uv Document

citrus88
citrus.mikan88@gmail.com
初版: 2025/08/05
更新日: 2025/08/05

1. uvの概略

uvは，複数あるPythonの開発ツールを1つのツールで置き換える包括的なソリューションとして設計されたツールである．Rustで書かれており，非常に高いパフォーマンスと多くの機能を持ったツールである．従来のPythonは，pip, pip-tools, pipx, poetry, pyenv, twine, virtualenvなど，複数のツールを使って開発を行っていた．ライセンスは，Apache License Version 2.0もしくはMIT Licenseのどちらかを選ぶことができる．
現在も開発が進められており，破壊的な変更が起こる可能性がある．日々最新の情報を取得しながら利用すると良い．

1.1 uvの特徴

- 高速性
  - 従来のpipなどのPythonパッケージ管理ツールと比較して，極めて高速な処理が可能である．
- 信頼性
  - 依存関係解決の信頼性が向上しており，より一貫性のある環境構築が可能である．
- 統合性
  - 複数のPythonツールを1つに統合することで，ツールチェーンの複雑さを大幅に軽減する．
- 自動管理
  - 必要に応じて不足しているパッケージを自動的にインストールし，環境のセットアップを簡素にできる

1.2 uvの主な機能

uvの用途とその機能を担っていた既存ツールを紹介する．

用途	既存のツール名
パッケージ管理	pip, pip-tools
仮想環境管理	virtualenv
プロジェクト管理	poetry
Pythonバージョン管理	pyenv
ツール実行	pipx
パッケージ管理	twine

2 uvのインストール方法

最新のインストール方法は，公式サイトを参照すること．

2.1 インストール

macOS・Linux

```
curl -LsSf https://astral.sh/uv/install.sh | sh
```

Windows

```
powershell -ExecutionPolicy ByPass -c "irm https://astral.sh/uv/install.ps1 | iex"
```

2.2 インストール確認

```
uv --version
```

エラーが出た場合，パスが正しく通っているかを確認してください．

3 uvの基本的な使い方

3.1 仮想環境の作成と管理

新しいプロジェクトの初期化

```
uv init my-project
cd my-project
```

3.2 ライブラリの追加・削除

パッケージの追加

```
uv add requests
```

パッケージの削除

```
uv remove requests
```

依存関係のインストール

`uv.lock`ファイルには正確な依存関係バージョンが自動的に記録されており，開発環境を簡単に再現することができる.

```
uv sync  # uv.lockから依存関係をインストール
```

3.3 Pythonスクリプトの実行

スクリプトの実行

```
uv run python script.py
```

4 uvの応用的な使い方

4.1 Pythonバージョン管理

```
# 特定のPythonバージョンのインストール
uv python install 3.11
# プロジェクトでのPythonバージョン指定
uv python pin 3.11
```

4.2 開発環境パッケージの管理

開発時だけ使用するパッケージは，実際にプログラムを動かす時は不要である．例えば，コードを整形するためのライブラリやテスト実行ツールなどは開発時に非常に重要であるが，プログラムを実行する時には不要である．uvでは，そのようなツールは開発用パッケージとして分けて管理することができる．

```
# 開発用パッケージの追加
uv add --dev pytest ruff mypy
# プロダクション環境でのインストール（開発依存関係を除外）
uv sync --no-dev
```

4.3 ツール管理

uvはPythonパッケージの便利なツールをグローバル環境にインストールできる．
他のプロジェクト間でツールを共有して利用できるため，一回のインストールで複数のプロジェクトで利用できる．
`uv add --dev`との明確な使い分けは難しいが，プロジェクトに依存しない汎用ツールを個人開発で利用したい場合に用いると良い．
チームで開発補助ツールを共有する必要がある場合，`uv add --dev`を利用する．

```
# グローバルツールのインストール
uv tool install ruff
# インストール済みツールの確認
uv tool list
# インストール済みツールの利用
uvx ruff    # uvx = uv tool run
```

5. トラブルシューティング

5.1 一般的な問題

- キャッシュクリア: `uv cache clean`
- 詳細ログ: `uv --verbose <command>`
- ヘルプ: `uv help <command>`

5.2 移行時の注意点

- 既存の`requirements.txt`からの移行: `uv add -r requirements.txt`
- `poetry.lock`からの移行: プロジェクト再初期化を推奨

参考文献

- uv
- uv: Python packaging in Rust
- Getting started - Astral Docs
- GitHub - astral-sh/uv