is missing!

```
[3]: from pyadv_course import algos
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[3], line 1
----> 1 from pyadv_course import algos

File ~/Desktop/SoftwarDev/pyadv_course/src/pyadv_course/algos.py:1
----> 1 import numpy as np
      3 def add_one(x):
      4     return np.array(x) + 1

ModuleNotFoundError: No module named 'numpy'
```

DSL

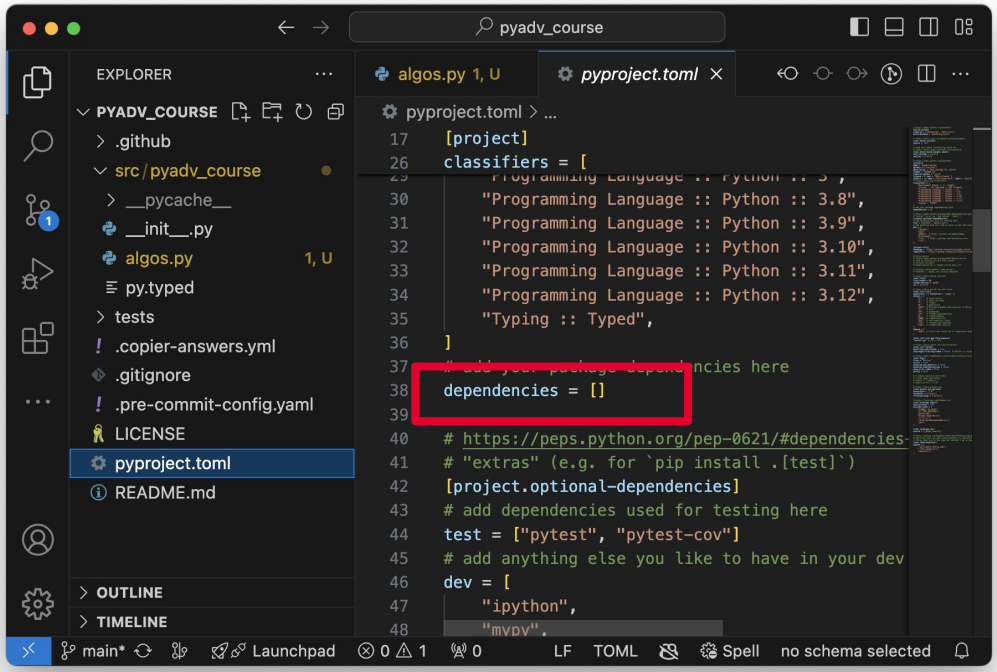
Dependencies

Adding dependencies

No dependencies specified.

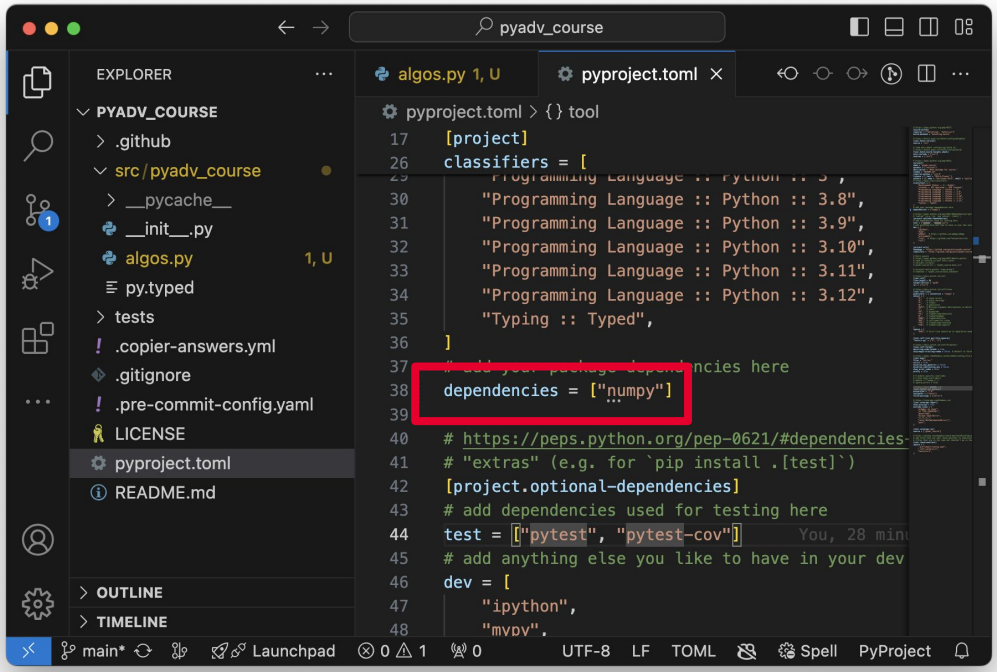
We can:

- install on the fly: cumbersome
- specify dependencies in the toml file -> dependency will always be available when installing package



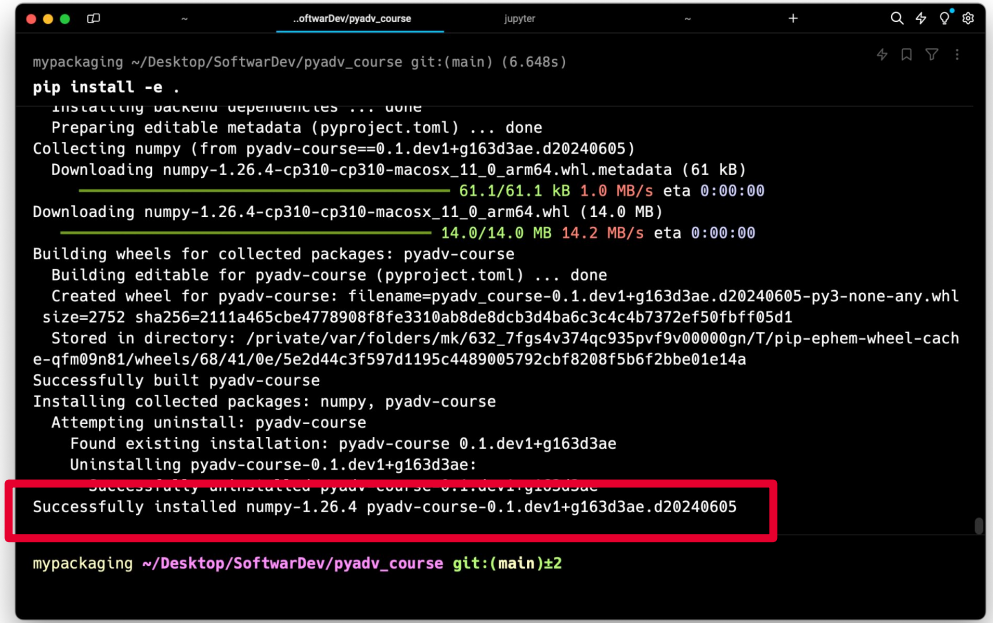
Adding dependencies

- After adding the dependency, we re-install the package:
- `pip install -e .`
- This triggers installation of dependencies!
- Dependencies are installed from PyPi



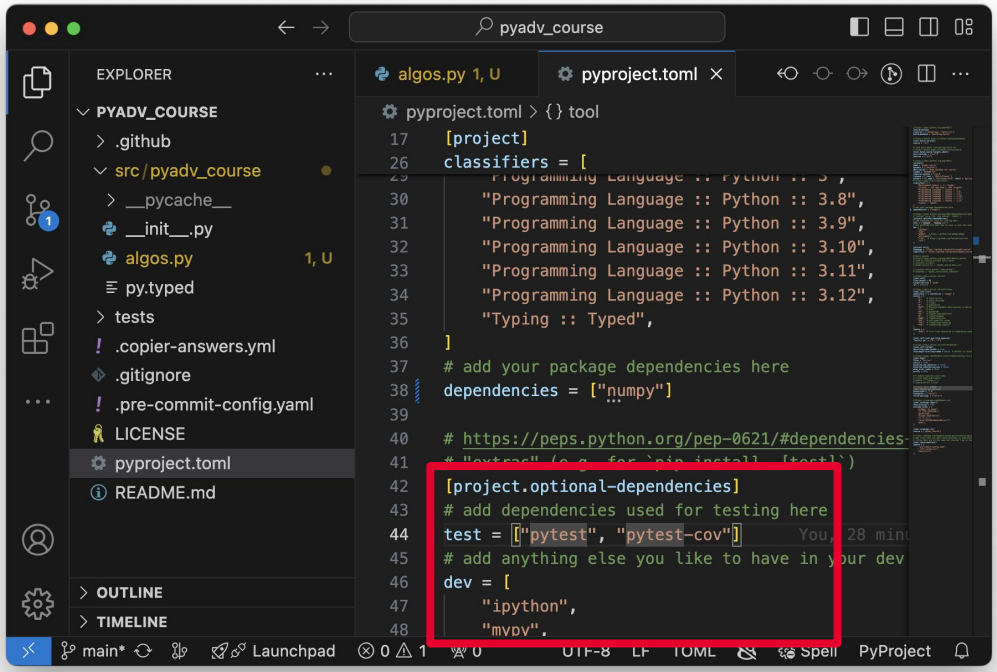
Adding dependencies

- After adding the dependency, we re-install the package:
- `pip install -e .`
- This triggers installation of dependencies!
- Dependencies are installed from PyPi



Adding optional dependencies

- On the model of `test` and `dev`, add you own name for optional dependencies, typically for functionalities used by only a fraction of users.



Adding dependencies

Our import now works!

```
[4]: from pyadv_course import algos
```

Hot reloading

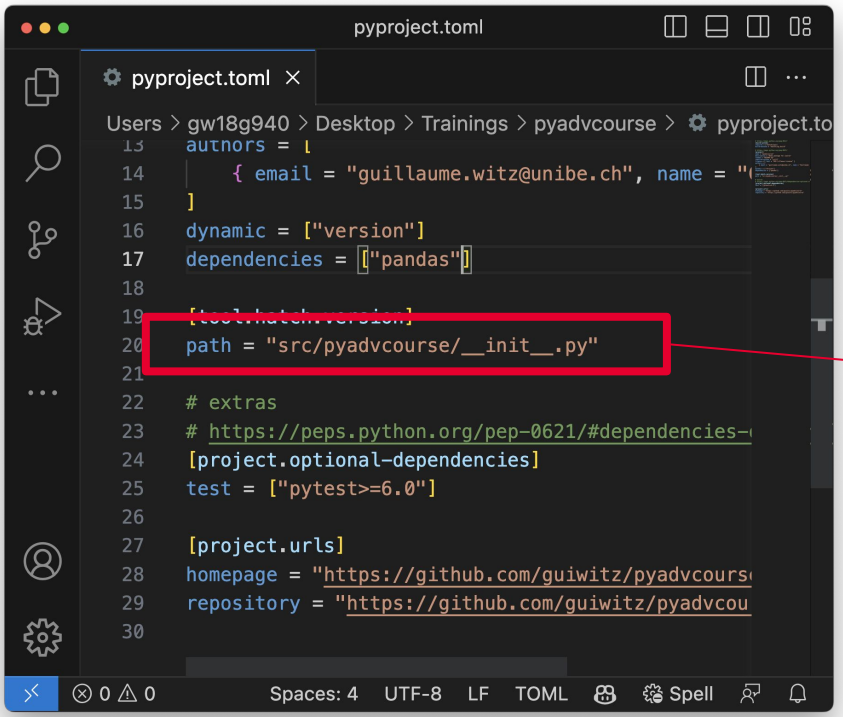
When modifying a package, one has to restart the notebook's kernel and import it for changes to take effect. To avoid this, one can add this before any import statement to be able to update packages "live":

```
%load_ext autoreload
%autoreload 2
```

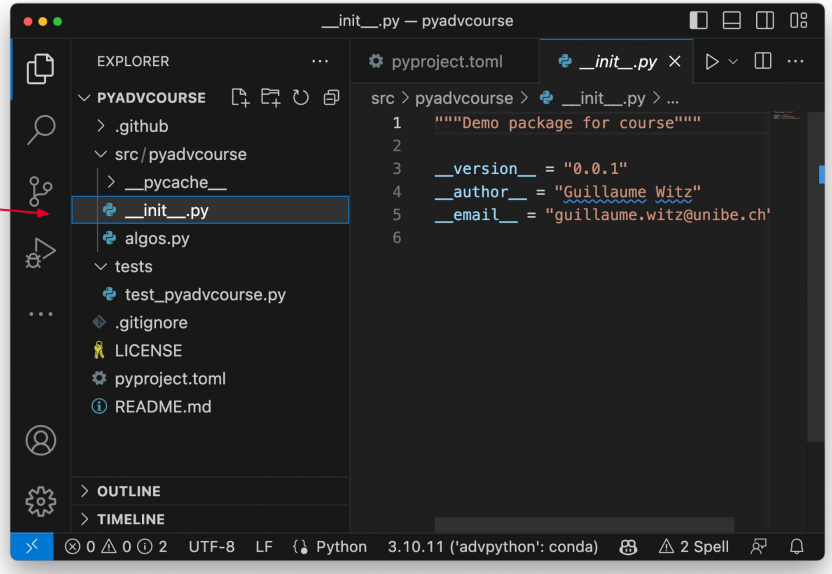
DSL

Versions

Hard-coded version




```
[7]: import pyadvcourse
     pyadvcourse.__version__

[7]: '0.0.1'
```



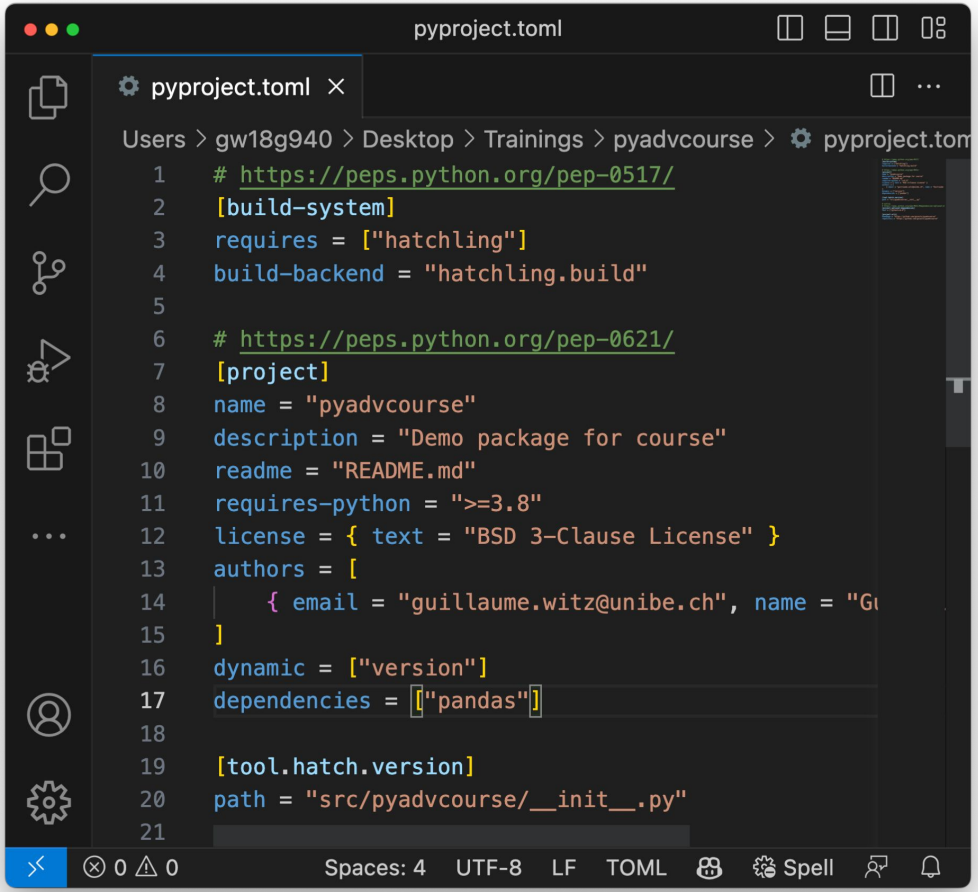
Dynamic version

Hard-coded versions are difficult to “maintain”

A better way is to dynamically update the version with changes

One solution: use git

For this we tell hatch to use git versioning to define the version



Dynamic version

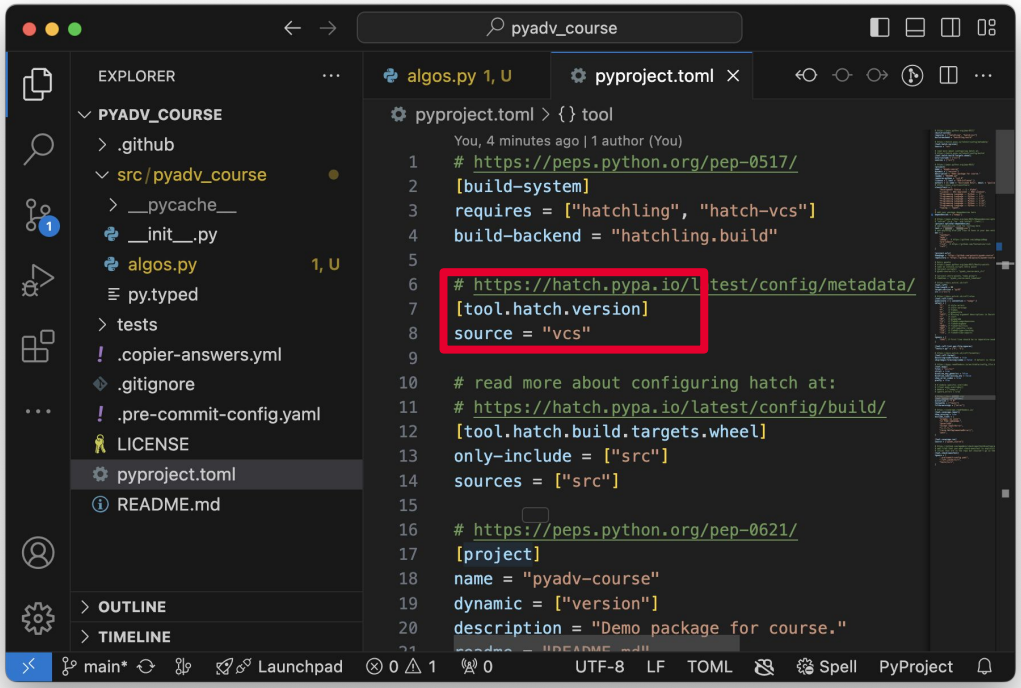
Hard-coded versions are difficult to “maintain”

A better way is to dynamically update the version with changes

One solution: use git

For this we tell hatch to use git versioning to define the version

From the copier, template, the project already uses VCS for versioning



Tagging the version

We add a git tag to the current version (comment about the status) and locally re-install:

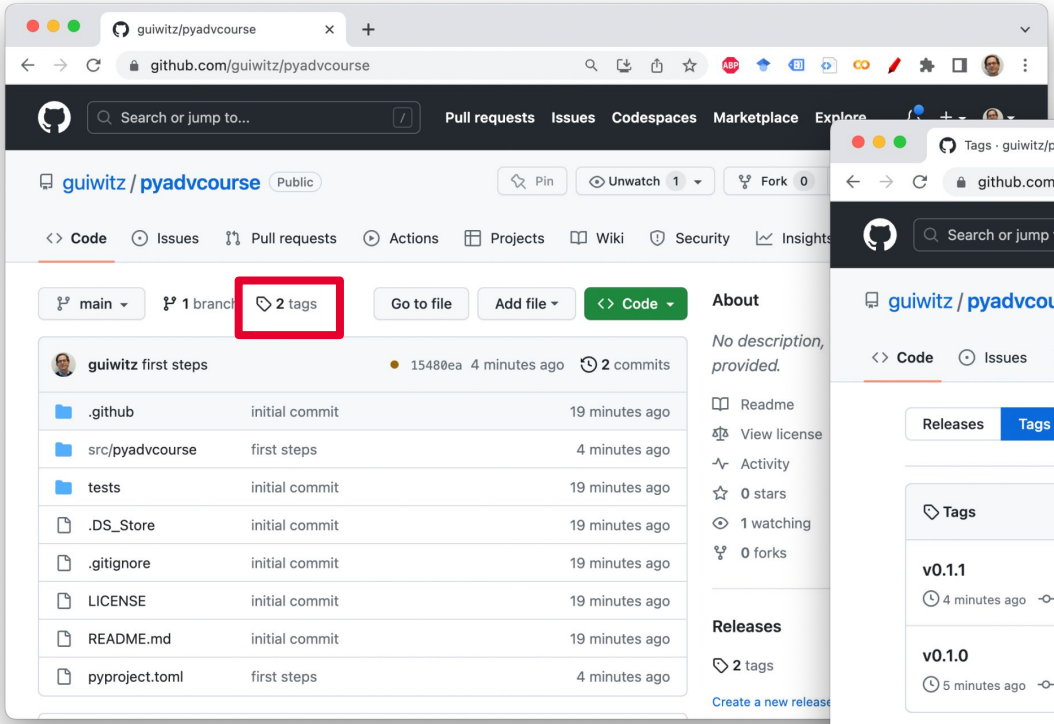
```
git tag -a v0.1.1 -m "v0.1.1"
pip install -e .
```

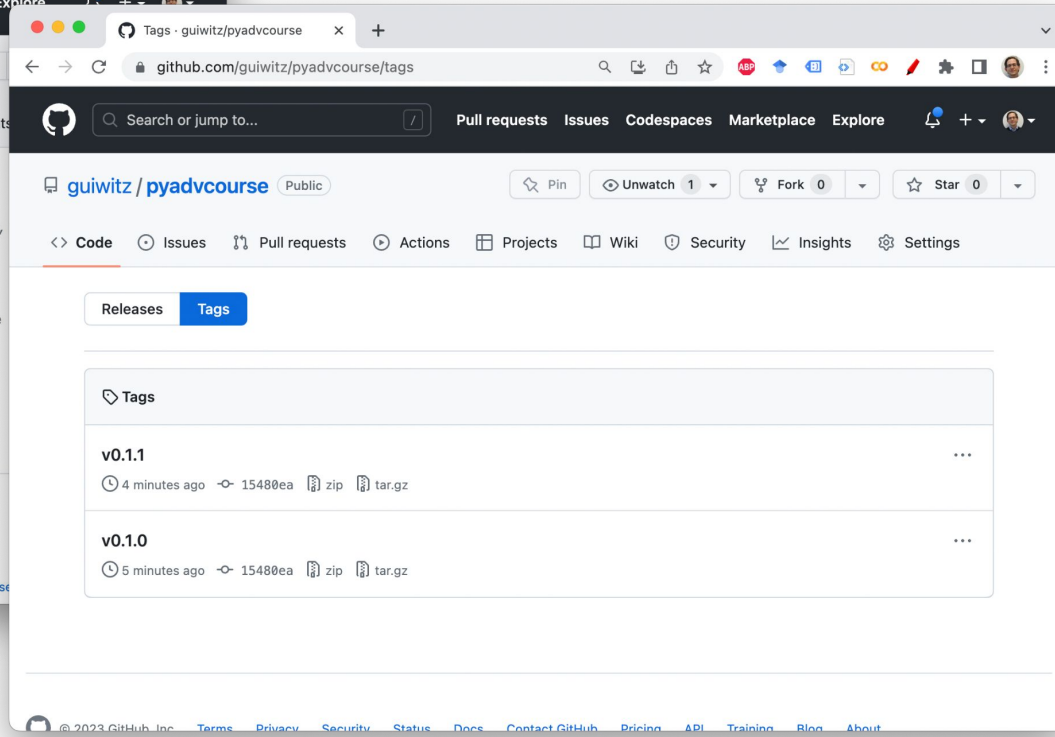
[1]:
```
import pyadv_course
```

[2]:
```
pyadv_course.__version__
```

[2]:
```
'0.1.2.dev0+g7c53523.d20240605'
```

Tags


guiwitz/pyadvcourse
github.com/guiwitz/pyadvcourse
Search or jump to...
Pull requests
Issues
Codespaces
Marketplace
Explore
guiwitz / pyadvcourse
Public
Pin
Unwatch 1
Fork 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
main
1 branch
2 tags
Go to file
Add file
Code
guiwitz first steps
15480ea 4 minutes ago
2 commits
.github initial commit 19 minutes ago
src/pyadvcourse first steps 4 minutes ago
tests initial commit 19 minutes ago
.DS_Store initial commit 19 minutes ago
.gitignore initial commit 19 minutes ago
LICENSE initial commit 19 minutes ago
README.md initial commit 19 minutes ago
pyproject.toml first steps 4 minutes ago
About
No description, provided.
Readme
View license
Activity
0 stars
1 watching
0 forks
Releases
2 tags
Create a new release


Tags · guiwitz/pyadvcourse
github.com/guiwitz/pyadvcourse/tags
Search or jump to...
Pull requests
Issues
Codespaces
Marketplace
Explore
guiwitz / pyadvcourse
Public
Pin
Unwatch 1
Fork 0
Star 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
Releases
Tags
Tags
v0.1.1
4 minutes ago 15480ea zip tar.gz
v0.1.0
5 minutes ago 15480ea zip tar.gz
© 2023 GitHub, Inc. Terms Privacy Security Status Docs Contact GitHub Pricing API Training Blog About