

Programming in Python

Week 8

Installing stuff

Syllabus:

http://briandorsey.github.com/uwpython_onsite/

Brian Dorsey
brian@dorseys.org
<http://briandorsey.info>

Introductions,
comments
(...)

assignment

wow

there sure are a lot of
ways to install modules

tools used:
pip
virtualenv
setup.py
apt-get
easy_install
make

mix of success and failure

sometimes

fail

fail

fail

success

interesting side note:

I had never heard of *any*
of the modules you
installed.

Python is glue with
superpowers

you can connect
anything with it

lots more discussion of
installation / distribution
later today

Unit A (25) debugging

debugging techniques?

interactive debuggers
print statements
logging
tests

interactive debuggers

not an expert here,
I almost never use them

pdb

**in stdlib
command line
local
in process**

demo

gui debuggers

nearly all IDEs include one

WingIDE for example:

[http://wingware.com/
wingide/debugger](http://wingware.com/wingide/debugger)

more

getting started with pdb (blog post)

<http://pythonconquerstheuniverse.wordpress.com/2009/09/10/debugging-in-python/>

use pdb to debug Django (screencast)

<http://ericholscher.com/blog/2008/aug/31/using-pdb-python-debugger-django-debugging-series-/>

python debugging techniques (blog post)

<http://aymanh.com/python-debugging-techniques>

print statements

tried and true

I use them all the time

with nose and py.test, you
can leave them in test
code forever

when things go wrong
again, you'll already have
a ton of detail printed

use print

logging

**“enterprise level print
statements”**

»

Table Of Contents

[15.7. logging — Logging facility for Python](#)

- [15.7.1. Logger Objects](#)
- [15.7.2. Handler Objects](#)
- [15.7.3. Formatter Objects](#)
- [15.7.4. Filter Objects](#)
- [15.7.5. LogRecord Objects](#)
- [15.7.6. LogRecord attributes](#)
- [15.7.7. LoggerAdapter Objects](#)
- [15.7.8. Thread Safety](#)
- [15.7.9. Module-Level Functions](#)
- [15.7.10. Integration with the warnings module](#)

Previous topic

[15.6. getopt — C-style parser for command line options](#)

Next topic

[15.8. logging.config — Logging configuration](#)

This Page

[Report a Bug](#)
[Show Source](#)

15.7. logging — Logging facility for Python

New in version 2.3.

This module defines functions and classes which implement a flexible event logging system for applications and libraries.

The key benefit of having the logging API provided by a standard library module is that all Python modules can participate in logging, so your application log can include your own messages integrated with messages from third-party modules.

The module provides a lot of functionality and flexibility. If you are unfamiliar with logging, the best way to get to grips with it is to see the tutorials (see the links on the right).

The basic classes defined by the module, together with their functions, are listed below.

- Loggers expose the interface that application code directly uses.
- Handlers send the log records (created by loggers) to the appropriate destination.
- Filters provide a finer grained facility for determining which log records to output.
- Formatters specify the layout of log records in the final output.

15.7.1. Logger Objects

Loggers have the following attributes and methods. Note that Loggers are never instantiated directly, but always through the module-level function `logging.getLogger(name)`.

Important

This page contains the API reference information. For tutorial information and discussion of more advanced topics, see

- [Basic Tutorial](#)
- [Advanced Tutorial](#)
- [Logging Cookbook](#)

logging module is
powerful, awesome,
and a bit annoying

»

Table Of Contents

15.7. `logging` — Logging facility for Python

- 15.7.1. Logger Objects
- 15.7.2. Handler Objects
- 15.7.3. Formatter Objects
- 15.7.4. Filter Objects
- 15.7.5. LogRecord Objects
- 15.7.6. LogRecord attributes
- 15.7.7. LoggerAdapter Objects
- 15.7.8. Thread Safety
- 15.7.9. Module-Level Functions
- 15.7.10. Integration with the warnings module

Previous topic

15.6. `getopt` — C-style parser for command line options

Next topic

15.8. `logging.config` — Logging configuration

This Page

[Report a Bug](#)
[Show Source](#)

15.7. `logging` — Logging facility for Python

New in version 2.3.

This module defines functions and classes which implement a flexible event logging system for applications and libraries.

The key benefit of having the logging API provided by a standard library module is that all Python modules can participate in logging, so your application log can include your own messages integrated with messages from third-party modules.

The module provides a lot of functionality and flexibility. If you are unfamiliar with logging, the best way to get to grips with it is to see the tutorials (see the links on the right).

The basic classes defined by the module, together with their functions, are listed below.

- Loggers expose the interface that application code directly uses.
- Handlers send the log records (created by loggers) to the appropriate destination.
- Filters provide a finer grained facility for determining which log records to output.
- Formatters specify the layout of log records in the final output.

15.7.1. Logger Objects

Loggers have the following attributes and methods. Note that Loggers are never instantiated directly, but always through the module-level function `logging.getLogger(name)`.

Important

This page contains the API reference information. For tutorial information and discussion of more advanced topics, see

- [Basic Tutorial](#)
- [Advanced Tutorial](#)
- [Logging Cookbook](#)

```
# myapp.py
import logging
import mylib

def main():
    logging.basicConfig(filename='myapp.log', level=logging.INFO)
    logging.info('Started')
    mylib.do_something()
    logging.info('Finished')

if __name__ == '__main__':
    main()
```

```
# mylib.py
import logging

def do_something():
    logging.info('Doing something')
```

If you run *myapp.py*, you should see this in *myapp.log*:

```
INFO:root:Started
INFO:root:Doing something
INFO:root:Finished
```

- **StreamHandler** instances send messages to streams (file-like objects).
- **FileHandler** instances send messages to disk files.
- **BaseRotatingHandler** is the base class for handlers that rotate log files at a certain point. It is not meant to be instantiated directly. Instead, use **RotatingFileHandler** or **TimedRotatingFileHandler**.
- **RotatingFileHandler** instances send messages to disk files, with support for maximum log file sizes and log file rotation.
- **TimedRotatingFileHandler** instances send messages to disk files, rotating the log file at certain timed intervals.
- **SocketHandler** instances send messages to TCP/IP sockets.
- **DatagramHandler** instances send messages to UDP sockets.
- **SMTPHandler** instances send messages to a designated email address.
- **SysLogHandler** instances send messages to a Unix syslog daemon, possibly on a remote machine.
- **NTEventLogHandler** instances send messages to a Windows NT/2000/XP event log.
- **MemoryHandler** instances send messages to a buffer in memory, which is flushed whenever specific criteria are met.
- **HTTPHandler** instances send messages to an HTTP server using either GET or POST semantics.
- **WatchedFileHandler** instances watch the file they are logging to. If the file changes, it is closed and reopened using the file name. This handler is only useful on Unix-like systems; Windows does not support the underlying mechanism used.
- **NullHandler** instances do nothing with error messages. They are used by library developers who want to use logging, but want to avoid the 'No handlers could be found for logger XXX' message which can be displayed if the library user has not configured logging. See

customized levels
string interpolation
on the fly config
etc, etc...

tests

test suites find bugs

Ever written repro steps which should lead directly to the source of the bug?

often get you pretty close
to the source of the bug

sometimes you can get
closer to the bug by
writing more tests

for heisenbugs this may
be your only option

heisenbug:
/hi-zen-buhg/, n.

A bug that disappears or alters its behavior when one attempts to probe or isolate it.

Lightning Talks!

(15)

Unit B (25) installation

a quick diversion

modules & packages

```
import sys  
from sys import path
```

demo

```
import xml
```

```
import xml.etree
```

```
__init__.py
```

packages
vs.
packaging

installing Python

usually already installed

maybe you can use it

system Python
vs
user installed Python

Windows is a special case

globally installed, shared,
user installed Python

...but you can install
multiple instances

every Python installation
has its' own stdlib and
site-packages folder

site-packages is the
default location for
installed packages

PyPI

catalog of packages

<http://pypi.python.org/pypi>

PyPI - the Python Package Index

The Python Package Index is a repository of software for the Python programming language. There are currently **21199** packages here.

To contact the PyPI admins, please use the [Get help](#) or [Bug reports](#) links.

Not Logged In

[Login](#)

[Register](#)

[Lost Login?](#)

[Use OpenID](#)

Get Packages

To use a package from this index either "`pip install package`" ([get pip](#)) or download, unpack and "`python setup.py install`" it. [Browse all packages](#) or use the search box above.

Package Authors

To submit a package use "[python setup.py upload](#)" ([full tutorial](#)). The index also [hosts documentation](#). You may submit packages using [SSH](#) or the [web form](#). You must [register](#).

Infrastructure

To interoperate with the index use the [JSON](#), [XML-RPC](#) or [HTTP](#) interfaces. Use the [mirror infrastructure](#) to download even if the primary site is unavailable.

Updated	Package	Description
2012-05-23	seriesoftubes 0.9.2	An extended pipeline for Solexa ChIP-seq data
2012-05-23	solr_cli 0.1	Command line client for Apache Solr
2012-05-23	pypi.trashfinder 0.0.4	This is a fork of zopyx.trashfinder, with modifications made to make integration with pythonpackages.com easier. It is not intended to be used outside of pythonpackages.com. For a command line trash finder, please install zopyx.trashfinder.
2012-05-23	fspyrpc 0.0.1	A simple RPC-framework written in Python
2012-05-23	raven 1.9.1	Raven is a client for Sentry (https://www.getsentry.com)
2012-05-23	rackspace-monitoring-cli 0.4.0	Command Line Utility for Rackspace Cloud Monitoring (MaaS)
2012-05-23	nerdy 0.01	
2012-05-23	django-modeldict 1.3.1	Stores a model as a dictionary
2012-05-23	Picatcha 0.1	Python library for Picatcha
2012-05-23	haiku-lang 0.0.6	An embedable LISP implemented on top of the Python interpreter.
2012-05-23	fatiando 0.0.1	Geophysical direct and inverse modeling
2012-05-23	metaTED 2.0.1	Creates metalink files of TED talks for easier downloading
2012-05-23	periscope_django 0.3	Periscope is an analysis tool for production SQL databases. This module provides the API for Periscope to communicate with your Django app.

installing
modules and
packages

python specific
vs
OS packaging

so...

just warning you...

I'm going to fail you
on this topic.

no easy answers

no single path to success

distutils
setup.py
setuptools
pip
virtualenv
distribute
distutils2
packaging

...

this is a
PUBLIC SERVICE ANNOUNCEMENT
from your friendly python packaging experts!

OLD and
BUSTED

setuptools

easy_install



distribute



pip

**NEW
HOTNESS**

Make the transition today!

```
$ curl -O http://python-distribute.org/distribute_setup.py  
$ python distribute_setup.py  
$ easy_install pip
```

You don't have to live with craptastic python packaging.

ATTENTION PYTHON COMRADES
NEW ORDERS FROM THE MINISTRY OF PACKAGING!

SONS AND DAUGHTERS OF THE GLORIOUS
PEOPLE'S PYTHONIC REPUBLIC
USE **DISTRIBUTE** AND **PIP**
THAT IS ALL.

THIS MESSAGE HAS BEEN APPROVED BY @JEZDEZ, HIGH CHANCELLOR OF PACKAGING





PYTHON KOMRADES!

НЕШ ØRDERS FRØM THE MINISTRY ØF PACKAGING

ИУЕТ: SETUPTOOLS / EASY_INSTALL

ДД: DISTRIBUTE / PIP

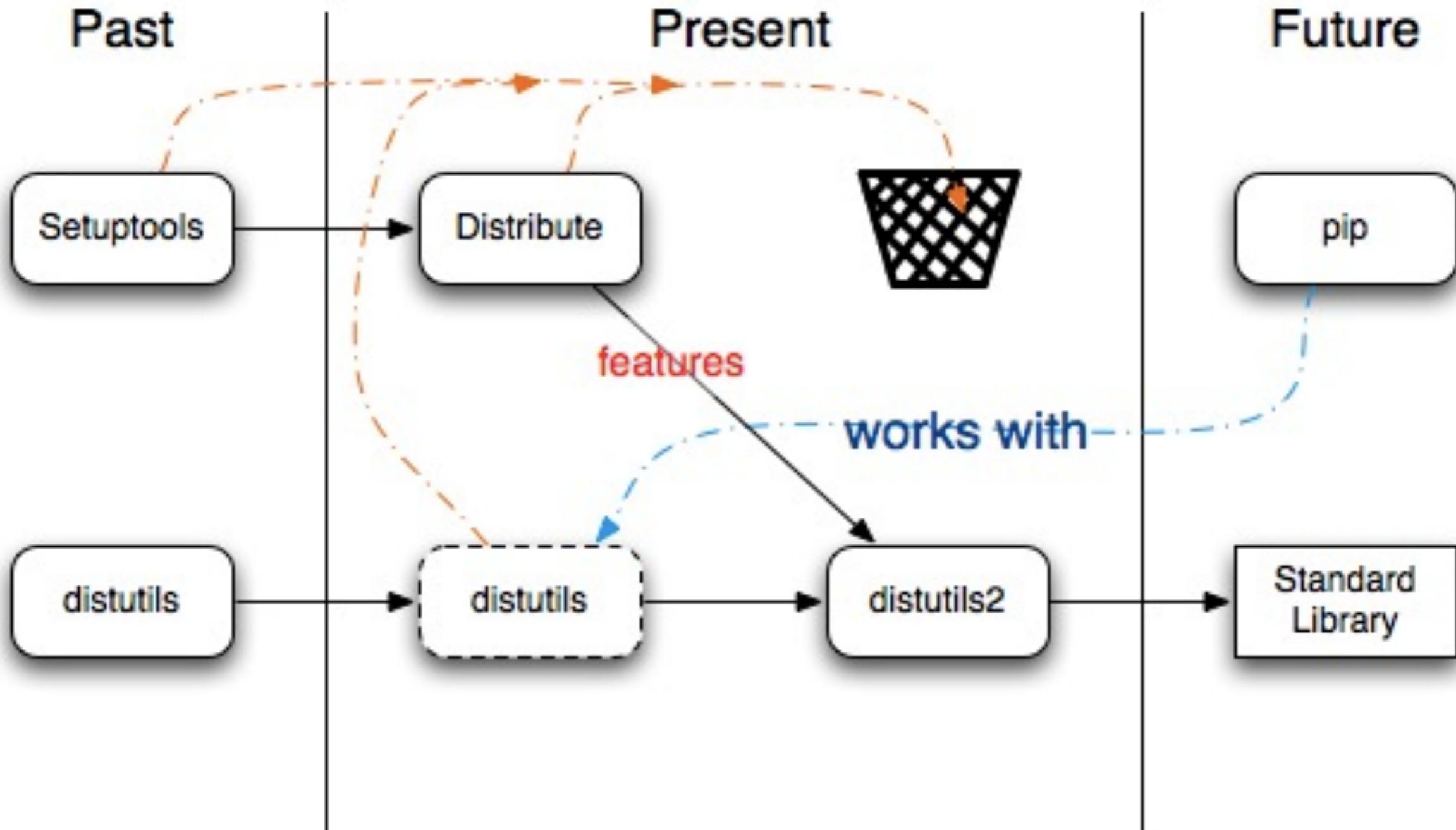
```
$ curl -O http://python-distribute.org/distribute_setup.py  
$ python distribute_setup.py  
$ easy_install pip
```

REMEMBER:

PØØR PACKAGING TØØLS TARNISH THE
BLAZING GLØRY ØF THE REPUBLIC

got it

Current State of Packaging



really?

timeline

hopefully this
will make some sense

issues

this is where most of the failures in your homework happened.

c modules

binary?

built at install time?

dependencies?

windows

binaries avail?
libraries assume unix?

osx

system python old?
permissions issues?
install to home dir?

linux

old system python?
python packaging vs.
system packaging fight?
install to home dir?

recommendations?

read the docs for
the package you
want to install

do what they say

pip install package

use virtualenv

IF you work on
multiple projects

keep an eye on

distutils2 / packaging /
pysetup

keep banging on it
until it works

Lab B
(20)

git pull

svn update

Lab B

- <http://docs.python.org/howto/logging.html#logging-basic-tutorial>
- Convert the print statements in logtacular.py to logging calls. Run the script and make sure you see logging output on the console.
- Wrap impossible_divide() in a try/except and log a full traceback on exceptions.
- bonus: Update the logging configuration to log to a file instead (or in addition to) the console.
- double bonus: Send yourself email for the exception case.

Unit C (25) distribution

**scripts
libraries
applications**

scripts

do this if you can

often you can just copy,
share, or check in the
script and call it good

when the script needs
more than just the stdlib
(or your company standard
environment)

you have an application,
not a script

libraries

when you read the
distutils docs, it's usually
libraries they're
talking about

for shared modules inside
an organization, consider
using source control for
sharing code

svn:externals

can pin to specific revisions as well: -r1234

**otherwise, it's time to
learn distutils**

applications

web apps

usually use virtualenv,
zc.buildout to setup
environment
(and maybe app)

VCS or package for main
application

web apps

also, I've seen
recommendations for
building system packages
of web apps for
deployment

desktop apps

system package system

or

bundle + installer

bundle?

basically:

**your app
python binary
all required modules**

**bonus: installer to put it
all in the right place**

py2exe
py2app
pyinstaller

...

Video: How I distribute Python apps on Windows

<http://blip.tv/pycon-us-videos-2009-2010-2011/how-i-distribute-python-applications-on-windows-py2exe-innosetup-2090661>

Lab C
(20)

git pull

svn update

Lab C

- discussion / question time

wrapup

Assignment

- Look at `bookdb.py`.
- Make a new class which connects to a database which actually has this data in it.
- Two database drivers are in the stdlib: `sqlite3`, `anydbm`. Use them, or another database you prefer.
- Send me your implementation, and the database file if you used `sqlite3` or `anydbm`.