
MINI-COURSE ON CODE DEVELOPMENT AND PACKAGING



RELEASING YOUR PACKAGE

VERSIONS

VERSIONING YOUR CODE

- Once you start releasing different versions of your package, you should explicitly give your code a *version string*
 - Users can use to know which version they are using
 - Installers know when to update the installation when asked (`pip install -U ...`)
 - Version string typically appears in
 - `setup.py`'s `setuptools.setup`
 - Your package's top-level `__init__.py` as the definition `__version__ = ...`
 - Perhaps in your `docs/source/conf.py`
 - Make sure to keep these in sync (can use `bump2version` for this, but not discussed here)
-

VERSION STRING FORMAT

- Format should be
 - $[N!]N(.N)^*[\{a|b|rc\}N][.postN][.devN]$
 - But in practice really at most $N(.N)^*[\{a|b|rc\}N][.postN][.devN]$
 - Doesn't include 'v'!
-

UNPACKING THE VERSION STRING FORMAT

- $N(.N)^*[\{a|b|rc\}N][.postN][.devN]$
 - $N(.N)^*$:
 - Main part: major.minor or major.minor.patch or ... (but typically one of the first two); e.g., 1.9.2
 - Increment as
 - 1.9.2 \rightarrow 1.9.3: small update to currently released version
 - 1.9.2 \rightarrow 1.10.0: most new releases with new features
 - 1.9.2 \rightarrow 2.0.0: major change in the code base, rare
 - $\{a|b|rc\}N$:
 - Use to indicate alpha (a), beta (b), or 'release candidate' (rc) for version specified by $N(.N)^*$
 - $[.devN]$:
 - Use to indicate in-development version of version specified by $N(.N)^*$, e.g., 1.9.3.dev0 is development version of 1.9.3
-

PARSING VERSIONS

- Using the standard format allows you to use standard tools to parse the version
- These are also to use in your code, if you need a certain version of a dependency for certain things

```
from pkg_resources import parse_version
import scipy
print(parse_version(scipy.__version__) >= parse_version('0.19'))
```

PREPARING FOR YOUR PACKAGE'S RELEASE

STEPS TO PREPARE

- Update the version number (e.g., remove the .dev part)
- May want to add a few more files related to the release:
 - HISTORY(.txt, .md): gives overview of main changes since the last version; good to start with release 1, where you can just say “first release”
 - MANIFEST.in: lists extra files to include in/exclude from source distribution, default only .py, README.x, files package_data, not LICENSE, so have file with, e.g.,

```
include LICENSE README.md
exclude tests/test*.py
```

RELEASING TO GITHUB

- Tag the version with git, using the version as the tag label

```
git tag x.y.z
```

- Push tag to GitHub

```
git push --tags
```

- Then go to GitHub to create the release
-

RELEASING TO GITHUB

The screenshot shows the GitHub repository page for 'jobovy/exampy'. The repository is described as 'A small example Python package'. The page includes navigation tabs for Code, Issues (0), Pull requests (0), Actions, Projects (0), Wiki, Security, Insights, and Settings. The repository statistics bar shows 64 commits, 1 branch, 0 packages, 1 release (circled in red), 1 contributor, and MIT license. Below the statistics, there are buttons for 'Branch: master', 'New pull request', 'Create new file', 'Upload files', 'Find file', and 'Clone or download'. The commit history table shows the following entries:

Commit	Message	Time
jobovy	Install nbsphinx on readthedocs	4 days ago
docs	Install nbsphinx on readthedocs	4 days ago
exampy	Catch TypeError in integrate.simps when function cannot be called on ...	8 days ago
tests	Make the expected failure strict and add a reason for the failure as ...	8 days ago
.appveyor.yml	AppVeyor run with multiple python and numpy versions, using Miniconda	6 days ago
.coveragerc	Don't omit most of the package from coverage	7 days ago

RELEASING TO GITHUB

The screenshot shows the GitHub interface for the repository 'jobovy / exampy'. At the top right, there are buttons for 'Unwatch' (1), 'Star' (1), and 'Fork' (0). Below this is a navigation bar with 'Code' selected, and other options like 'Issues 0', 'Pull requests 0', 'Actions', 'Projects 0', 'Wiki', 'Security', 'Insights', and 'Settings'. Underneath the navigation bar, there are two tabs: 'Releases' (which is active and highlighted in blue) and 'Tags'. On the right side of the page, a button labeled 'Draft a new release' is circled in red. Below the navigation, a release entry is visible with the version '0.1' and a commit hash '036893a', along with download links for 'zip' and 'tar.gz'.

RELEASING TO GITHUB

The screenshot shows the GitHub interface for a repository named 'jobovy / exampy'. At the top right, there are buttons for 'Unwatch' (1), 'Star' (1), and 'Fork' (0). Below this is a navigation bar with links for 'Code', 'Issues' (0), 'Pull requests' (0), 'Actions', 'Projects' (0), 'Wiki', 'Security', 'Insights', and 'Settings'. The 'Releases' tab is selected, and the 'Tags' tab is also visible. A text input field contains '0.1', which is marked as an 'Existing tag'. Below this, a larger text input field contains 'Version 0.1'. There are 'Write' and 'Preview' buttons. The main content area contains the text: 'First release of the "exampy" package, a very simple example Python package that accompanies a set of notes on how to create a scientific Python package.' At the bottom of the content area, there is a dashed line and the text: 'Attach files by dragging & dropping, selecting or pasting them.' To the right of the main content area, there are two informational sections: 'Tagging suggestions' and 'Semantic versioning'.

jobovy / exampy

Unwatch 1 Star 1 Fork 0

Code Issues 0 Pull requests 0 Actions Projects 0 Wiki Security Insights Settings

Releases Tags

0.1

✓ Existing tag

Version 0.1

Write Preview

First release of the "exampy" package, a very simple example Python package that accompanies a set of notes on how to create a scientific Python package.

Attach files by dragging & dropping, selecting or pasting them.

Tagging suggestions

It's common practice to prefix your version names with the letter v. Some good tag names might be v1.0 or v2.3.4.

If the tag isn't meant for production use, add a pre-release version after the version name. Some good pre-release versions might be v0.2-alpha or v5.9-beta.3.

Semantic versioning

If you're new to releasing software, we highly recommend reading about [semantic versioning](#).

RELEASING TO PyPI

RELEASING TO PyPI

- Python Package Index (PyPI): main repository for releases of Python packages, source of `pip install PACKAGE`
 - TestPyPI: fully independent version of PyPI that can be used to test whether release looks okay before making it (because hard to change once released to PyPI)
 - To get started, make accounts on <https://pypi.org/> and <https://test.pypi.org/>
 - Interacting with PyPI uses `twine`, so `pip install twine`
-

CREATING THE SOURCE DISTRIBUTION

- First we'll create a source distribution, a “tarball” that has the source code that users can use to build the code themselves (e.g., through `pip`)
- Check out a clean version of your repository, to make sure you don't accidentally include files that shouldn't be included (e.g., files left over from a merge), then check out the tag for version `x.y.z`

```
git checkout x.y.z
```

- Create the source distribution in `dist/` with,

```
python setup.py sdist
```

TEST YOUR RELEASE ON TestPyPI

⚠ You are using TestPyPI – a separate instance of the Python Package Index that allows you to try distribution tools and processes without affecting the real index.

[Help](#)[Donate](#)[Log in](#)[Register](#)

exampy 0.1

[Latest version](#)

```
pip install -i https://test.pypi.org/simple/ exampy
```



Released: 2 minutes ago

A small example Python package

Navigation

[Project description](#)[Release history](#)

Project description

The author of this package has not provided a project description

TEST YOUR RELEASE ON TestPyPI

⚠ You are using TestPyPI – a separate instance of the Python Package Index that allows you to try distribution tools and processes without affecting the real index.

exampy 0.2



```
pip install -i https://test.pypi.org/simple/ exampy
```

Released: less than 10 seconds ago

A small example Python package

Navigation

☰ Project description

🕒 Release history

📄 Download files

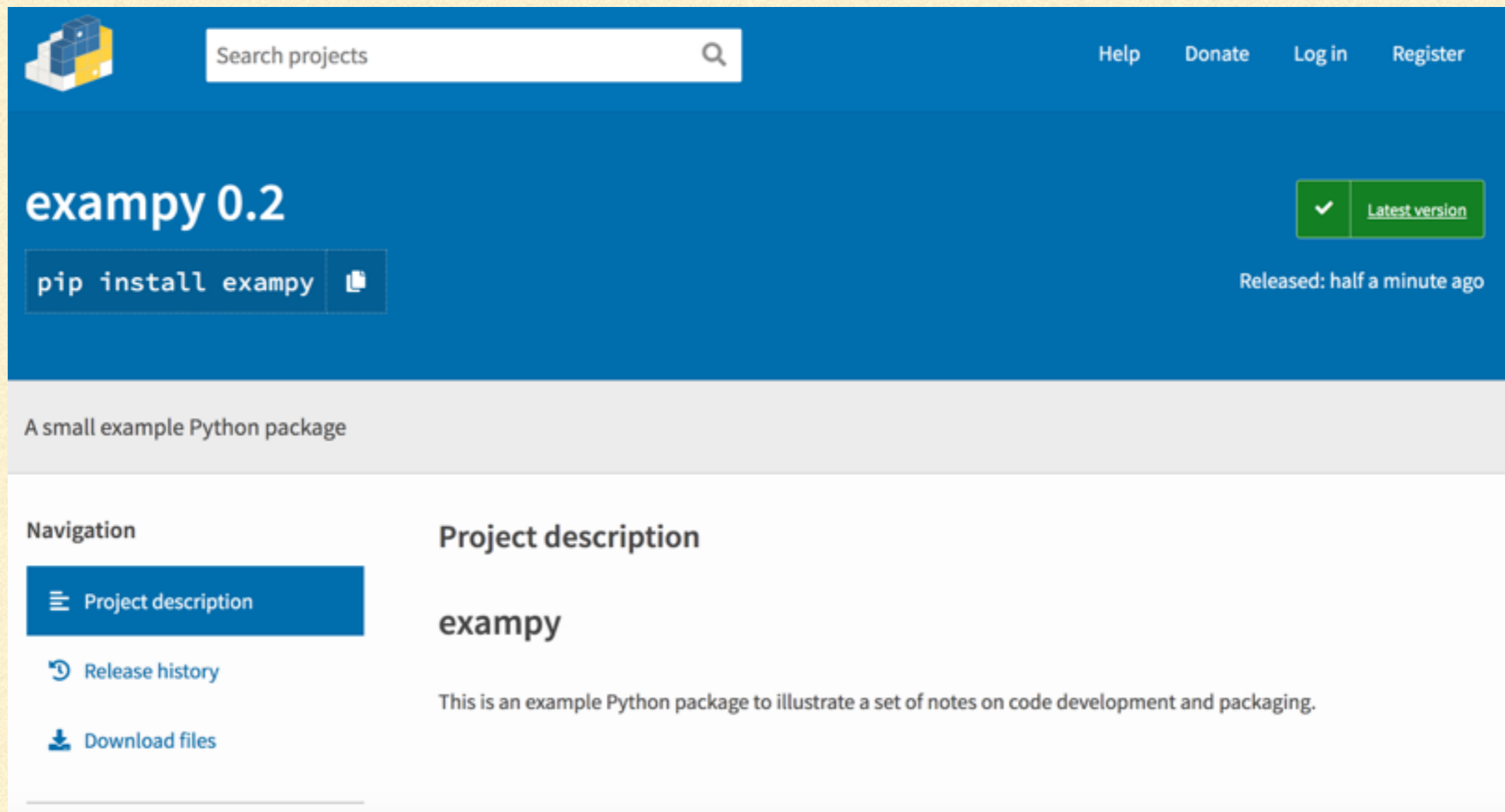
Project description

exampy

This is an example Python package to illustrate a set of notes on code development and packaging.

ALL LOOKS GOOD? —> PyPI

```
twine upload dist/*
```



The screenshot shows the PyPI project page for 'exampy 0.2'. At the top, there is a search bar and navigation links for 'Help', 'Donate', 'Log in', and 'Register'. The project name 'exampy 0.2' is prominently displayed, along with a green checkmark and a 'Latest version' button. Below this, a button shows the command 'pip install exampy' with a copy icon. The release date is noted as 'Released: half a minute ago'. The main content area is divided into two columns: 'Navigation' on the left with links for 'Project description', 'Release history', and 'Download files'; and 'Project description' on the right, which includes the project name 'exampy' and a brief description: 'This is an example Python package to illustrate a set of notes on code development and packaging.'

BUILDING “WHEELS”: BINARY DISTRIBUTIONS

WHEELS

- If all you upload to PyPI is the source code, then every time users install your code, they need to build it locally (automatically done by `pip`, but still)
 - Can lead to long build times if your code needs to be compiled, and make it difficult to install on Windows
 - Solution: build “wheels”: binary distributions
 - To create them, first `pip install wheel`
-

TYPES OF WHEEL

- Wheels are created using

```
python setup.py bdist_wheel
```

- If your code is pure Python, a platform-agnostic wheel is built (still useful for faster installation!)

```
exampy-0.2-py3-none-any.whl
```

- With compiled code, wheel becomes specific to

- Python version

```
exampy-0.2-cp37-cp37m-macosx_10_9_x86_64.whl
```

- CPython version (typically the same...)

- Operating system
-

WHEELS AND `pip`

- You can upload wheels using the same command as before, e.g.,

```
twine upload dist/* --skip-existing
```

- When users then `pip install YOUR_PACKAGE`, if the Python version and operating system match the wheel, the wheel is installed, which is fast
-

BUILDING AND HOSTING
DOCUMENTATION ON `readthedocs.io`

DOCUMENTATION RECAP

- Two weeks ago, we discussed the basics of documentation and how to create documentation using sphinx
 - `docs/` directory in your top-level package directory
 - `docs/source/` with documentation source
 - Documentation consists of `.rst` files (or jupyter notebooks), API using autodoc
 - with sphinx do `make html/pdflatex/...` to make the documentation
 - Skipped how to host the documentation online, will cover this now
-

GETTING STARTED WITH `readthedocs.io`

- Go to <https://readthedocs.io/> and sign up with GitHub
 - Go to your dashboard and click to import a project
 - Find the project you want to import from your GitHub repositories
 - Then you get to the admin page for your documentation
-

- [Overview](#)
- [Downloads](#)
- [Search](#)
- [Builds](#)
- [Versions](#)
- [Admin](#)

Versions

[Edit](#)

Build a version

[Build version](#)

Click to build first version
(will fail, but will connect to GitHub)

Repository

<https://github.com/jobovy/exampy.git>

Project Slug

exampy

Last Built

No builds yet

Maintainers



Badge

[docs](#) [unknown](#) [i](#)

Tags

Project has no tags. Add some in your [project settings](#).

Project Privacy Level

Public

Short URLs

exampy.readthedocs.io

exampy.rtd.io

CONFIGURING YOUR DOCUMENTATION ON `readthedocs.io`

- Add a file `.readthedocs.yml` to your repository with the configuration parameters

- E.g.,

```
version: 2

sphinx:
  configuration: docs/source/conf.py

python:
  version: 3
  install:
    - method: pip
      path: .
```


exampy

Navigation

Contents:

[Installation instructions](#)

[Introduction](#)

[API reference](#)

Quick search

```
function countSpaces(str) {  
  // what's missing?  
}  
a) return str.indexOf(' ');  
b) return str.split(' ').length - 1;  
c) return str.count(' ') - 1;  
d) return str.words().length - 1;
```

TAKE THE QUIZ!

Beat Triplebyte's online coding quiz. Get offers from top companies. Skip resumes & recruiters.

Sponsored · Ads served ethically

Welcome to exampy's documentation!

Contents:

- [Installation instructions](#)
 - [Dependencies](#)
 - [Installation](#)
- [Introduction](#)
- [API reference](#)
 - [exampy](#)
 - [exampy.integrate](#)

Indices and tables ¶

- [Index](#)
- [Module Index](#)
- [Search Page](#)

MORE DETAILED CONFIGURATIONS

- If you need to install additional Python packages to build your documentation (e.g., `nbsphinx` if you include `jupyter` notebooks)
 - Add a `docs/requirements.txt` file (not `requirements.txt`, because then that would appear to users as the dependencies necessary to run your package, but these are to build the docs)
 - Then use `.readthedocs.yml`:
-

MORE DETAILED CONFIGURATIONS

```
version: 2
```

```
sphinx:
```

```
  configuration: docs/source/conf.py
```

```
python:
```

```
  version: 3
```

```
  install:
```

```
    - method: pip
```

```
      path: .
```

```
    - requirements: docs/requirements.txt
```
