The Packaging Gradient
A Holistic Guide to Software Distribution

Mahmoud Hashemi
August, 2017
Lessons from 350+ FOSS applications

PyGothenburg
Mahmoud Hashemi
2019
LET’S TALK GOALS

Why do we code?
Why we code

Common inspirations:

1. Sales said your product does something it doesn’t
2. Google just deprecated another API
3. GitHub called, it needs one more unit testing util
4. Last quarter your manager said you needed to work on setting more aggressive OKRs

Right?
THERE ARE TWO REASONS TO START WANTING TO CODE

I want to make a video game.

I want to be free of Excel.
Recognize any of these names?

- Anki
- BleachBit
- Deluge
- FreeCAD
- Home Assistant
- Odoo
- Reddit
- Unknown Horizons
- MusicBrainz Picard
- youtube-dl
They are in fact...

1. **Popular** software
2. Targeted at a **non-programmer** audience
3. Written in **open-source Python**

Pretty inspiring, right?
We want to build!

Wait. How do we do that again?
How do I (X) ?

In the Python community we get a lot of questions about:

1. Testing
2. Packaging
3. Architecture
4. Performance
5. Documentation

etc.
How you should learn (X)

So the community responds with:
1. Stack Overflow answers
2. Blog posts
3. Video tutorials
4. Tweet rants
5. Conference talks

detc.
But what about...

All the other stuff?
An Alternative

1. Figure out **what kind** of application you’re building.
2. Find **other** applications like that.
3. Explore and **reuse**!
AWESOME PYTHON APPLICATIONS

Case studies to complement your building (and learning)
Awesome Python Applications

As developers, we spend our days with code. The site you’re reading this on is mostly modules, packages, libraries, frameworks, and the like. But users see applications.

When building our own applications, open-source Python applications are a gold mine of practical patterns that we know work together. A production application is worth a thousand blog posts and Stack Overflow answers.

This document is an always-growing list of 360 open-source Python applications arranged by topic, with links to repositories, docs, and more, generated from structured data using aptaine. If you have one to add or find some information missing, please let us know!

Read the announcement post to learn more about this list.
Subscribe to the RSS/Atom feed to see new applications added.

Contents
1. Internet (30)
2. Audio (16)
3. Video (7)
4. Graphics (18)
5. Games (10)
6. Productivity (20)
7. Organization (35)
8. Communication (32)
9. Education (7)
10. Science (19)
11. CMS (11)
12. ERP/5 (5)
13. Static Site (6)
14. Dev (150)
   i. SCM (17)
   ii. Code Review (4)
   iii. Storage (13)
   iv. CI/CD (9)
   v. Make/Build (7)
   vi. Ticketing (10)
   vii. Monitoring (10)
1. AWESOME

How does one list awesome?
Awesome™ Lists

- GitHub’s greatest meme!
- moinmoin but with Pull Requests
- Resource hubs
- Chock full of mostly-working links
2. PYTHON

The obvious choice.
Are you making the most of it?

You’re part of the biggest, most-successful software platform ever.
3. APPLICATIONS

As opposed to?
**Libraries vs Applications**

**Libraries**
- Developer-facing
- Compiled (imported)
- `pip install`

**Applications**
- User-facing
- Configured
- Just... install.
APA: More than just a README

○ Descriptions
○ Links
○ Tags
○ Structured YAML
LET’S LOOK AT THE DATA

Data last pulled: 2019-10-01T04:46:13Z
Even across 360 repos, that’s a lot of code!

(It takes git, hg, and bzr about 6 hours to clone in parallel.)
69,202,309
Lines of code

2,519,532
Commits

50,760
Committers
95%

Of application repositories have commits in 2019.
1. Architecture

What hath FOSS wrought?
Application Architecture

- 43% Server
- 57% Desktop
Application Architecture Over Time

![Chart showing application architecture over time with categories: desktop and server.]
SPOTLIGHT:

ganeti

- Cluster management tool focused on long-lived VMs used for workloads without built-in redundancy
- 15,964 commits since 2007-07-16
- Widely deployed, including at Wikimedia
- Developed at Google
- 60% Python
- 20% Haskell
2. Dependencies

Which shoulders/turtles are we standing on?
Desktop GUI Frameworks

- Qt: 52%
- WX: 17%
- GTK: 27%
- Pygame: 2%
- Kivy: 2%
GUI Frameworks Over Time

- gtk
- qt
- wx
- pygame
- kivy

Start Date:
- 2000
- 2004
- 2008
- 2012
- 2016

Ask the Ecosystem - October 2019 - bit.ly/AskTheEco
Server Frameworks

- django: 54%
- flask: 26%
- pyramid: 4%
- pylons: 3%
- zope: 10%
- bottle: 3%
Server Frameworks Over Time

The diagram shows the popularity of various server frameworks over time, with a focus on the years 2000 to 2020. Notable frameworks include Django, Zope, Flask, Pylons, Pyramid, and Bottle. The data points indicate the start dates for each framework's usage, with a trend towards increased adoption over time.
Concurrency

- tornado (18%)
- twisted (19%)
- gevent (19%)
- concurrent\futures (17%)
- asyncio (27%)
Concurrency Over Time

- twisted
- asyncio
- gevent
- concurrent\futures
- tornado

Start Date

SPOTLIGHT: GNU Mailman

- The original listserv, a web application and email server for managing subscriptions and discussion archives.
- 9,403 commits since 1998-01-06
- [https://www.list.org/](https://www.list.org/)
- Oldest user of asyncio
- Oldest APA project (by 3 months)
- One of five to do Python 1 → 2
- One of two to do Python 1 → 2 → 3
- Python 3.5
Python 3 Compatibility

- <3.5+: 48%
- 3.5+: 25%
- 3.6+: 21%
- 3.7+: 6%
Python Compatibility

- py3: 60%
- py2: 29%
- py23: 11%
Python Compatibility Over Time

### Python 3 (py3)
- Data points spread from 2000 to 2020, with a concentration around 2008-2012.

### Python 2 (py2)
- Data points spread from 2000 to 2020, with a concentration around 2004-2012.

### Python 2.3 (py23)
- Data points spread from 2000 to 2020, with a concentration around 2000-2007.
Python 2 vs 3 Committers
3. Maintainability

Coping with the commitment.
55
Median number of committers

51%
Of applications are mostly written by one committer

16,000
Median lines of Python written by the primary maintainer
SPOTLIGHT:

edx-platform

- **51,750** commits since 2011-12-07
- Platform for massively open online courses, powering edx.org
- **Third-largest** Django project on the APA
- 300 committers
- One of only **2 projects** where no one developer has >10% of the commit history
4.

Licensing

Spoiler alert: No one wrote their own.
Licenses Over Time
Hereditary Licenses

- True: 68%
- False: 32%
Hereditary Licenses Over Time

The diagram illustrates the distribution of hereditary licenses over time. The x-axis represents the start date, ranging from 2000 to 2020, with specific markers at every four years. The y-axis indicates whether the license is hereditary (True) or not (False). The data points are color-coded, with True licenses in blue and False licenses in yellow. The distribution shows a trend of increasing hereditary licenses from 2008 onwards.
**SPOTLIGHT:**

sentry

- Web service and frontend for cross-platform application monitoring, with a focus on **error reporting**.
- 26,801 commits since 2008-05-12
- The **largest** FOSS Django project
  - 1 million lines of Python
  - (including 120,000 vendored)
  - Largest flask app is **Pagure** (110k)
- **BSD-3** Licensed
opbeat is copyright (c) Opbeat, David Cramer and individual contributors.

opbeat is forked from Raven by David Cramer:
Copyright (c) 2009 David Cramer and individual contributors.
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted

Redistributions of source code must retain the above copyright notice, this list of conditions
5. Packaging

Two years in the making.
62%

90 out of 146 GUI applications use freezers.
Freezers

- pyinstaller: 42%
- py2exe: 35%
- cx_Freeze: 9%
- py2app: 13%
- pynsist: 2%
Freezers Over Time

The diagram illustrates the usage of different freezing technologies over time. The x-axis represents the start date, ranging from 2002 to 2018, and the y-axis indicates the freezing technologies, including `py2app`, `py2exe`, `pyinstaller`, `cx_Freeze`, and `pynsist`. The data points show the prevalence of these technologies over the years.
**SPOTLIGHT:**

*OnionShare*

- Secure and anonymous file sharing over Tor services.
- **2,694** commits since 2014-05-20
- Linux, Windows, and Mac
- Built on qt5
- Ported from py2exe/py2app to PyInstaller
Containerization Over Time
SPOTLIGHT:
The Median Python App

- Something related to communication, collaboration, or development
- 3,000 commits since 2011-12-16
  - 8 years old
- 27 drive-by committers with 1 commit
- Mostly written by one person
- Python 3.4+
- 65% Python
- Hereditary License (GPL, MPL, etc.)
Methods

I didn’t make this all up, I swear.
METHODS:

apatite

- 89 commits since 2019-08-05 (2 months ago)
- CLI for managing and analyzing Awesome™ lists
- Plugin support
  - `tokei` for SLOC count
  - `go-license-detector` for licenses
  - `vermin` for minimum Python detection
- Dozens of heuristics and lots of manual tagging
- Jupyter + pandas for graphs (thanks Maya)
- [https://github.com/mahmoud/apatite](https://github.com/mahmoud/apatite)
What next?

We’re only getting started with our ecosystem.
USING THE APA

Build
Reference the list for similar applications when building your application.

2700+ years of maintenance.

Cite
Research your talk, blog post, or tweet for examples of patterns you’re trying to highlight.

Recruit
Not all developers have an idea for an original application or library, especially when just starting out.
BUILDING THE APA

Fix Bugs
Apatite is far from complete. The potential for more metrics is limitless, but also:
- CI / Auto-link checking (~980 APA links atm)
- Project archiving
- Static site generation

Find Applications
We’ve got a big list of sources for popular applications that needs review.

Share
Together, we can close the loop on FOSS development.
Questions?

THANKS!

github.com/mahmoud/awesome-python-applications
github.com/mahmoud/apatite
twitter.com/mhashemi
sedimental.org
yak.party

Ask the Ecosystem - October 2019 - bit.ly/AskTheEco