PythonパッケージをPyPIに登録する手順

PythonパッケージをPyPIに登録することで、他のユーザーがpipを使用して簡単にインストールできるようになります。以下は、パッケージをPyPIに登録する基本的な手順です。

1. PyPIアカウントの作成

まず最初に、PyPIのウェブサイトでアカウントを作成します。アカウントがない場合は、Registerから新しいアカウントを作成してください。

2. パッケージの準備

自分のスクリプトをパッケージとしてまとめる必要があります。

Poetry

```
poetry init # Poetryプロジェクトを初期化します。 -n をつけるとノーインタラクション

poetry add your_script_name
cat requirements.txt | xargs poetry add  # 依存関係をpyproject.tomlファイルに記述します

poetry version patch # バージョンを更新します。
```

```
# pyproject.toml

[tool.poetry]
name = "my_package"
version = "0.1.0"
description = ""
authors = ["admin"]
packages = [{include = "src"}]
readme = "docs/README.md"

[tool.poetry.scripts]
regrex = 'src.app:main'

[build-system]
requires = ["poetry-core>=1.0.0"]
build-backend = "poetry.core.masonry.api"

[tool.poetry.dependencies]
python = "^3.8"

[tool.poetry.dev-dependencies]
pytest = "^5.2"
```

```
poetry update --dry-run
touch src/__init__.py
```

以下は、基本的な構造の例です。

```
my-package/
├ src/
│  ├ __init__.py
│  ├ __main__.py
│  ├ app.py
├ tests/
│  ├ __init__.py
├ pyproject.toml
├ poetry.lock
├ requirements.txt
├ docs/readme.md
├ dist/
```

3. 仮想環境のセットアップ

仮想環境を構築する場合、次の手順に従います。

```
poetry install
poetry install --no-dev (optional)

poetry run python <python-file> # 仮想環境で実行 (optional)
poetry run pytest # ユニットテストを実行する (optional)
poetry shell # Shell を起動 (optional)
```

4. パッケージのビルド

次に、以下のコマンドでパッケージをビルドします。

```
poetry build # パッケージをビルドします。
```

これにより、dist/ ディレクトリに my_package-0.1.tar.gz のようなアーカイブファイルが生成されます。

5. PyPIへのアップロード

以下のコマンドでパッケージをPyPIにアップロードします。アップロード時に、PyPIのユーザー名とパスワードが必要です。

```
set url https://pypi.org/legacy
set PYPI_AUTH_TOKEN_1 ***

poetry config repositories.pypi $url # PyPIのリポジトリを設定ファイルへ記入します。
poetry config pypi-token.pypi $PYPI_AUTH_TOKEN_1 # PyPIのアクセストークンを設定します。

poetry publish
# poetry publish --build # build same time
```

- TestPyPI:

```
set url https://testpypi.org/legacy
set TEST_PYPI_AUTH_TOKEN_1 ***

poetry config repositories.test-pypi $url # PyPIのリポジトリを設定ファイルへ記入します。
poetry config pypi-token.pypi $TEST_PYPI_AUTH_TOKEN_1 # PyPIのアクセストークンを設定します。

poetry publish
# poetry publish --build # build same time
```

以上で、自作スクリプトがPyPI/TestPyPIに登録され、他のユーザーが pip install my_package で簡単にインストールできるようになります。

```
pip install my_package
pip show my_package
pip install -U --no-cache-dir my_package # update
my_package -h
```

- Test:

```
set pkg python-demo
set url https://test.pypi.org/simple/$pkg
pip install --index-url $url
```

See also

pip-tools · PyPI

Using pip-compile to manage dependencies in your Python packages | by Christopher Davies | Packagr | Medium

GitHub - tox-dev/pipdeptree: A command line utility to display dependency tree of the installed Python packages

- 依存関係の自動解決

```
pipreqs --savepath=requirements.txt --use-local && pip-compile -o requirements.txt pyproject.toml

cat requirements.txt | xargs poetry update
```