josbohde-event-notes Documentation

Release 0.1

Josh Bohde

November 12, 2016

Contents

Contents:

CHAPTER 1

PyCodeConf

1.1 The Future Is Bright

1.1.1 Author

Talk by Jesse Noller

- PSF Member
- Pycon Chair

1.1.2 Intro

- Aimed at all communities.
- “Kids are okay”.

1.1.3 What is Python?

- Python = python language + community.
- Includes IronPython, Jython, Pypy.
- Heroku blog post says it really well http://blog.heroku.com/archives/2011/9/28/python_and_django/

1.1.4 Who uses Python

- Everyone.
- Scales up to 100k users, to a script to calculate budget.
- Easy to teach. Fits in head. Scales up well, and scales down.

1.1.5 Where is the language

- ~123 accepted PEPs.
- ~80 builtin functions
- ~285 documented modules.

1.1.6 What's the Future?

- 2.7 is the last release of 2.x
- Py3k
- Proposals for Coroutines, async IO, cofunctions, daemon, asyncIO for subprocess, OS and Exception heirarchy.

1.1.7 What should Python be?

- Should be the Borg, by borrowing the good ideas from other languages.
- Ease of use, simplicity.
- Adopt, but make it Pythonic.
- Must continue to look outside the language

1.1.8 Things Jesse Wants

- Communication (messaging)
- Lightweight processes
- Actors.
- Gevent, libevent,

1.1.9 What we need

- Cleaner, more Pythonic APIs.
- Don't leak.
- Balance advanced users versus keeping it simple.
- Modernize standard library.
- Python is a conservative language. Doesn't blaze the path.
- Core language must be able to fit in one's head.

1.1.10 Interpreters

- **Pypy**
  - It's fast. blazingly fast.
  - Magic in the RPython.
- **CPython**
  - It's the cockroach.
  - Simple, uncomplicated C code.
  - Battle tested.

1.1.11 Predictions

- Pypy will become the dominant interpreter. CPython won't die, and it will be reliable.
- PyPy, CPython, Jython, and ironPython are BFFs.

1.1.12 Python3

- Keep Calm, and Carry On.
- Python is over 21 years old.
- 5 years is nothing for migrating a community so big.

1.1.13 Community

- Cannot afford to be idle.
- Cannot be hostile, but welcoming.
- Get involved locally.
- Be open to criticism, especially constructive.
- Look through the pile of vitrol, to find the core of truth.

1.2 Embracing The GIL

1.2.1 Author

- David Beazley (@dabeaz)

1.2.2 Intro

- Railed on GIL at PyCon, thought it deserved some Love
- Godwin's Law of Python

1.2.3 Interest

- Fun hard systems problem
- Likes to break GILs as a hobby

1.2.4 Threads are Useful

- People love to hate on threads...
- Because they are being used.
- They solve tricky problems.

1.2.5 In A Nutshell

- Python code -> VM instructions
- Can't execute VM instructions concurrently, therefore locking
- Keep things safe * Ref counts * Mutable types * Internal bookkepping * Thread safety
- All low level.

1.2.6 An Experiment in Messaging

- Comes up in a lot of contexts
- Involves IO
- Foundation for working around the GIL
- **Shows an experiment in messaging using 5 implementations**
  - C + ZeroMQ
  - Python + ZeroMQ (C extensions)
  - Python + multiprocessing
  - Python + blocking sockets
  - Python + nonblocking sockets
- Tested on xlarge EC2 instance.

1.2.7 Scenario 1

- Unloaded server
- Expect ~10 seconds (10 seconds of sleep in there)
- All roughly the same (~13 seconds for each)
- Shows a real time example

1.2.8 Scenario 2

- Implement a thread to calculate Fib(200) (Referencing Node.js is a cancer)
- C version is barely affected.
- Python blocking goes to 142 seconds.
- Real time example takes a long time.

1.2.9 Commentary

- This aggression will not stand.

1.2.10 Thoughts

- **Try Pypy**
  - Test on Pypy (6k seconds)
  - fixed in trunk
- **Try 2.7**
  - Within 2x

1.2.11 GUI

- Uses Idle with threads (esp CPU bound)
- Kills performance to the point of completely unusable
- Can barely type into Idle

1.2.12 Thread Switching

- GIL aquisition based on timeout
- Thread that want the GIL must wait 5ms
- Causes a problem on release
- 5ms delays build up

1.2.13 What's Really Happening

- Before send and recv, acquire GIL
- After release

1.2.14 How to Fix

- Thread priorities
- Was in the original "New GIL" patch
- Should be revisited

1.2.15 Experiment

- Has an experimental python 3.2 with priorities
- Really minimal
- Threads can set their priority
- Performance that is comparable to version without threads.
- Makes GUI completely usable.
- Tried with 1.4k threads.

1.2.16 More Thoughts

- Huge boost in performance with few modifications
- Not the only way to improve the GIL
- Example: Should the GIL release on nonblocking IO?
- Currently releases on every IO
- If you are doing nonblocking IO, you aren't blocking.

1.2.17 Wrapping Up

- Python programmers should be interested in improving the GIL
- Doesn't have to be huge, incremental.
- http://www.dabeaz.com/talks/EmbraceGIL
- Code available via version control

1.2.18 Questions

- **Did you do an academic paper on this?**
  - No, but I think there is room for it.
  - The interesting question is if the OS thread library gives enough help to languages with a GIL.
  - Could it cooperate to tell the thread that it will be context switched.
  - At Pycon, OS kernel hackers came to talk about this.
  - Should say "Fixing the GIL is impossible".
- **Is priority code production runnable?**
  - No
  - Threads cannot quit.

1.3 What Makes Python AWESOME?

1.3.1 Author

- Raymond Hettinger (@raymondh)

1.3.2 Context for Success

- OS License
- **Commercial distributions**
  - Sponsor advancements
- **Zen**
  - Guides the language and community

- **Community**
  - Killer feature
- **Repositories (Pypi)**
  - Solved problems are a *pip install* away.

1.3.3 High level qualities

- **Ease of learning**
  - Can build a Python programmer in a week and a half
- **Rapid Dev Cycle**
  - Used in a high frequency trading company
  - More important to react to market
- Economy of Expression
- **Readability and Beauty**
  - Makes it easy to work in, and less tiring
- **One way to do it**
  - Once you learn an aspect, you can apply it somewhere else

1.3.4 A bit of Awesomeness

- Five minutes to write code to find duplicate files.
- Can throw away.
- **How long to write in C?**
  - Infinite
  - You won't write it
  - Python programmers write things C programmers won't.
- Just the same as any other scripting language?

Why is Python Awesome?

1.3.5 Indentation

- How we write psuedocode
- Contributes to readability
- Shows an example of indentation in C lying

1.3.6 Iterator protocol

- Lots of stuf is iterable
- Hold the language together
- sets, lists, dicts, files
- shows *sorted(set('abracadabra'))*
- *sorted(set(open(filename)))*
- Like legos: fit together perfectly
- Shows an analogy between that and Unix pipes.
- Not enough, GOF pattern

1.3.7 List Comprehension

- More flexibile than functional style

1.3.8 Generators

- Easiest way to write an iterator
- Adds one keyword (yield)
- Makes tricky iterators easy

1.3.9 Generator Expressions

- Produce values just-in-time
- *sum(x**3 for x in xrange(1000000))*
- In Pypy, roughly C speed
- setcomps and dictcomps

1.3.10 Generators that accept Input

- generators support send(), throw(), and close()
- Unique to Python
- Can make Twisted's inline deferreds using this
- A state machine with callbacks.
- Write code that looks procedural, but uses callbacks
- Monocle (https://github.com/saucelabs/monocle), Twisted inline deferred
- Fantastic improvement of callback code.

1.3.11 Decorators

- Expressive
- Always worked for function
- Initial response: Syntactic sugar
- Community rose up and demanded it from Guido.
- Easy on the eyes
- Shows example using itty (https://github.com/toastdriven/itty) using decorators for routing.
- Ping into another machine using curl to lookup environment variables in 3 lines.
- Web service in 20 lines, made possible by decorators
- Thanks Django!

1.3.12 With Statement

- Clean, elegant
- Profoundly important
- Sandwich analogy
- Subroutines factor out the 'meat' of the code
- With statments factor out the 'bread' of the code
- Factors out common setup and teardown methods.

1.3.13 Abstract Base Classes

- Uniform definition of what it means to be a sequence, mapping, etc.
- Ability to override *isinstance()* and *issubclass()*
- Duck-typing says: "If it says it's a duck..."
- Mixin capability (DictMixin)
- **Can provide the base of a class**
  - shows using a list-based set with *__iter__*, *__contains__*, and something else
  - Mixin provides the rest

1.4 Backbone.js and Django for a Faster WebUI

1.4.1 Author

- Leah Culver (http://twitter.com/leahculver)
- Cofounder of Pownce, one of the first big Django applications.
- Works at Convore (https://convore.com/)

1.4.2 Leafy Chat

- Web frontend for IRC
- Done in Django Dash (2008?)

1.4.3 Grove

- New project
- Internal IRC for your company
- https://grove.io

1.4.4 Chat Systems

- Built a lot of them
- Leafy chat - only used jQuery, lots of javascript
- Using Backbone in Grove

1.4.5 Examples

- Show an example of using jQuery to build UI.
- Embedded HTML in javascript.

1.4.6 Backbone and Grove

- The UI looks the same
- Backbone gives MVC style, in a single file.
- You can roll it yourself, making it easy to get started.
- Not actually MVC, actually Models, Templates, and Views

1.4.7 Models

- Shows *Backbone.Model.extend({})*
- http://documentcloud.github.com/backbone/#Model

1.4.8 Collections

- Shows *Backbone.Collection.extend({})*
- http://documentcloud.github.com/backbone/#Collection

1.4.9 Views

- Highlight the Backbone Views on the Grove app page.
- **Demonstrates Backbone Event binding**
  - Creates the view from the model data
  - Bind updating view when the model changes

1.4.10 Templates

- Uses handlebars.js (http://www.handlebarsjs.com/)
- Looks like Django
- Specify templates to a view.
- use include_raw templatetag http://djangosnippets.org/snippets/1684/

Additional Goodies

1.4.11 Sync

- Used to synchronize data on Django server
- Shows *request.raw_post_data* to get JSON objects.
- https://gist.github.com/1265346

1.4.12 Events

- Can update multiple views for a single model.
- *App.trigger('messageAdded', ...)*

1.4.13 Router

- Will trigger Events based upon the hash

1.4.14 Questions

- **Do you feel bad that your Django app is now Javascript?**
  - No, this is how apps are going.

1.5 PyPy is your Past, Present, and Future

1.5.1 Author

- Alex Gaynor (http://twitter.com/alex_gaynor)
- Still in school, core Django, CPython, and Pypy committer

1.5.2 Intro

- **There are 2 things faster than C**
  - Neutrinos
  - Pypy

1.5.3 Story

- Armin wanted to write a JIT for Python (Psyco)
- Psyco was the written by Armin.
- Kind of messing.
- Generators came along, and not supported
- 64-bit computers weren't supported either
- Started writing Python in Python
- About 2000x slower than CPython
- Somethings in the standard library were in python
- Copied some optimizations over (TimSort)
- Writing JITs sucked.
- Writing a JIT generator for arbitrary languages is much simpler than writing a JIT for Python
- ~2-3 years ago Alex got into Pypy
- Beat C in str_cmp ~1 month ago
- http://speed.pypy.org
- Tries to show example of real time video analysis, mplayer broke.

1.5.4 Numpy

- Science likes big datasets, use Numpy
- Numpy is in C
- Numpy likes speed, so does pypy
- Started reimplementing Numpy in Pypy

1.5.5 Hotspot Detection

- Humans are bad at detecting slow downs
- **Pypy has a JITViewer**
  - http://morepypy.blogspot.com/2011/08/visualization-of-jitted-code.html
  - Allows you to view code in levels
  - Python, Assembler, etc.
- Shows demo fo JITViewer

- **Look into code**
  - "I think that's too many instructions"
  - Optimize code!
- Shows example of *sum(x**3 for x in xrange(10000))*
- JVM Community has good tooling
- Python could use that too.

1.5.6 Current

- Usually benchmark against C
- Experimenting with using C extensions.

1.5.7 Where we're going

- **Many projects are being migrated**
  - Django
- Porting to Python3

1.5.8 Architecture

- Because they use a JIT Generator, can improve constantly
- Speedups in Python3 will improve Python2

1.5.9 GIL

- **Wrote a blogpost on STM (Software Transactional Memory).**
  - http://morepypy.blogspot.com/2011/08/we-need-software-transactional-memory.html
- Think STM would be slower for single core
- STM for multicore workloads
- GIL for single core.

1.5.10 What People Are Doing with Pypy

- Researchers getting results over lunch, instead of over night.
- Financial company for market analysis
- Engineers at CERN

1.5.11 What Pypy Needs from the Community

- Encourages use of pypy if you are CPU bound
- Requests for slow code, and they'll use it in benchmarks
- Want to make Python the right tool for the job in more places
- Work on the ecosystem and tools

1.6 Processing Firefox Crash Reports With Python

1.6.1 Author

- Laura Thompson (http://twitter.com/lxt)
- Works on lots of internal tools at Mozilla

1.6.2 Socorro

- Named after an array in New Mexico
- Is today's browser more or less crashy than yesterday's?
- If you see the Firefo crash popup, please submit it.
- Shows Mozilla Crash Stats https://crash-stats.mozilla.com/products/Firefox

1.6.3 Architecture

- **Collector**
  - web.py app behind apache
  - Puts on disk
- Store in HBase (crashmover)
- Write to Postgres by monitor
- Webapp and API
- All Python

1.6.4 Lifetime of a Crash

- Raw dump submitted by POST, JSON + minidump
- Stored
- Processed

1.6.5 Processing

- Processing spins off minidumpstackwalk (msdws)
- Tries to regenerate stack
- Processor generates a signature
- Tries avoid things like malloc
- Writes to Postgres, which acts like a large, relational cache.

1.6.6 Backend Processing

- Cron
- **Calculate aggregates**
  - Top crashers by signature
  - URL
  - Domain (hates Farmville)
- Process incoming builds
- Match known crashes to mozilla bugs
- Dupe detection
- Match up crash pairs, e.g. plugin containers and browsers
- Generate CSV extracts for engineers for analysis

1.6.7 Middleware

- **Move data access to REST API**
  - Allow engineers to build apps against the data
- Enable to rewrite app in Django in 2012

1.6.8 Webapp

- **How to visualize?**
  - Many builds: release channel, nightly, hourly
- **Reporting in build time**
  - Rebuilding in Django in 2012, because it's Crufty
  - Maybe Flast
- Almost all new Mozilla apps are Django
- Don't need models, though

1.6.9 Implementation

- Use Python2.6
- Postgres 9.1, some stored procedures
- memcached
- **Thrift for HBase access**
  - HBase written in Java
  - Thought about rewriting Hbase parts on JVM
  - Decided not to, Clojure not common, Jython for various reasons

1.6.10 Scaling

- **Different**
  - Usually scale to millions of users.
  - Crash Center has terrabytes of data, ~100 users.
- **2300 crashes per minute**
  - Going down
- 2.5 million per day
- Median size 150k
- Max 20MB
- Reject bigger, since probably not useful since mem dump
- ~110TB in HDFS (3x replicatoin)

1.6.11 What Can We Do?

- Compare beta null signature crashes.
- Analyze Flash versions crashes
- Detect duplicate crashes
- Detect explosive crashes
- **Find "frankeninstalls"**
  - Some Windows updaters don't work properly
  - Keep duplicate but out of version dlls

1.6.12 Implementation Scale

- **>115 Physical Boxes**
  - About to rollout Elastic Search
- **8 Devs, sysadmins, qa, hadoop ops, analysts**
  - Hiring

1.6.13 Managing Complexity

- **Fork**
  - Hard to install
  - Use version control VMs
  - Found to help with complex dev environments
- Pull requests with bugfix features
- **Jenkins polls master on github**
  - Runs tests
  - Build package
  - Push out to dev environment
  - builds release branch
  - manual push staging
  - *missed rest of this*

1.6.14 Continous Deployment

- Critical
- Build machinery for Continuous Deploy, even if you don't
- Can deploy at 10 a.m.
- Everyone relaxed
- Deployment is not a big deal

1.6.15 Config Management

- **Automate configs**
  - Managed through Puppet

1.6.16 Virtualization

- Don't want to bulid HBase
- Use Vagrant (http://vagrantup.com/)
- Jenkins builds Vagrant VMs
- Puppet configures VMs.
- Tricky to get data
- This + Github increased community activity

1.6.17 Upcoming

- **ElasticSearch**
  - Lucene, distributed flexible search engine
  - Don't know how to tune
- **Analytics**
  - Detect explosive crashes
  - Detect malware
- **Better queueing**
  - Sagrada queue
  - Mozilla Services - Ben Bangert (https://github.com/bbangert/moz_mq ?)

1.6.18 Open Source

- Almost everything is open

1.7 The Future of Collaboration in the Python community and beyond

1.7.1 Author

- Daniel Greenfield (http://twitter.com/pydanny)
- Cartwheel Web/ Revsys
- Django Packages (http://djangopackages.com)
- Whitespace Jobs (https://whitespacejobs.org)
- Fiance of Audrey Row

1.7.2 Mark Pilgrim is Gone

- **He did feedparser, httplib2**
  - aside: httplib2 was actually by Joe Gregorio
- Dive into Python
- Dive into HTML5
- Other things
- We lost a lot with him leaving, sad to see him go.

1.7.3 What Happened to his Projects?

- **What is the copyright?**
  - A: CC-SA
- **What about his code?**

  - httplib2 is a big dependency of many projects
  - That's how found out he was gone
  - Pypi didn't host it
  - Google Code didn't host it anymore.

1.7.4 PyPI issues

- Too easy to delete a package * Dependency checks for that package * Request a project handoff * Other projects need to be notified * RSS feeds
- Human moderation * Some can be automate * Burdens PyPI team

1.7.5 Repeating History

- Django-lint
- **Django-Piston**
  - social factors caused no release in years
- python.org
- **opencomparison.org**
  - Host djangopackages.com
  - How does this get maintained?

1.7.6 Dark Future

- Critical Packages Breakdown
- Python packages vanish
- Build scripts fail
- Replace from caches/backups

1.7.7 Repercussions

- Lose domain knowledge
- Python can't move forward.
- Social Issues
- 3rd Party Community is just as critical as Python core

1.7.8 Not the Future

- It's today
- Legacy code with legacy packages
- Build scripts fail

- **Example of NASA issue**
  - caused project to go to ColdFusion
- **We have lost works of antiquity**
  - Blame is moot
- Stuff we make today is legacy in 5 years

1.7.9 Trust Issues

- This causes a lack of trust in Python
- **Without trust, we can't collaborate as well**
  - The disease that will trigger zombie apocolypse

1.7.10 Solutions

- **Money!**
  - Sponsorships
  - Problems getting money
  - Applications
  - Focus on sprints
  - Code quality issue from sprints
  - Ongoing maintenance

1.7.11 Future is still dark

- Community Managers
- Ticket triage, etc.
- Needs core/senior developers
- **They are already busy**
  - Examples pay people to do this
  - Volunteers may have life get in the way
- Determining authority

1.7.12 PSF Paid Commmunity Manager

- Proposed solution
- Paid via the PSF

1.7.13 Repercussions

- Fixes some problems
- Mitigate social issues
- Can still lose domain knowledge

1.7.14 Precedents

- Ubuntu
- Fedory
- Twilio
- Github

1.7.15 Wants

- More reasons to trust
- More reasons to contribute
- Keep projects operating

1.7.16 Call to Action

- This is a proposal
- Wants to see PSF project incubation
- **PSF provides seed funding for OS projects**
  - Should return on investment
  - Preferably to Python community
  - Needs a viable business model
  - PSF is an investor
- Choose from particapants in Django Dash & coding contests

1.7.17 Return

- Gives OS code
- Gives money back to the PSF

1.7.18 What this isn't

- Covering < $100 for hosting
- Things without a self-supporting business model

1.7.19 Examples Projects

- **djangolint.com**
  - Little setup requires
  - Uses github
  - Wants for all Python
  - Wants syndication
  - **How does it make money?**
    - Pay to analyze privately?
  - Easy linting increases trust
- **readthedocs.org**
  - Places in the 2010 Django Dash
  - Documentation increases trust
  - **Business model?**
    - Pay for private doc hosting would be good.
    - Clients don't want to host docs.
- **depot.io**
  - Freeze your python dependencies
  - Doesn't replace PyPI
  - Provides additional security
  - **Possible Advantages**
    - Archive legacy packages
    - Leave PyPI as the canonical source
    - Adds dependablility, trust
- **PyPI**
  - Pay for a PyPI Appliance?
  - Github makes "giant" profits on Enterprise Appliance
- **djangopackages.com**
  - Just launched pyramid version
  - Plone?
  - Python?
  - http://bit.ly/django-reg
  - Compare and contrast packages
  - Helped determine a package to use
  - Gives metrics
  - Metrics give trust
  - **As opencomparison, support more things**

    - Languages
    - Syndication
    - OAuth
    - What's the business model?

1.7.20 Results

- Don't have packages vanish.
- Let Python move forward
- Have new social issues.

1.7.21 Paid Community Manager

- Maybe the PSF shouldn't be involved
- Outside factors?

1.7.22 Project Incubations

- Already exists, just not with PSF
- How much code comes out of these?
- Energy of startup giving back?

1.8 The State of Packaging & Dependency Management

1.8.1 Author

Craig Kerstiens (http://twitter.com/craigkerstiens) Works at Heroku

1.8.2 Packaging

- Need to release it

Where To Release

1.8.3 Your Server

- Full flexibility
- People rely on you being up
- Breaks deploys
- Don't do this, unless you want to provide better uptime than PyPI

1.8.4 Github

- Awesome for dev
- Not for release
- Not mean to packages, but source code

1.8.5 PyPI

- Please release it here
- Complain about it being down
- 5 mirrors that are well updated

1.8.6 Managing Dependencies

- **Use pip**
  - Supports uninstalling
  - Lots of small improvements
  - Supports version control
  - Don't use this in production
- **Use virtualenv**
  - Great for sandboxing
  - Destroy and recreate it often
  - Pin your dependencies

1.8.7 Pinning

- Only deploy specific versions
- *pip freeze > requirements.txt*
- It's explicit (see the Zen)

1.8.8 Version Control

- Having a github/bitbucket source is good for dev...
- Not for prod.
- Put tarballs on internal servers.

PyPI is Down ————oG

- *pip install –use-mirrors*, problem solved

1.8.9 Whats Missing

- Not as good as Bundler from Ruby community
- Pip upgrade needs to be better

1.8.10 Recap

- Use PyPI
- Explicit versions
- Use mirrors
- Need to use the tools more effectively

1.8.11 Questions

- A frozen requirement may have unfrozen dependencies
- May need to tweak requirements.txt

1.9 Python For Humans

1.9.1 Author

- Kenneth Reitz (http://twitter.com/Kennethreitz)
- Works for Readability
- Works on the Github Reflog
- Used to be part of the Changelog
- Authored Requests, Tablib, Legit, OSX-GCC_installer, Clint, Evnoy, Httpbin
- Makes software for humans

1.9.2 Talk Slides

- https://github.com/kennethreitz/python-for-humans

1.9.3 Philosphy

- **What people like about Python**
  - Simplicity
  - Speed to develop
  - Pypy
- *import this*
- The Zen of Python, our manifesto

- **Beautiful is better than ugly**
  - Syntax
- **Explicit is better than implicit**
  - Compared to Ruby
- **If the implemenatation is hard to explain, it's a bad idea**
  - Unless you're pypy
- This talk will focus on there should only be one obvious way to do it.

1.9.4 Messing Around

- Using Github API
- Show's Ruby code, not beautiful but straightforward
- **When trying it in Python we get confused about what library to use**
  - Python 3 helps this naming issue
- **Shows code using *urllib2***
  - Too many actions to just use basic auth
  - And there's more!
  - Github API uses 404 instead of 401, need to write our own BasicAuthHandler
  - Need to force it to send basic auth, took 3 hours
- This would prevent people from using Python.

1.9.5 Problems

- Unclear on what module to use
- "HTTP should be simple as the print statement"

1.9.6 Solution

- We need pragmatic, elegant tools.

1.9.7 HTTP

- Has methods
- Very simple
- Urrllib2 is very complex, and therefore toxic

1.9.8 Requests

- For humans
- Simple solution for a simple problem

1.9.9 Litmus Test

- You should not have to refer to the docs everytime you want to do something simple
- API is the most important thing
- Handle the 95% case elegantly

1.9.10 Building

- Requests was very simple at first, but it resonated with people
- Grew to handle more stuff
- 17th most watched project on Github

1.9.11 Subprocesses

- Powerful, effective, second worst API
- Docs lacking
- Follows C API
- Mostly docs that are lacking

1.9.12 Proposed Solution

- Envoy
- Mostly the same API as Requests
- Pipe, read stdout, etc.
- Get it done quickly and effectively

1.9.13 File and System Ops

- Surveyed dev ops
- Shutil, sys, etc. are confusing
- Limits adoption by dev ops guys

1.9.14 Install Python

- Surveying room on installation methods on OSX
- Many chosen
- “What happened to one obvious way to do it?”

1.9.15 XML

- *etree* is terrible
- *lxml* is awesome
- We need to adopt a better standard

1.9.16 Packaging and Dependencies

- *pip* or *easy_install*
- *setuptools*?
- ***Distribute***
  - How is it better than setuptools?
- We need simple instructions on how to install, and release packages

1.9.17 Dates

- Some good 3rd parties
- Stdlib not good enough

1.9.18 Unicode

- **It's a simple problem**
  - Room erupts in "No it's not!"
- Should be easy

1.9.19 Testing

- Unittests
- Didn't get the downside

1.9.20 Installing Dependencies

- Asked room about difficulties
- Almost everyone had difficulties

1.9.21 Hitchiker's Guide to Python

- http://python-guide.org
- Teach the best practices
- "There should be one– and preferably only one –obvious way to do it"
- **Brief overview**

    - Idioms
    - Freezing Code
    - Installing code
- Up for debate, collaboration
- Aimed to be a reference guide, and to lower the barrier of entry

1.9.22 Manifesto

- Simplify APIs
- Document Best Practices

1.10 Python is Only Slow If You Use it Wrong

1.10.1 Author

- Avery Pennarun (http://twitter.com/apenwarr)
- Works at Google

1.10.2 Bup

- Written in Python
- Backup software
- Uses Git as a data store
- 80 megs/second

1.10.3 sshuttle

- VPN that handles wireless speeds
- Also in Python

1.10.4 How to Use Python Wrong

- Tight Inner Loops
- In compiled languages, you have these often
- Really bad in Python
- Line of code in Python is 80-100x slower than C
- Keep it in a higher level

1.10.5 Ways to Make it Fast

- **Use Regex and C modules**
  - Word based instead of char based ~5x faster
  - Will run it in C
  - Most of bup is Python, small bit in C to speed it up
- 100% Pure is not pragmatic
- **CPython has a really good C API**
  - Java doesn't, it's super painful
- **Python + C is winning so far**
  - C is for tight inner loops
  - Python for the higher level

1.10.6 Threads

- **Computation threads are useless, because of GIL**
  - Sometimes worse than single threaded
- **Okay for I/O**
  - GIL will release for I/O
- **fork() works great for both**
  - Recommend to use it all the time
  - No GIL
  - Trick is getting info from process to process
  - Bup uses this
  - No weird locking interactions
- **C modules can use threads**
  - Can release GIL when you get objects
  - Run threads
  - Get GIL when computations are done
  - Can get high performance
- CPU Bound threads in Python is doing it wrong
- Question from audience: Scipy has Weave, which will allow you to inline C code. * Dynamic compilation
- There are workarounds for the GIL

1.10.7 Garbage Collection

- Python is both refcounting and gc
- **Refcounting**

    - Whenever you use a variable, increase reference count
    - Whenever you stop, decrease the reference count
    - Terrible, terrible thing with threads
    - Need to lock on refcounts
    - GIL solves this problem
- **Shows graphs of programs memory and time**
    - Allocates 10k of space a lot
    - Refcounting sematics allow Python lower mem usage than Java
- **Testing Java**
    - 3 different tests
    - Shows one where it allocates as much memory as possible
- **Sometimes Python is Garbage Collected**
    - Mutual referencing objects that have ref count of one
    - Backup GC finds this, and collects them
    - Shows example on how to do this
    - Pretty complicated in order to get across the GC
    - Then it relies on sucking up tons of memory, and getting it later

1.10.8 Advice: Stay away from GC

- Break circular references
- **Most common, trees with reference to parents**
    - Full tree need to be GC'ed
- Better: use the *weakref* module

1.10.9 Deterministic Destructors

- Win32 example of two writers to a file
- Win32 doesn't allow two writers
- CPython allows it because it closes the writer because of refcounting
- **This causes deterministic behavior, unlike 'real' gc**
    - In Python you don't need to manage many resources
    - Files, database handles, etc.
- **Some people are trying to take this away**
    - Pypy?
    - with statement isn't a desirable alternative

1.10.10 HelloMark

- Fork and exec "Hello World" 20x
- Demonstrates startup times
- Jython takes 15 seconds, slower than C+valgrind
- Shows what you want to write command line tools in
- **pyc + CPython files are awesome for this**
  - Django and Tornado can reload really quickly
- Pypy loses in this regard

1.10.11 Summary

- Love refcounting
- Don't use tight inner loops in Python
- Don't need the JIT
- Work on startup time
- benchmarks: https://github.com/apenwarr/avebench

1.11 Amazing Things In Open Source

1.11.1 Author

- Audrey Roy (http://twitter.com/audreyr)
- Python volunteer, Django Packages

1.11.2 Overview

- Community recognizes work you do (meritocracy)

1.11.3 Meritocracy

- People will use your work if it has merit
- **Anyone can build or be a leader**
  - If they put in the work
- **Permission isn't (usually) needed**
  - We allow experiments

1.11.4 Open Comparison

- Writing Comparison Grids for sub communities
- Compare packages for Django, Pyramid, etc.

1.11.5 Call to Action

- Build it!
- Be Nice
- Others probaby won't build it, so you should

1.11.6 Early Decisions

- Django Packages
- Made during Django Dash
- Decided to only manually add packages
- **Good decision?**
  - Doesn't matter
  - 900 packages right now
- Action is better than having something get debated
- Probably better in the hands of the core devs
- **Gut instinct is often right**
  - Can always change it later

1.11.7 Ecosystem Patterns

- **Mostly from Django experience**
  - Django has many 3rd party packages
  - Compared to Legos
- **Django Core vs. Apps**
  - Many batteries included
  - This approach is good and bad
  - Can get stuck with a heavy core
  - Promotes "one obvious way"
- **Django has well defined patterns for apps**
  - App structure
  - App settings
  - Overridable templates
- Reuse encourages innovations as 3rd party packages
- Core is conservative
- Best 3rd party apps get added to core
- **Grow fastest when there is a pattern for extensions**
  - jQuery

    - CPAN
- **Pyramid**
    - Smaller core
    - Core functionality as add-ons
    - Endorsed add-ons
    - Potential for rapid growth
    - Can deprecate, and allow add-ons to evolve
    - Don't need to wait on core
- **Pyramid's Ecosystem developed over time**
    - Came from Pylons, Repoze, Turbogears

1.11.8 How to Grow an Ecosystem

- **Write "Best Practices" on how to write 3rd party packages**
    - There is a big gap in this
- **Well-defined specs**
    - Allow others to write upon a base
- Sample code
- Active community
- Mailing list/ IRC
- Docs
- 3rd-Party packages catalog

1.11.9 Too Many Options?

- "There should be one– and preferably only one –obvious way to do it."
- There can be many web frameworks
- But there is often too much clutter
- Document the differences
- **Deprecate bad packages**
    - Hard to do in some cases
    - Recommend replacements

1.11.10 Fragmentation

- Not all web
- Science, games, etc.
- **Can't have too many interest groups**

    - Diversity of ideas

1.11.11 3rd Party Packages

- Best: Do one thing well
- **Usability**
    - Good docs
    - Easy to install
- **Reliability**
    - Tests
    - Help
- Antipatterns are viral
- **Snippets is the biggest anti-pattern**
    - Copy and paste code
- Don't over-engineer though
- **Don't make the "kitchen-sink" package**
    - Utility functions
    - Unrelated problems
    - More visible in HTML/CSS world
- **Do Be Pythoic**
    - Elegance
    - Ease of use
    - Explicitness
    - Simplicity is why we use Python

1.11.12 Mentorship

- Provide positive encouragement
- Put yourself out there

1.11.13 Diversity of Ideas

- Differ from country to country
- Other types of diversity
- PyLadies vs. SoCal Python Interest Group

1.12 The Prejudgement of Programming Languages

1.12.1 Author

- Gary Bernhardt (http://twitter.com/garybernhardt)

1.12.2 Intro

- 10 Years of Failures and Bad Ideas
- Pre-2001: Ignorant of Software
- ~2001: C is the best thing, Java sucks
- ~2003: Learned Lisp
- Designed a "more modern" C
- Had curly braces, static types, but basically Python
- ~2006 Built BitBacker in ~98% Python
- Arc: C -> Lisp -> Python
- ~2009: Ruby and Python 50/50
- Tweet about frustration of integrating libraries in Ruby + Javascript
- Frustrated by Python's lack of blocks
- Shows a conversation between _why and Ryan
- "Ruby isn't serious"
- Frustrated with programming
- q2 2010: Writing Tests
- **Show TDD using Ruby**
  - Crazy Vim action

1.12.3 Testing

- Claim: RSpec is confusing
- Never had this problem
- Python based on SUnit from 1994
- Thought Django views are not as advanced as Rails
- Ruby is the serious one?
- "A Python programmer rejects a new idea without considering its value. A Ruby programmer accepts a new idea without considering its value."

1.12.4 Choose Ruby or Python

- Ruby community more willing to pay
- Move to that full time
- **Shows examples of ugliness in Ruby**
  - @foo ||= bar
  - realization, it's how you do memoization
- Maybe Ruby is well designed?
- Generators, Comprehensions, Decorators, and Context Managers are easy to implement with blocks
- Which language is complicated?

1.12.5 Emprically

- Realized back to ignorance
- Judged languages before he should
- Ruby's community is serious about testing
- Rare opportunity to work with both

1.13 Cherry-picking for Huge Success

1.13.1 Author

- Armin Ronacher (http://twitter.com/mitsuhiko)
- Part of the Pocoo Team
- Notable work: Flask, Jinja2, Werkzueg

1.13.2 Preface

- Framework/Language fights are boring. Just use the best tool for the job.

1.13.3 Twitter

- 2006: Rails, XML API
- Now: JS Frontend, Erlang/Java

1.13.4 Does Ruby Suck?

- No, and neither does Python
- Both are great for prototyping
- Application changes over time
- Will rewrite

1.13.5 Solution

- Build small applications
- Combine into a larger one
- Builds foundation to experiment * Move dbs, etc.
- **Crossing language boundaries**
  - Rewrite
  - Use a different library
  - Implement a service

1.13.6 Agnostic Code

- Example of depending on Django too much
- **Instead of importing from Django, pass it in**
  - Class instance, parameter
  - Make it specific, but not more

1.13.7 Example

- Drop down to WSGI
- Usually too specific, if you only need just the url

1.13.8 Protocol Example

- Compared to Python iterables
- Flask views return wsgi apps
- Can dispatch to a Django application, for example

1.13.9 Difflib

- Compares any iterable that is hashable and comparable
- Overly specific would be strings, though that's the main use case
- Real world use to diff HTML docs
- Plugin Genshi to difflib to accomplish this

1.13.10 Interface Examples

- **Serializers**
  - Missed examples

1.13.11 Mergepoint

- To build apps we need merge points for smaller apps

1.13.12 WSGI

- Used with most Python web frameworks
- Often not enough
- Provides a framework independent environment
- Middleware can be useful mergepoints, though overused
- Cannot consume form data in WSGI, inject uniform html, etc.
- **Libraries that help with this**
  - Werkzeug
  - WebOb
  - Paste
- Can write short helpers to dispatch from e.g. Django to WSGI

1.13.13 HTTP

- Language independent
- Cacheable
- Harder to work with, complex
- Can do proxying, nginx
- Caching layers for scalability
- Problem: Need to keep them running
- Language independent library
- cUrl

1.13.14 ZeroMQ

- More modern TCP Socket
- Language independent
- **Different topologies**
  - push/pull
  - pub/sub
- Easier than HTTP
- No caching
- Non gracefully dies
- No broker infrastructure

1.13.15 Message Queues

- Similar to ZeroMQ
- In reality, a different problem
- Can run tasks outside request/response
- Different codes, languages to run code
- Accessor Library: Celery
- Don't assume code to be nonblocking
- Greatly simplifies testing
- **Redis queues are a good start**
  - ~20 lines of code to build your own

1.13.16 Data Store

- Using the same db for different apps
- Works well as long as everyone plays nice

1.13.17 Redis

- Remote datastructures
- Shows bash example of a queue worker

1.13.18 Javascript

- It's awesome
- Geeks hate it
- ugly, can be abused
- Use Coffeescript
- Decouples frontend by using different services
- Examples: xbox.com, Battlefield 3 game lobby
- Can efficiently transform the DOM
- Backbone.js
- Testing sucks for others

1.13.19 Processes

- Daemons can be annoying to run
- Processes can have different privileges
- Tune individual processes
- Upgrade parts to python3

- ZeroMQ/HTTP to operate together

1.14 Breakdancer

1.14.1 Author

- Dustin Sallings (http://twitter.com/dlsspy)
- Memcached contributor

1.14.2 Testing

- Few constructs not mentioned in the past day
- Someone submitted a bug
- "I have tests"
- Straightforward bug that wasn't tested
- All the individual items work, but sequences can fail.
- Testing all sequences is a large number of combinations

1.14.3 Breakdancer Overview

- Conditions, Actions, Effects
- Driver to run things
- Shows how *add* command can be decomposed into conditions
- All Conditions, Actions, and Effects are composable
- Driver holds the boilerplate
- Python makes boilerplate minimal
- itertools makes combinations simple.
- Generate test case combinations automatically
- Do preconditions, postconditions.

1.15 The Many Hats of Building and Launching a Web Startup

1.15.1 Author

- Tracy Osborn (http://twitter.com/limedaring)
- Founder of WeddingLovely.com
- Considers herself a designer

1.15.2 Overview

- Quit job as designer
- Failed to found co-founder
- Learned Python

1.15.3 Start Out

- **Have good runway**
  - 1 year+
- Health and relationships
- Quit your job

1.15.4 What is Success?

- Don't want to build Google
- Just build something that makes you some money
- Take a step back
- Love your job
- Concentrate on small successes

1.15.5 Background

- Knew HTML
- Hated CS courses
- Got a job at a startup
- Got bored
- Started freelancing

1.15.6 Entreprenuer

- No cofounder is better than a bad cofounder
- Applied to YC
- Things didn't go well
- Used Learn Python the Hard Way (http://learnpythonthehardway.org/)
- Used Django
- Six weeks later, launched

1.15.7 Launch as Fast as Possible

- You need customers
- It helps morale
- Allows you to iterate
- “Good enough”
- You can add features later
- Work on the hard parts first
- For her, programming part was hard
- It was okay to launch with bad code.
- Violates DRY.
- Got picked up by Swiss Miss with MVP

1.15.8 Monetization

- Have a plan.
- Don’t think about it later or rely on funding

1.15.9 Don’t Be Alone

- Surround yourself in a community
- Find people who are smarter than you to help you out
- No NDAs
- Inhibits advice
- People stealing your ideas is a good thing
- Use Twitter/HN to talk
- Attend Hacker Events, SuperHappyDevHouse, PyLadies

1.15.10 Take Shortcuts

- Django ecosystem is awesome
- Doesn’t know databases at all, South makes it easy
- Dotcloud makes servers easy
- Themeforest for design
- Design for Non Designers
- You can always iterate later
- Launchrock.com

1.16 Future of Python and NumPy for array-oriented computing

1.16.1 Author

- Travis Oliphant (http://twitter.com/teoliphant)
- Made NumPy

1.16.2 Why Python?

- Fits your brain
- Doesn't get in your way
- Software engineering is more about neuroscience than code.
- Fibonacci is just an Unstable Infinite Impulse Response linear filter
- Shows numpy example, which is fast, but wraps hardware integer
- Wants to make Python faster than C, as in a GPU or FPGA

1.16.3 Conway's Game of Life

- Interesting excercies
- Shows an example of it
- Array oriented
- **APL**
  - Grandfather of most array oriented languages
  - J,K,Matlab are descendents
  - Numpy is a descendent
  - Unicode glyphs
- Game of Life is one line in APL
- Array-oriented programming deals with arrays as a block
- Shows numpy example

1.16.4 Numpy/Scipy History

- Numeric around ~1994
- **More features for array oriented computing**
  - a[0,1], a[::2]
  - Ellipsis object
  - Complex numbers
- Syntax matters
- Aside: We need more numpy/scipy and core collaboration

- Derivative Calculations in 1997
- Came from MATLAB, but it wasn't memory efficient enough
- Iterative update loop made Python nice
- 1999 Scipy emerges
- Python was better language than MATLAB, but lacked scientific libraries
- **Community Effort**
  - Mostly from academics
- Numpy emerged from Numeric in 2005

1.16.5 Numpy

- **Data types**
  - Collections of objects
  - Arrays
- Statistics functions
- **Arbitrary Arrays**
  - Column oriented calculations

1.16.6 Scipy

- Stats
- Data fitting
- Interpolation
- Brownian Motion

1.16.7 Zen of Numpy

- http://technicaldiscovery.blogspot.com/2010/11/zen-of-numpy.html

1.16.8 Pypy

- Let's not chase C, let's chase Fortran 90.
- Example where Fortran 90 is 7 times faster than Numpy and Pypy

1.16.9 Question

- **Coolest thing seen with NumPy?**
  - Implant surgery planning tool
  - CT Scans, 3d vis

1.17 Lightning Talks

1.17.1 Vagrant

- Vagrant loves Python
- Building and distributing VMs
- Gives isolation, repeatability, and verification
- Move dev to virtual machines
- Move production ops scripts to setup environment
- Vagrant command line, to manage life cycle
- Designers can use it too
- http://vagrantup.com

1.17.2 Testing CSS

- Needle
- Takes screen shots
- Checks them
- Looks like normal unit tests
- uses css selectors
- Extension to nose, with selenium
- https://github.com/bfirsh/needle

1.17.3 Pyparsing

- Time trial using Pypy
- Search for integers in a string of random alphas and numbers
- Pypy ~10x faster
- Verilog parser (~16k lines)
- Cpython (500 lines/sec)
- Pypy (1131 lines/sec)

1.17.4 Pandas

- @wesmckinn
- Agile Tooling for Small Data
- First need to small the small data problem before big data
- DBs, Flat files, time series, mean you may want it
- indexed data structures for relation data

- Fast manipulation tool
- Data alignment
- Join merge
- group by
- Reshaping/pivot
- In memory and fast
- Meant for quant finance application backbone
- ~26k loc
- In productions since 2008
- Data Analysis is dominated by thing like SAS
- Lots of people want to expand in these areas
- Operations to naturally select portions of data
- Can plot data
- Would love collaborators

1.17.5 DSLs

- Peter Wang (@pwang)
- Crazy crazy ideas
- Would like Python to ignore some syntax where we can do whatever the hell we want
- It might be awesome
- Calling it extern
- Just syntactic sugar
- Hacking import hooks to make it work
- .pydsl file
- uses pyparsing under the hood to transform the dsl
- Aimed at scientists
- People want it: weave, numexpr
- Everyone needs it
- Let's Python assimilate into existing systems

1.17.6 stackful

- @erikrose
- This is a hack
- Wish things weren't global
- Dynamic variables like in Perl
- Perl has *local* variables which leaks onto things it calls

- *stackful* implemented as with statement
- Thread safe
- Implementation is funny
- No hook in Python for reference
- Just override every single magic method in Python
- Should be able to be used

CHAPTER 2

PyCon 2012

2.1 Keynote with Stormy Peters

2.1.1 Author

- Stormy Peters

2.1.2 Web

- We should make people aware of how their info is being used

2.1.3 Growing a Community

- As companies get involved we wonder about the direction of the community
- Reach out to new people, because it can be intimidating
- When you meet someone, you have 3 seconds to make an impression * Based on your hair * And then shoes
- When you respond to a bug report, or mailing list post, this is their first impression * Make it a good one
- Python groups are great for this outreach
- Study says learning something new is worth a 20% raise * old job needs 20% more money vs. new job with new tech
- Some like to be famous (cue chuckles)
- Some get involved because they are paid to
- Some for ideals of free and available
- Stay because of the community
- Community is better when you can measure the impact of members

2.1.4 Open Web

- Believes in an open web * Shows phone that boots to Gecko * Someone in Mongolia wrote about how excited they were for access to books * Could send html books instead of text messages

- People made huge sacrifices to make ease of use with open and free software * Stay up all night to get a modem working
- Free != open * Just because it's free, doesn't mean it has the ideals of open software
- We haven't defined what it means to have an open web service
- I want you to host my data, but what kind of access do I need to make it open?
- You may create a web service that puts you into a position you don't want to * Give users tools along the way so that they don't feeel disempowered
- We need to help change the world so we get fewer phone calls
- Things to help this (Mozilla examples) * Do not track movement * Browserid (now Persona)
- Backup is important, as well as delete
- "Are you sober enough to publish this picture?"

2.2 Paul Graham Keynote

2.2.1 Author

- Paul Graham
- YCombinator

2.2.2 Silicon Valley

- The center of SV moves around the peopl who make the next generation of stuff * So, this room is right now
- The frightening-ness of big startup ideas
- List of 7 gigantic startup ideas
- Scary, maybe I should do that recipe site instead
-

2.2.3 Next Google

- Start next Google
- Microsoft lost their way when they got into the search business
- Google has been getting into the social network business
- Nostalgic for the right answer from google * Seems based on Scientologist: "What's true is what's true for you"
- Find tiny idea that turns big idea * Dinosaur egg
- Search engine for top 10k hackers
- Make the search engine the one you want
- Don't worry about something that constrains you in the long term

2.2.4 Replace Email

- Any big idea has a bunch of people nibbling around it
- Not designed to be used the way it is now * Bell labs “Want to go to lunch?”
- Now a shitty todo list
- Tweaking the inbox is not enough
- Todo list protocol insteayd of messaging protocol
- Sending emails to yourself
- Want to know what they want you to do
- When does it need to be done?
- Whenever powerful people are in pain, that is the way to make lots of money
- Gmail has gotten painfully slow
- People will pay for faster email

2.2.5 Replace Universities

- *claps*
- Last couple of decades, universities seem to have gone down the wrong path
- Expensive country clubs

2.2.6 Kill Hollywood

- Hollywood was slow to embrace the internet
- Internet beat cable
- Bolted an iMac to the wall, found it better than a TV
- TV seemed like it was designed by the same people who designed the thermostat
- How do you deliver drama via the internet?
- You kind of want to know what you’re going to get with a show

2.2.7 A New Apple

- If Apple won’t make the next iPad, who will? * Empirically, it’s none of the incumbents
- It will be a startup * Not crazy, Apple did it
- Steve Jobs showed us what one person can do
- “Steve Jobs unrolled the future like a carpet”
- The next CEO might not live up to Steve Jobs, but doesn’t need to * Just needs to be better than HP, Samsung, Motorola

2.2.8 Bring Back Moore's Law

- Circuits are going to get twice as dense, not twice as fast
- Hardware would just solve software's problems
- Need to rewrite it to be parallel
- It would be really great by making a lot of CPUs look like one
- The most ambitious is to do it automatically via a compiler * "Sufficiently smart compiler"
- If not impossible, expected value is really high
- Less ambitious is to start from the bottom * Build programs out of more parallizable lego blocks * Programmer still does a lot of the work
- Middle ground is a semi automatic weapon * Looks like a sufficiently smart compiles, but there are humans in there
- Make a market place, let people do it * Maybe make bots that will do it

2.2.9 Ongoing Diagnosis

- Imagine the ways we will seem backwards to people in the future
- Seem barbaric to wait for symptoms to be diagnosed
- Bill Clinton had to wait for arteries to be 90% blocked to find out
- Launch fast and iterate may not work for medical. * Work on pigs first * Sausage company on the side
- The medical profession will be an obstacle to this
- Doctors are alarmed to look for problems that aren't there
- If you start testing people all the time, you may get a lot of terrifying false alarms
- Think this is an artifact of current limitations
- Going against medical tradition

2.2.10 Tactical Advice

- For big problems, don't make a frontal attack
- "Are we there yet?", Haters
- Notice that you replaced email when it's done
- Start with small things, let them get big * Facebook
- Maybe big ambitions are a bad thing * The bigger they are, more likely to be wrong * Don't identify, just think there is something out there * When the opportunity comes to move, move there
- Blurry vision may be better

2.3 Graph Processing in Python

2.3.1 Author

- Van Lindberg

2.3.2 Graphs

- Uoniversal datatype
- Probably not the best fit if you don't have a relationship

2.3.3 Python-Dev

- "Who talks to whom?"
- Nodes are people
- Edges are "responded to on Python-dev"
- Centrality * Intuitively, the more central, tend to connect others * Dict to map person to how central they are * There's a fairly tight knit community, with smallers around the edge * Antoine Pitrou was the most likely to respond
- Topics
- Nodes are people and topics
- Edges are "commented on"
- Filter out too-common topics

2.4 Fast Test, Slow Test

2.4.1 Author

- Gary Bernhardt

2.4.2 Suites

- Prevent Regression * Weakest, doesn't change how you build
- Prevent Fear * Being able to change things minute to minute, and have test verify * Where speed comes in
- Prevent Bad Design * Holy Grail of Testing

2.5 Stop Writing Classes

2.5.1 Author

- Jack Diederich

2.5.2 When Should I refactor

- When there are two methods, and one is __init__
- When you write functions around classes

2.5.3 Evolution of an API

- MuffinHash replaces a dict
- Was two lines, and obfuscated the code
- 1 package, 20 modules

2.5.4 Version II

- Easy to read
- Two methods, __init__ and call

2.5.5 Version III

- stdlib parts, 6 lines
- 1 function

2.5.6 Namespaces

- Preven collisions
- Not taxonomies
- Otherwise extra things to type, remember
- Anytime you make a class, ask “What am I using it for?”
- Reuse stdlib exceptions
- Don’t complicate the names of your exceptions

2.5.7 stdlib

- 200k sloc
- avg 10 files per package
- 165 exceptions

2.5.8 Classes

- great for containers
- heapq doesn’t use a class
- Probably should be a class, since functions looke like methods * first param is data

2.5.9 Game of Life

- Cell and Board classes
- Board has two methods
- Refactor to dictionary and function
- Well, cell can be refactored to the key of the dict
- Two functions and a dict

2.6 Code Generation in Python: Dismantling Jinja

2.6.1 Author

- Armin Ronacher

2.6.2 Why?

- Isn't it evil?
- A security problem?
- Bad for performance?
- Not if you do it right.

2.6.3 Security

- Code Injection
- **Pollute namespace**
  - Change local variables
  - Can evaluate code in different namespace

2.6.4 Performance

- Alternative: Write an interpreter
- Too slow
- Not suitable

2.6.5 Eval 101

- Compile function to make code objects
- evan can work on a namespace
- Using ast module, can alter underlying structure
- Can use ast to add in line numbers to nodes

- Don't pass strings to eval/exec, but use code objects
- Explicit compliation and namespaces, to fix problems

2.6.6 Jinja

- Jinja and Django have C inspired scoping rules
- **Pipeline**
  - Lexer
  - Parser
  - Identifier analyzer
  - Code generator
  - Python source
  - Bytecode
  - Runtime
- Only runtime is necessary

2.6.7 Scoping

- Context objects are dict-alike
- Slow
- Resolve in context ahead of time

2.6.8 Code Generation

- Low level
- Target byte-code
- High level
- AST generation
- Bytecode doesn't work on appengine, and is implementation specific
- Would be nice to map jinja to bytecode
- Ast is limited, easier to debug, and doesn't segfault

2.6.9 Tale of Two Pieces of Code

- scope in a function is faster than global scope
- lookup via index instead of name
- local dictionary isn't generally used
- semantics can be mapped to fast execution environment
- Jinja context is data source

- Django context is data store
- You cannot modify context in Jinja

2.6.10 jsonjinja

- Semantics of jinja, in javascript
- https://github.com/mitsuhiko/jsonjinja

2.6.11 Q&A

- If you had the chance to redo would you use ast? * Yes, there are utility libraries that help this
- ctypes for line numbers? * put special line number variables, monkey patch traceback * works in everything tested, including pypy * Some problems on some architectures.

2.7 Putting Python in PostgreSQL

2.7.1 Author

- Frank Wiles

2.7.2 Why

- Usually you want pl/pgsql
- Sometimes you want a scripting, with libraries, etc.

2.7.3 Installing

- Aptitude: *postgresql-plpython*
- homebrew

2.7.4 Setting up the database

- createlang plpythonu <databasename>
- Check with *SELECT * FROM pg_language*
- Python is untrusted
- Can set this up in templates

2.7.5 Writing your first function

- CREATE OR REPLACE FUNCTION

2.7.6 Debugging

- plpy.notice, debug, error, and fatal
- Will access the log file directly
- Can use logging

2.7.7 Problems

- Pain to maintain and debug
- Can confuse the dba
- Not free, cached

2.7.8 When

- Rolling up/aggregating data * Remove network, sql parsing to keep runtime low
- Enforce new constraints that aren't in SQL
- Protect data integrity

2.7.9 Triggers

- CREATE TRIGGER...
- Throw a Python exception
- The TD variable has a lot of stuff in it

2.7.10 Redis

- Can use system libraries
- Update Redis unread count automatically

2.7.11 What can you do?

- Executing other sql, create materialized views
- plpy namespace has execute

2.7.12 Ideas

- Lots of them
- Celery tasks, caches, backups, apis, zeromq
- Emails, inserts into another system, send an sms

2.7.13 Q&A

- Limit the runtime of the procedure? * Don't think so
- Test Python Code? * Fake it outside
- Automatically cache? * Have to say it's immutable
- How easy is it to specify a python binary? * Can specify per Postgres cluster
- Run postgres queries inside query, infinite loop? * Will time out eventually? * Not yet, will be in 9.2
- Interpreter external or internal? * Didn't hear
- Timeout kill trigger? * Could have connection timeout in code
- PGSQL v. Python was a magnitude difference * Not surprisingly
- Pypy or Jython? * Probably not * Not yet
- Table functions? * Haven't done much with that, mostly just materialized views

2.8 pandas: Powerful data analysis tools for Python

2.8.1 Author

- Wes McKinney
- Recovering mathematician
- 3 years quant
- Building Lambda Foundry
- writing "Python for Data Analysis" * coming out later this year

2.8.2 Pandas

- pandas.pydata.org
- rich relational data on numpy
- high performance tools
- consistent api

2.8.3 Data Wrangling

- Simplify the tools on processing the data
- Don't transfer from R to Python

2.8.4 Testing

- >98% coverage
- Battle tested

2.8.5 Demos

- iPython transformed development
- Good outside of science

2.8.6 Table

- DataFrame is the core structure
- Axis indexing allows rich data alignment
- Alignment free programming * Often does munging for you

2.9 Lightning Talks

2.9.1 Numba

- Travis Oliphant
- Python compiler
- **For numpy and C extensions**
  - Pypy not good enough
- Dynamic compilation
- **Scipy needs a python compiler**
  - Allows higher level SciPy
- Numba
- Replaces byte-code with type inference
- Uses LLVM
- Dothoes codegen
- Uses C function pointers
- LLVM works with everything
- Uses a decorator to compile
- High bandwidth communication to llvm
- Python for high level, LLVM for low level
- DSLs based upon these
- https://github.com/numba/numba

2.9.2 I has a money

- Chad Whittaker
- Mint stores passwords in cleartext.
- ihasamoney.com

- Personal finance for geeks
- j/k to navigate, no mouse

2.9.3 Brain Hacking

- Talks are bad (but not here)
- Code for brain
- No spec for the brain
- Tell a story
- Implausible story better than plausible story
- Make them care * Babies are better than code
- Show puzzles not solutions * If you show the solution, they won't care
- Have to practice in order to get good

2.9.4 Python 3 on Pypi

- Brett Cannon
- "Pie-pee-eye"
- 54-58% of the top projects support py3k
- Some are under dev, like Django
- The goal was 5 years
- 3 years was the stretch
- Update your metadata, e.g. "Programming Language :: Python :: 3.2"
- Public shame
- pyporting guide
- added u" prefix to make it easier

2.9.5 Python on IBooks

- Luke Gotzling
- Can run interpreter in an ebook
- Embed an interpreter in javascript in an html widget
- 4.8 mg overhead
- Runs on vanilla ipads

2.10 Keynote: David Beazley

2.10.1 Author

- David Beazley

2.10.2 Let's Talk About (something diabolical)

- Let's talk about Pypy
- Python implemented in Python
- Quite a bit faster because of magic
- Mandlebrot runs 34x faster
- Which one can you adjust with a pocketknife?

2.10.3 Thinking about Tinkering

- CPython has patches, extensions, ideas
- Talking about GIL, etc, wouldn't be possible without tinkering
- iPython notebook is an examples of this
- Is it just "evil geniuses"?
- Can you tinker with PyPy?
- Can I teach myself to tinker with it using just resources available, part-time?
- Building PyPy is challenging
- Takes hours, > 4gbs of memory, might break C compiler
- RPython is a restricted subset of the language, but can run as valid Python
- RPython is defined by the translation toolchain
- If you love Python, you will hate RPython
- Uses type inference
- Lists need to be of a single type
- Pypy uses the bytecode interpreter and an abstract runtime to compile to C code

2.11 Why PyPy By Example

2.11.1 Authors

- Maciej Fijalkowski
- Alex Gaynor
- Armin Rigo

2.11.2 What is PyPy

- Can't convince that they are not crazy
- Python in Python
- No longer speed of interpreter, speed of running program
- Measuring memory is important

2.11.3 Edge Detection

- Use dynamic objects with __get__ overridden to act like a list
- Do edge detection on a web cam in real time
- Implemented in Python
- In cPython, ~7 seconds per frame

2.11.4 Tracebin

- Successor to JITViewer
- Expose performance information without understanding how PyPy works

2.11.5 Numpy

- Believe easier to add numpy to JIT than a JIT to numpy
- Some good initial results, but not complete

2.11.6 Garbage Collection

- Don't have to call *free*
- History of talk for Pascal
- Everywhere now

2.11.7 Transactional Memory

- How do we use multiple cores? * Semaphores, events, etc.
- Multicore usage
- Two times the execution time * Where we were with GC years ago
- Hard work

2.11.8 Sprints

- Come sprint on PyPy
- We'll help with getting projects working on PyPy

2.12 Flexing SQLAlchemy's Relational Power

2.12.1 Author

- Brandon Rhodes

2.12.2 Denormalization

- Quick to render, hard to update * e.g. IMDB updating an actor where it's stored with movies

2.12.3 Normalization

- Only store data once
- Easier update
- Need to pull data from multiple places

2.12.4 SQL

- Need to model relationships through intermediary table
- **No composite data types**
  - If you see fields like actor_1, actor_2, etc. something is wrong

2.12.5 Storage is Slow

- Indexes let us jump to right part faster
- Keeping records sorted on disk is slow
- Indexes make this faster

2.12.6 How to make it fast?

- Ask one question
- Use explain and indexes
- Domain knowledge can tell us how we can optimize a query * Postgres has an analyzer that does this well

2.12.7 The O Error

- misconception: An ORM just deals with objects, and hides the relational
- You need to know relational

2.13 Hand Coded Applications with SQLAlchemy

2.13.1 Author

- Michael Bayer

2.13.2 What's a Database

- We can put data in and get it out
- Can do queries that allow us to find records with attributes

2.13.3 Relational Database

- Can create derived tables with suqueries
- Set operations
- ACID

2.13.4 How Talk to DB

- DBAPI
- Abstraction layers

2.13.5 ORM

- Maps to relations
- Can map to multiple relations
- Can map object heirarchies to tables
- How abstract should these be? * Should document stores work?
- Relational features are under/misused which causes the mismatch
- Best to not hide, but to automate
- Explicit decisions and automation is "hand-coded"

2.13.6 Hand-Coded

- Make decisions about everything
- Automate these decisions for ease
- Opposite of "wizards", "plugins", and APIs that make implementation decisions
- Can still use libraries and frameworks

2.13.7 Polymorphic Association

- Map multiple classes to something using GenericReferences
- Does magic for us
- Sometimes called GenericForeignKey
- This breaks the C in ACID * Can generate FK that doesn't point to anything
- Implicit design decisions * Magic tables * source code stored as data, which is coupling * Application layer responsible for consistency

2.13.8 SQLAlchemy's Response

- Declarative Base * Composable patterns
- HasOwner, and PortfolioAssets defaults
- Define convention for polymorphic association

2.14 Advanced Celery

2.14.1 Author

- Ask Solem
- Work at VMware, on RabbitMQ team

2.14.2 Overview

- Flexible and Reliable message queue system
- Granularity: the less computation, the more fine grained the task is * Can reuse connections, etc
- Chunking * Grouping fine-grained tasks to reuse resources

2.14.3 Chords

- Sync primitive
- Known as a barier
- Callback the body with the results of the headers
- Native support in Redis, with good enough fallbacks for others
- demo of parallel summariazation using chords
- Can use this to implement MapReduce

2.14.4 Blocking

- Is bad
- Timeouts

2.14.5 Routing

- Smart routing
- CPU based routers would be nice

2.14.6 Cyme

- https://github.com/celery/cyme
- A distributed Celery instance manager
- HTTP based API

CHAPTER 3

Indices and tables

- genindex
- modindex
- search