PyCon2012 Schedule & Notes Documentation

Release 1.0

Andrew Schoen

November 20, 2015

Contents

CHAPTER 1

Friday

1.1 Keynote: Stormy Peters, Mozilla Corporation

1.1.1 The Challenge

Everybody in this room is defining the future. Think about how you put the user in control of the web - the average user.

Make a future with cool robots, not a future where companies take you're information without your knowlege. Keep the web open and accessible.

Dancing robots!!

1.1.2 Agena

- How we can create responsible websites
- The Growing Community
- Making tools that are responsible

1.1.3 Growing Community

- Grow in ways the include the new people and existing.
- 136 sponsors this year, 2004 was a fraction of the sponsors
- Tons of people are here for their first PyCon ever
- Veteran PyCon members should find new attendees and welcome them
- **When you meet somebody you have 3 seconds to make an impression**
  - First thing they notice is your hair
  - Second is your shoes (she wears toe shoes)
  - Active memebers in the community should encourage people who make the effort to be involved. Give them a good first impression
- **Why people get involved in the community**
  - Because they have an itch to scratch

    - **But not all of us.**
        * People love to learn something new
        * Solving problems
        * To become famous (laughs)
        * They get paid to (typically more diversity is created by this route)
        * Because they believe it should be free and availble to all people
- Everybody stays because of the **Community**
- Community is much more likely to stay involved if you can measure their impact

1.1.4 How can we create responsible websites

- Mozilla and Stormy believes in building these tools
- Mozilla is completely open with the web as the platform
- **She get's an email from a person from Mongolia**
    - He's excited because he's a teacher and in Mongolia you can't checkout books from the library, but his students can get these books on their phone because of html books from Mozilla.
    - Kids in Mongolia have cell phones, but not very many books
- By keeping the web open and software free we can change the world.
- People make huge sacrifices to make this open web
- The challenge is to keep pushing, keep going. Still more to be done.
- Free != Open
- How can we make websites and services to offer people freedoom

1.1.5 Making Tools

- You have to give users tools along the way so that they don't feel disempowered when they can't to their data
- We need fewer 'it doesn't work'
- **The do not track movement**
    - BrowserId
    - Logging into websites with profiles
    - Give people ways to delete and remove those profiles
    - It's hard to delete your online identity
- We need to rethink the way we build websites
- **How these interact with our online identity**
    - Our different devices
- Keep the user in control and responsble
- **Education is important, but it needs to be more than this**
    - Let user know what they are actually doing on the web, what data they are putting out there

1.1.6 Closing

- Grow the community responsibly
- Create tools the give users control
- Make nice friendly robots

How do we make sure the future is a place where the users have control

1.2 Keynote: Paul Graham, YCombinator

This is way bigger than the last PyCon we talked at. It was 2002. Just a small fraction of the attendance today.

What is silicon valley like?

- The center of silicon valley moves around.
- Where is the greatest collection of people who want to make stuff happen

We are in the center of silicon valley right now, right here

A list of 7 Giant startup ideas (be a billionaire!!)

- The biggest startup ideas are frightening
- The threaten your identity
- Start the next google
- The right ideas are just on the right side of impossible
- **Microsoft really lost their way when they got into the search business**
  - The did this because they were afraid of Google
- Google may have peaked, but doesn't mean there is a room for a new search engine
- Paul is nostalgic to the old google.
- Now results are like scientology results. What's true is true for you (big laughs)

Find the tiny thing that turns into the giant idea. Find the dinosaur egg.

1.2.1 Replace Google

- make a search engine that all the hackers use
- Maybe only 10,000 users but you'd be in a powerful position
- Make the search engine you want

1.2.2 Replace Email

- Email is a to-do list not communication
- But it's a shitty to-do list
- Tweaking the inbox is not enough
- Make a to-do list protocal

- Gives more power to the receipent
- People send emails to themselves is proof of this (I do this all the time)
- Entreanched protocols are almost impossible to replace
- But it's time
- **Powerful people are in pain because of email**
  - This means you can make money on this idea!!
  - You have a dinosaur chicken!
  - Build it and make it fast
- He could justify spending tons of money on this

1.2.3 Replace Universities

- Is reluctant to suggest this.
- The last couple decades universities have went down a bad path
- They are like expensive country clubs
- **Might get replaced by a whole bunch of little things**
  - Many will not look like universities at all
- Change the way people learn
- **Universities are kinda like a credential at this point**
  - So maybe crenditals have to be done outside of this

1.2.4 Kill Hollywood

Huge Applause

- Slow to embrace the internet
- **Can now call the winner for delievry mechanism for entertainment**
  - It is the Internet
- Hates TVs
- Some of the attention people spend watching TV can be stole by other things
- Still a residual demand for people who want to watch a story unfold
- **How do we deliver drama via the Internet**
  - Must be on a larger scale than YouTube
- Two ways delivery and payment can play out

1.2.5 A new Apple

- Was talking to somebody who knew Apple well. Asks the question, "Post Steve Jobs can Apple still innovate" Answer: No
- **Who is going to make the next iPad**
  - Maybe none of the existing players
- Only want to get a product visionary as CEO is for that person to start the company. (And that person doesn't get fired)
- The next Apple is a start up
- Who's going to make the future
- **There is a vacuum. Ready for a new company to lead the way.**
  - People are used to following

1.2.6 Bring back the old Moore's Law

- Hardware was solving softwares problems
- Not anymore
- Would be great if a start up could give back something of the old Moore's Law
- Make many CPU's look and act like one super fast one
- Programmers like convience
- In world of web services we don't see processors anymore
- Instead of the compiler, build stuff out of smaller components. (hadoop, map / reduce)
- **Create a market place for optimization**
  - Write bots to do the optimization

1.2.7 Ongoing Diagnosis

- Imagine the ways we will seem backwards to people in the future
- **Ridiculous we have to wait for symptoms to know we have problems**
  - Example: heart problems
  - Have to wait until your arteries are 90% blocked to know there is a problem
- Surely, in the future this will happen. People will know how blocked their arteries are like their weight
- Next example, Cancer
- Traditionally you feel symptoms, go to the Dr. Need to start looking for problems pro-actively.
- If people are scanned all the time - there will be less freak-out moments
- Going against thousand of years of medical history

1.2.8 Conclusion

- If you want to take on a problem, don't make a frontal attack.
- Don't say you're replacing email, say you're building to-do list software
- Start with small things, make them bigger
- Start small for your own sake, not for other people
- **Bigger your ambitions the longer they take to realize**
  - The more you have to look into the future, the more you'll be wrong
- Columbus "There is something in the West"
- Blurry view of the future is the best

1.2.9 Questions

- The idea of what is property? What is convient to be called property. Land didn't used to be, but it is now.
- Now files move around like smells. Doesn't make sense to charge people for copies anymore.
- **Question about Replacing universities. Other value to them than learning.**
  - Would it be bad if people don't physically meet? Things have to happen in person.
  - Things can be made that involve people meeting in person.
- Get a degree from people, not institutions
- Maybe universities used to be like this

1.3 Introduction to Metaclasses

Presenter: Luke Sneeringer

Track: IV

Description:

> Python's metaclasses grant the Python OOP ecosystem all the power of more complex object inheritance systems in other languages, while retaining for most uses the simplicity of the straightforward class structures most developers learn when being introduced to object-oriented programming. This talk is an explanation of metaclasses: first, what they are, and second, how to use them.
>
> https://us.pycon.org/2012/schedule/presentation/64/

- Metaclasses can be a big scary word
- Typically newcomers are told to stay away from metaclasses
- Some truth to this, but not completely
- This talk is an attempt to teach people about metaclasses who don't understand them and make them understand at a basic level
- When to use them and why

1.3.1 Terminology

Know the difference between class and instances. For this talk object and instance is interchangeable.

- Classes are objects too
- They are first class objects (can be passed around)
- you can pass callables around
- The ability to pass callables is the key to reusable code

1.3.2 Instances

- Class variables are assigned as references back to the class
- **Any method on the class are bound to the instance**
  - This makes `self` work
- Instances are constructed at runtime from the classes
- Classes work the same way (they are instances)
- Objects are instances of their class; classes are instances of their metaclass

Terminology

- Meta is a greek prefix meaning "after"
- Now we use the prefix to refer to a special kind of self-reference
- Therefore metaclasses are classes about classes
- Metaclasses provide code for the creation and execution of classes
- instance are to classes are classes are to metaclasses

1.3.3 The Details

- When making a class inherit from `object`
- Metaclasses inherit from `type`
- Python has multi-inheritance
- First calls __new__ and then __init__

1.3.4 Writing Metaclasses

- `type` is the true top of the chain
- type(type) is type
- To create a metaclass inherit from type
- Use __metaclass__ in python 2
- Most straightforward case is to override __new__ or __init__
- These run when the class is created, not the instances
- This is how Django models work

1.3.5 When Metaclasses

- If you're writing an API or interface of classes or subclasses
- They are useful for making other classes more straightforward
- Sometimes metclasses aren't the right answer
- **Class factory functions are more expensive than metaclasses**
  - But still often useful if you don't know something until execution

1.3.6 Practicing Wisdom

- Some justification for running away from metaclasses
- Don't go meta if you don't need to - there is complexity added
- But use it if you need to
- The key advantage is that they can me be much clearer and cleaning than the alternatives
- Used inproperly it's the exact opposite
- If you're not sure when to use them, ask or research
- The problem defines the solution, not your skill set
- If it is the right solution, make sure you know them well

1.4 Extracting musical information from sound

Presenter: Adrian Holovaty

Track: III

Description:

> Music Information Retrieval technology has gotten good enough that you extract musical metadata from your sound files with some degree of accuracy. Find out how to use Python (along with third-party APIs) to determine everything from the key/tempo of a song to the pitch/timbre of individual notes. Then we'll do some amusing analysis of popular tunes.
>
> https://us.pycon.org/2012/schedule/presentation/6/

1.4.1 What does it mean?

- taking an audio file (bits and bytes) and turning it into musical information (notes)
- use python to take an mp3 and extract musical information from it
- Adrian spends a ton of free time trying to figure out songs on his guitar
- Would be awesome if we could do this automatically

1.4.2 What are we going to use?

- Using python and echonest (free api, I think)
- Using tip toe through the tulips as the example song; a solo guitar piece

1.4.3 Getting Basic Information

- pip install pyechonest
- get api key from developer.echonest.com
- import stuff and use track_from_file from echonest
- **Lots of stuff available about the track by using that function**
  - beats, sample rate, encoding quality, tempo, key
- get confidence numbers on the returned values
- **sections tell you when the song is changing**
  - not perfect, but close
- Tatums (the lowest regular pulse of the music)
- **Segments ("relatively uniform" pieces of the sound)**
  - In practice, every note or drum beat
  - This is awesome, love it (my comment)
  - Loudness (in decibles) - part of a segment
  - **Pitches - can almost extract the notes from this**
    - good start for automated music transcription
  - **Timbre - not sure exactly what this means or represents (the color of sound)**
    - loudness, flatness, attack, brightness

1.4.4 What can we do with this?

- HacKey - shows what keys your songs are in from your last.fm profile
- The Loudness War (Paul Lamere)
- **Click track detection (labs.echonest.com/click)**
  - Django Reinhardt (a jazz musican named after a programming framework)
- LOL, nickelback (Nickelback to Bickelnack)
- moreCowbell.dj
- Swinger and the Deswinger

1.5 Practical Machine Learning in Python

Presenter: Matt Spitz

Track: II

Description:

> There are a plethora of options when it comes to deciding how to add a machine learning component to your python application. In this talk, I'll discuss why python as a language is well-suited to solving these particular problems, the tradeoffs of different machine learning solutions for python applications, and some tricks you can use to get the most out of whatever package you decide to use.
>
> https://us.pycon.org/2012/schedule/presentation/119/

1.5.1 Why Python?

- Python is great for data processing and analytics
- Versatile
- Also has Mature ML Packages

1.5.2 Terms

- **Data points**
  - feature sets and label
- **Classification**
  - training set

1.5.3 SluggerML

- baseball stats and machine learning
- **feature sets represent game state for a plate appearance**
  - day game, night game, at-bat, batter/pitcher
- using training sets and classifier predictions
- **Gathering Data**
  - retrosheet.org
  - sean lahman's baseball archive
- **Coalescing**
  - 1st pass, Lahman: create player database
  - 2nd pass, Retrosheet: track game state, join on player db
- **Scrubbing**
  - ensure consistency
- **Training Set**
  - regular season games from 1980 - 2011

- **Selecting a Toolkit, tradeoffs**
  - Speed
  - Transparency
  - Support
- **High-Level Options for Toolkit**
  - External bindings
  - Python implementations
  - DIY

1.5.4 Python Toolkits

- **Python Implementations**
  - **nltk**
    - focus on NLP
  - **mlpy**
    - regression, classification, clustering
  - **PyML**
    - foscus on SVM
  - PyBrain
  - mdp-toolkit

1.5.5 Feature Selection

- Identifies predictive features
- uses scikit-learn
- Visualizing significance

1.5.6 Tips and Tricks

- **Persistent classifier internals**
  - once trained, save and reuse
  - depends on implementation
- **Using generators where possible**
  - avoid keeping data in memory
- Multicore text processing

1.6 Code Generation in Python: Dismantling Jinja

Presenter: Armin Ronacher

Track: III

Description:

> For many DSLs such as templating languages it's important to use code generation to achieve acceptable performance in Python. The current version of Jinja went through many different iterations to end up where it is currently. This talk walks through the design of Jinja2's compiler infrastructure and why it works the way it works and how one can use newer Python features for better results.
>
> https://us.pycon.org/2012/schedule/presentation/246/

bit.ly/codegeneration for feedback

1.6.1 Why Code Generation?

- **Why is eval evil?**
  - Security & Performance
- Can be avoided if done properly
- **Possible security issues**
  - Code Injection
  - Namespace pollution
- **Performance**
  - no bytecode
  - code makes code that code runs
- use responsibly

1.6.2 Eval 101

- compile an arbitrary string into code
- you get a code object back
- must exec code in a namespace
- in Python 2.6+ you can use ast. import ast
- **Need to specify line numbers and columns on every note**
  - use ast.fix_missing_locations
- **don't pass strings directly to eval**
  - can keep the code object around

1.6.3 Template engine architecture (Jinja)

- 2nd iteration of jinja
- generate python code
- python semantics
- different scoping

How it works

- Lexer > Parser > identier analyzer > code generator

Complexities

- Different scoping
- WSGI & Generating
- Debug-ability
- Restricted Environments

1.6.4 How does it work?

- art of code generation. low level vs high level
- ast was introduced after jinja was created
- high level would be using ast
- low level is bytecode
- **biggest problems with bytecode is that it's undocumented and implemention specific**
  - doesn't work on GAE
- ast is more limited, but easier to debug
- as of python 2.7 you can not segfault the interpreter with ast

Jinja is fast because it operates on the fast aspects of the python virtual machine. Code run at the global scope is slower. Local scope is much faster.

1.6.5 Semantics

- jinja tracks every identifier
- context in jinja is a data source
- context in django is a data store
- impossible in jinja to have a function that modifies context

1.7 Throwing Together Distributed Services With Gevent

Presenter: Jeff Lindsay (@progrium)

Track: V

Description:

> In this talk we learn how to throw together a distributed system using gevent and a simple framework called gservice. We'll go from nothing to a distributed messaging system ready for production deployment based on experiences building scalable, distributed systems at Twilio.
>
> https://us.pycon.org/2012/schedule/presentation/288/

Jeff works at twilio. twilio uses a service oriented architecture. Twilio as a high level service is made up of many subservices - sms, voice, client. Each made up of other services. Mostly written in tornado, gevent. Going to use a framework called Ginko.

- services are nested modules than can start, stop and reload
- going to build a scalable gateway around a self-organizing messaging system.
- first we'll build a simple number client
- next a pubsub service
- then combine both into a gateway service
- and finally make it all distributed

1.7.1 Number Server

- a basic gevent StreamServer
- for every connection generate a random number, sleep for 60 seconds
- configuration is a python file
- ginko has start / stop services
- start the number server in the background

1.7.2 Number Client

- spawns a greenlet relative to your service
- makes services self contained
- gets the numbers from the server and puts it in a queue

1.7.3 PUBSub

- uses httpstreamer
- subscription wraps a queue
- posts are publishes, gets are subscribes

1.7.4 MessageHub

- hub is now responsible for managing subscriptions

If you're in a loop that doesn't use IO, run gevent.sleep(0) to make sure it yields

Code for the example is up here: https://github.com/progrium/ginkgotutorial

1.8 Fake It Til You Make It: Unit Testing Patterns With Mocks and Fakes

Presenter: Brian K. Jones

Track: I

Description:

> In this talk, aimed at intermediate Pythonistas, we'll have a look at some common, simple patterns in code, and the testing patterns that go with them. We'll also discover what makes some code more testable than others, and how mocks and fakes can help isolate the code to be tested (and why you want to do that). Finally, we'll touch on some tools to help make writing and running tests easier.
>
> https://us.pycon.org/2012/schedule/presentation/336/

github.com/bkjones jonesy on freenode (join #python-testing)

1.8.1 What's Covered

- Definitions
- Patterns
- Tools

unittest module is not going to be covered.

1.8.2 What is a unit test?

A unit test is a test that does not require larger parts of the system to run.

- unit tests are granular
- unit tests are isolated
- unit tests are localized
- **not a unit test if something outside of the code you care about has to be included for it to pass**
  - if so, it's an intergration test

Coverage is a loaded term. Generically referred to as "code coverage". Cover all the conditions around the execution of the code, not lines of code.

coverage.py

- Integrates well with nosetest
- super easy to use

Nose

Nose is a great discovery tool. Running nosetests –with-coverage will use coverage.py. Nice HTML reporting and integrates well with Jenkins.

1.8.3 Why Unit Tests?

- They're fast
- They're simple
- They provide greater localization of problems than other types of tests often can.
- running unit tests create the entire execution environment.

Unit Tests Aren't Enough

- They're one component of the complete testing regime. They don't test how things work together.

1.8.4 Mock

Mock is cool. Use it.

http://mock.readthedocs.org/en/latest/index.html

Mock handles harder stuff

- make assertions about what methods on a mock were called, with arguments
- puts objects back the way it found them after a test
- tons more, check out mock
- makes code more testable

1.8.5 Low-hanging Fruit

Making unit tests easier

- **limit the scope of responsibility**
  - will make code flexible and easier to unit test
- **create local wrappers**
  - that talk to external resources
  - better encapsulation, switching libraries without chaning api
- deduplicate the code

Well factored code should be easy to test.

Yes! You do need to test!

Unit tests can be a great form of documentation

Follow the one test, one assertion rule. Overall unit test will be better off. Care for your unit tests like you do your production code.

Tox is cool, check it out later. Testing across multiple python versions.

1.9 Python Metaprogramming for Mad Scientists and Evil Geniuses

Presenter: Walker Hale

Track: IV

Description:

> This talk covers the power and metaprogramming features of Python that cater to mad scientists and evil geniuses. This will also be of interest to others who just want to use of Python in a more power (hungry) way. The core concept is that you can synthesize functions, classes and modules without a direct correspondence to source code. You can also mutate third-party objects and apps.
>
> https://us.pycon.org/2012/schedule/presentation/380/

Python is an ideal language for both Mad Scientists and Evil Geniuses.

Mad Scientist: creating new things because it's cool

Evil Genius: has a goal and will get there no matter who gets in their way

Mad Scientist Goal:

- Create new living code from scraps of other open source code
- Mutate third party code to suit our needs

A class is just a dictionary wearing a class. Python is just dictionaires all the way down.

type's job is to construct other classes. When initiating a class Python is basically just handing you an empty dictionary.

A module is even less than a class.

Applications

- Integrating user-specific formulae
- Domain-specific languages
- GUI Frameworks
- Object-Relational Mappers
- Monkey patching

Monkey Patching

- functions, classes and modules are just objects in memory
- and they can be modified
- Very similar to Aspect-Oriented Programming
- **Patching third-party objects is more robust than patching third-party code**
  - altering the code on the fly

Monkey Patching Modules

- Modify an existing module
- Synthesize a replacement

With either approach you must do this before other code imports the module. Modify the code an then insert that back into sys.modules

Monkey Patching a Class

- use new.instancemethod to add a method to a class

- operates at the class level.
- can monkey patch a specific instance of a class by using new.instancemethod as well

If sitecustomize.py exists anywhere in your code python will import that very early. Can be used to monkey patch executables

When to do this:

- patch a module written in C
- Patch just a few functions you think you need.
- Create an object that actls like a module but really synthesizes teh functions it needs by implementing a custom __getattr__

Limitations

- **Patching third-party code can confuse co-workers**
  - Do it seldom
  - Do it publicly
- Updates to third-party code can still break a monkey-patch
- You can't monkey patch most code that rsides below Python(C, Java) but you can synthesize a replacement

For the evil geniuses

- Networks code - ww.metasploit.com
- Selenium is a man-in-the-middle attack
- Speed up with things like Cython, ctypes, pypy

We live in a golden age where reason and science can triumph over convention and superstition.

Todo:

- Revel in your power
- Laugh your most diabolical laugh
- Bend code to your will

1.10 Practicing Continuous Deployment

Presenter: David Cramer (@zeeg)

Track: I

Description:

Practice iterative development like the pros. Release sooner, faster, and more often.

https://us.pycon.org/2012/schedule/presentation/12/

Definition

Shipping new code as soon as it's ready

1.10.1 When is it ready?

- Review by peers
- **Passes automated tests**
  - none of this works without testing
- Some level of QA

Focus on stability and integration

Workflow

Commit > review > integration (repeat if fails) > deploy > reporting (if bad, rollback)

1.10.2 The Good

- Develop features incrementally
- release frequently
- smallers does of QA

1.10.3 The Bad

- **Culture Shock**
  - can be very hard to implement
  - this is a mindset
- Stability depends on test coverage
- Initial time investment

Keep your development simple

1.10.4 Development

- Automate testing of complicated processes and architecture
- **Simple can be better than complete**
  - especially locally
- **Packaging your app is a must**
  - puppet, chef, buildout, fabric, etc.

Different environments can be setup a bit differently.

Bootstrapping local

- **simplify local setup**
  - git clone, then run a command to take care of everything
  - then spin up the services (django, flask, etc)
- **Need to test dependancies?**
  - virtualbox + vagrant up

    - makes it easy for the developer to make a production like environment locally

Progressive Rollout

Gargoyle https://www.github.com/disqus/gargoyle

Early adopters are free QA - use them.

All commits must go through code review. Disqus uses phabricator. phabricator.org

Integration Requirements

- Developers must know they broke something
- **Support proper reporting**
    - jenkins does a ton of this stuff for you
- Painless setup

Shortcomings

- False positives
- Test coverage
- Feedback delay

Speeding up tests

- Write true unit tests
- Mock external services (lots of love for Mock today)
- Distributed and parallel testing

Meaningful Metrics

- **Rate of traffic (not just hits)**
    - can find bugs this way
    - Sentry, graphite
- Response time (database, web)
- Twitter activity can let you know if your site is down

Getting Started

- Package your app
- Value code review
- Ease deployment; fast rollbacks
- Setup automated tests
- Gather some easy metrics

Going further

- Build an immune system
- **Adjust to your culture**
    - there is no "right way"
- SOA == great success

Disqus deploys atleast once a day but usually around 6 times a day.

Make sure schema changes can be null. First commit make the change, second backfill data and third commit code to use schema change.

CHAPTER 2

Saturday

2.1 Lightning Talks

Missed these. Was at the pycon 5k.

2.2 Keynote: David Beazley

Let's talk about something diabolical. The best part of PyCon is learning about new stuff.

We're going to talk about PyPy. PyPy is python implemented in Python. Faster than normal python by some types of magic.

- Just in time compilation
- Translation to C

2.2.1 Thinking about Tinkering

Tinkering is something that is very important. Using a VW bug has an example of different types of Python.

- Started with Python because they could hack on it
- Easy to work with and change

Is PyPy something you can tinker with?

A Confession

- PyPy scares him. Because of complexity.
- **Started an experiment to answer that question.**
  - Use nothing but the source

Tinkering with PyPy != Using PyPy

- Start by downloading the source. pypy.org. build it yourself
- type python py.py in the bin directory of source to run it
- **To get the "real" version you have to translate it.**
  - Go to /translator/goal/ - run translate with some options
  - Builds pypy from source

Building PyPy

- Takes a few hours to build
- It takes > 4GB of ram to build
- Translation of PyPy to C ~10.5 million lines of C
- It might break your C compiler

This is Amazing

- **Implemented in a subset of python called RPython**
  - it is a restricted subset of Python
  - "RPython is everything that our translation toolchain can accept"
  - Documentation are on readthedocs.org
- 454 directories in source
- ~1.2 million lines

Running a live example of RPython

- Had a def target to set the entry point in the code
- **Must translate the script to run it. Using translate.py from PyPy**
  - creates a C file
- **Trying a fibonachi sequence as an example**
  - takes about 8 seconds to run in Python
  - Now runs the file through the translate.py
  - .17 seconds to run after translation to C
  - a raw C example is slower than the rpython (pypy) version

Limitations of RPython

- Do not get the dynamic features of Python
- Must have consistent types
- Can call external functions in C
- **Uses Type inference**
  - Looks at your code and applies types across it
  - This is what's happening in the build process

Understnading Translation

- Will blow your mind
- **Insight: Python already parsed the code.**
  - doesn't even operate off source code
  - operates entirely of of code objects
  - translater code looks at the bytecode
- CPython has a bytecode interpreter
- PyPy does this as well, the bytecode interpreter is written in Python

- Bytecode interpreter is modular
- PyPy compiles itself using itself
- Runs the bytecode of the object "in the abstract"
- The full details are "hairy"

Problems

- As an outsider coming to PyPy. Everything is Python, makes it difficult to sort his head around it.
- Two different languages co-exist (Python, RPython)

What if PHP has the same syntax as HTML, that's PyPy

Uses docstrings to tell the difference between RPython and Python in the source

Looking at some RPython

- opens ll_thread.py
- specifies C header files, calling out to C
- Lots of decorator usage

A realization, he still doesn't know how it works

PyPy: 1, Dave: 0

Doesn't know CPython either, but he does know how to use the tools that make up CPython

PyPy has a different set of tools. Need to learn about the tools before you can understand how PyPy works. It's a different vocabulary.

Figuring out how PyPY works is left as an exercise of us

- He loves breaking GILS
- Showing an example of Ruby, Ruby is 3600x slower than Python

Still not sure if you can tinker with PyPy but you should try.

2.3 Why PyPy by example

Presenters: Maciej Fijalkowski , Alex Gaynor , Armin Rigo

Track: II

Description:

> One of the goals of PyPy is to make existing Python code faster, however an even broader goal was to make it possible to write things in Python that previous would needed to be written in C or other low-level language. This talk will show examples of this, and describe how they represent the tremendous progress PyPy has made, and what it means for people looking to use PyPy.
>
> https://us.pycon.org/2012/schedule/presentation/244/

We've failed to convince you all that we aren't crazy

PyPy project has been around for 9 years. Used to be slower than CPython, now is faster.

If it's not faster, it's a bug. How much is dependant on your program

PyPy is fast, but is not a silver bullet. Measure and complain so that PyPy can get better.

Going to show some programs written in PyPy

Showing an example of edge detection with a webcam in PyPy. Then shows the example in Python. Took over 10 seconds to get a single frame.

Numerics in PyPy is pretty fast

Next example is Tracebin

Tools for understanding what our programs are doing is very important. Tracebin is a JIT viewer tool.

- Records every instruction the JIT compiled.
- For each line of python code
- Being able to expose this type of information to developers will be super powerful

numpy

They are working on implementing numpy. Stay tuned, will be a few months until it's complete.

transactional memory

This was given 30 years ago at PascalCon

"solution": put everything in the GC-managed memory.

the theoretical answer: horrible for performance (memory and speed)

real programers can deal with explicit garbage collection

Now, 30 years later we don't even talk about garbage collection instead we talk about how to use multiple cores.

mess with locks, reentrant locks, semaphorse, events...

in Python we have the GIL, but also the threading module

transactional memory promised to give multi-core usage

using pypy has as reasearch on software transactional memory

2.4 Django Form Processing Deep Dive

Presenter: Nathan Yergler (@nyergler)

Track: V

Description:

> Django Form processing often takes a back seat to flashier, more visible parts of the framework. But Django forms, fully leveraged, can help developers be more productive and write more cohesive code. This talk will dive deep into the stock Django forms package, as well as discuss a strategy for abstracting validation for forms, and the use of unit and integration tests with forms.
>
> https://us.pycon.org/2012/schedule/presentation/420/

Eventbrite uses django forms quite extensively.

The problem with a deep dive is that sometimes you hit your head at the bottom

2.4.1 Form Basics

- Wasn't obvious at first where they fit in the stack of django
- **A form converts inmport into Python objects**

    - Can be used even when you don't have HTML involved at all
- Forms are composed of fields, which have a widget
- Can instantiate a form with or without data
- data does not have to be request.POST, can be json, etc.
- **Two ways to access a field**
    - fields dictionary form.fields['name'] returns Field object
    - form['name'] returns a BoundField
- Can give forms initial data
- form['name'].value gives you the current value

2.4.2 Validation

- Only bound forms can be validated
- calling form.is_valid() starts validation
- Three phases for fields: To Python, Validation and Cleaning
- If validation raises an error, cleaning is skipped
- Django provides some basic validation rules (URL, email, max / min length, etc.)
- **To clean a specific field clean_fieldname()**
    - Must returned the cleaned_data
- **.clean() performs cross-field validation**
    - Must return cleaned_data dictionary
- inital != default data
- **Track changes**
    - form.has_changed()
    - formchanged_fields gives you any fields that have changed
    - show_hidden_initial=True on a form field. will render a hidden input with the initial value

2.4.3 Testing

- **Testing strategies**
    - initial states
    - field validation
    - final state of cleaned_data
- **Unit Tests**
    - **Just open sourced a library called rebar**
        * flattens form data into a dict

2.4.4 Rendering Forms

- Class based views are nice for using forms
- **Three output modes for forms**
  - .as_p(), .as_ul(), .as_table()
- Can iterate over the form if you want to control form output
- **Customize rendering**
  - can override widgets when defining a field
  - can give the field attrs like {'class': 'custom'}
  - can also specify form-wide CSS
- **Build in validators have default error messages**
  - error_messages on the field let's you customize these messages
- **ValidationErrors are wrapped in a class**
  - by default ErrorList outputs as a UL
  - **specify the error_class kwarg when consructing form to override**
    - extend ErrorList from django.forms.util
- **Multiple forms**
  - can add the arg `prefix` when instantiating form to give different prefixes to each form on page

Form Sets give you a way to use multiple forms of the same type on the page. They have the same validation structure. Overriding clean works the same way on formsets.

localize=True can localize a form field

http://yergler.net/2012/pycon-forms

2.5 Testing and Django

Presenter: Carl Meyer (@carljm)

Track: V

Description:

> A deep dive into writing tests with Django, covering Django's custom test-suite-runner and the testing utilities in Django, what all they actually do, how you should and shouldn't use them (and some you shouldn't use at all!). Also, guidelines for writing good tests (with or without Django), and my least favorite things about testing in Django (and how I'd like to fix them).
>
> https://us.pycon.org/2012/schedule/presentation/412/

Examples will be using the lastest django 1.4 rc

Running the tests for django 1.4 takes 14 seconds, get's two errors. Not very good. The tests fail because the tests assume two databases.

Not all apps are created equal

It's a waste of time to run tests for apps you don't really care about.

- Django insists that all your tests exists in a tests module

- **Django's test discovery**
  - Wastes times with apps I don't care about
  - Forces intermingling of tests and non-test code
  - **It's easy to change**
    - unittest2, nose
    - TEST_RUNNER settings

Types of test

- **Unit tests**
  - test one unit of code. fast, focused
  - helps you structure your code better
- **Integration test**
  - tests that the whole integrated system works
  - tend to be slow
  - less useful failures
  - write fewer of these
- **The database makes your tests slow**
  - try to write tests that don't hit it at all
  - separate db-independent model-layer functionality from db-dependant functionality
  - mocking the database usually isn't worth it
  - don't run tests on a method that does a self.save(). remove the code you want to test into a different method and test that method

TransactionTestCase is slow.

Don't use fixtures to test with, just say no

- fixtures are hard to maintain and update
- fixtures increase test interdependence
- fixtures are slow to load

Alternatives to fixtures are Model Factories

- Using a factory you can give just the data you care about
- Use a factory that can create a saved object or not

Use factory_boy as an alternative

- FactoryObject.create (saves) FactoryObject.build (does not save)
- Test data is lcoal to test code (explicit) using factories
- easy to maintain
- dont' create any data you don't need for that test
- works great even on large/complex test data sets

Imposing no-DB discipline

Use Mock

Testing Views

- Unit testing vies is hard
- Views have many collaborators / dependancies
- Write less view code is the solution
- Use RequestFactory
- **Call the view callable directly**
  - unit tests do not use the test client
- Set up dependancies manually (request.user, etc.)
- **or just don't test views**
  - I write less view code, and cover it via functional tests

Integration testing views

- use WebTest
- **The markup matters**
  - it can especially break forms
- django-webtest provides integration

In-browser testing

- using something like Selenium
- Is easier than you think
- inherit from LiveServerTestCase (takes care of running a server for you)

What type of tests to write

- Write system tests for your views
- Write Selenium tests for Ajax, other JS/server interactions
- Write unit tests for everything else
- Test each case (code branch) where it occurs
- One assert/action per test case method
- Should avoid multi-step tests

Testing Documentation

"We have always been at war with doctests"

- not entirely fair
- doctest are great
- you can turn code examples in Sphinx into tests. DocFileSuite

@override_settings - check this out in 1.4

2.6 Web Server Bottlenecks And Performance Tuning

Presenter: Graham Dumpleton

Track: V

Description:

> New Python web developers seem to love running benchmarks on WSGI servers. Reality is that they often have no idea what they are doing or what to look at. This talk will look at a range of factors which can influence the performance of your Python web application. This includes the impact of using threads vs processes, number of processors, memory available, the GIL and slow HTTP clients.
>
> https://us.pycon.org/2012/schedule/presentation/275/

2.7 RESTful APIs With Tastypie

Presenter: Daniel Lindsley

Track: V

Description:

> Providing full-featured REST APIs is an increasingly popular request. Tastypie allows you to easily implement a customizable REST API for your Python or Django applications.
>
> https://us.pycon.org/2012/schedule/presentation/61/

How I learned to stop worrying and love JSON.

A REST framework for Django. Provides a web based API off of django.

- Designed for extension
- Supports both Model and non-Model data

2.7.1 Philosophy

- **Make good use of HTTP**
  - Was written with servers in mind
- Tries to be "of the internet" & use the REST methods/status codes properly
- Graceful degradation - your API should be backwards compatible
- **Flexible serialization - not everybody wants JSON**
  - Actually flexible everything. Customizability is a core feature
- **Data can round-trip**
  - Anything you can GET, you should be able to POST/PUT
- **Reasonable defaults**
  - But easy to extend
- URIs everywhere!

HATEOAS - Hypermedia as teh engine of teh application state

- the user shouldn't have to know anything in advance

- All about explore-ability
- deep linking

2.7.2 What about Tastypie

- Build ontop of Django - is a third party app and should play nicely
- GET/POST/PUT/DELETE/PATCH
- Any datasource (not just models)
- designed to be extended
- **Supports a wide variety of serialization formats**
  - json, xml, yaml, bplist
- Well tested (80% coverage) and documented

The setup

```
pip install django-tastypie
```

- once installed just add to isntalled apps and syncdb

2.7.3 Auth API

- code goes in our apps, not Django
- Don't fork Django!!
- make an api directory, make a resources.py in it with __init__.py
- Set up a tastypie Resource for User
- Next, set up URLConf importing the Resource urls
- **Pull up the url in a browser, you're done**
  - you get lists, specific users, the user schema "/schema" and get multiple users with "/multiple"
- needs lxml, pyaml for the other formats
- pagination by default
- everyone has full read-only GET access
- To exclude fields use the exclude attr on your Resource to pass a list of excluded fields
- To add authentication, use the authentication attr
- **There is a filtering meta option as well**
  - filter useing a querystring

Authorization

- Not who you are, but can you do that
- tastypie.authorization - in class Meta add in the authorization attr
- tastypie.cache import SimpleCache - using the attr cache in class Meta for the Resource
- throttling works the same as well

2.7.4 Extensibility

- goal of the project was the give API developers lots of tools
- classes make extending behavior trivial
- composition > inheritance is the reason for the many classes in tastypie
- hooks, hooks, hooks
- tries to use reasonable defaults
- serialization, authorization, authentication, pagination, caching
- Resource has many methods, override or extend to your needs
- Can specify what formats are available by using the serializer attr in Meta

HTML Serialization

- can write a custom TemplateSerializer to output in HTML, override to_html()
- to read the data back override from_html()
- hook it up by using serializer attr in class Meta on Resource

2.7.5 Fields

- maybe you don't want to show your database schema, use fields for this
- use ModelResource and can define fields just like a ModelForm
- class Meta needs a queryset attr. Use exclude just like a ModelForm as well
- you can control how data gets prepared (dehydrate) or accepted from the user (hydrate)
- Can provide methods on your resource for non-simple things
- dehydrate and hydrate work just like clean methods on ModelForms dehydrate_field_name - hydrate_field_name
- ModelResource uses introspection to find the fields for you

2.7.6 Caching

- caching is very simple. you should be using varnish

The talk got cut short, but was awesome. Would have like to have seen the rest of it.

2.8 Using fabric to standardize the development process

Presenter: Ricardo Kirkner

Track: IV

Description:

> By ensuring consistency and repeatability in setting up the development environments of a team of developers, errors can be avoided (by automating repetitive tasks). It also helps by lowering the entry barrier for new developers, and letting existing developers focus on development tasks without having to worry about infrastructure or process issues.
>
> https://us.pycon.org/2012/schedule/presentation/25/

Currently employeed at Cononical

2.8.1 What is Fabric?

- fabric is a python library for streamling the use of SSH for application deployment and system tasks
- **use a run command from fabric.api to run commands on remote servers**
    - use fab from the command line and run method written in fabfile.py

2.8.2 Why Fabric?

- choose fabric for 1) lazyness - already in Python a language he knows
- and 2) legacy - was being used already in legacy applications

2.8.3 How to use Fabric

- you can do many things with fabric. anything you can do on the command line you can do in fabric
- **Bootstrap**
    - do sanity checks
    - setup virtualenv
    - install dependancies
    - setup configuration
    - will help other developers to set up the project locally and on production systems
- local() runs local and run() is on a remote machine - set by your host
- **Database**
    - setup database
    - start/stop database
    - create/drop database
    - sync database
    - migrate database

***Continous Integration**

- fabric can be used during CI.
- works with Jenkins

- **Deployment**
    - fabric is typically used for deployment tasks as well.

2.9 Building a Robot that Can Play Angry Birds on a Smartphone, (or Robots are the Future of Testing)

Presenter: Jason Huggins

Track: V

Description:

> Can your robot play Angry Birds? On an iPhone? Mine can. I call it "BitbeamBot". It started as an art project, but it has a much more serious practical application: mobile web testing. To trust that your mobile app truly works, you need an end-to-end test on the actual device. BitbeamBot is an Arduino-powered open-source hardware CNC robot that can test any application on any mobile device.
>
> https://us.pycon.org/2012/schedule/presentation/470/

Gave a demo of the robot playing angry birds (big apploause)

The name of the project is bitbeam (bitbeam.org)

The original idea was to build a robot fish, after the robot dog from sony in 1998

- fish swim in a sine wave
- while researching sine waves he became fascinated with 3d simulations
- **liked the motorized pin art and had the idea of building 3d simulations using them**
  - to buy the parts was very expensive, so he designed his own linear actuator
  - realized the linear actuator could be used for clicking buttons / testing
- currently works at Selenium and is hoping to make bitbeambot selenium powered

Why?

- selenium is a software-based robot
- it's mission is to mimic and automate user actions
- for mobile you this means you have to integrate with the device
- this is an experiment to take Selenium out of the screen and into the real world
- **using opencv is find the 'bird' in angry birds**
  - eventually wants to use this to have the bitbeambot be more robust in playing the game and in future uses
- shows an example of the bot playing tic-tac-toe

This talk was full of demos, check out the video when it's posted.

2.10 Designing Embedded Systems with Linux and Python

Presenter: Mark Kohler

Track: I

Description:

The continual decrease in the cost of computer hardware is allowing more embedded systems to be built with Linux and Python, instead of the traditional approach of a real-time operating system and C. This talk reviews the differences between those approaches and describes problems, solutions, and tools that can be used when building embedded systems with Python.

https://us.pycon.org/2012/schedule/presentation/373/

CHAPTER 3

Sunday

3.1 Lightning Talks

- pickle
- **ggplot**
  - charting tool in python
  - using a library called bokeh with panda
  - @pwang
- **Mike Muhler - PyCon DE**
  - german speaking python conference
  - 200 people at their first conference
  - This year will be in October
  - EuroSciPy 2012 in August
- **Carl Meyer**
  - ls -F in virtualenv
  - virtualenv “I can’t beleive this thing even works at all”
  - the new virtualenv (pyvenv)
  - pyvenv has a config file pyenv.cfg
  - PEP 405
- **Ian Oswald**
  - works with high performance computing
  - showing mandaldroit (sp) demos
  - pypy on original code gets a 10x increase
  - shedskin - 50x speedup
  - numpy gets behind the GIL - crazy fast with cython
- **Dan Frank**
  - asyncdynamo

- **Nathan Yergler**
  - hieroglyph - and extension for sphinx to write html5 slides with rst and sphinx
  - add an an extension to sphinx
  - this is cool, I'll be using this
  - https://github.com/nyergler/hieroglyph

3.2 Keynote: Guido Van Rossum

- Thank you PSF!
- There are alot of Trolls in the real world
- **What is a Troll?**
  - a question that isn't meant as a question, but to heckle
- **What have trolls said about python?**
  - used to be most common "you gotta be kidding about the whitespace"
  - trolls have evolved now
  - **"Python Sucks. Ruby Rules"**
    - apples and oranges
    - this is mostly all bullshit
    - python, ruby, perl are mostly the same language at 10k feet
    - don't get pulled into language comparissons
    - be proud of your language, but don't diss other languages
  - don't worry, be happy - there are tons of people here
  - **"When will you admit Python 3 is a mistake?"**
    - unicode literals are back in Python 3
    - Python 3 is doing fine
    - numpy is being ported to Python 3
  - **"Since PyPy is so much faster than CPython, why not abandon CPython"**
    - asks people to raise their hands who use PyPy in production - almost nobody
    - now CPython - everybody
    - guido sees many possibilities with PyPy
  - **Dynamic Typing**
    - "I don't want my app to fail witn an AttributeError after running for 4 hours"
    - type checking in compilers is actually very weak
    - **don't trust your compiler to find all your errors for you**
      - if you do, you haven't done software engineering that long
    - the only way to fix this is to test, think about it, audit it, hire smart people. especially smart people who know they can still get smarter. work with a team with different capabilities.

    - dynamic typing is incredibly important and useful in many ways
    - dynamic languages have become so important that the static community is designing new languages with dynamic features

- **Speed**
    - "Python is to slow for real work"
    - "Why don't you write a compiler?"
    - Anytime he writes something in Python, it's fast enough
    - In languages that are fast, the development is too slow
- **The GIL**
    - "Python is useless because of the GIL"
    - "You can't use iton a multi-core computer"
- **Async I/O and Concurrency**
    - Likes gevent, but is hesitant to add it to the stdlib
    - guido is not a callback fan
- **The Browser**
    - why don't you put Python in the browser
    - because nobody would want to use it
    - introducting a new language to replace javascript is a huge political problem
- **Mobile**
    - "You work for Google. Why didn't you make Python the language of choice for Android"
    - There are enough tools out there that people can use to make this happen
- **Functional Programming**
    - "Why did you kill reduce()"
    - "Please fix lambda"
    - "Please add more functional features"
    - **Python is not a functional language for a very specific reason**
        - it's very pragmatic
    - What you can do is learn a functional language, get excited about the functional features. And when you are writing python and when you see an oppurtunity to write in a functional style, use it.
    - Guido likes functional programming as a challenge and a fresh idea
- **The standard Library**
    - "Why hasn't <my favorite package> been added to teh standard library yet?"
    - You're better off. Once it's in stdlib you're stuck with the API you have because of backwards compatability. You're stuck on a release cycle
- **Garbage Collection**
    - Python has gotten a long way with reference counting

        * Python does have garbage collection
    - **Language Evolution**
        * "Stop changing the language already!"
        * will never be solved for everybody
        * as your userbase grows you become more conservative with changes

3.3 Poster Sessions

3.4 Sketching a Better Product

Presenter: Idan Gazit

Track: I

Description:

> If writing is a means for organizing your thoughts, then sketching is a means for organizing your thoughts visually. Just as good writing requires drafts, good design requires sketches: low-investment, low-resolution braindumps. Learn how to use ugly sketching to iterate your way to a better product.
>
> https://us.pycon.org/2012/schedule/presentation/301/

Django's Benevolent Designer for Life

- Sketching is not drawing
- **sketching is a tool that serves a purpose**
    - to explore visual idea
- drawing is art, it serves itself
- Doesn't matter if you can't draw, sketch
- Sketching is for ideas as drafts are to writing
- **sketching is the act of getting out of your head and brin git back through your eyeballs**
    - this uses a whole different part of your brain
- **sketches should be cheap**
    - try stuff, throw bad stuff away
    - try different ideas
- **sketches shoudl be fast**
    - quick to create
    - low resolution
- **sketches should be ugly**
    - communicate unfinished nature - make sure to communicate the unfinishedness
    - people might think it's complete if not
- **The design funnel**
    - sketching user experiences - a book to check out

    - go down the path of ideation to usability
    - sketches -> prototype
    - it's a gradual process
- **resolution vs. time**
    - spend time after you figure out the good idea
    - don't waste time with polishing bad ideas, sketch them
- **what do you need?**
    - paper, pen and a wall
    - you need a wall to show off sketches to the rest of the team
- **sketching, the 37 signals way**
    - two fat markers - black and red
    - epicenter design - sketch the parts of the heart of the page
    - sketch the content, that's the heart of the page
- **how to sketch**
    - it's easy.
    - you need to be able to draw lines
    - you need to be able to draw a box
    - that's all you need, really
    - **don't draw text**
        * use boxes if it's really big, if smaller use lines
    - greeked text - two lines filled in with squiggles
    - **images**
        * represent with boxes with an X
    - **iconic elements**
        * calendars can be grids
- **don't worry about it looking bad, just do it**
    - and make it quick
    - a great way to think visually
- **stencils**
    - can be used to draw various icons
    - use when you need to refine sketches
- can use grid systems in the sketch
- **rulers are awesome, use them**
    - but later in the process
- uses an app called penultimate on ipad - and cosmonaut

3.5 Building A Python-Based Search Engine

Presenter: Daniel Lindsley

Track: IV

Description:

> Search is an increasingly common request in all types of applications as the amount of data all of us deal with continues to grow. The technology/architecture behind search engines is wildly different from what many developers expect. This talk will give a solid grounding in the fundamentals of providing search using Python to flesh out these concepts in a simple library.
>
> https://us.pycon.org/2012/schedule/presentation/66/

Primary author of Haystack

The Goal

- teach you how search works
- increase your comfort with other engines

Why should you care about search?

- **Standard crawlers have to scrape HTML**
  - getting good content out of that can be a nightmare
- You know the data model better than they do
- Maybe it's not a web app at all

Core Concepts

- **Document-based**
  - never grep through a string
- Inverted Index

Terms

Engine - the black box you hand a query to get results back

Documents - a text blob with optional metadata

Corpus - the collection of all the documents

Stopword - words that are filler (and, the, of)

Steming - finding the root of the word

Segments - the data that make up the overall index

Relevance - the algorithm used to rank the results

Faceting - providing results matching a certain criteria

Boost - a way to push up certain type of results to the top

Indexing

- **Documents**
  - The are **not** a row in the db
  - think blob of text + metadata

  - text quality is the most important thing
  - flat, not relational
  - denormalize, denormalize, denormalize!!!!
- **Tokenization**
  - the process of taking the text blob and making it smaller word sized chunks
  - split on whitespace, lowercase, strip stopwords
  - the point is to normalize the tokens to work with when querying
- **Stemming**
  - the process of finding the root word
  - find root works because of spelling mistakes or pluralization
  - these become the terms in the inverted index
  - Cons: really only works on the language it was built for
  - very hard to make work cross language
  - how do we solve this? - n-grams
- **n-grams**
  - passes a "window" over the tokenized data
  - these become terms in the index
  - **example gram size of 3 on hello**
    - ['hel', 'ell', 'llo', 'wor', 'ld']
  - **edge n-grams - typically uses multiple gram sizes**
    - this works great for autocomplete
    - can also work across other languages
    - cons: generate a lot more terms and storage can be a problem
    - initially quality of search can suffer a little
- **inverted index**
  - the heart of the engine - it's how things work
  - like a dictionary - keys matter (terms from all docs)
  - stores position & document IDs
- **Segments**
  - with a huge data structure it really can't be in one file
  - lots of ways to do this
  - many follow Lucene
  - flat files and then hash the keys
  - terms will always be sorted
- **Searching**
  - Three main components

    - **Query Parser**
        * take a users hand written query and parse it out to something the engine can tackle
        * do this the same way as you do with the index
    - **Index reading**
        * per-term, hash the term to get the right file
        * rip through & collect position and document collection
    - **Scoring**
        * we have terms now, but the are out of order
        * **lots of choices**
            · bm25, phased, google's pagerank

- **Faceting**
    - for a given field, collect all terms
    - count the length of the unique document ids for each
    - order by descending count
- **Boost**
    - during the scoring process
    - If a condition is met, alter the score according
- **More like this**
    - collect all the terms for a given document
    - can find other like terms in other documents
    - sotr based on how many times a document is seen in the set
    - more complete solutions use NLP to increase quality

https://github.com/toastdriven/microsearch http://speakerdeck.com/daniellindsley

3.6 Diversity in practice: How the Boston Python User Group grew to 1700 people and over 15% women

Presenter: Jessica McKellar , Asheesh Laroia

Track: III

Description:

> How do you bring more women into programming communities with long-term, measurable results? In this talk we'll analyze our successful effort, the Boston Python Workshop, which brought over 200 women into Boston's Python community this year. We'll talk about lessons learned running the workshop, the dramatic effect it has had on the local user group, and how to run a workshop in your city.
>
> https://us.pycon.org/2012/schedule/presentation/168/O

Diverse membership makes user groups better

It's more dynamic, you can run different event styles. And you can learn from a larger set of skills

Diversity outreach can help user groups grow.

The Boston Python Workshop

An intro to python event mainly focused on women. The are maintainers of the OpenHatch project.

Motivation

- A simple observation that there were almost no women in the group
- no pipeline for newcomers / beginners
- instead of creating a new group, change the existing group from within for more diversity

Goals

- Get more women in the community - target was 15% women
- Use the workshops has a platform to showcase great women programmers
- Encourage other user groups to think about diversity

History so far

- been doing these for the past year
- almost 200 women have went through and became part of the group
- created a large volunteer base and beginner's pipeline
- a 'project night' is the next follow up meeting to a workshop

Workshop structure

- it's a 2 day workshop
- the first day you get the computer setup. the next day is more hands on in learning Python.
- **Friday is the setup day**
    - they have install instructions on how to install python on multiple operating systems
    - get a basic text editor setup
    - get the basics of the interpreter
    - all the information is on OpenHatch online
- **Saturday is a lecture**
    - learn about the basic information they need to do the practices
    - after the lecture the do a serious of hands-on projects
    - **a choice of 3 different projects**
        * color wall, twitter api and a wordplay exercise
    - **projects are part lecture and part hands-on**
        * first half is the basics
        * second half is giving them excercises to work through
    - **wrap-up**
        * how to learn python on their own
        * fill out a survey
- **this has had a huge impact on the user group**

  - 3 organizers now
  - 1800 members
  - lightning talks
  - hack nights
  - lots of social engagments
  - bi-montly workshops and monthly project nights
- **Relfection & Sharing**
  - **Why'd you sign up (from the surveys)**
    - women focused, judgment-free, free to attend
  - "Feeling welcome in what felt like a closed world to me" - feedback after workshop
- **Now there is more coding practice and more simplified projects**
  - this needs to be beginner friendly
- **How we share?**
  - curriculum on wiki - everything is opensource
  - events mailing list
  - OpenHatch blog
- **Scaling up**
  - from 25 person events to 80 person events
  - be good at content refinement and reuse
  - uses codingbat.com at the workshops
- **at the project-nights there is always a beginners corner**
  - using the same material from the workshops
- **Scaling out: impact beyond boston**
  - other groups are using the content
  - pyladies
  - PyStar philly + PhillyPUG
- has had a big impact on pycon 2012
- **Next steps**
  - sustain things in boston
  - go from local impact to global impact
- **What's your next step**
  - **run a project night**
    - easy to organize, beginners' corner, set the tone
  - **run an intro event**
    - pick a goal and stick to it
    - utilize the existing user group memebers

        * reuse material!
    - iterate

openhatch.org/wiki/

bostonpythonworkshops.org

@jessicamckellar

@asheeshlaroia

3.7 Afternoon Lightning Talks

Missed these, had to catch the plane.