WI-Project: Open source project

Introduction to Python (3): Creating a Python package

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

Learning objectives:

- Understand the purpose and structure of a modern Python package.
- Use standard tools (`uv`, `pip`, `pytest`) to create and manage a package.
- Write and run automated tests for your code.
- See how these skills apply to extending `Colrev`.

Information on this session:

- This session is a hands-on tutorial.
- We will build a complete, working Python package from scratch.
- The corresponding Jupyter Notebook contains all the commands and code.

Prerequisites: Installing `uv`

`uv` is a modern, fast tool for managing Python projects. It does not come with Python.

To install `uv`:

(Only needs to be done once)

```
# For macOS, Linux, and Windows WSL
curl -LsSf https://astral.sh/uv/install.sh | sh

# For Windows (Powershell)
irm https://astral.sh/uv/install.ps1 | iex
```

After installing, restart your terminal and check the version: `uv --version`

The anatomy of a Python package

A standard structure is key for tools and collaborators to understand your project.

```
colrev-journal-formatter/
├── src/
│   ├── __init__.py
│   └── formatter.py
├── tests/
│   └── test_formatter.py
└── pyproject.toml
```

- **`pyproject.toml`**: The project's control center.
- **`src/`**: Contains your actual Python source code.
- **`tests/`**: Contains your test code.

Step 1: Initialize the project with `uv`

The `uv init` command creates the `pyproject.toml` file, which is the heart of your package.

```
mkdir colrev-journal-formatter
cd colrev-journal-formatter

uv init
```

This file contains your package's name, version, dependencies, and build instructions.

Step 2: Install in editable mode

To test your code as you write it, install your package in "editable" mode.

```
# Run from the project's root directory
pip install -e .
```

The `-e` flag creates a link to your source files. Any changes you make to the code are immediately available without reinstalling.

Step 3: Add code and tests

1. **Add your function** in `src/colrev_journal_formatter/formatter.py`.
2. **Add** `pytest` as a development dependency:

```
uv add pytest --dev
```

3. **Write your test** in `tests/test_formatter.py` using `assert` statements.
4. **Run your tests** from the project's root directory:

```
pytest
```

`pytest` automatically discovers and runs your test functions.

Next steps: Applying to `Colrev`

You now have the fundamental skills to create any Python package.

- A `Colrev` plugin **is** a standard Python package.
- The functions you write (like data cleaning or formatting) are the core logic of a plugin.
- By learning this, you've acquired the most important skill for extending existing tools like `Colrev`.