Python Open Source Projects Setup

Python Milano Meetup 2016-03-23

Speakers: @cstrap - @tw_lgiordani

Assaggio di Pycon7 alla Milanese

Pycon Sette
Firenze
15-17 Aprile 2016

https://www.pycon.it/

SABATO 2 APRILE 2016
% VENINI 42 (MIKAMAI)

Agenda

- Which Python?
- pip
- virtualenv(wrapper)
- package patterns
- cookiecutter
- bumpversion
- PyPI
- tox

Which Python?

Python 2 or Python 3?
Python 2 ending support on 2020.

Install different interpreters:

- https://github.com/yyuu/pyenv lets you easily switch between multiple versions of Python. It's simple, unobtrusive, and follows the UNIX tradition of single-purpose tools that do one thing well.

Compatibility solutions:

- http://python-future.org is the missing compatibility layer between Python 2 and Python 3. It allows you to use a single, clean Python 3.x-compatible codebase to support both Python 2 and Python 3 with minimal overhead.
- `Six` provides simple utilities for wrapping over differences between Python 2 and Python 3. It is intended to support codebases that work on both Python 2 and 3 without modification. six consists of only one Python file, so it is painless to copy into a project. https://pythonhosted.org/six/

pip

`pip` is a tool for installing and managing Python packages, such as those found in the Python Package Index (PyPI).

`pip` is already installed if you're using Python 2 >=2.7.9 or Python 3 >=3.4 downloaded from python.org, but you'll need to upgrade pip.

https://pip.pypa.io/en/stable/

```
$ pip install SomePackage

$ pip install --upgrade package-name

$ pip uninstall package-name

$ pip search "query" # yolk or pipdeptree are two
packages to view dependencies

$ pip freeze > requirements.txt

$ pip install -r requirements.txt
```

```
pip install /path/to/SomePackage.1.1.1.tar.gz

pip install http://myrepo.it/SomePakage-1.4.0.zip

pip install -e \
     git+http://github.com/django/django.git#egg=django

pip install -e path/to/SomePackage

pip install django>=1.7 # ==, >=, >, <, <=
```

virtualenv

`virtualenv` is a tool to create isolated Python environments.
Every virtualenv has pip installed in it automatically. Does not require root access or modify your system.

Why?

Isolation - Python packages and even version live in their own space
Permissions - No sudoers, the environment is mine!
Organization - each project can maintain its own requirements file of Python packages
No-Globalization - don't require installing stuff globally on the system.

https://virtualenv.readthedocs.org/en/latest/

```
$ virtualenv ENV
      # This creates a folder ENV in the $PWD
      # You'll find python packages on
      #    ENV/lib/pythonX.X/site-packages

$ virtualenv ENV --python=/path/to/bin/python
      # Doesn't inherit global site-packages
      # Use a different Python interpreter

$ cd ENV ; . bin/activate
(ENV) $ pip install django
 Downloading/unpacking django
      ...
```

`virtualenvwrapper` is is a set of extensions to `virtualenv` tool. https://virtualenvwrapper.readthedocs.org/en/latest/

- Organizes all of your virtual environments in one place.
- Wrappers for managing your virtual environments (create, delete, copy).
- Use a single command to switch between environments.
- Tab completion for commands that take a virtual environment as argument.
- User-configurable hooks for all operations.
- Plugin system for more creating sharable extensions.

entropiae 12:08 PM

L'impostazione generale dei progetti non rispetta lo Zen di Python 😢

"There should be one— and preferably only one —obvious way to do it."

```
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

Package Patterns

keep test code in a separate root directory, commonly used for standalone modules as it allows us to distribute the code and tests together

```
root
|
+- package
|   |
|   +- file1
|   +- file2
|
+- tests
    |
    +- test_file1
    +- test_file2
```

```
python_boilerplate
|
+- python_boilerplate
|   |
|   +- __init__.py
|   +- python_boilerplate.py
|
+- tests
    |
    +- __init__.py
    +- test_python_boilerplate.py
```

keep test code as a submodule of the main code and has an advantage when the application has to be packaged without the test code (e.g. deploying to production servers or distributing to customers)

```
root
|
+- package
    |
    +- file1
    +- file2
    +- tests
        |
        +- test_file1
        +- test_file2
```

```
src
|
+- python_boilerplate
    |
    +- __init__.py
    +- python_boilerplate.py
    +- tests
        |
        +- __init__.py
        +- test_python_boilerplate.py
```

Excerpt From: “Test-Driven Python Development.”

cookiecutter

`cookiecutter` is a command-line utility that creates projects from **cookiecutters** (project templates).

Some features:

Cross-platform: Windows, Mac, and Linux are officially supported.
Works with Python 2.7, 3.3, 3.4, 3.5, and PyPy. (But you don't have to know/write Python code to use Cookiecutter.)
Project templates can be in any programming language or markup format: Python, JavaScript, Ruby,...
Simple command line usage.

https://github.com/audreyr/cookiecutter

```
$ pip install cookiecutter
(env) $ cookiecutter gh:audreyr/cookiecutter-pypackage
...

$ cat ~/.cookiecutterrc
default_context:
     full_name: "Antani Tatablinda"
     email: "come.se.fosse@ntani.org"
     github_username: "antani"
cookiecutters_dir: "~/.cookiecutters/"
```

There're some **cookiecutters** (aka Cookiecutter project templates) available for Python, Python-Django, C, C++, C#, Common Lisp, JavaScript, Kotlin, LaTeX/XeTeX, Berkshelf-Vagrant, HTML, Scala, 6502 Assembly

Similar projects: Paste, Diecutter, Django's startproject and startapp commands, python-packager, Yeoman, Pyramid's pcreate command, mr.bob, grunt-init, scaffolt, init-skeleton, Cog, Skaffold and maybe others :-)

Dissecting a package

```
python_boilerplate
+- AUTHORS.rst
+- CONTRIBUTING.rst
+- HISTORY.rst
+- LICENSE
+- MANIFEST.in
+- Makefile
+- README.rst
+- python_boilerplate
|   |
|   +- __init__.py
|   +- python_boilerplate.py
+- requirements_dev.txt
+- setup.cfg
+- setup.py
+- tests
|   |
|   +- __init__.py
|   +- test_python_boilerplate.py
+- tox.ini
+- travis_pypi_setup.py
```

Let’s analyze all files created by **cookiecutter**

Dissecting a package

```
python_boilerplate
+- AUTHORS.rst
+- CONTRIBUTING.rst
+- HISTORY.rst
+- LICENSE
+- MANIFEST.in
+- Makefile
+- README.rst
+- python_boilerplate
|   |
|   +- __init__.py
|   +- python_boilerplate.py
+- requirements_dev.txt
+- setup.cfg
+- setup.py
+- tests
|   |
|   +- __init__.py
|   +- test_python_boilerplate.py
+- tox.ini
+- travis_pypi_setup.py
```

Info on authors, how to contribute, history and license. Some infos provided by cookiecutter.

Dissecting a package

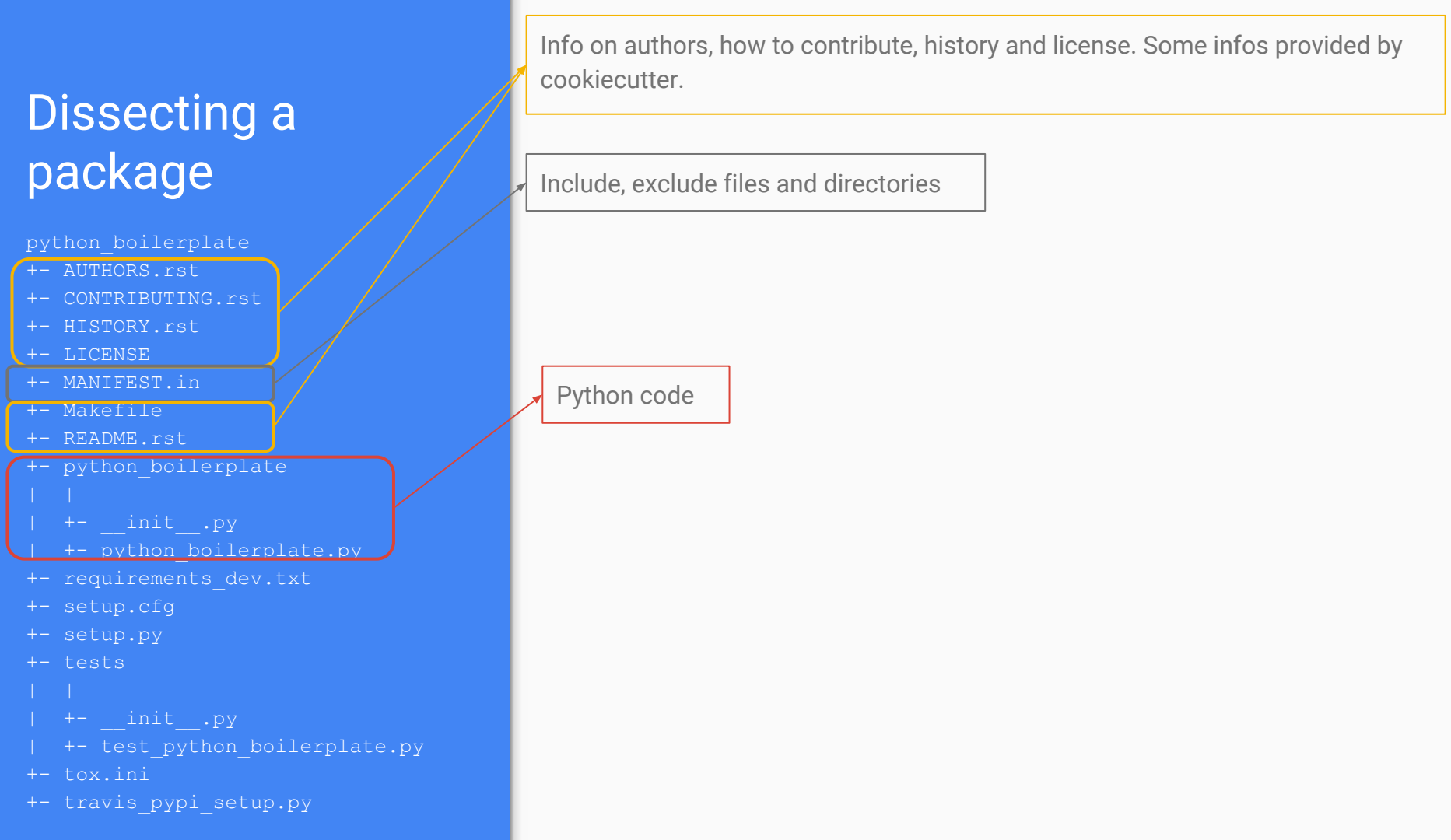
```
python_boilerplate
+- AUTHORS.rst
+- CONTRIBUTING.rst
+- HISTORY.rst
+- LICENSE
+- MANIFEST.in
+- Makefile
+- README.rst
+- python_boilerplate
|   |
|   +- __init__.py
|   +- python_boilerplate.py
+- requirements_dev.txt
+- setup.cfg
+- setup.py
+- tests
|   |
|   +- __init__.py
|   +- test_python_boilerplate.py
+- tox.ini
+- travis_pypi_setup.py
```

Info on authors, how to contribute, history and license. Some infos provided by cookiecutter.

Include, exclude files and directories

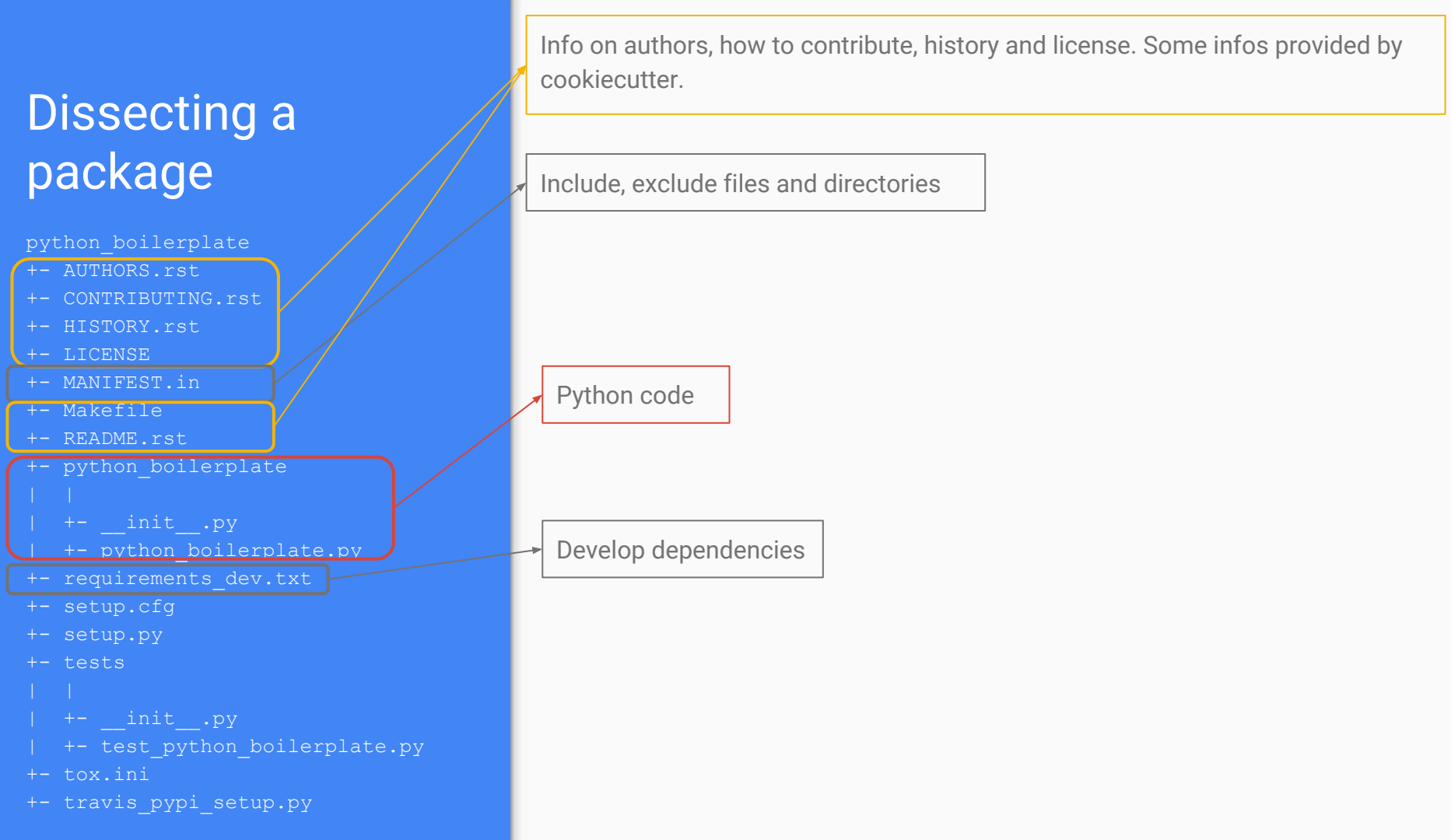
Dissecting a package

```
python_boilerplate
+- AUTHORS.rst
+- CONTRIBUTING.rst
+- HISTORY.rst
+- LICENSE
+- MANIFEST.in
+- Makefile
+- README.rst
+- python_boilerplate
|   |
|   +- __init__.py
|   +- python_boilerplate.py
+- requirements_dev.txt
+- setup.cfg
+- setup.py
+- tests
|   |
|   +- __init__.py
|   +- test_python_boilerplate.py
+- tox.ini
+- travis_pypi_setup.py
```



Dissecting a package

```
python_boilerplate
+- AUTHORS.rst
+- CONTRIBUTING.rst
+- HISTORY.rst
+- LICENSE
+- MANIFEST.in
+- Makefile
+- README.rst
+- python_boilerplate
|   |
|   +- __init__.py
|   +- python_boilerplate.py
+- requirements_dev.txt
+- setup.cfg
+- setup.py
+- tests
|   |
|   +- __init__.py
|   +- test_python_boilerplate.py
+- tox.ini
+- travis_pypi_setup.py
```



Dissecting a package

```
python_boilerplate
+- AUTHORS.rst
+- CONTRIBUTING.rst
+- HISTORY.rst
+- LICENSE
+- MANIFEST.in
+- Makefile
+- README.rst
+- python_boilerplate
|   |
|   +- __init__.py
|   +- python_boilerplate.py
+- requirements_dev.txt
+- setup.cfg
+- setup.py
+- tests
|   |
|   +- __init__.py
|   +- test_python_boilerplate.py
+- tox.ini
+- travis_pypi_setup.py
```

Info on authors, how to contribute, history and license. Some infos provided by cookiecutter.

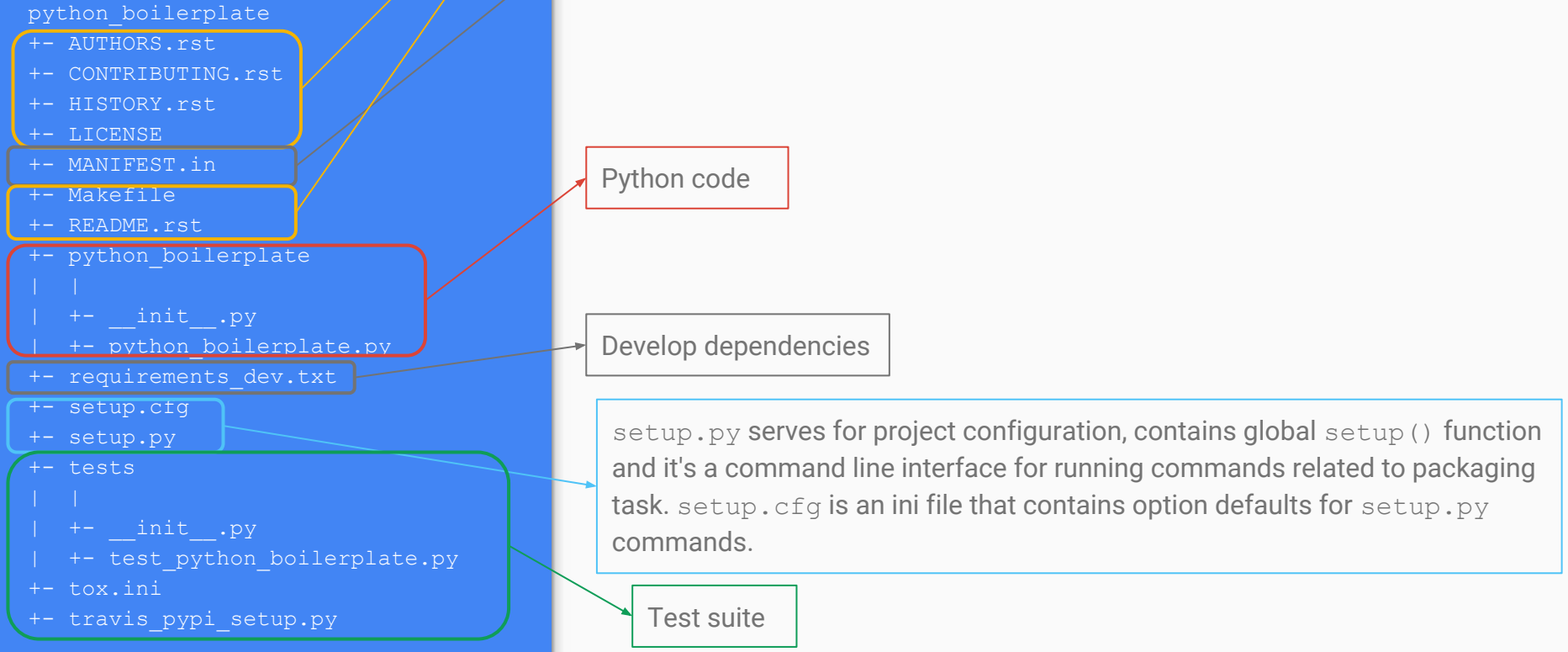
Include, exclude files and directories

Python code

Develop dependencies

`setup.py` serves for project configuration, contains global `setup()` function and it's a command line interface for running commands related to packaging task. `setup.cfg` is an ini file that contains option defaults for `setup.py` commands.

Dissecting a package



Dissecting a package

```
python_boilerplate
+- AUTHORS.rst
+- CONTRIBUTING.rst
+- HISTORY.rst
+- LICENSE
+- MANIFEST.in
+- Makefile
+- README.rst
+- python_boilerplate
|   |
|   +- __init__.py
|   +- python_boilerplate.py
+- requirements_dev.txt
+- setup.cfg
+- setup.py
+- tests
|   |
|   +- __init__.py
|   +- test_python_boilerplate.py
+- tox.ini
+- travis_pypi_setup.py
```

Info on authors, how to contribute, history and license. Some infos provided by cookiecutter.

Include, exclude files and directories

Because we want write docs

```
+- docs
|   |
|   +- Makefile
|   +- authors.rst
|   +- conf.py
|   +- contributing.rst
|   +- history.rst
|   +- index.rst
|   +- installation.rst
|   +- make.bat
|   +- readme.rst
|   +- usage.rst
```

Python code

Develop dependencies

`setup.py` serves for project configuration, contains global `setup()` function and it's a command line interface for running commands related to packaging task. `setup.cfg` is an ini file that contains option defaults for `setup.py` commands.

Test suite

bumpversion

`bumpversion` is a small command line tool to simplify releasing software by updating all version strings in your source code by the correct increment. Also creates commits and tags.

Features:

- version formats are highly configurable
- works without any VCS, but happily reads tag information from and writes commits and tags to Git and Mercurial if available
- just handles text files, so it's not specific to any programming language.

https://github.com/peritus/bumpversion/

```
$ pip install --upgrade bumpversion

$ bumpversion patch
$ bumpversion minor
$ bumpversion major
```

PyPI

`PyPI` is a repository of software for Python and currently count more than 76000 packages.

The PyPI repository provides alternative locations that store the packages.

You can create your own mirror, following the PEP 381 or using a tool such as `pep381client`

You can host your own private repository: e.g. `pypiserver` can easly do the job.

https://pypi.python.org/pypi

```
$ virtualenv cheeseshop
$ cd cheeshop ; . bin/activate
(cheeseshop) $ pip install pypiserver passlib
...
(cheeseshop) $ mkdir packages
(cheeseshop) $ pypi-server -p 8080 ~/packages
```

```
(env) $ pip install  --extra-index-url http://localhost:8080/simple/...
```

```
$ cat .pypirc
[distutils]
index-servers =
      pypi
      myPyPi

[pypi]
repository=https://pypi.python.org/pypi
username=
password=

[myPyPi]
repository=http://csgs1.pythonanywhere.com
username=foobar
password=barfoo
```

Demo

with disclaimer ;-)

Thanks!

Contact us:

@cstrap on twitter, github, bitbucket, linkendIn

Next: @tw_lgiordani with `tox` and testing