Frost Ming @ PyCon China 2025

Python Packaging PEPs

打包生态的最新进展

关于我

Frost Ming

Python Developer · PyPA Member

🛠️ PDM 作者 - 现代 Python 包管理器
📦 Python 打包生态贡献者
✨ 开源项目：Unearth, Marko, Monas

@frostming frostming.com mas.to@frostming

什么是 PEP?

PYTHON ENHANCEMENT PROPOSAL

- Python 改进提案
- 描述新特性或流程的技术规范
- 社区驱动的决策过程
- 推动 Python 生态系统演进

打包相关的 PEP

- 定义打包标准和规范
- 改进依赖管理
- 提升用户体验
- 促进工具互操作性

已完成的 PEP

Final Status

PEP 723：内联脚本元数据

```python
# /// script
# requires-python = ">=3.11"
# dependencies = [
#   "requests",
#   "rich",
# ]
# ///

import requests
from rich import print

print(requests.get("https://api.github.com"))
```

主要特性

- 在单文件脚本中声明依赖
- 标准化的元数据格式
- 工具可以自动处理依赖安装

PEP 723：运行脚本

使用现代工具直接运行

```
# 使用 PDM - 自动安装依赖并运行
pdm run script.py

# 使用 uv - 自动安装依赖并运行
uv run script.py
```

特点

- 🚀 自动检测并安装脚本中声明的依赖
- 📦 创建隔离的虚拟环境
- ✨ 无需手动管理依赖
- 🔄 支持依赖版本锁定和更新

PEP 735：依赖组

```toml
[dependency-groups]
test = [
  "pytest>=7.0",
  "pytest-cov",
]
docs = [
  "sphinx>=5.0",
  "sphinx-rtd-theme",
]
dev = [
  { include-group = "test" },
  { include-group = "docs" },
  "pre-commit",
]
```

优势

- 标准化的依赖分组方式
- 支持组合和继承
- 替代各种工具特定的格式

依赖组 vs 可选依赖

🎁 可选依赖 (OPTIONAL DEPENDENCIES)

```toml
[project.optional-dependencies]
aws = ["boto3>=1.26", "s3fs>=2023.1"]
viz = ["matplotlib>=3.7", "seaborn"]
```

用户功能

- 发布到 PyPI
- 面向最终用户
- `pip install pkg[aws]`
- 扩展包功能

🔧 依赖组 (DEPENDENCY GROUPS)

```toml
[dependency-groups]
test = ["pytest>=7.0", "pytest-cov"]
lint = ["ruff>=0.1", "mypy>=1.0"]
```

开发工具

- 不发布到 PyPI
- 面向开发者
- `pip install --dependency-groups test`
- 管理开发环境

关键区别: 可选依赖是包的一部分，依赖组是开发环境的一部分

📌 **PDM 已原生支持**: PDM 现已采用标准的 `dependency-groups`，取代了原有的 `tool.pdm.dev-dependencies`

PEP 751：锁文件格式

```toml
[[packages]]
name = "requests"
version = "2.31.0"
source = { registry = "https://pypi.org/simple" }
sdist = { url = "...", hash = "sha256:..." }

[[packages.wheels]]
url = "..."
hash = "sha256:..."
tags = ["cp39-cp39-manylinux_2_17_x86_64"]
requires-python = ">=3.7"
dependencies = ["certifi", "charset-normalizer", "idna", "urllib3"]
```

特点

- 记录精确的依赖版本
- 包含完整的哈希值
- 支持跨平台锁定
- 确保可重现的安装

PEP 751：各工具支持情况

当前各工具锁文件

工具	当前格式	导出 pylock.toml	从 pylock.toml 安装
Poetry	poetry.lock	否	否
PDM	pdm.lock/**pylock.toml**	是	是
uv	uv.lock	是	是
pip-tools	requirements.txt	否	否
pip	无	是	否

另外，Dependabot 也已开始支持 pylock.toml。

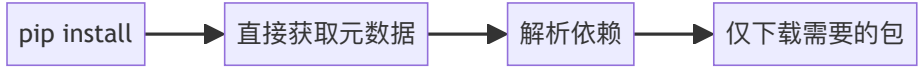
已接受的 PEP

Accepted Status

PEP 658：简单 API 中的元数据

改进前

改进后



✅ 减少下载量 ✅ 加快依赖解析速度 ✅ 改善用户体验

```
<a href="https://files.pythonhosted.org/packages/36/42/015c23096649b908c809c69388a805a571a3bea44362fe87e33fc3afa01f/
    data-requires-python="&gt;=3.8"
    data-dist-info-metadata="sha256=d365cfebd5538b09f96e5711807732b9243670b4ee24557c157c36c78427c4aa"
    data-core-metadata="sha256=d365cfebd5538b09f96e5711807732b9243670b4ee24557c157c36c78427c4aa">
        flask-3.0.0-py3-none-any.whl
</a>
```

元数据链接：

```
https://files.pythonhosted.org/packages/.../flask-3.0.0-py3-none-any.whl.metadata
```

PEP 691：JSON 格式的简单 API

HTML 格式（旧）

```
<a href="django-4.2.tar.gz">
  django-4.2.tar.gz
</a>
<a href="django-4.2-py3-none-any.whl">
  django-4.2-py3-none-any.whl
</a>
```

JSON 格式（新）

```
{
  "files": [{
    "filename": "django-4.2.tar.gz",
    "url": "...",
    "hashes": {"sha256": "..."},
    "requires-python": ">=3.8",
    "yanked": false
  }]
}
```

方法：

```
GET https://pypi.org/simple/django/

Accept: application/vnd.pypi.simple.v1+json
```

更结构化 · 更易解析 · 更多元数据

讨论中的 PEP

Under Discussion

PEP 771：默认额外依赖

```toml
[project]
name = "mypackage"
default-optional-dependency-keys = [
    "recommended",
]

[project.optional-dependencies]
lite = []
recommended = [
  "rich", "httpx"
]
```

使用场景

```bash
# 安装包及其默认额外依赖
pip install mypackage

# 仅安装核心依赖
pip install mypackage[lite]
```

PEP 794：导入名元数据

问题：包名 ≠ 导入名

包名 `beautifulsoup4`	包名 `pillow`	包名 `python-dateutil`
导入名 `bs4`	导入名 `PIL`	导入名 `dateutil`

解决方案

```
[project]
name = "beautifulsoup4"
import-names = ["bs4"]
```

💡 工具可以自动检测未安装的依赖 💡 改善开发体验

谢谢！

问题与讨论

相关链接

PEPs 官网 · Python Packaging User Guide · PyPA 讨论区