# Virtualenv, Packaging and Useful Tools

March 30, 2015

Department of Aerospace Engineering, IIT Bombay

**Author:** Prabhu Ramachandran

## Agenda

- Introduction to virtual env and its use
- Python packaging
    - What are Packages?
    - Distributing Packages
    - PyPI
- Hosting documentation
- Continuous integration
- Test coverage

## Preamble: what's a package?

A collection of Python modules organized into a hierarchy:

```
$ tree jedi
 jedi/
 ├── __init__.py
 ├── api.py
 ├── builtin.py
 ...
 ├── debug.py
 ├── refactoring.py
 └── settings.py
```

## Virtualenvs

A self-contained virtual Python environment which **you** control!

- Requires a "system" Python distribution

## Motivation

- Need to install Python packages
- No root access
- Don’t want to mess up system
- Create “isolated” environment

## Virtualenv

- www.virtualenv.org
- pypi.python.org/pypi/virtualenv
- Either install virtualenv
- Or use the virtualenv.py

```
$ easy_install virtualenv
```

Or download tarball and:

```
$ python setup.py install [--prefix=/usr/local]
```

## Environments

```
$ python virtualenv.py --help
```

```
$ python virtualenv.py ENV
```

```
$ virtualenv --system-site-packages ENV
```

- `ENV` is a directory name of your choice
- Self-contained universe
- Can inherit system packages
- Look in `ENV/bin` (or `ENV\Scripts`)
- Look in `ENV/lib` (or `ENV\Lib`)

## Activation

```
# Activation (Linux/Mac OSX)
$ source ENV/bin/activate
```

```
$ ENV\Scripts\activate.bat
```

```
# Deactivate
(ENV)$ deactivate
```

## Usage

```
(ENV)$ deactivate
$ python virtualenv.py ANOTHER
(ANOTHER)$ source ANOTHER/bin/activate
(ANOTHER)$ pip install PKG
# installs in ANOTHER

(ANOTHER)$ deactivate
$ source ENV/bin/activate

# To remove ANOTHER
$ rm -rf ANOTHER
```

Easy and convenient!

## Python Packages

- Organized hierarchy of modules
- Can include data
- Documentation
- Installation and distribution

## Installing packages

```
$ pip install requests

$ pip list

$ pip uninstall requests
```

## Making your own

Let us look at a simple package

```
$ tree my_project
my_project/
├── README.md
└── sees
    ├── __init__.py
    └── hello.py
```

## Packaging the package

- We want to "install" this
- Want others to install it
- Want to share it
- Make sure you have `setuptools` available

```
$ python -c "import setuptools"
```

No import error means you are good!

## Writing a `setup.py`

Create `setup.py`

```
from setuptools import setup, find_packages

setup(
    name="sees",
    version="0.1",
    description="Utility code for SEES course",
    author="FOSSEE developers",
    author_email="python@fossee.in",
    packages=find_packages(),
)
```

## Using it

```
$ python setup.py --help
$ python setup.py sdist
$ python setup.py develop
$ python setup.py install
```

## More on distutils

- Many more options
- Support to compile extensions
- Including package data
- See setuptools documentation
- See distutils documentation.

## Distributing and PyPI

- Register on http://pypi.python.org

```
$ python setup.py register
```

- Create a `~/.pypirc`

```
[pypirc]
servers = pypi
[server-login]
username:your_awesome_username
password:your_awesome_password
```

## Hosting Documentation

- Bitbucket/Github
- Better: http://www.readthedocs.org
    - http://read-the-docs.readthedocs.org/

## Continuous integration

- Run tests automatically
    - On each commit
    - For each PR
    - For all branches
- Potentially on multiple platforms

## Continuous integration

- Jenkins/Hudson
- Travis-ci.org
- Drone.io
- Shippable.com
- Appveyor

## Travis-CI example

- Uses a YAML file for config
- Most others are similar
- Look at some examples

## Additional Reading

Very nice document with examples:

https://pythonhosted.org/an_example_pypi_project/index.html