Python Packaging Tools

Distribution metadata management

- bumpversion - Version-bump your software with a single command (Github)
- check-manifest - Check that your MANIFEST.in rules are complete and include all your files in the source package (Github)
- restview - Preview your package's `long_description` in ReST format in the browser, rendered in the same way as PyPI would (Home page)

Dependency management

- bumper - Bump (pin/manage) your dependency requirements with ease (Github)
- checkmyreqs - Check your project requirements for Python version compatibility (Github)
- pipreqs - Generate requirements.txt file for any project based on imports (Github)
- pipdeptree - Command line utility for displaying the Python packages installed in an virtualenv in form of a dependency tree (Github)
- piprot - Check the requirements defined in your requirements files for freshness (Github)
- pip-review - A similar tool to piprot, but reports on all packages installed in a virtualenv (Github)
- pip-tools - A set of tools to keep your pinned Python dependencies fresh (Github)
- pipwrap - Manage pip requirements files for multiple deployment environments, e.g. production and development (Github)

Package index and repository

- pip-accel - A wrapper for pip, which accelerates initializing virtual envs by building a local package cache (Github)
- devpi - A PyPI-compliant local, caching package index server and related clients and tools (Homepage)
- twine - Twine uploads distributions to PyPI via SSL and, unlike plain distutils, allows pre-creating and signing packages in a separate step (Github)