Introduction to setuptools, Eggs, and Easy Install

Jim C. McDonald

Michigan Python Users Group

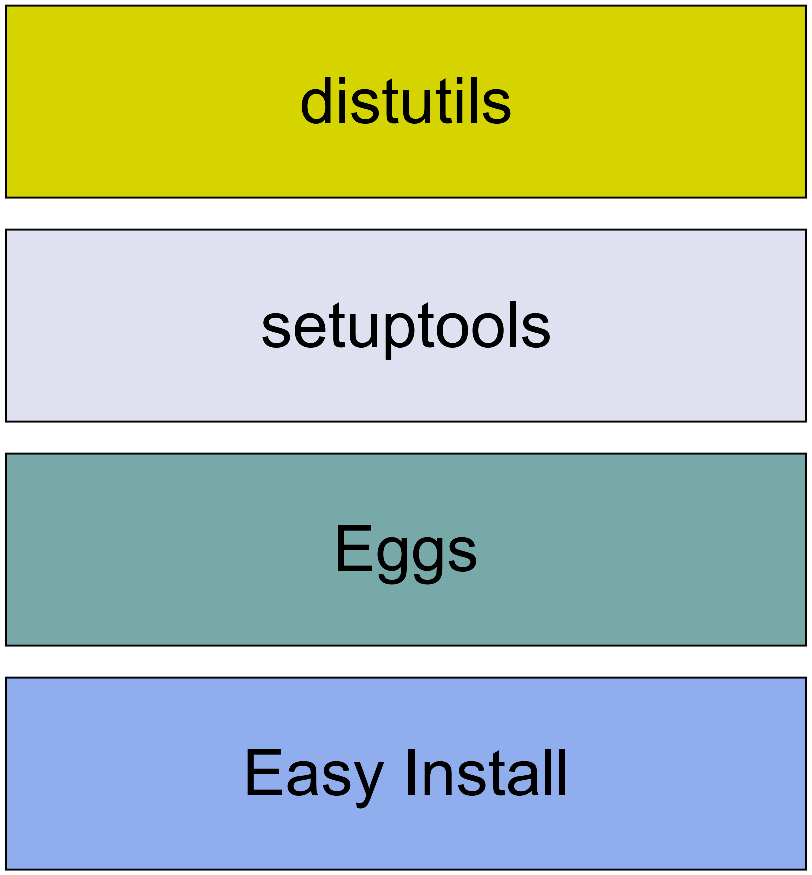
February 2, 2006

Attributions

Most of the material in this presentation
is a repackaging of information in the following documents.
Any errors in the presentation are my own.

- Phillip J. Eby and the Official Eggs, setuptools, Easy Install documentation http://peak.telecommunity.com/
- Ian Bicking Python Packaging with SetupTools http://ianbicking.org/docs/setuptools-presentation
- Titus Brown Blog: The 30-second Guide to Making Eggs http://www.advogato.org/person/titus/diary.html?start=148

The Ensemble



distutils

- Part of the Python standard library since version 1.6
- The standard way of building and installing packages
- Primary functionality in `distutils.core`
- Developer or packager creates `setup.py`

```
#!/usr/bin/env python
from distutils.core import setup
setup (name = "foo",
       version = "1.0",
       py_modules = ["foo"])

$ python setup.py sdist     (create a source distribution)
$ python setup.py bdist     (create a build distribution)
$ python setup.py install   (install using defaults)

* Other --install-* options (remember to update $PYTHONPATH)
```

setuptools

- A collection of enhancements to the Python `distutils` package that allow one to more easily build and distribute python packages
- Additional set of keyword arguments to `setup()`
- Includes `easy_install.py`
- Creates eggs (.egg)
- Features for developers (e.g. support for data files, MANIFEST, Pyrex, PyPI upload,…)
- Ability to deploy project in “development mode” via `setup.py develop` command

Eggs

“Eggs are to Pythons as Jars are to Java…”

Phillip J. Eby

- Single-file importable distribution format
- Eggs are Zipfiles using the .egg extension, that support including data and C extensions as well as Python code
- Requires Python 2.3 or above
- Eggs are built using the `setuptools` package
- The published plan is to propose inclusion in the Python 2.5 standard library

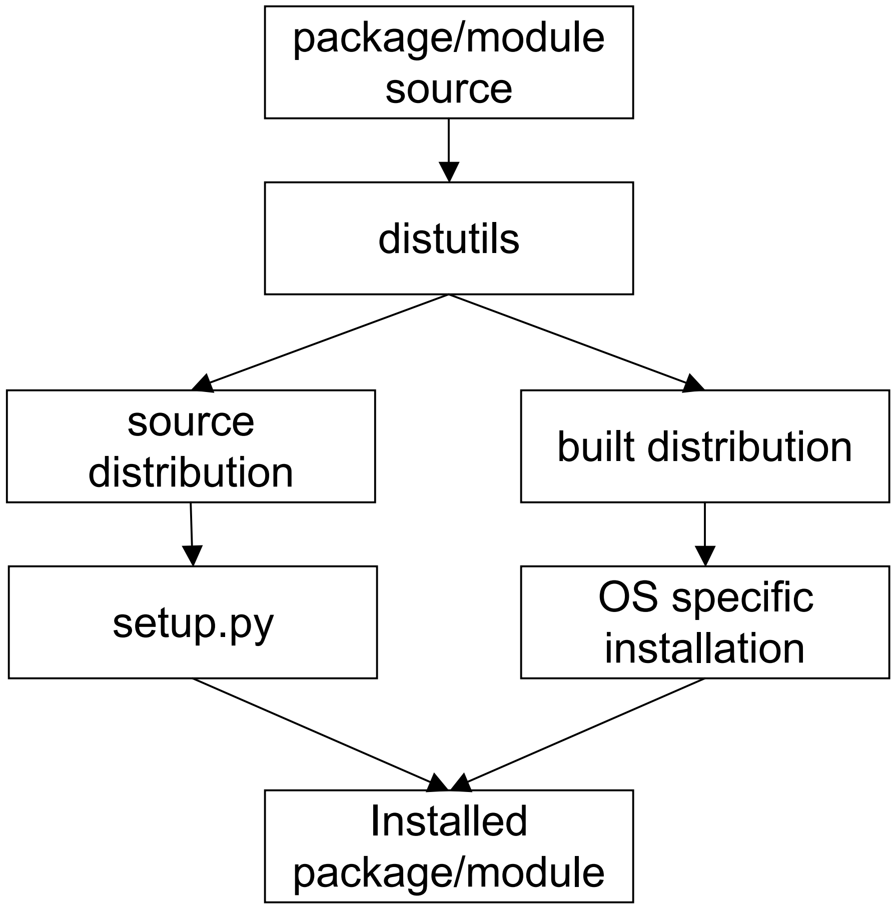
Why bother with Eggs?

- Enable tools like “Easy Install” Python package manager
- They are a “zero installation” format for pure Python packages (put them on `PYTHONPATH` or `sys.path`)
- They can include package metadata (e.g. dependencies)
- They allow namespace packages (packages that contain other packages) to be split into separate distributions
- They allow applications or libraries to specify the needed version of a library before doing an import (e.g. `require(“Twisted-Internet>=2.0”)`)
- They provide a framework for plug-ins (similar to Eclipse’s extension point)
- Enables one to distribute a project that depends on other software available via PyPI a.k.a. Cheese Shop

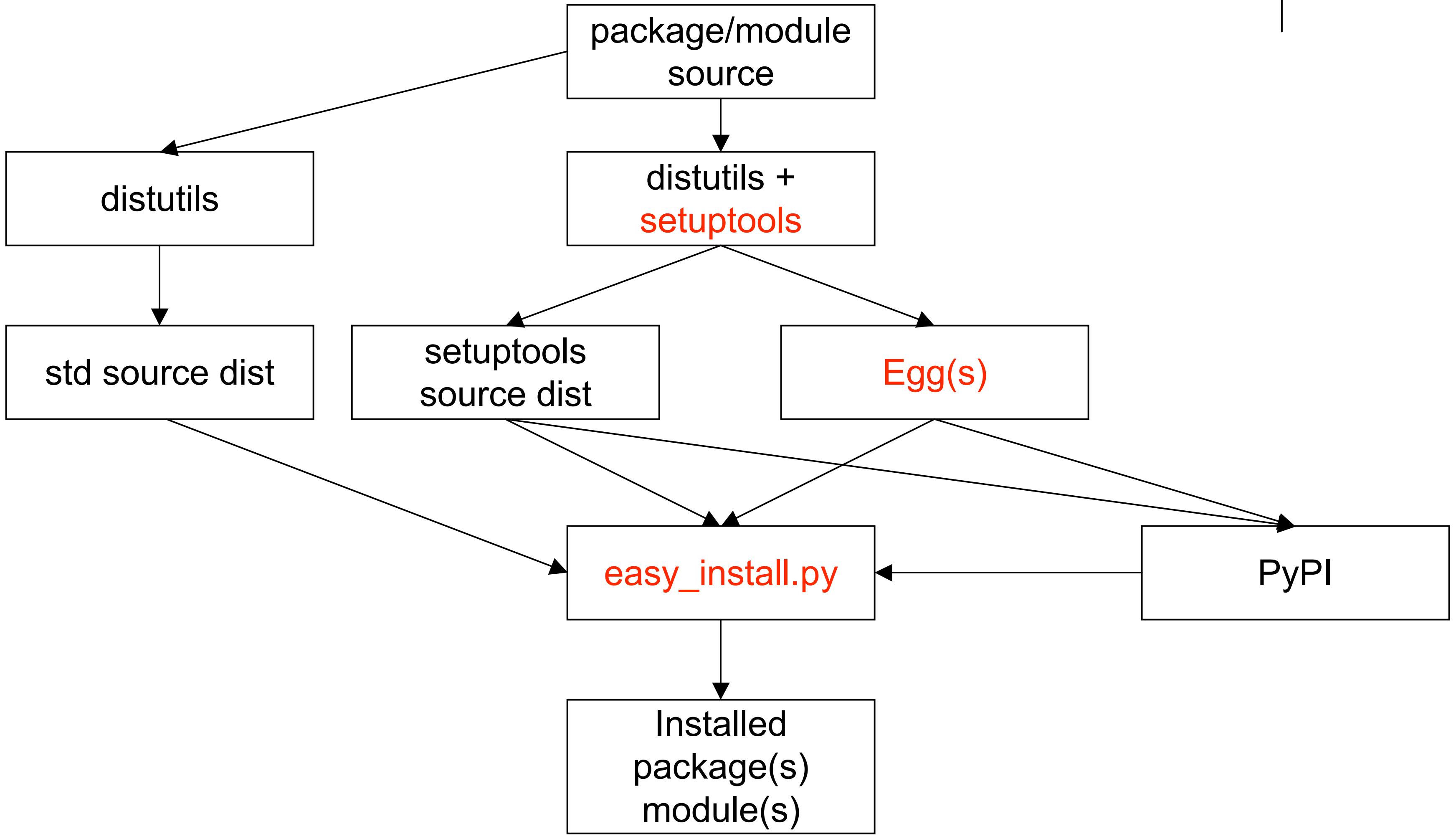
Easy Install / `easy_install.py`

- A Python module bundled with `setuptools` that lets one automatically build, install, and manage python packages
- Part of the `setuptools` package
- Installs any distutils-based package
- Can find packages on PyPI
- Handles dependencies via arguments to `setup()` (e.g. `install_requires = [‘foo>=1.4’,’Bar’]`)

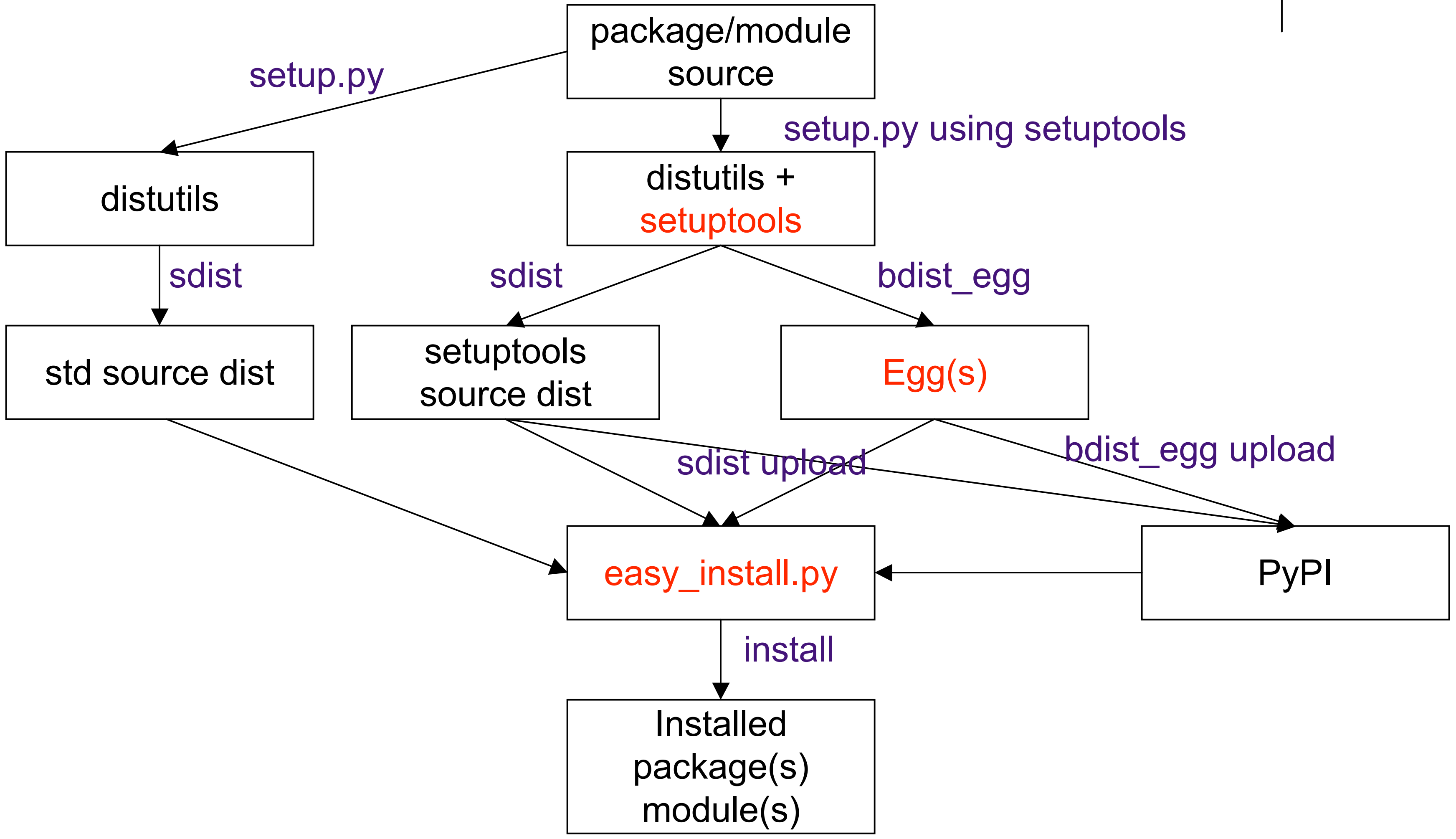
distutils-based workflow



setuptools-based workflow



setuptools-based workflow



How To Install setuptools

1. Download ez_setup.py (http://peak.telecommunity.com/dist/ez_setup.py)

2. Run `ez_setup.py` to download and install the `setuptools` egg

```
  $ python ez_setup.py
```

After installation take a look at

```
   /site-packages/easy-install.pth
```

Recommended Reference: Bob Ippolito's Using .pth for Python Development
http://bob.pythonmac.org/archives/2005/02/06/using-pth-files-for-python-development/

Easy Install Examples

Example 1. Install a package by name, searching PyPI for the latest version, and automatically downloading, building, and installing it:

```
$ easy_install SQLObject
```

Example 2. Install or upgrade a package by name and version by finding links on a given "download page":

```
$ easy_install -f http://pythonpaste.org/package_index.html SQLObject
```

Example 3. Download a source distribution from a specified URL, automatically building and installing it:

```
$ easy_install http://example.com/path/to/MyPackage-1.2.3.tgz
```

Example 4. Install an already-downloaded .egg file:

```
$ easy_install /my_downloads/OtherPackage-3.2.1-py2.3.egg
```

More Easy Install Examples

Example 5. Upgrade an already-installed package to the latest version listed on PyPI:

```
$ easy_install --upgrade PyProtocols
```

Example 6. Install a source distribution that's already downloaded and extracted in the current directory:

```
$ easy_install .
```

Example 7. Find a source distribution or Subversion checkout URL for a package, and extract it or check it out to ~/projects/sqlobject (the name will always be in all-lowercase), where it can be examined or edited. (The package will not be installed, but it can easily be installed with easy_install ~/projects/sqlobject. :

```
$ easy_install --editable --build-directory ~/projects SQLObject
```

setuptools/Eggs Example

Titus Brown's The 30-second Guide to Making Eggs
http://www.advogato.org/person/titus/diary.html?start=148

```
Add the following to your setup.py file

### ez_setup.py and 'use_setuptools()' will automagically download and
### install setuptools. The downside to this code is that then
### you need to include ez_setup.py in your distribution, too.

try:
   from ez_setup import use_setuptools
   use_setuptools()
except ImportError:
   pass

### this is the critical line
from setuptools import setup # instead of the 'distutils.core' setup

### also import Extension, etc -- anything else you need -- from setuptools.
```

setuptools/Eggs Example

Titus Brown’s The 30-second Guide to Making Eggs (continued)
http://www.advogato.org/person/titus/diary.html?start=148

```
then you can make eggs with the bdist_egg command. Try:

% python2.3 setup.py bdist_egg
% python2.4 setup.py bdist_egg

to build eggs for each version of Python you have installed.

The eggs will end up in build/. If you're distributing precompiled
binary code, you'll need to make an egg for each platform/Python version.
```

Eggs from distutils pkgs

Often eggs may be built from `distutils` source distributions (distributions that do not import from `setuptools`).

Recipe for building eggs from `disutils` packages in Python 2.4 or higher:

```
$ python setup.py --command-packages=setuptools.command bdist_egg
```

Per the documentation this will work for most packages.
- YMMV

pkg_resources module

- Need `pkg_resources` module to use eggs
  - Provides runtime support for eggs
  - API for automatic locating eggs and their dependencies and adding them to `sys.path` at runtime
- Allows one to install and keep multiple versions of the same package on your system
- If dependency is not met will raise a `DistributionNotFound` exception

Automatic Discovery

Place an egg in a directory already on `sys.path` such as `site-packages`:

```
from pkg_resources import require
require(“FooBar>=1.2”)
```

`pkg_resources` understands typical version numbering schemes

Additional setuptools topics

- Using `setuptools` in “Development Mode” including running eggs from source
- Namespace Packages
- Accessing Package Resources (including data files)
- Specifics of Declaring Dependencies
- `setuptools.find_packages()`
- Recipes for custom installations
- Review of Plug-in architecture, discussion of how to build pluggable software
- Runtime API

Key Messages

- [setuptools, Eggs, Easy Install] leverage and enhance the value of `distutils` and PyPI
- There is a low threshold of effort beyond `distutils` to begin to use [setuptools, Eggs, Easy Install]
- [setuptools, Eggs, Easy Install] offer substantive, valuable improvements over `distutils` alone

Questions?