Event Notes Documentation

Release 1.1

Jathan McCollum

Sep 27, 2017

Contents

This is a collection of my notes from various events, meetups, conferences, tutorials, or other interesting things. I hope you find them useful.

CHAPTER 1

Conferences

OSCon 2012

Dates July 17, 2012 - July 20, 2012

Venue Oregon Convention Center

Site http://oscon.com/oscon2012/

Talks

Assholes are Killing Your Project

Date 2012-07-20

Speaker Donnie Berkholz

Slides http://www.slideshare.net/dberkholz/assholes-are-killing-your-project

The Gist

- Community is critical
- Results depends on community
- Reputation -> Contributors -> Results -> Reputation...
- “The best” of the community convince you to join
- Basically 5/6 people have to not be assholes
- 1 bad is “offset” by 5 good
- Provide expectations and lead by example!

How Not to Confuse Your Open Source Community with Your Customers

Date 2012-07-17

Speaker Stephen Walli

The Gist

- Community: People with time, but no money.
- Customers: People with money, but no time.
- You *give* to the community because they are your testers, evanglists.
- Community provides momentum
- "He who owns the developer base owns the market."
- What do you want from the community?
- Bees vs. Community - James Dixon's beekeeper model: The Bees and the Trees.

Django Doesn't Scale! (And what you can do about it.)

Date 2012-07-19

Speaker Jacob Kaplan-Moss

Slides https://speakerdeck.com/u/jacobian/p/django-doesnt-scale

The Gist

- Large, successful services are using Django (Pinterest, Instagram)
- "Django doesn't scale"
- Frameworks don't scale!
- Frameworks: Easy to start, then you hit a wall
- See Cal Henderson's "Why I hate Django" on YouTube
- "Measure twice, cut once."

Five Ways Django Fails

1. Data collection

- Sentry
- Python's `logging` module
- python-statsd (used by Graphite)
- mmstats (django-mmstats)
- Metrology
- Graphite

- New Relic - not free (Graham @ New Relic works on Python support)

2. Caching

- "Cache rules everything around me" (http://pyvideo.org/video/679)
- "My pinstafacegramisqus is slow!"
- Technique: Resource decomposition
  - edge-side includes (django-esi) ala Akamai, Varnish, et al.
  - 2-phase template rendering (django-phased)
  - client-side composition (e.g. using jQuery)
- "There are only two hard problems in computer science: Cache invalidation, naming things, and off-by-one errors."
- Technique: Serve everything from cache
  - regenerate cache in the background

3. Perceived Performance

- aka background tasks
- A lie you should tell yourself: "Writes are 10x as expensive as reads."
- Use celery to delay write operations
  - less efficient in aggregate, but...
  - better front-end performance
- "Eventual consistency"

4. Metrics

- If you track one metric, track *query count*
- View execution stops to wait for each query
  - Network latency -> db -> network latency
  - Queries happen synchronously
  - Every query blocks
- Use these:
  - `.select_related()`
  - `.prefetch_related()`
  - raw queries

5. Database Optimization

- "ORM is the Vietnam of Computer Science" –Ted Neward
- ORM Inefficiencies
  - Queryset cloning (chaining) is very heavy
    - Use the `Q` object (better)
    - Use raw queries
  - Model instantiation is *slow*
    - ~40k inits per second
    - Use `.values()` or `.values_list()`
  - Saving models
    - Each row is updated in full (all columns)
    - Use `.update()`
  - Bulk inserts are *slow*
- Django needs to be db-agnostic, but *you* don't.
  - Push your db to the max
  - Don't rely on Django to optimize

Effective Code Review

Date 2012-07-18

Speaker Dan Menard

Slides http://cdn.oreillystatic.com/en/assets/1/event/80/Effective%20Code%20Review%20Presentation.ppt

Everyone should do code review!

- Everyone who codes should be involved

Why?

- You write better code when you know it will be judged
- More than one person **understands** the code
- A great way to learn the codebase

What to look for

- Bad design
- Lack fo clarity or conformity (style)
- Performance hazards (iterators, leaks)

What's not important

- Optimization (premature!)
- Skill & experience gaps (e.g. novice reviewing expert's code)
- Personal style ("that's now how *I* would have done it")

When

- Fit code review into the dev cycle
- Ad hoc ("is there a better way to ...")
- Fisheye is a great tool

Remote Teams

- Acts as a REAL status update
- Helps manage async coding
- Builds trust & morale

How to sell it to others

- Not as hard as unit tests!
- Bottom-up approach ("hey check this code, let me know")
- JUST DO IT

Tips

- Solo projects are no exception!
- Try providing feedback to others and asking for feedback in trade
- Don't try to define goals, but rather track your direction
  - "This is the direction we're going... are we ok with it?" vs.
  - "We should have 173 lines of code per developer per day"

Instantly Better Vim

Date 2012-07-20

Speaker Damian Conway

Slides http://tinyurl.com/IBV2012 (also includes vim snippets)

The Gist

- Handy snippets to make vim awesomer

Help

- `:helpgrep` - search help for patterns
- `:vimgrep /pat/ files...` - grep results into vim

Navigating

- `ctrl-W T` - turn a window into a tab
- `:set ruler` - shows line num/pos
- `:help statusline` - customize the ruler
- `nnoremap <SPACE> <PAGEDOWN>` - hit space to page down in normal mode

Persistent Undos

- `ctrl-R` - redo
- `earlier 30s` - temporal undo (30s)
- `later 1m` - temporal redo (1m)
- Normally the undo buffer is lost
  - `set undodir=$HOME/tmp/.VIM_UNDO_FILES` - set undo dir
  - `set undolevels=5000` (default 1000)
- `:help undo-persistence`
- Plugin to warn undo into previous session (see PDF)

Visual block mode

- Specify area to be affected; then specify command
- `ctrl-V`, navigate to select rectangular area
- `:set virtualedit=block` - always stay in block
- Persistent visual selections (in tarball: `plugin/persistentvisuals.vim`)
  - `gv` - restore previous selection
- Column highlighting (in tarball: `plugin/visualguide.vim`)

Searching

- `:set ignorecase` - ignore case
- `:set smartcase` - partial sensitivity: If string has a capital letter, search case-sensitive
- Search highlighting
  - `:highlight search ctermfg=white ctermbg=red`
  - `:set hlsearch` - enable search highlighting

  - `:nohlsearch` - disable highlighted results

Regex

- Metasyntax must be \-escaped, e.g. `\\t`
- Start pattern with `\v` and all metasyntax will be treated as literals
- `nmap / /\v` - make literal search the default
- Search folding to fold buffer on search (in tarball: `plugin/foldsearches.vim`)

Marks

- When you jump, vim leaves a mark
- Mark hotness
  - With persistent undos these persist too!
  - ``` `` ``` - go back to previous mark
  - ``` `. ``` - jump to last place you modified the buffer
  - ``` `" ``` - jump to last place from last session
  - `ctrl-O` - walk history of all jumps - backwards
  - `ctrl-I` - walk history of all jumps - forwards
  - `g;` - backward thru modifications
  - `g,` - forwards thru modifications

Advanced editing

- `y}` - yank paragraph
- `diw` - delete *surrounding* word
- `di(` - delete between `(...)`
- `di"` - delete between `"..."`
- `dit` - delete between (x)html tags `<...>`
- `vipJ` - select paragraph, join all lines together

Large Django sites at Mozilla

Date 2012-07-19

Speaker Andy McKay

Details http://www.oscon.com/oscon2012/public/schedule/detail/24099

The Gist

- Mozilla's addons site is a Django app
- The upcoming App Store is too
- zamboni - Mozilla's Django app

How Big?

- 300k+ addons
- 150M unique views/month
- 2B+ API hits/day
- Backend:
  - MySQL
  - Memcache
  - Elasticsearch
  - Celery
  - RabbitMQ
  - Redis (being deprecated because clustering not in yet)

Monitoring

- Nagios (zamboni-monitor)
- Ganglia (hardware monitor)
- Graphite (django-statsd)
- Navigation timing API (http://w3.org/TR/navigation-timing)

Templating

- Jinja2 used for templating
- jingo - template minification (css, js, etc.)
  - Adds build ids to file imports, too (e.g. `foo.css?build_id=12345`)

Model Caching

- cache-machine
  - connections, models cached in memcache/redis
  - hashes query to map object to query
  - doesn't cache empty querysets

Search

- Elasticsearch (https://github.com/mozilla/elasticutils)

Performance

- Async anything non-user-time
- Be mindful of things that don't complete (use timeouts)
- Process chunks asynchronously too using `.values_list('pk')`:

```
for chunk in chunks:
    task.delay(chunk)
```

- queryset-transform - collapse chained queries
- Firefox update URL gets ~8000 requests/sec
  - 1 hit per addon
  - addons are cached
  - 6400/sec cached; ~1600/sec raw
- Legacy PHP app could handle 550 req/sec, stock Django only 150/sec
- What if we use connection pooling (django-db-pool)?
  - Raw Django: 201/sec (wsgi was capped at 200/sec)
  - WSGI, no Django: 350/sec
  - Pooling, optimized queries: 700/sec (no Django, SQLAlchemy conn. pool)

Deployments

- Big Red Button that logs to IRC
- django-waffle - flip features in your app
- Email errors DO NOT SCALE (use Sentry)

Marketing: Necessary (But Not Evil)

Date 2012-07-17

Speaker VM Brasseur

Slides http://cdn.oreillystatic.com/en/assets/1/event/80/Marketing_%20Necessary%20_But%20Not%20Evil_%20Presentation%201.pdf

The Gist

- Your project needs a marketing statement
- "Our market is ____."
- You are certified by the community you serve

- Demographics (what people are) vs. psychographics (what people think)
- Doble via! (both ways) - no idea why I wrote that

Moving from Apps to Services

Date 2012-07-18

Speaker Craig Kerstiens

Slides https://speakerdeck.com/u/craigkerstiens/p/django-apps-to-services

The Gist

- Build all your Django apps to expose a service
- All inter-application chatter should be via services
- Choose an API that works best for you (REST, SOAP, XMLRPC, etc.)
  - If you do, do it right: Follow best practices
  - Don't make up your own

MVC to API

Application

URLs Controls entry points to views

Views Renders content using templates

Models Maps content to stored data

Service

Provider Controls entry point to endpoints

Endpoint Renders content

Contract Maps to stored data

Compared

Application	Service
URLs	Provider
Views	Endpoint
Models	Contract

Open Source Community Growth as a User Experience Problem

Date 2012-07-18

Speaker Asheesh Laroia, Karen Rustad

Slides http://cdn.oreillystatic.com/en/assets/1/event/80/Open%20source%20community%20growth%20as%20a%20user%20experience%20problem%20Presentation.pdf

Open Hatch

- Open source community help
- User experience design
- Links
  - http://bit.ly/oh-cookbook
  - http://openhatch.org

Improving User Experience

- Foster and encourage volunteer enthusiasm
- Create clear, welcoming entry points
- "User" in this context: contributor
- Know for whom you're designing!
- Know the "workflow" for newcomers (shopping cart metaphor)
- It's like conversions in e-commerce, but for code users
- See: "funnel effect"

Goal

Get more people down the funnel

Solutions

- Get more people (hard)
- Remove steps (aka remove friction)
  - Research why a step eliminates lots of people
- Or make steps easier/faster/more informative
- Where in the process would *you* give up!

Encourage People To Edit

- Identify tasks to work on!
- Publish a wishlist/TODO list
- Make it very clear how to get started, setup, install
  - Test your docs!
  - Get people to follow the docs to install

Case Studies

nano

The text editor (that replaced pico)

- For patch docs: "send 'it' to the nano address"
  - Send what exactly? The patch?

GNOME

GNOME Love project

- "Get involved" link: Good
- Index page: too much detail, confusing/overwhelming
- Get involved steps are in the wrong order

LibreOffice

Easy Hacks project

- "Developer" and "Contribute" links: Good (both link to same page)
- Build step is optional and last resort: Good
- List skills required for each step: Good
- List of "easy" hacks for suggested fixes: Good

UX Cookbook of Strategies

Fedora Design Bounties

- "One-click shopping" for OSS contributors
- Quarterly blog post announcing winner w/ prize
- Winner gets committer perms
- Specific tasks: reward is LOVE

Open Hatch Training Missions

- Cross-project skill checks
- Automated training sessions
- Learning without embarrassment
- Clear progress tracking, improved feedback

Gentoo GSoC Tracking

- Quantitative community mgmt
- Interns check in regularly w/ mentors
- Iterative tracking of the progress to a goal

Wikipedia Revert Message A/B Testing

- Auto-detection of vandalism to pages
  - Discourages legit new/undeducated users
- Rejection sans rejection
- How to reject patches without crushing souls?
- Rejection email links to "Talk" page

Summary

- Better retention is possible
- Diversity is key!
  - Consider accidental bias by high barriers to contribution
- If you ask for **specific help**, you'll probably get it!
  - Delegate to users ("You: fix this!")
  - Wishlist/todo list is a Good Thing™.

Open Stack: The Future of Computing

Date 2012-07-17

Speaker Joshua McKenty

Detail http://www.oscon.com/oscon2012/public/schedule/detail/24164

The Gist

1. IT is dead
2. Convergence
3. Profit!!

The Meat

- Traditional SysAdmins are like human bowling pin-setters: obsolete
- "Doing things right" vs. "doing the right things"
- Effects in play
  - The Network Effect (aka Metcalf's Law)
  - The Edge Effect (the more different things, the better)
  - Meatware (real people)
- Laws in effect
  - Nielson's Law (bandwidth) vs. Kryder's Law (storage density)
  - "The Mobility Gap" between storage and bandwidth
  - Moore's Law (computing power)
- The Internet is a DAG (Direct Acyclic Graph)
- If we have quantum computing, we have no crypto: Time to rethink!
- Post-convergence for PC (e.g. Apple -> Pagemaker)

Summary

- Things are interesting, new, and exciting in this era.
- The goal is to make your platform boring!

OSI - More Relevant Than Ever

Date 2012-07-20

Speaker Tony Wasserman

Slides http://cdn.oreillystatic.com/en/assets/1/event/80/OSI%20-%20More%20Relevant%20Than%20Ever%20Presentation.pdf

The Gist

- 14 years of OSI (founded in 1998)
- There are 66 OSI-approved licenes: Good luck getting a new one approved!
- Focus on education, advocating, bridging communities
- Stakeholders:

  - Communities
  - Individuals (new!!)
  - Businesses
- Foundations as affiliates
- Individual membership is new! ($40/year)
- San Francisco city has an open source policy
- Educating people about FLOSS

Exhibitors

DRBD - High availability network disk replication

The Gist

DRBD refers to block devices designed as a building block to form high availability (HA) clusters. This is done by mirroring a whole block device via an assigned network. DRBD can be understood as network based raid-1.

http://drbd.org

Highlights

- Open source
- Synchronously replicate storage over the network using commodity hardware

OW2 - The open source community for infrastructure software

The Gist

OW2 hosts 100+ open source projects spanning application platforms, components, middleware, and tools for the development, deployment, and management of distributed applications.

http://ow2.org

Black Hat 2012

Dates July 25, 2012 - July 26, 2012

Venue Caesars Palace

Site http://www.blackhat.com/usa/

Talks

DDoS Black and White "Kung Fu" Revealead

Date 2012-07-28

Speakers Tony Miu, Anthony Lai, &c;

Slides https://media.defcon.org/dc-20/presentations/Lai-Miu-Wong-Chung/DEFCON-20-Lai-Miu-Wong-Chung-DDoS-Kungfu.pdf

How to DDoS a Site

- Analyze web apps for high-cost methods (GET vs. POST)
- Calculations of resources w/ high cost, esp. db ops, large files
- Test for Referer, Keep-Alive, Pipelining
- Force to not load from e.g. (e.g. /uri/?foo)
- Spoof Content-Length, set to irregular value
- Focus on TCP & HTTP (TCP x HTTP killer)
- Focus on TCP *state*
- Use HTTP as example to control TCP state
  - Server reserves resources to control TCP state
  - Focus on reply of server: FINACK, RST, HTTP 302, etc.

Target TCP States

TCP State	Defense
established	lower timeout
FIN_WAIT_1	RST timeouts
CLOSE_WAIT	TCP timeouts

FakeNet

Date 2012-07-26

Speaker Andrew Honig

Link http://practicalmalwareanalysis.com/fakenet/

The Gist

- Fakethat is intended to run on Windows XP.
- Allows you to hijack all socket connections received by the system
- Has an embedded Python 2.7.3 interpreter (custom modules!!)
- Feasible for a single-system dummy test network.

Caveats

- Custom socket I/O; _socket.so module was intentionally excluded
- In order to do socket calls you must import FakeNet.
- Greatly simplified interface, handles buildup/teardown of session
- You worry about send/recv only

Hacking with WebSockets

Date 2012-07-26

Speakers Mike Shema, Sergey Shekyan, Vaagn Toukharian

Slides http://media.blackhat.com/bh-us-12/Briefings/Shekyan/BH_US_12_Shekyan_Toukharian_Hacking_Websocket_Slides.pdf

The Gist

- "Behold the bi-directional browser"
- 2-way comm
- Untrusted code
- Forcing persistence on a non-persistent protocol!
- RFC 6455
- Tunnel arbitrary data (JSON, XML, HTML, images, video, sound... ANOTHER PROTOCOL)

WebSocket Emulation

- web-socket.js - Flash raw sockets with Flash "security"
- sockjs-client - Pure JS
- Force HTML5 in non-HTML5 browser

Why Worry!

- 0.15% of sites today use WebSockets
- Most are for support chat (95%)
- Among remaining 5%, < 1% using crypto
- OLD THREATS (DoS, MitM)

Analysis of NFC Attack Surface

Date 2012-07-25

Speakers Charlie Miller

Slides http://media.blackhat.com/bh-us-12/Briefings/C_Miller/BH_US_12_Miller_NFC_attack_surface_Slides.pdf

The Gist

- Fuzzing NFC stacks
- Potential attacks and demos
- Many Android phones have NFC. Not iPhone yet.
- He broke into a Google Nexus S and Nokia N9 using NFC!

Motivation

- NFC coming to a phone near you!
- "Server-side" attack vector
- Very hard to test NFC implementations

NFC Attack Surface

- New way to test NFC stacks
- Examples
  - Google wallt PIN brute force
  - parking meters
  - bus passes, gym memberships
  - URL spoofing, vending machines

NFC Basics

- Based on RFID (ISO 14443)
- 13.56 MHz (+/- 7kHz)
- Range: < 4cm
- Data rates: 106, 212, 424 kbps
- Typically on when phone screen is on (not when "asleep")
- Modes: Passive, Active (P2P)

Passive Bluetooth Monitoring in Scapy

Date 2012-07-26

Speaker Ryan Holeman

Slides http://hackgnar.com/article/slides-libraries-and-tutorials-my-defcon-and-black/

The Gist

- scapy-btbb - Open source Bluetooth scanner
- Bluetooth is a frequency-hopping protocol

BTBB

- BTBB = Bluetooth Baseband
- Everyday devices cannot acess the baseband

Address Parts

- NAP - Non-significant Address Parts
- UAP - Upper Address Parts
- LAP - Lower Address Parts

NAP	UAP	LAP
AA:BB	CC	DD:EE:FF

Tools

- BTBB hardware: Ubertooth
  - Kismet plugin: dump BT to pcap!!
- libbtbb - Wireshark plugin

Goal

Get BTBB into Python

- btbb layer in Scapy
- load BT pcap into PcapReader
- read pcap files as they are written
- vendor/metadata support (resolution)
- Use iPython w/ iPython Notebook (!!)
- Pandas for graphing/plotting

A Plan for Permanent Network Compromise

Date 2012-07-28

Speakers Phil Purviance, Josh Brashars

Slides http://media.blackhat.com/bh-us-12/Briefings/Purviance/BH_US_12_Purviance_Blended_Threats_Slides.pdf

Browser-Based Attacks

Old Skool

- Exploit Windows
- Exfiltrate data
- Detected/removed by AV

Nu Skool

- aka “Blended threats”
- multiple vectors (worm gets email, back-door for infection)
- Break free of the browser and into the network

Why Attack Network Devices?

- Hard to detect w/ AV
- Non-standard upgrade model
- Ignored by users if service keeps running

Compromising Network Devices

- Rogue SOHO/wifi routers (!!)
  - More common than you think
  - Engineers, careless QA plugging into Enterprise
  - Default settings!!
- Bridging enterprise via VPN from compromised home users (!!)
- Worst case scenario:
  - Make browser do as much as possible
  - Make end-user do all the work
- Proof-of-concept: 1 JavaScript program
  - Hijack ad networks, upload sites, online surveys
  - Social network sites
  - Exploiting non-technical friends/family with spam posts

Network Scanning w/ JS

- JSScan
- JS-Recon
- jslanscanner

- Enumerate IP addreses/ports with dynamic element creation (to load an image) - code makes a request on the LAN to see if reachable
- WebSockets
- Pwning SOHO/home routers w/ default credentials
- HTTP Basic Authentication

SCaLE 11x

Dates February 22, 2013 - February 24, 2013

Venue Hilton Los Angeles Airport Hotel

Site http://www.socallinuxexpo.org/scale11x

Talks

Hockeypuck: Open source GPG keyserver

Speaker Casey Marshall

Date 2013-02-24

Site http://launchpad.net/hockeypuck

Intro

- Public Key Servers to validate identities
- GnuPG Keyserver example:

```
gpg --keyserver ketserver.foo.com --search-keys me@foo.com
```

- Secure email (SMIME), e.g. Thunderbird + enigmail plugin
- Verifying software packages (apt, yum, App Store, etc.)
- OS packages:

```
sudo apt-key adv --keyserver keyserver.foo.com --recev 7F0CEB10
```

- Search by name:

```
gpg --keyserver ks.foo.com --search-keys barack obama
gpg: searching for "barack obama" from hkp server ks.foo.com
```

What is a Key Server?

- HKP (HTTP Keyserver Protocol) - RESTful HTTP API
- Read Ops
  - `/pks/lookup?`

  - ‘‘search={kw} - fingerprint or shortened suffix
  - `op=get`
  - `op=index`, `op=vindex`
- Write Ops
  - `/pks/add`
  - Merge key material (new sigs, revocations)
- PKS (SMTP Protocol)

SKS: Sync Key Server

- Berkley DB storage
- Global distributed db
- SKS global pool, highly distributed and sharded
- Makes it difficult for anyone to control or forge a key

Gazzang zTrustee

- Commercial KS, closed source
- Opaque object store
- “You get what you put”
- Built on OpenPGP
- Cannot decrypt
- Multi-factor authorization (email or API)

Hockeypuck

- Written in go
  - Simplicity, modularity
  - “Static duck typing”
  - Concurrency
- Backed by MongoDB
- Indexd by reversed fingerprint
- Full-text search
- Pretty fast: 2M keys in ~24 hours
  - OpenPGP v3 sigs not yet supported
  - Database is the bottleneck (thanks MongoDB)

PGP Self-Signatures

- Signature-checking "optional"?
- If you don't check it, others can add attributes (BAD)
- GPG will throw it out because it is not self-signed and doesn't match

SKS Reconciliation

- Distributed db
- Reperesent db as polynomial, diffs as a ratio
- Share sample points and ... interpolate??
  - See: Conflux (github cmars/conflux)
  - ...not just for keys anymore!
- Content-addressable things (digitally sign all the things)
  - Messages, images, contacts
  - File/folder sync
  - Location servers

Building Trust

- Add multiple factos to a key exch
  - rendezvous for key exchange
  - identites beyond email (e.g. facebook)
- Keep keyservers distributed
  - Promote diversity, avoid monoculture

Identity Management

- Certify yourself
  - Provide id to ks
  - Sign something, upload to an account you control
  - Link a public key to OpenID or OAuth accounts?
- Hide yourself
  - From all searches?
  - Unless search request is signed by X keys?
- Delete yourself!
  - Store this key on zero or more keyservers
  - "Hey, this key server burned me!"
- Decide for yourself

  - Group-level visibility, certifications

SSH

Host Authenticity

- Trust established the first time you connect (accepting host key)
- SSHFP DNS records
- What about cloud?
- J-PAKE mediated by Hockeypuck
  - Secure remote handshaking by intermediary
  - Authenticate key out-of-band!
- SSH, shell script wrappers and Hockeypuck

User Authenticity

- Some SSH key server you might not know...
  - Launchpad (trusted users can see each others SSH keys!)
  - GitHub
- `ssh-import-id gh:cmars lp:kirkland`
- Add authenticity with PGP
  - `ssh-import-id pgp:0x44A2D1Db`
  - "Server-side Monkeysphere"?

Certificate Authorities

- ONE KEY TO RULE THEM ALL AND IN THE DARKNESS SIGN THEM (lulz)
- convergence.io protocol, act as notary
- PGP sigs on certs
- "Too big to fail?"
  - Translation: "Brings failure to others"
  - Or: "Youre' not trying hard enough"

logstash - Open Source Log Parsing

Speaker Jordan Sissel

Date 2013-02-24

Site http://logstash.net

Blog http://sysadvent.blopgspot.com

Why

- Mascot is a log with a mustache!
- Because logging sucks.
- You wrote some crazy-ass regex and now you're covered in birds
- Other options:
  - graylog2
  - flume
  - storm
  - elsa
- Distributed as a Jar, written in Java.

Case Study: Email

Old Solution

- reduced support per bad tooling

New Solution

- Central logstash cluster
- One web search interface
- Faster, better, etc, etc.

Implementation

- Logging agent running on systems
- 7-node logstash/elasticsearch cluster
- Stats
  - 4TB * 7
  - 500M ev/day
  - ~10k events/sec
  - 4 B ev/wek
  - 1/TB/wk
  - 10% peak CPU

How can it help you?

- powerful, flexible
- search, anyalytics
- integrates well
- logstash is a pipe for events (a timestamp and some data)

Inputs

- Where logs come from
- 30 formats supported today
- gemfire, redis, logs, files, etc, etc.

Filters

- 25 built-in filters
- grok: “Describe the shape of your events” (key/value pairs)
- date: THEY ARE ALWAYS DIFFRENT FORMATS
- geoip: “Where is 24.22.31.135?”
- anonymize: Sanitize PII from logs
- complex: k/v-pairs, json, xml, csv, url, multi-line
- mutate: modify events (like sed)
- translate: map values (301 -> “Redirect Permanent”)

Outputs

- ElasticSearch
- Graphite
- Pagerduty
- Redis, ZMQ, RabbitMQ, STOMP, XMPP, IRC... WHATEVER YOU WANT

Principles

- If a newbie ahs a bad time, it is a bug.
- Make it possible, make it correct, make it fast. In that order.
- Open Source: APL 2.0.
- Should be easy to integrate.

Extensions

- Kibana: Recommened web-interface - Flippin' awesome.
- logstash-cli: CLI interface... :P
- cookbook.logstash.net: Useful patterns and docs

Managing the Release Life Cycle of an OSS Project

Speaker Robyn Bergeron

Date 2013-02-22

Features and Change

- Features (feature-based) vs. Sprints (time-based)
- Changes *are* features
- Features without documentation might as well not exist
- Avoid surprises for your end-users
  - Surprises turn people away
  - Surprised users may not come back

Managing Change

- Change is contentious
- Change things sanely, with ample warning
- Document **everything**
- You don't know everything, and new-comers definitely do not
- Learn to live w/ Murphy's Law
  - Document retrospectives/post-mortems
  - Opportunities to share what you learned
- Set expectations
  - Transparency is essential
  - Be forthcoming in your communication
  - Surprises: scary; Invitations: awesome

Full Throttle Database: PostgreSQL 9.2

Speaker Josh Berkus

Date 2013-02-22

New Features

- Latest: 9.2.2
- Read Scalability (350,000+ QPS?!)
- Write Performance
- Parallel Bulk Load - FASTEST POSTGRES EVER
  - 10-15 paralell streams (depending on CPU)
  - ~3x load speed
- Index-Only Scans
  - Problem: "count(*)" is slow in PG
  - Looks at "Visibility Map", skip table lookup
  - Default: sequential scan, index-only: ~50% faster FOR FREE
- Cascading Replication
  - Traditional replication puts heavy load on master's network interface
  - You may now replicate from a replica ("replica master")
  - p2p replication!
  - It's easy:
    - % pg_basebackup -h replica-master -P -x -D .
    - primary_conninfo=
    - from replica-master: "select client_addr from pg_stat_replication;"
    - from master-master: "select client_addr from pg_stat_replication;"
- Replication Improvements
  - recv vs. write modes for synchronous replication
  - satandby-only backup
- JSON
  - JSON data type: STORE JSON TOO!!
  - Array_to_json, row_to_json functions
  - Get query results as JSON!
  - "select row_to_json(books.*) from books;"
  - Cut out the middle-man transcoding between JSON!
- PL/v8 - Pluggable JavaScript engine
  - Create PL plugins using... JS
  - Create indexes based on JS functions!
  - "create index bibrec_author on json_val('author')..."
  - PL/coffee!
- Indexing
  - 7.1: GiST - Generalized Search Tree (ranges, boxes)

  - 8.1: GIN - Generalized Inverted Index (default for full-text search, arrays)
  - 9.1: KNN - K-Neartest Neighbor (promixity, GIS, text similarity/trigrams)
  - 9.2: Space-GiST!
    - "Space-Partitioning Trees"
    - Faster to read/update than GiST!
    - "create index pt_gist_idx on geo using gist(point);"
    - "create index pt_gist_idx on geo using spgist(point);"
- Range Types
  - Temporal range: [2012-04-10, 2012-04-12]
    - Types: tstzrange, timestamptz
    - "select * from copy_hostory where period @> timestamptz '2010-05-01';"
  - Alpha index: [Abbe, Babel]
  - Linear distance: [375.453, 374.441]
- DDL pit stop
  - ALTER IF EXISTS
  - DROP INDEX CONCURRENTLY
  - NOT VALID CHECK constraints
- Instrumentation
  - Get good knowledge of what your db is doing and how it do
  - more autovacuum logging
  - "pg_stat_statements" extension (must be installed) - Real-time stats
  - "select * from pg_stat_statements order by total_time desc;"
    - Not as rich as parsing query logs, but instantly available!
- Better EXPLAIN
  - report filtered-out rows
  - no-TIMING option
  - "explain (analyze on, buffers on) select * from pegbench_accounts;"
- Other - lower CPU wakeups (power saving) - pg_hba.conf improvements (readability mostly) - XML "improvements"

PostgreSQL as a Schemaless Database

Speaker Chris Pettus

Date 2013-02-22

Site thebuild.com

Twitter @xof

Basics

- Requires PostgreSQL 9.2
- NoSQL == Schemaless
- Documents vs. rows
- Persistent object storage

Types

- XML, JSON, hstore, relational

XML

- Documents up to 2GB
- Supports Xpath
- No indexing :(

hstore

- Heirarchial (docments link to documents)
- Key/value store (strongs or other hstore objects/values)
- GiST or GIN indexing on hstore values

JSON

- Validates JSON going in
- Indexable as text for strict comparison
- Uses PL/V8
- PL/V8 Pro-tips:
  - Use V8 that comes w/ PL/V8
  - Functions are not compiled by PL/V8 until first use
  - JSON injection is possible, stay vigilant
  - PL invocation is non-trivial (tangible overhead)

Comparison

- Side-by-side comparison of various types
  - Stock 9.2.2 install
  - Will compare against MongoDB
  - No tuning on app or system

Strategy

- Single-field tables w/ single type
- Wrap queries in object-mapper
- Index the column
- Profit!

Performance

- Test 1.78M records against MongoDB
- Load/Disk/Query results ranked by fastest-to-slowest
- For hstore, raw, GiST, and GIN are tested
- For XML, JSON, MongoDB index type is expression index

Type	Index	Load	Disk	Query (pk)
relational	pk	1	6	2
hstore	GiST	2,5,7	2,3,7	5,6,7
XML	expr	6	4	3
JSON	expr	3	5	1
MongoDB	expr	4	1	4

- Query by name: hstore(GIN) is #1, JSON is #6
- Query by pk: JSON is #1! (Thanks V8!)

Conclusions

- Stock MongoDB is doesn't seem to be more performant than PostgreSQL at querying documents when using accessor functions
  - Be realistic, test w/ real data
- Index your accessor functions!
- Avoid full-table scans
- Consider hstore (most flexible)
- GiST + GIN rock on high-entropy data (b-tree)
  - GIN outperforms GiST as dataset grows

Automated Cloud Provisioning with Salt Cloud

Speaker Christer Edwards

Date 2013-02-23

What is it?

- Disclaimer: THIS IS BETA!
- Public cloud provisioning tool
- Integrate into cloud providers cleanly, quickly, easily
- Manage via maps and profiles
- Pre-installs Salt, but doesn't have to be used
- Meant to be a generic cloud mgmt thingy

Configuration

- Configs are yaml
- Specify provider settings here, too.
- Multiple providers works too

`/etc/salt/cloud`:

```
minion:
    master: salt.domain.tld
```

Profiles

- Designate VMs inside profile config

`/etc/salt/cloud.profiles`:

```
centos_linode:
    provider: linode
    image: CentoOS 6.2 64bit
    size: Linode 512
    minion:
        master.salt.domain.tld
    grains:
        role: webserver
```

- Define Salt master and custom grains
- Allows you to assing roles, or different masters per providers
  - See: Salt `syndics`
  - Syndics are tiered masters (master -> sub-masters -> minions)
  - e.g. one syndic master per cloud provider
  - Controlled by master-master

Maps

- Specify a prfile and then machines to make from it

- Example:

```
fedora_high:
    - redis1
    - redis2
    - redis3
centos_high:
    - mysql1
    - mysql2
    - mysql3
```

Then run it:

```
salt-cloud -m /path/to/map.file -P
```

- `-P` = parallel exec
- Map files can also include grains!

Bootstrapping Salt

- Supports a few ways for strapping Salt onto new machines
- `script:` (<0.8.4) config setting:

```
fedora_linode:
    ...
    script: Fedora
```

- `salt-bootstrap` (>=0.8.4) utility
- src:`saltcloud/deploy` has distro-specific insall scripts (e.g. `Fedora.sh`)
- >=0.8.4 if you omit `script:`, `salt-bootstrap` is used
- Update manually:

```
salt-cloud [-u|--update-boostrap]
```

No Deployment

- You can provision without deploying Salt
- `deploy: False` or `script: None` in profile
- CLI: `salt-cloud --no-deploy -p <profile> <instance>`

What's coming?

- Salt 0.14.0 will include `salt-virt`
- KVM support (libvirt)
- `salt.run virt.query`
- `salt.run virt.init $name $cpu $ram $image`
- Serve images from `salt://`, `http://`, `ftp://`

The Secure Boot Journey

Speaker Matthew Garrett

Date 2013-02-23

The Past

- UEFI is cross-vendor ($3000/year to contrib)
- 2006: UEFI 2.0 - describes method for signing drivers
- 2008: UEFI 2.2 - describes method for image validation
- 2011: Windows 8 client hardware must default ot enforcing UEFI Secure Boot.
  - WTF MSFT?!
  - Will this mean that ONLY Windows 8 can run?

UEFI Secure Boot

- Hash a binary
- Sign it with private key
- On bot, check hash
- Check sig was made w/ trusted key
- Refuse to boot if checks fail

But why?

- If you hijack the bootloader, you get total control.
- Perfectly designed malware could be virtually impossible to detect
- Command & control == Profit
- Therefore: Anti-terrorism (Obviously)

What are our options?

- Drink
- No, really. Drink.
- Break RSA, taking down SSL with it.
- Ok, back to drinking.

Cause Trouble

- Sept 2011: Matthew blogs about the Win8 requirements (aka WTF MSFT?!)
- PANIC
- It's easy to cause trouble:
  - Just tell people Microsoft is trying to take Linux away from you.
- 2 days later, MSFT responds:
  - "Secure Boot is a UEFI standard, not a Windows 8 feature."
  - Translation: Blame the system vendors (yeah, right)
- Bullets are also "standards"
  - Aren't you glad you're being shot at with a standard?

Now What?

- Don't just run around: Run around and scream.
- Try to convince vendors to ship a Linux key?
  - Who would control it?
  - Whould get access to it?
  - What baout licensing implications of shipping objects signed with it?
- What if there was a Red Hat key?
  - What about the PR issues?
  - What about Debian, Ubuntu, etc...
- Can we get aaccess to MSFT key? (Heh.)

Other options

- Can we avoid pre-instaled keys entirely?
- What about installing a new signing key when you install the OS?
- Give up this Linux thing, take up goat farming.

December 2012

- Vendors must provide a mechanism to disable Secure Boot
- Vendors must provide a mechanism for users to install their own keys

Did we win?

- No standard way of key mgmt
- No standard way of disabling Secure Boot

- No way of remote deployments
- Complexity & documentation nightmare - how do you get end-users to do this themselves?

Mcirosft Plays Ball

- Commtited to provide open access ot the UEFI signing service
- Sigs are contingent upon not being used to attack Windows (Derp)
- Potential revocatoin of existing sigs (e.g. Red Hat key exploited to attack)

Licenese

- "Everyone knows" GPLv3 requires you to release signing keys
- GPLv3 material must be configurable by the end-user.
- (Everyone is wrong)

Two Getouts

- If it's possible to replace it, you don't need to ship the keys
- If it's not a User Product, you don't need to ship the keys
- (Software isn't a User Product, e.g. a physical object)

User Control

- User freedom is essential
- Users need key mgmt
- Machine Operator Key
  - variables can be limited ot pre-boot
  - keys stored in them can't nbe modified by the OS
  - key install can be limited to physically presetn users
  - code written and contributed by Suse

Secure Boot Support

- Ubuntu 12.10 and 12.04.2
- Fedora 18
- A few smaller distros.

Pre-signed Shim

- Signed binary w/ no intrinsic trust
- Insall distro key as first step of install process
- NO MST!!
- No risk of revocation

Now what?

- Linux can be installed w/out disabling Secure Boot or changing firmware settings
- Users can install and manage their own keys

Exhibitors

PyCon 2013

Dates March 15, 2013 - March 17, 2013

Venue Santa Clara Convention Center

Site https://us.pycon.org/2013/

Talks

Dynamic Code Patterns: Extending Your Applications with Plugins

Date 2013-03-16

Speaker Doug Hellman

The Gist

- Comparison of dynamic code loading in various projects

What's a Plugin?

- Loaded dynamically
- Extends core
- Possibly (usually) unknown source

Why Plugins?

- Better API abstraction
- Separate between core + extensions
- Reduce core deps.
- Strategy Pattern vs. Visitor Pattern
  - e.g. Device drivers
- Indirect code contributions

Ceilometer

- Metering component for OpenStack
- Measuring clouds, Billing
- Extend/customize by deployers
- Components communicate using notification message bus
  - Meter data has event listener on master message bus
- Plugin types
  - message bus
  - reciving notif
  - polling
  - storage

Research

- Sphinx
- Django
- Pyramid
- Mercurial
- Nose
- Trac
- SQLAlchemy
- Nova - primary component for OpenStack
- cliff - Speaker wrote this (sub-commands for main app)
- virtualenvwrapper - Speaker wrote this too :)

Discovery

- Explicit vs. Implicit
- Import reference vs. File

Enabling

- Explicit vs. Implicit

Importing

- Custom
  - Django, Sphinx

    - This is prone to errors
- pkg_resources
    - cliff, virtualenvwrapper
    - This is saver

Integration

- Prompt vs. Inpsect
- Granularity: Fine vs. Coarse
    - Fine: single class or function
    - Coarse: Single plugin might include hooks into multiple parts of app
        * Django is coarse

API Enforcement

- Convention vs. Base Class/Interface
    - Convention: Django
    - Class: cliff (abc), Trace (zope.interface)
    - Duck-typing probably a good idea for classes

Invocation

- Driver: Per use-case
- Dispatcher: slkk;ds
- Iterator: All data is given to all plugins

Designing a Plugin

Discovery/Importing

- Think about entry points
- Consider distribute and pkg_resources
- Be consistent!

Decisions made in Ceilometer

Enabling

- Explicit disabling (e.g. config file)
- Automatic disabling

Integration

- Fine
- Inspect
- App owns relationship

API Enforcement

- Abstract Base Classes
- Duck typing

Invocation

- storage: river
- notif: dispather
- pollsters: iterators

stevedore

- Download: https://github.com/dreamhost/stevedore
  - Docs: http://stevedore.readthedocs.org/en/latest/
- Plugin lib that you should use!!
- Implements plugin patterns
- Wraps pkg_resources
- NamedExtensionManager
  - multple plugins
  - only loads named plugins
  - map() them
- EnabledExtensionManager
  - multiple lugins
  - cheakcs each w/ func on load
  - map() them
- DispatchExtensionManager
  - multple plguins
  - invokes subset on map()
- DriverManager
  - single plugin
  - direct access

Encapsulation with Descriptors

Date 2013-03-15

Speaker Luciano Ramalho

Assumptions

- You know the basics of classes/objects
- New- vs old-style classes

The Scenario

- Selling organic bulk foods
- An order has several items
- Each item has desc, weight, prices, subtotal (weight * price)

A Simple object

- Too simple? What about a negative weight?
- Amazon found customers could order negative quantity and it would credit their cards!!

Classic Solution

- Getters/setters
  - Breaks existing code
  - Protected attributes are no fun
- Protected attributes in Pyhton exist for safety
  - to avoid accidental assignment/override
  - to prevent intentional use/misuse

Validation with property

- Implement `weight` as a property:

```
@property
def weightself):
    return self.__weight

@weight.setter:
def weight(self, value):
    if value > 0:
        self.__weight = value
    else:
        raise ValueError('value must be > 0')
```

- What if we want to do the same thing to `price`?
- Duplicate the getter/setter but for `price`?

Descriptors!

- Abstraction of getters/setters
- Manage access to weight/price attrs:

```
class Quantity(object):
    __counter = 0

    def __init__(self):
        prefix = '_' + self.__class__.__name__
        key = self.__class__.__counter
        self.target_name = '%s_%s' % (prefix, key)
        self.__class__.__counter += 1

    def __get__(self, instance, owner):
        return getattr(instance, self.target_name)

    def __set__(self, instance, value):
        if value > 0:
            setattr(instance, self.target_name, value)
        else:
            raise ValueError('value must be > 0')

class LineItem(object):
    weight = Quantity()
    price = Quantity()
```

- Get/set logic moved to `Quantity` descriptor class.
- These instances are created at import time
- Each access goes thru their descriptor

What is a descriptor?

- A descriptor is any class that defines `__get__`, `__set__`, or `__delete__`
- `target_name` is the name of the attribute from the caller (aka instance)
- Each `Quantity` instance must create and use a unique a target attribute name
- e.g. `LineItem._Quantity_0`, `LineItem._Quantity_1`
  - Instead of using a counter, perhaps you can use `id()` of the object

Room for improvement

- What about making them more descriptive: `LineItem__weight`.
- The challenge
  - When descriptor instantiated, LineItem class does not exist and attributes don't exist either.

What now?

- If descirptor needs to know the name of the managed class attr...
- ... Then you need to control construction using a METACLASS!
- COMPLEXITY!

References

- Raymond Hettinger's "Descriptor HowTo Guide"
- Alex Martelli's "Python in a Nutshell 2e"
- Dave Beazley "Python Essential Reference 4e"

By the Way

- All Python functions are descriptors
  - They implement `__get__`
- That's how a function becomes bound to an instance
  - `fn.__get__` returns a partial
  - this fixes the first arg (`self`) to the target instance

Getting Started with Automated Testing

Date 2013-03-16

Speaker Carl Meyer

Slides http://oddbird.net/start-testing-preso

What is it?

- Untested code is legacy code
- This guy LOVES to test!
- Even the slides have tests! Whoa, man.

Let's Make a Thing!

Git recommendations engine

- gitrecs.py
  - similarity() - Return similarity score for two users
- Testing things manually gets old fast.
  - Updating your tests with your revisions gets older faster.
  - Repetitive and boring... Therefore you skip them.

  - Not easily-reproducable
  - Error-prone
- ... And you ship broken code because it's not fully tested

Automate a test!

- test_gitrecs.py
  - from gitrecs import similarity
- `assert similarity(blah1, blah2) == expected_value`
- Got ZeroDivisionError
  - New test: `assert similarity({}, {}) == 0.0`
- pytest - Test runner
  - Knows how to discover & run test (if name is `test_foo.py`

Test runners

- py.test
- Nose
- unittest (stdlib)
- twisted.trial
- zope.testrunner

Why Tests?

- So you knw when your code is broken.
- Help later devs grok your code.
- Improve the design of your code.
- Testing things that rely on network information is hard
  - You must fake it! (Mock objects with boilerplated data?)
  - These tests are less clear
- Testable code is maintainable
  - The less a function knows about the world...
  - The more robust it is against changes
  - See: Principal of Least Knowledge
  - Therefore: The less of the world you have to setup to test it

Types of Tests

- Doctest is not recommended
  - Fragile
  - No isolation
  - Harder to maintain
  - Good for testing your documentation, but that's it!

Unit Tests

- Test one "unit" of code (func/method)
- Small & fast
- Focused: Informative failures
- Con: Require more refactoring w/ changes:w

Integration Tests

- Test that components talk to each other correctly
- Slower, exercises more code.
- "Black box" tests aka "system", "functional", or "acceptance" tests.

Workflows for Testing

- Make it a habit, not a chore.
- Testing first is more fun (objective!)

Adding a feature

- Write an end-to-end test describing the working feature.
- Start implementation from the outside in.
- Program by wish: "I wish I had a thing that did..."
  - Stub it.
- For each stubbed func write tests describing how it *should* work.

The "spike" workflow

- Write exploratory code to figure out a problem
- Strict TDD says delete your spikes and rewrite them test-first...
- Try it... It can be illuminating.

Retrofitting workflow

- Codebase with out test? (*cough* Trigger)
  - Probably not structured for tests. Oh, bother.
- Start /w system tests.
- Use code coverage as a metric for testable areas.
  - `pip install coverage`

See Also

- tox - test your lib accross multiple Python versions/configs
- mock - easily create fakes for testing (included in Python 3.3)
- WebTest - request/response for WSGI apps
- Selenium - browser automation
- http://pytest.org

Coding with Tests

- Fun and satisfying!
- Replaces fear w/ moxy.
- Results in better code.

Let Them Configure!

Date 2013-03-16

Speaker Lukasz Langa

A Sermon on Reusability

- Let's talk about cars.
- Cars break easily.
- They require special tools.
- Standard tool: "English key" aka crescent wrench
  - A highly-configurable tool

Four Desirable Characteristics of Configuration

1. Composability
   - Operating system configuration
     - user defaults, system defaults, runtime defaults, etc.

2. Readability
    - By humans AND computers
    - To know when an option is toggled
3. Exchangeability
    - Program can write changes back
4. Discoverability
    - Self-documenting
    - Easy to use

World Tour

.ini format

- Not really a standard; informatl standard
- Implementations vary (white space, blank lines, etc.)
- Status:
    - Composable? Highly!
    - Readable? If you're consistent
    - Exchangable? If you keep the encoding the same
    - Discoverable? Mostly, depending on developer

Custom formats

- Apache: Confusing, not the most readable
- Nginx: Kind of c-like (braces), supports nesting, context, scriptable

JSON

- Wasn't designed to be a config format
- It's a transmission format
- Simple, human-readable, unicode, x-platform
- No includes, no cascading, must not output comments

TOML?

- New kid on the block
- By Tom from GitHub (which may lend to popularity)
    - http://github.com/mojombo/toml
- Like ini, but supports cascading/nesting

- Inline comments, datetimes, integers, lists, booleans
- Strings should be UTF-8, single-line and double-quoted

Complex formats

- XML :(
  - Schema validation
  - Not human-readable
- YAML
  - Actually a transmissoin format
  - Lends itself well to configuration
  - Superest of JSON, has native data types/structures
    - Lends itself to security problems :(
  - Not very extensible, very complicated
    - YAML spec is 80 pgs long
  - Significant whitespace is fragile for configuration
- So what about Python?
  - UNLIMITED POWER!!
  - Problematic if user is not a programmer (a contrived problem IMO)

Awkward formats

- DSLs, such as "plain language" formats
- Prone to strange complexity, bugs
- SQLite
  - Postfix can use SQLite databases instead of strings
  - Mongrel2 stores all config using SQLite
  - Ensures only programmers would be able to config your app
- Windows Registry (the horror)
  - Not exchangeable
  - Windows only
  - User vs. program settings is... difficult
  - Single point of failure
  - "kind of" readable
  - Only format that feeds an economy to clean it up

Worst format ever

- The one you made up yourself
- You impose a learning curve on your users
- Your design decisions will be unintuitive
- You will fail at parsing!
- Configuration written once has to be supported forever!

How much configuration do you want?

- One-size fits all
- Dropbox has very weird server-side config
- Config != Data
  - Config: Describes behavior
  - Code: Defines and executes behavior
  - Data: Subject to behavior
- Hard-coding is an anti-pattern
  - Embedding source-code in configuration is too

Practical Configurability

Django!

- Django mixes different kinds of settings
  - Framework behavior
  - App behavior
  - Deployment settings
- Using `execfile()` for config includes?
- Class-based configuratoin templates (e.g. dev, prod, caching)
- .ini configuration for heirarchial configs

File-based configs

- Use configglue
- ConfArgParse
- Configopt
- Python 3's new ConfigParser
  - Dictionary-like API
  - Fetch config from dict

  - Highly-customizable!
  - Stuck on Python 2.6? `pip install configparser`

Logical Failures

Date 2013-03-16

Speaker Luke Sneeringer

Why this talk?

- Identifying logical steps in your thinking.
- All programmers are professional logicians.
- Logical mistakes are easy to make; easier than you may think.

A Question

- Linda is 3 years old, single, outspoken, and very bright. She majored in philosophy. As a student, she was deeply concerned with issues of discrimination and social justice, and also participated in anti-nuclear demonstrations.
- Which is more likely?
  - She's a bank teller. (Yes. 90% of people choose this)
  - Shes a bank teller in a womans rights movement. (No. 10%)
- NO conjunction can be more probably than any of its conjuncts.
  - It's more likely that she's a bank teller than that she is both.

Just Enough Logic

- Logical languages look a lot like programming
  - Booleans: True vs. False
  - Operators: not, and, or, xor, if, iff (if-and-only-if)

Validity

- A set of statements if any of its values are true: NO
- A statement is valid if all permises are true and conclusions are also true
- Validity does not entail truth
  - Invalid conclusions may be true
- Necessary & Sufficient conditions
  - Necessary: if any condition not met: can't be true
  - Sufficient: the oppositeo
- Epistemology: The study of how we know what we know.

  - True belief + faulty reasoning is still true

Fallacies

1. Asserting the Consequent
   - Given a conditional, concluding its converse.
   - If P, then Q
     - Assume P, Conclude Q
   - Inverse (modus tollens)
   - Converse isn't true
   - Example: "If it's raining, then the (uncovered) grass will be wet."
     - Valid: "'Grass not wet, therefore not raining."
     - Invalid: "Grass wee, therefore raining"
     - Invalid: "Grass dry, not raining."
2. Questionable Cause
   - A group of fallacies centered on misidentifying caues
   - P occured, therefore Q happened.
   - "We never had a problem with the air conditioner until you moved into the house."
   - Sequence is necessary but insufficient condition for causality.
   - "The code hasn't changed, therefore it can't be the cause."
3. Hasty Generalization
   - Reaching a conclusion w/ insufficient evidence
   - "3 is prime, 5 is prime, 7 is prime, therefore all odd numbers are prime."
   - NM has towns named "Pie Town" and "Truth or Consequences", therefore all cities in NM have awesome names.
   - "It works on my machine, therefore not a code problem."
   - User inputs that break because we didn't expect that input type.
   - NoSQL for *every solution*!!
4. False Compromise
   - Assuming that a compromise between two statements is correct.
   - If John wants to build a bridge across a 10-mile river, and I don't.
     - You don't build have the bridge.
   - Incrementalism (let's do some of all the things we want)
5. Regression Fallacy
   - Misattribution of causality.
   - When a statistically extreme circumstance occurs, it is usually followed by a return to normal circumstances.

- Misinterpreting this return to normalcy as being the result of a response.
- "Traffic cameras stop accidents."
  - Often installed after a seris of traffic fatalities
- "Observe high cpu, ctake action, CPU goes down."

6. Argument From Fallacy
   - Concluding that because an argument is invalid, its conclusion must be false.
   - Invalid args may nonetheless have true conclusions.
   - Take what you learn, expand it, and learn to spot poor reasoning.
   - Don't throw out the conlsusion, correct the reasoning

Python Profiling

Date 2013-03-15

Speaker Amjith Ramanujam

What is it?

- This is nothing like racial profiling
- All about measuring performance

cProfile

- The jam, part of the standard library
- `python -m cProfile lcm.py`
- Dump to a .pstats file

GUI Profiler: RunSnakeRun

- Makes pretty square maps of stuff
- Sort by column, drill down, awesome.

Tips

- Huge overhead
- "Don't run in production!"
- Slow
- Not realistic

Targeted Profiling

- Critical functions
- Start profiler, profile critical thing, stop profiler
- `from profile_func import profile_func`:

```
@profile_func
def lcm(arg1, arg2):
    # ...
```

- You don't want to affect your critical function!
- Pros: less overhead, finer resolution
- Cons: still slow, manual, littering code w/ decorators

New Relic

- Targeted, hybrid profiling
- Web apps only :/
- (They don't use cProfile)
- Targeted profiling:
  - Web frameworks (django, etc)
  - View handlers
  - SQL calls
  - Monkey patch calls you care about

Hybrid profiling

- Top-level function only
- Timing data
- Capture args
- Waterfall diagram/graphing
- Pros: fast, function args
- Cons: semi-manual, limited instrumentation

Statistical Profiling

- Non-deterministic
- Kind of like overly-attached girlfriend
- Interrupt, inquire, collate = OVERLY ATTACHED
  - Who are you with
  - Where are you

    - When are you coming home?
    - I miss you!
- Interrupt (timer):
    - UNIX signals
    - Threads
- Inquire (trace)
    - Stack frame of every thread
    - import sys, traceback
        * frames = sys._current_frames()
        * traceback.extract_stack(frame)
- 3rd party libs
    - StatProf (UNIX signals, CLI)
    - PLOP (Unix Signals, D3 Call Graph), by DropBox
        * Size of the bubble indicates CPU time
    - New Relic (threads, GUI)

X-Ray Sessions

- New Relic SECRET BETA (until now)
- Deterministic
- Targeted transactions (e.g. checkout page), count, gather, profile threads
- Measure/graph the response & throughput
- PROFILE ALL THE THINGS

Python's Class Development Toolkit

Date 2013-03-16

Speaker Raymond Hettinger

The Challenge

- You write a class, test it, DONE.
- What about how users use it?
- Every new user will stretch (abuse) your code in ways you never conceived.

Our Plan

- Learn the dev toolkist
- See how users exercise your code
- Have fun

Agile vs. Lean

Agile Methodology

- Out with waterfall: design, code, test, ship
- In with: tight iterations
- Core idea: iterate and adapt rapidly

Lean Startup Methodology

- Out with: raise capital, spend it, go to market, fail
- In with: ship early, get feedback, pivot, iterate
- Agile applied to busines

Start coding

- Start with the documentation
- Use new-style classes (inherit from object)
  - This is the default in Python 3
- Variables not unique to instance should not be instance variables
- `__init__()` is not a constructor!
  - Its job is to init intance variables.
- `self` is a convention, use it.
- Class definition is in itself like a module namespace
- Pi is not a constant, it's a variable that never changes. (hrm?)
- YAGNI: You ain't gonna need it!
  - aka don't bog your code down with features you don't even need yet
  - Stay minimal!
- Shared data should be at class level (class variables)
- Don't use floats for your version numbers. Strings or tuples, plz.
- Always use iterators (e.g. xrange) to conserve memory!
  - Stay in L1 cache where possible
- If you expose an attribute, expect users to all kinds of interesting things with it.

  - In Python this is common and normal. Accept it.
- Adapt init/constructor for commoon use-cases
  - CONSTRUCTOR WARS!
    - e.g. Converter functions passed to init, are bad.
    - Everyone should win.
  - Provider alternate constructors
  - classmethods make for great alternate constructors
    - dict.fromkeys(), datetime.fromtimestamp(), etc.
  - And they should always work from subclasses
- Always plan for subclassing! (use `super()`!)
- Use staticmethods to attach functions to classes
- Use dunder prefix for class-local variables (e.g. subclasses)
- Slots save memory, but you lose the ability to inspect, modify
  - Flyweight design pattern
  - Always do them LAST, as a memory efficiency step
  - Cache miss is as expensive as a floating point divide
  - Slots are not inherited by subclasses

The Code

```
"""
Circles, Inc.
"""

import math # module for code reuse

class Circle(object):
    """An advvanced circle analytics toolkit"""
    version = '0.6' # class variable

    __slots__ = ['diameter']

    def __init__(self, radius):
        self.radius = radius # instance variable

    @property
    def radius(self):
        return self.diameter / 2.0

    @radius.setter
    def radius(self, radius):
        self.diameter = radius * 2.0

    def area(self):
        "Perform quadrature on shape of unofrm radius"
        p = self.__perimeter()
        r = p / math.pi / 2.0
```

```
        return math.pi * r ** 2.0

    def perimeter(self):
        "Return the perimeter"
        return 2.0 * math.pi * self.radius
    __perimeter = perimeter

    @classmethod
    def from_bbd(cls, bbd):
        "Construct from a bounding box diagonal"
        radius = bbd / 2.0 / math.sqrt(2.0)
        return cls(radius)

    def dump(self):
        print '   radius:', self.radius
        print '     area:', self.area()
        print 'perimiter:', self.perimeter()
        print

    @staticmethod
    def angle_to_grade(self, angle):
        'Convert angle in degree to a % grade'
        return math.tan(math.radians(angle)) * 100.0

class Tire(Circle):
    "Tires are circles with a corrected perimiter"

    def perimeter(self):
        "Circumference corrected for the rubber"
        return Circle.perimeter(self) * 1.25

if __name__ == '__main__':
    print 'Version', Circle.version
    c = Circle(10)
    c.dump()

    t = Tire(10)
    t.dump()
```

Summary

1. Always inherit from object
2. Use instance vars for info unique to instances.
3. Use class vars for shared info across instances.
4. Regular (instance) methods need `self` to access instance data.
5. Use classmethods for alternative constructors. They need `cls`.
6. Use staticmethods to attach funcs to classes, They don't need `self` or `cls`.
7. `property()` lets getter/setters be invoked auomatically w/ attr access
8. `__slots__` implements Flyweight Design Pattern by suppressing instance dict.

Solid Python Application Deployments for Everybody

Date 2013-03-16

Speaker Hynek Schlawack

Slides http://ox.cx/d

Warnings

- Heavy Opinions ahead
- We're not talking about PaaS, schema migrations

Key Concepts

- Responsible Deployment Cycle
- Easy != Simple
- "Simplicity is a prerequisite for realiablity." - Dijkstra
  - "... and security." - Every security expert ever
- Put **effort** into making your deployments **simple**.

Development

- Always develop your app on the target platform
- Avoid inevitable differences!
- See: Vagrant - Maci VM tool for devs.
- What if?
  - Target platform: CentOS 5 (python 2.4.3 =( )
  - "Python 2.4 is not supported. It came out 8 years ago. Upgrade." - Kenneth Reitz

Stability

- You want a stable platform. Key infrastructure?
- Use pre-built packages?
  - NO!
  - Unless you're writing a package FOR a distribution.
  - System packages are often spotty, usually outdated
- Use virtualenv!
  - Pin your deps hard: "Djang==1.4.3"
  - Don't rely on SemVer!!
  - Update w/ pip-tools

Security

- It's **your** job, no one else knows the app like you do.
- You must be responsible for your own service.

Deployment Prequisites

Package it!

- Git + Fabric is "cool" but not stable.
- Why?
  - Build tools on production services could be dangerous
    - resources
    - exploitation
  - Repetitive (compiling C extensions on every server? zzz)
  - Duplication
- Use native packages (.rpm, .deb, .pkg)
  - You can tell "this server is on this version"
  - Introspection
  - CM integration
  - Versatility
- You want reproducabliity.
  - So you can scale.
- Use FPM
- You don't want to run your own repo server?
- Use CLI tools (`rpm -i`, `dpkg -i`)

Workflow

1. check outfrom git
2. create virtualenv
3. install deps
4. do whatever you want
5. profit

- Abuse the Pipeline?
  - Run tests
  - less/sass/cofeescript
  - compression

  - cache busting

Automate!

- Deployment (see his blog post):

```
from _ import Deployment
def deb(branch=None):
    deploy = Deployment('whois', build_deps=['libp-dev'],
                        run_deps=['libpq5'])
    deploy.prepare_app(branch=branch)
    # ...
```

- Parcel: 3rd party lib for x-platform packaging

Configuration

- Don't put your configs in the package
- Use config mgmt
  - Declarative, describe the goal
  - Let CM choose the path
  - Salt, Puppet, Chef, etc.
- Not easy, make an informed choice.

Security

- Never use the same credentials between dev and prod.
  - You know, just in case.
- Never run anything as root. Seriously, just don't.
  - Priveleged ports (<1024)? Drop privs after launch, use authbind.
- Use single purpose wokrers (celery, rq)
  - e.g. "User creation worker", "Service restarting worker"
  - Isolate volatile tasks where logical
- Be paranoid.
  - Use iptables to lock down *everything*
  - Use file sockets (file perms), no listening ports.
- Each app should have its own user/group
  - Set shell to `/bin/false`
  - Use fail2ban
- Each app should have its own datbase and user

Test it in Staging

- Staging must be an **exact** replica of production environment
- Same platform, versions, etc.

Stability

- Don't run it in foreground using screen/tmux!
- Daemonize everything
  - upstart (Ubuntu)
  - systemd
  - supervisord
  - circus
- Apache + mod_wsgi is not your only choic!
  - mod_wsgi is overkill for most cases
- (uWSGI or NGiNX) + gunicorn have better separation of duties
  - HTTP: uWSGI/nginx
  - WSGI: gunicorn
  - ssl, http -> https rediction, headers, etc.

Actually Deploying

- Given you've done everything above...
- ... Once you get to this point, it should be easy.
- It didn't work. Undo! Undo! Rollback! Oh no!!
  - Should also be as easy as reverting the package.

Metrics

- Predict problems before you have to solve them
- Choices:
  - statsd
  - graphite
  - ganglia
  - scales
  - StatHat
- MEASURE ALL THE THINGS
  - CPU, memory, requests in/out, db calls, login failures, etc.

Monitoring

- Pingdom, Nagios, etc.
- Alerting/reporting

The Magic of Metaprogramming

Date 2013-03-15

Speaker Jeff Rush

What it is?

- Code that writes, analyzes, or adjusts other code
- Makes uses of metaclasses, decorators, descriptors
- Python 2.x, new-style classes

Model

- Code -> Data -> Events
- Metaprogramming injects Metacode
  - adding attributes
  - adjusting values
  - registering instances
  - tagging objects
  - latching onto events

Example

- Subclass Request object without modifying the original code
- Catch the import!
- Redfine class to be subclass
- Hook to after import
- Inspect which module is being imported
- Re-arrange module
- Return the module
- `from tau.metaservices import MetaServices`

You can subclass a module!!

- stdlib: `ihooks` module
  - `ihooks.ModuleImporter`
  - `ihooks.FancyModuleLoader`
- `import types; types.ModuleType`
- `class MyModule(types.ModuleType): ...`

How we got here

- Variables, functions, code, blah...
- Rise of C structs, grouping variables into namespaces
- Promotion of functions to "variables"
- Namespace "assembly function" aka constructors
- Prototype pattern: stamp out copies (or instances of objects)
- Shared vs. non-shared namespaces (module vs. instance)
- Rise of iterative lookup (aka scoping)
- Stuff that things have in common (aka classes)
- Classes of classes (aka inheritance)
- Parent classses (aka multiple inheritance)
- Who is creating the classes (aka metaclasses)
- Subclass your metaclasses!

Metaclasses

- Metaclass is a "kind of class"
- New kinds useful for:
  - wrapping complexity
  - domain specific stuff
  - generate classes dynamically (e.g. XML DTDs)
- Example: Create class object from database table

The Naming of Ducks: Where Dynamic Types Meet Smart Conventions

Date 2013-03-15

Speaker Brandon Rhodes

The Gist

- Text-width of 76-79 is perfecto!

Consistent style

- Breaking function calls/defs that are multiline into
- Always add a trailing comma to all containers:

```
foo(
    thing1='thing1',
    thing2=thing2,
)

- Revision control loves this!
```

Type Checking

- Refactor your names and you get "free type checking" by way of consistency

Explicitness

Things to Make Writing Tests Easier

Date 2013-03-16

Speaker Chris Withers

Something to Test

- Parsing CSV data
- tempfile.NamedTemporaryFile
  - Write stuff to it
- `python -m unittest discover`

Reduce Boilerplate

- use `setUp()` and `tearDown()` (w/ `unittest.TestCase` objects)

Libs

- unittest (stdlib)
- testfixtures
- mock (hello, again)
- unittest2

Rich comparisons with testfixtures

Basic comparisons

- testfixtures.compare
  - Gives you a diff on long strings
  - Displays differences when comparing collections (sets, dicts, etc.)

Generators

- testfixtures.generator
  - Rich comparison of generators/iterators!
  - Be careful of just comparing by id
  - Be careful when unwinding

Strings

- testfixtures.StringComparison
  - good for process ids, thread ids
  - Costly (regex): use sparingly

Complex Objects

- testfixtures.Comparison
  - Compare objects that don't support comparison (cool?)
  - `Comparison('module.SomeClass' == SomeClass(1,2)`
  - Useful post-comparison representation
  - You don't have to compare every attribute!

Registering your own comparison objects

- testfixtures.comparison.register
- testfixtures.comparison.compare_sequence
- Strict comparison
  - Relaxed and useful by default
  - Not always what oyu want, so you canbe strict (`strict=True`)
- What about context?
  - No contextual information provided
  - Use the prefix arg:

    ```
    * compare(1, 2, prefix='this is what happened:')
    ```

Things that print

- Lots of code writes to stdout/stderr
- We should test that!
- testfixtures.OutputCapture
  - `with OutputCapture() as output:`
  - later... `output.compare(...)`

Exceptions

- testfixtures.ShouldRaise
  - `with ShouldRaise():  as s`
  - later... `compare(s.raised, ...`

Files and Directories

- Annoying to setup
- Have to clean up
- Difficult to make x-platform
- Reading/Writing files
  - textfixtures.TempDirectory
  - `with TempDirectory() as dir:`
  - `dir.read()`, `dir.write()`, `dir.makedir()`
  - `dir.check_dir()` - Check the dir
  - `dir.check_all()` - Check contents
  - truly x-platform,

Logging

- testfixtures.LogCapture
- Captures logs..
- `with LogCapture() as log`
- later... `log.check(..)`
- Capture a certain level, or logger, etc.
- Uninstall and reinstall to only cap certain code/log events.

Mocking

- Where do you mock?
- testfixtures.Replacer
- testfixtures.not_there
- testfixtures.replace
  - Mock all kinds of stuff: dict keys, list elements, object attributes
- mock.Mock, mock.call

Datetimes

- testfixtures.test_datetime
  - Supports deltas, timezones

Transforming Code into Beautiful, Idiomatic Python

Date 2013-03-15

Speaker Raymond Hettinger

Slides tbd

The Gist

- "Don't do this, do this instead."
- "Everywhere you see this, replace it with this."
- Replace all examples in these slides in your real code. Do it.

Replace things..

- Replace index manip with core looping idioms
- Use for..else and two arg form of iter()

Looping

- Use xrange vs. range; Python 3: range IS xrange
- Use enumerate() instead of range(len(iterable)) for indices
- To loop backwards use reversed(iterable)
- Use zip to loop over 2 collections
  - Use itertools.izip for large collections (or just default to it)
- Use sorted(iterable) to loop in order
- To loop until sential match, use iter(iterable, sentinel)

- Track multiple exit points, use for..else vs. match + break

Dictionary skills

- Fundamental Python skill
- Linking, counting, grouping
- Construct a dict from pairs using zip/izip, pass it to dict()
- Count with a dict:
  - collections.defaultdict
  - collections.Counter
- Group with dicts:
  - dict.setdefault: `d.setdefault(key, []).append(name)`
  - collections.defaultdict(list)
- Is dict.popitem() atomic?
  - Yes. It's thread-safe!!
- Linking dicts:

```
defaults = {'a': 1, 'b': 2}
d = defaults.copy)(
d.update(os.environ)
d.update(cli_args)
```

  - Python3 : ChainMap(cli_args, os.environ, defaults)

Improving Clarity

- Clarify func calls w/ kwargs
  - hours of programmer time > microseconds of performance
- Everywhere you use tuples use collections.namedtuple instead
- Use sequence unpacking instead of indexing!
- Use tuple packing/unpacking for multiple variables (state, assignment in one)
  - `x, y = 0, 1` vs. `x = 0; y = 1`
  - Simultaneous state updates!

Efficiency

- Don't use + to concatenate, use .join()
- Don't use pop, insert, del to update lists; use collections.deque

Decorators & Context Managers

- Separate business logic from admin logic
  - business: Open URL, return page
  - admin logic: cache lookup
  - from functools import lru_cache (Python 2.7, 3.x)
- Factor out temporary contexts
  - If you're repeating setup/teardown, use context managers
  - __enter__, __exit__, with statement
  - Release locks using try..finally (or puppies die every time)
    - Or `with lock` BOOM!
- `with irgnored(OSError)` (Python 3.4):

```
from contextlib iport contextmanager
@contextmanager
def ignored(*exceptions):
    ## do a thing...
```

- `with redirect_stdout(f):` (not even real yet!)

Expressive One-Liners

- Don't put too much on one line; don't put too little either...
- One logical line of code equals one sentence in English.
  - No run-on sentences (statements) !
- List comprehensions/generator expressions are more declarative.
- Everywhere you use [] (list comp) try to use genexp instead

Twisted Logic

Date 2013-03-15

Speaker Ashwini Oruganti

What it is?

- Async event-driven networking framework
- aka... HARD

Endpoints

- Interface w/ a single method that takes an argument
- Use-case, make v4 server work w/ v6.

- Next-case: stdio endpoint.
- You don't have to write your own endpoints. Write interfaces instead.

Don't be afeared

- It's just code.
- Or something...

Deferreds

- Callbacks vs. errbacks
- Flow is not obvious
- Debugging is tricky
- Firing a Deferred is like putting an item into a list with one method, and then returning the value from another

Twisted is HARD

' + It's 'X', when it isn't really 'X'. + It's full of highly complex objects + The problem is how we typically view programs

- It's like Russian stacking dolls.
- Async code doesn't work that way.
- camelCase (PEP-8: 2001-07-05, Twisted: 2001-05-02)
- It's huge!!

Why You Should Use Python 3 for Text Processing

Date 2013-03-16

Speaker David Mertz

Why?

- Native unicode (str is unicode, bytes is str)
- String and bytes have grown handy methods
- Wrote "Text Processing in Python" (download: http://gnosis.cx/TPiP)
- Impressionistic review of nice-to-have improvements
- These features are backported into 2.7
  - 3.1 -> 2.7
  - 3.2 -> 2.7.3
  - 3.3 -> 2.7.4?

Cool Stuff in collections

- namedtuple, OrderedDict, HashMap

namedtuple

- Useful for dealing with CSV and database rows:

```
import csv, collections
users = open('user'csv')
headers = users.readline()
UserRecord = collections.namedtuple('UserRecord', headers)
for row in csv.reader(users, rename=True):
    print(UserRecord(*row))
```

- If you set `rename=True` it renames attributes that may be reserved types
- Less memory than dicts (because of __slots__)

Counters

- Useful for histograms (such as commonality of letters):

```
import collections
c1 = collections.Counter('abracadabra')
print c1.most_common(4)
# 3.3 only:
# c1['d'] -= 10
```

- Pseudo-arithmetic stuff, basically defaultdict to value 0:

```
c2 = Counter('ramalama bim boom')
(c1 + c2).most_common(4)
# +c1 Increment
```

ChainMap

- New in Python 3.3
- Collection of mappings... "container of containers"
- Sneaky equiv. to dnyamic inheritance and MRO
- ChainMaps can include ChainMaps:

```
d1 = {'a':, 1, 'b': 2}
d2 = {'c':, 3, 'd': 4}
chian = ChainMap(d1, d1)
print chain['a'], chain['d]
# => (1, 4)
```

Unicode is hard

- *Most* (not all!) Unicode is in the BMP (Basic Multilingual Plane)
- All of Latin-1 is in range 00 of the BMP
- Internal encoding matters
  - Fixed-width (UTF-32/UCS-4): Uses a lot of memory
  - Variable-width (UTF-8): positing indexing is very slow
  - With UTF-16/UCS-2 you get the worst of everything:
    - Not strictly fixed-width (i.e. surrogate pairs)
    - Usually wasted memory
- I have no idea what any of this means (fixed- vs. variable- width)

PEP-393

- Strings are normally Latin-1
- Python encodes everything in the most compact form it can.
- v3.3 adds back explicit unicode literals `u'bacon'`.

Pro Tips

- `str.startswith()` and `str.endswith()` take tuples as well a string
  - (But not lists or other iterables)
- Module `textwrap`
  - People reimplment this over-and-over (*cough* Trigger)
  - Don't roll your own:

```
textwrap.fill(s, width=35, initial_indent='| ', subsequent_indent='| '))
```

  - Dedent multiline strings:

```
multiline = """
foo
bar
bacon"""
multi_line = textwrap.dedent(multi_line)
```

  - `textwrap.indent()` new in Python 3.3:

```
def my_pred(line):
  return not line.endswith('wrote:\n')
print(textwrap.indent(s, '| ', predicate=my_pred))
```

- Module html.entities:

```
from html.entities import h5ml5, entitydefs, codepoint2name
print html5['Exists;']
```

- Module unicodedata
  - Get names of glyphs
  - Validate, inspect
- Proper quoting
  - Hidden in `pipes.quote()`
  - In Python 3.3 it's moved to `shlex.quote()`
  - Useful for generating shell scripts or CLI/subprocess args
- Use `format()`
  - Mini language
  - More powerful and robust than '%s' style.
  - e.g. Thousand separator (locale aware)
    - `'${:,.2f}'.format(1000000) # => '$1,000,000.00'`
    - This is in Python 2.7, also
- Module email
  - `msg = email.message_from_file(...)`
  - `payload = msg.get_payload()`
  - `payload.get_content_type()`

Exhibitors

CHAPTER 2

Credits

Special thanks to PyDanny for the inspiration for this project. (See: PyDanny's Event Notes)

CHAPTER 3

Indices and tables

- genindex
- modindex
- search