Amazing things in open source
3rd-party Python package ecosystems

Audrey Roy
@audreyr
About me

- Python volunteer work: Django Packages/OpenComparison co-BDFL, PyLadies co-organizer
- Entrepreneur: co-founder of Whitespace Jobs
- Django consultant: Cartwheel Web & Revsys
- Fiancée of Daniel Greenfeld (@pydanny)
Overview

- Python community is a meritocracy
- Python ecosystem patterns
- 3rd-party packages: Quantity & quality
- Community-building
Meritocracy

If your work has merit, people use it
We are a meritocracy

- Python open-source community is a meritocracy
  - Anyone can build anything
  - Anyone can start a user group
  - Anyone can be a leader
    - ...as long as they put in the work
No permission needed

• Just implement or experiment with what you want
• The more useful or helpful your work is, the more you succeed
• (A few exceptions)
It's fun

• Rewarding to see others using your work
Meritocracies are about what you actually do

- Call to action: Build whatever you desire/need
  - Don’t worry about politics. Just be nice
  - If you don’t build X, no one will
  - You can solve your own problems
Who’s in charge?

If you’re doing the coding, you’re in charge*. So make the decisions that you want.

*when it’s your own project or 3rd-party Python package
Early decisions for djangopackages.com

• Forced to make quick decisions during Django Dash
• All packages are added manually, using:
  • package name
  • PyPI URL
  • repo URL
Decisions, decisions

- No automatic package-finding spiders, by design.
- Was it a good decision?
  - Who knows, doesn’t matter
- Currently 900+ packages
Decisions, decisions

- Sometimes you just have to pick something
- As opposed to asking a mailing list and having X get debated for 2 weeks
  - Mailing lists can be useful, but be careful
The Future is in Your Hands
(or is it?)

- Future of your Python project: In your hands/in the hands of your core dev team
- or...
- Future of your Python project: Decided by other people on a mailing list
Your Gut Instinct Is Often Right

• Fun fact: many of the grid items.descriptions on Django Packages come from “throwaway” fixtures that I created in a rush

• Don’t like them? You have edit permissions ;)}
Ecosystem Patterns

Add-ons, plugins, contrib, core size, etc.
What makes a Python project or framework grow?
Why I Use Mostly Django These Days

- Many 3rd-party packages
  - Candy store
  - Bucket of Legos
- Not having to write everything myself is fun
Why I Use Mostly Django These Days

• (also, I’m just used to it)
Django Core vs. Apps

- Many, many batteries included
  - Gives you one obvious way to do things
- Third-party apps: “Django packages”
- Good and bad
Django’s Reusable App Structure

- What an app looks like

```
houses/
  __init__.py
  models.py
  views.py
  urls.py
  tests.py

templates/houses/
  house_list.html
  house_detail.html
```
Installing Django Apps

• How to install an app

# settings.py
INSTALLED_APPS += ('houses')
YOUR_CUSTOM_SETTING = 'your value'

# urls.py
url(r'^houses/', include('houses.urls')), 
Django’s Ecosystem Over Time

- More and more new innovations implemented as 3rd-party packages.
- The best innovations get added to core
Observation

- Ecosystems grow fastest when end-users can write packages
  - Look at jQuery plugins
  - Look at PyPI
  - Look at CPAN
Pyramid Core vs. Add-Ons

- Smaller core, more add-ons
- Anyone can write add-ons
- Some add-ons are “officially endorsed”
- Still young, but potential for rapid growth
Pyramid’s Ecosystem Over Time

- Past: Pylons, Repoze.BFG, Turbogears
- Present: Core-like functionality implemented as add-ons.
  - Few but growing # of add-ons
Checklist: What 3rd-Party Package Devs Need

- “Best practices” doc on how to write 3rd party packages
- Well-defined, easy-to-understand spec
- Sample code (as much as possible)
- Active community
Checklist: What Every Micro-Community Needs

- Mailing list for announcements/questions
- IRC channel
- Docs, tutorials, sample projects
- Spec or “Best practices” doc for package authors.
- Add-on, plugin, or other 3rd-party package system
- Actively-updated 3rd-party package catalog
Too many options?
Dealing with a (not always) happy problem
The One True Way?

- The Zen of Python says: “There should be one—and preferably only one--obvious way to do it.”
- This is about Python language constructs

- Not applicable to 3rd-party packages
  - e.g. okay to have multiple web frameworks
Too Much Clutter
Can’t find the right package?

• What if 2 packages have the same purpose?
• Differences should be documented
  • Within the package docs
  • In a catalog
• Deprecate bad packages
Too Much Fragmentation?

- So many Python micro-communities
  - Web frameworks with their own user bases
  - SciPy, PyGame, wxPython, etc.
Too Much Fragmentation?

- You can never have too many Python interest groups
- Diversity of ideas
- More communities == more Python developers
- Share as much code as possible
3rd Party Package Quality
Patterns and Anti-Patterns
What Makes a Python Package Useful?

- Solving problems elegantly, clearly, efficiently
- Does one thing well
- Usability: thorough docs, pip install from PyPI
- Reliability: tests, support
Package Anti-Patterns Are Viral

- Don’t underestimate the influence of your patterns
- When there is no best practice, people find the anti-pattern code samples
Package Anti-Pattern: Not Packaging

- Releasing snippets instead of packages
- Heavy reliance on copying and pasting snippets
  - project-to-project
  - from old blog posts
  - snippet directories
Package Anti-Pattern: Not Packaging

- But there are exceptions to this!
- Don’t over-engineer to be pluggable
Package Anti-Pattern: Not Packaging

- Tricks that you have to be “in the know” to understand
Package Anti-Pattern: Too Much Functionality

- Kitchen-sink base platforms
- Utility, do-everything packages
- Duct-tape packages to correct a large variety of unrelated problems
Package Pattern: Be Pythonic

• Why do we love Python?
  • Elegance
  • Ease of use
  • Explicitness, clarity
  • Simplicity
Community-building

Groups, leaders, and diversity of ideas
Mentorship

- Today’s new users are tomorrow’s contributors & leaders
- Mentorship groups: Positive encouragement
  - PyLadies
  - Python Core Mentorship
  - (need more like this)
Diversity of Ideas

- Look at schedules & slides from PyCons & PUGs around the world
- Ideas differ country-to-country
- Same goes for other types of diversity besides geographic
  - LA PyLadies is very different from SoCal Python Interest Group & LA Django
Summary
Summary

• Build what you want
• Encourage a community of 3rd-party package developers
• Be helpful
Thank you

- Twitter: @audreyr
- Find me in IRC - #pyladies, others

- By the way: