24 – Creating pip-installable Python packages

Bálint Aradi

Scientific Programming in Python (2026)

https://atticlectures.net/scipro/python-2026/

Prerequisites

- Build tool for creating installable Python packages

```
conda install build
```

Distributing Python projects

Developers

- Full project, with source code, tests, development related files, development history
- **Git-repository**
- Distribution: **publicly hosted git repository** (e.g. GitHub, GitLab, …)

Consumers / Users

- Exported functionality only
- **Installable package (e.g. wheel)**
- Distribution: **publicly hosted package repositories**

PyPI

- standard Python package repository
- Package manager (**pip**) part of Python

Conda

- Complex software package repository
- used for non-Python software as well
- Package manager (**conda**) must be installed first

Virtual environments

- Virtual environments are lightweight Python environments (similar to Conda environments)
- They allow the installation / testing of Python packages separated from each other
- Virtual environments are created by importing the Python module **venv**

```
python3 -m venv myenv
```

Creates a directory *myenv* with a new (default) Python environment

- The virtual environment uses the actual Python interpreter but without any third party packages

- Virtual environments are **activated** via their activation script:

```
source myenv/bin/activate
```

- Virtual environments are **deactivated** by the deactivate command:

```
deactivate
```

Installing Python packages into virtual environments

- Packages installed via **pip**
- **pip** installs packages from the Python Package Index (PyPI) or optionally local packages into the active virtual environment

```
pip install numpy
```

Installs package from “numpy” from PyPI

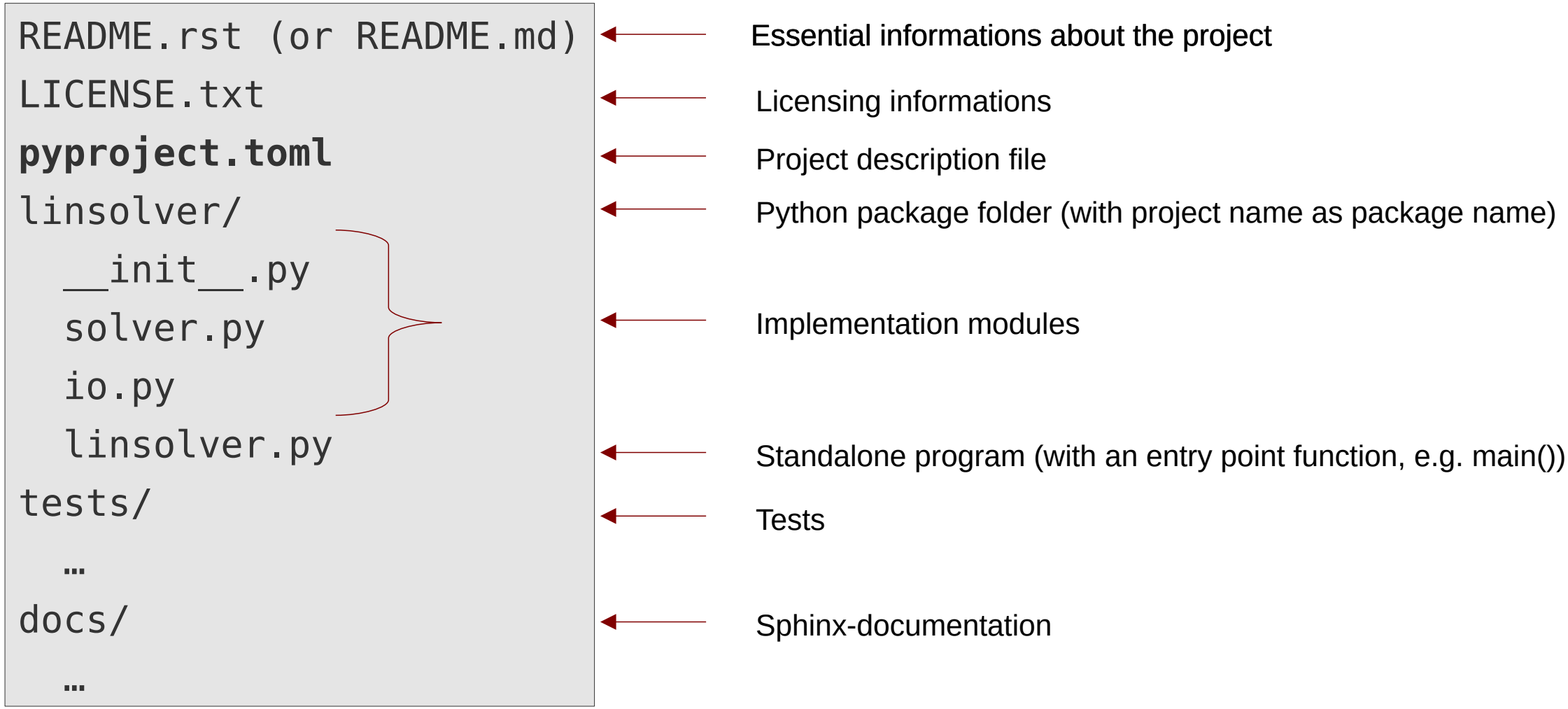
```
pip install ~/Downloads/linsolver-0.9-py3-none-any.whl
```

Installs a locally stored Python-wheel (distributable Python package)

- Installable packages use (among others) the **Python-wheel package format**

Project layout for installable packages

Suggested flat-layout for installable packages



- Alternative **src-layout** with has an additional **src/** folder, see comparison of the two layouts for details

Project description file

pyproject.toml

```
[build-system]
requires = ["hatchling"]
build-backend = "hatchling.build"
```

Build system, use hatchling if unsure

```
[project]
name = "linsolver"
version = "0.9"
requires-python = ">= 3.12"
dependencies = ["numpy"]
maintainers = [{name = "Bálint Aradi", email = "aradi@uni-bremen.de"}]
description = "Demo implementation of a solver …"
readme = "README.rst"
license = {file = "LICENSE.txt"}
```

project details

which dependencies are needed for this project?

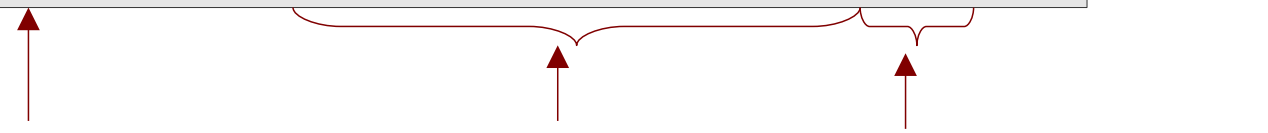
```
[project.scripts]
linsolver = "linsolver.linsolver:main"
```

Standalone programs to be created

Creating standalone scripts

- The installer can create executable standalone scripts during installation

```
[project.scripts]
linsolver = "linsolver.linsolver:main"
```



Name of the standalone script

Module to load when the standalone script is executed

Function to call when the standalone script is executed

- When the installed script is executed (e.g. from the command line), it will simply import the specified module and call the specified function in it

```
linsolver
```

equiv.

```
python3 -c 'import linsolver.linsolver; linsolver.linsolver.main()'
```

Creating a distributable package (wheel)

- Distributable packages (wheels) can be created with the **build** tool

```
python3 -m build -w
```

(issued in the projects root folder – linsolver)

```
dist/linsolver-0.9-py3-none-
any.whl
```

- The created wheel can be **installed** into a virtual environment via **pip**

```
python3 -m venv linsolver.venv
source linsolver.venv/bin/activate
pip install linsolver/dist/linsolver-0.9-….whl
```

(issued **outside** of the project folder!)

new clean virtual environment is active now

install package from wheel into venv

```
linsolver -h
```

standalone program is now available in the virtual environment
when you run Python, you can also import and use the installed packages / modules

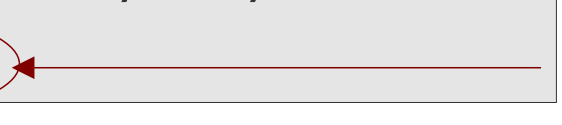
```
deactivate
rm -rf linsolver.venv
```

Deactivate (and optionally delete) virtual environment if not needed any more

Editable installation (for developers)

- Avoids the need for building and installing the project after every change in order to test it
- Only a **pointer to the project source folder** (where development happens) will be installed

```
python3 -m venv linsolver-devtest.venv
source linsolver-devtest.venv/bin/activate
pip install -e linsolver
```



path to your project folder

```
linsolver -h
```

standalone program is available in the virtual environment.
when you run python, you can import and use the installed packages / modules

- If a source file is changed in the development folder, it can be immediately tested in the virtual environment (no need to build a wheel and install it again)
- If you add a new file to the project, you might need to **reinstall** it, to make the editable installation aware of it:

```
pip install -e linsolver
```

path to your project folder

Further information

- Distributing wheels via PyPI and further details on packaging
- Building Conda packages for conda-forge
  (quite complex, take an existing Conda project as example)
- …