

MPPT

Modern Python Package Template

沈祥壮 (Mathew Shen)

2023 年 12 月 15 日

- A set of navigation icons typically found in Beamer presentations, including symbols for back, forward, search, and other slide controls.

Introduction

- Package Management: Poetry
- Documentation: Mkdocs with Material theme
- Linters & Formatter: Black, Isort, Flake8, Ruff, Mypy, Pre-commit, SonarLint
- Testing: Pytest, Hypothesis, Codecov
- Task runner: Makefile, Duty, Taskfile
- Miscellaneous: Changelog, License, Semantic Versioning, Contributing

Most challenging problems

It's not a simple question... ¹

Most challenging problems in software engineering

Dependency management—the management of networks of libraries, packages, and dependencies that we don't control—is one of the least understood and most challenging problems in software engineering.

¹The resources in the section are mostly from the book *Software engineering at google: Lessons learned from programming over time* [1]

Most challenging problems

It's not a simple question... ¹

Most challenging problems in software engineering

Dependency management—the management of networks of libraries, packages, and dependencies that we don't control—is one of the least understood and most challenging problems in software engineering.

A network of dependencies and their changes over time

The trick isn't just finding a way to manage one dependency—the trick is how to manage a network of dependencies and their changes over time.

¹The resources in the section are mostly from the book *Software engineering at google: Lessons learned from programming over time* [1]

Conflicting Requirements and Diamond Dependencies

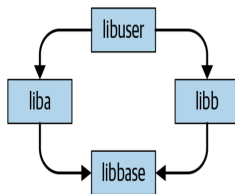
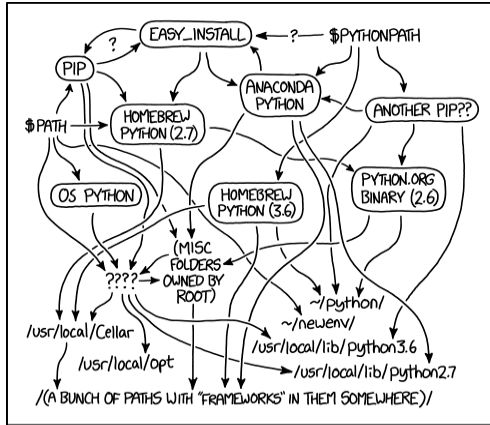


图 1: Diamond Dependencies

Conflicting Requirements

If libbase ever introduces an incompatible change, there is a chance that liba and libb , as products of separate organizations, don't update simultaneously. If liba depends on the new libbase version and libb depends on the old version, there's no general way for libuser (aka your code) to put everything together.

Python Environment



MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED
THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

图 2: XKCD: Python Environment

Why Poetry?

- More popular: Poetry(27.6K Star), Rye(7.6K Star), PDM(5.6K Star)
- Faster: **Python Package Manager Shootout**

More details

See: <https://datahonor.com/mppt/package/>

Documentation

Mkdocs with Material theme

- Markdown is easier/better(in a way) than reStructuredText(which is used by Sphinx)
- Material theme is Awesome!



Martin Donath liked your post

MPPT: A Modern Python Package Template

Package Management: Poetry

Documentation: Mkdocs with Material theme

Linters: Black, Isort, Flake8, Mypy, Pre-commit

Testing: Pytest, Hypothesis, Codecov

Task runner: Makefile, Duty, Taskfile pic.twitter.com/KP6at6tsnh

 3: Like from Martin Donath²

²Author of Material for MkDocs

More details

See: <https://datahonor.com/mppt/doc/>

Linters & Formatters

Break code style is easy in Python

How many code style mistakes in the code in `hello.py`?

```
1 print ( "Hello, World" )
```

Break code style is easy in Python

How many code style mistakes in the code in `hello.py`?

```
1 print ( "Hello, World" )
```

We got four code style mistakes in a one line hello world code.

- `hello.py:1:6: E211 whitespace before '('`
- `hello.py:1:8: E201 whitespace after '('`
- `hello.py:1:23: E202 whitespace before ')'`
- `hello.py:1:25: W292 no newline at end of file`

Linter & Formatter

Solution1: Black, Isort, Flake8, MyPy

- Black: Code Formatter
- Isort: Style Linter for Import Statements
- Flake8: Error & Style Linter & Complexity Analysis
- MyPy: Type Checker

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ◻ ↺ 🔍 ↻

- Black: Code Formatter
- Isort: Style Linter for Import Statements
- Flake8: Error & Style Linter & Complexity Analysis
- MyPy: Type Checker

Ruff: Linter & Formatter

We can use pre-commit to manage all the linters and formatters together.

More details

See: <https://datahonor.com/mppt/linter/>

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ↺ 🔍 ↻

Testing

- Pytest
- Hypothesis
- Codecov

Hypothesis for property-based testing

Property-based testing is popularised by the Haskell library Quickcheck.³

```

1 from hypothesis import given
2 from hypothesis.strategies import text
3
4
5 @given(text())
6 def test_decode_inverts_encode(s):
7     assert decode(encode(s)) == s
    
```

³ [Hypothesis doc](#)

More details

See: <https://datahonor.com/mppt/test/>

Task runner

Task runner

- Makefile
- Taskfile
- Duty
- Typer

More details

See: <https://datahonor.com/mppt/task/>

Miscellaneous

Miscellaneous

- Keep a Changelog
- Semantic Versioning
- Choose an open source license
- Badge: Shield.io
- Contributing to Open Source on GitHub

More details

See: <https://datahonor.com/mppt/miscellaneous/>

References

- [1] Titus Winters, Tom Manshreck, and Hyrum Wright. *Software engineering at google: Lessons learned from programming over time*. O'Reilly Media, 2020.

QA