PyCon 2012 Recap

Wesley Chun
@wescpy/+wescpy
2012 Mar 22
SF Bay Area/Silicon Valley
Python users group (BayPIGgies)

Frighteningly Ambitious Startup Ideas

Paul Graham
PyCon 2012 keynote
1000, F Mar 9
http://pyvideo.org/video/626
http://paulgraham.com/ambitious.html

Introduction

- Center of silicon valley moves around. It is where the...
- Greatest concentration of programmers building next big thing
  - Right now... it's here (points at crowd)
  - Hope you're not disappointed (laughter)

Biggest startup ideas...

- Threaten your life as it exists
- Repel you... need to attack into that wind
- Note: starting "the next Google" is just on right side of impossible

1. Making a new search engine to compete w/Google

- Harken back to early days of Google
- A Hacker's Tool: simple, fast, done

2. Replace Email

- Not a messaging protocol, but a to-do list
  - …and a bad one

3. Replace Universities

- Heading down wrong path
- Not fun for students nor profs
- Expensive country clubs where you don't learn much
- Replaced w/smaller versions where people "learn stuff"

4. Kill Hollywood/Internet Drama.

- Big mistake: too slow to embrace Internet
- Delivery of entertainment via Internet not cable
- Larger scale than YouTube clip
- Netflix, iTunes, etc. will be enterainment store
- Will need infrastructure companies

5. A new Apple/The next Steve Jobs

- Someone said Apple won't exist (well) after current pipeline
- Next revolutionary product? (not from existing players)
  - None run by product visionaries
- Next Steve Jobs will have to start a company & not get fired
  - Don't need to be better than Jobs...
  - At least better than Samsung, HP, Motorola
  - Doesn't sound so hard now does it?

6. Bring back old Moore's Law

- Not computers twice as fast in 18 mos
- Circuit densities double in 18 mos
- Made lazy by old way (no need to optimize; wait for next gen CPUs)
- Can't give us faster CPUs... now only getting **more** CPUs/cores
- Need to make software run efficiently on multicores

7. Ongoing (medical) diagnosis

- We'll seem backwards to those in future
- Barbaric to wait for symptoms to diagnose heart disease, cancer, etc.
  - Clinton had to wait until breathing problem due to 90% heart blockage
- Shouldn't be that hard without destructive testing
  - Maybe v1 can be destructive
  - Try heart disease for pigs 1st, run sausage company on side
- Similar for cancer, shouldn't need to wait till have a lump
  - Need "radar" screen where it shows up more quickly
  - 10/100s of microcancers at any given time, which ones to watch
- Medical profession doesn't work this way
  - Symptoms -> doctor
  - "Fishing" for problems that aren't even there
  - Terrifying false alarms

Tactics

- Don't make frontal attack: "I will replace email"
  - VCs skeptical and ppl wait for you to fail
  - Just say you're doing a new "to-do" list
- Start small, go big.
  - Want biggest software company on Earth? Port DOS & write BASIC
  - Want biggest social network? Start w/undergrad project to stalk each other, rate babes
- Best way is to not try to predict future...
  - Be like Columbus: think there's something to the West
  - Plan, strategize, wait
  - When moment is right, set sail

Trolls/Questions & Answers

Guido van Rossum
PyCon 2012 BDFL keynote
0910, Su Mar 11
http://pyvideo.org/video/656

Keynote

- Some thought Guido's talk depressing
  - New to Python community
- I found it uplifting & encouraging, deep & meaningful
  - At least more positive than in the past
  - At least more interesting than Twitter stream
  - Felt 23 years of bullets on bulletproof vest
- About trolls and how to deal w/them mixed w/FAQ&A

Trolls

- Leading question towards an argument (or a person trolling)
- Ignore trolls
- Most common troll used to be
  - "You gotta be kidding about the whitespace"

Python sucks. Ruby rules.

- Apples and oranges
- Python, Ruby, Perl are the same; they should be our friends
- Java fundamentally different, traditions, philosophy, community
- C++ really different
- Lisp not all that different (looks different w/many similarities)
- Don't get pulled into language comparisons
  - Be proud of your lang but don't diss another language
- Room for all of them; not one kicking butt over others

When you admit Python 3 is a mistake?

- "When will we get a Python 2.8 release?" (see PEP 404)
- "Why don't we add all the features back to Python 3?"
- Python 3 is doing fine
  - Companies considering 3.x as logical step after 2.7
  - NumPy being ported to 3.x
  - More projects seeing advantages to have a 3.x port

Knew it was going to take long but now light at end of tunnel

- 5 yrs from now will conclude: we dealt with it, got over it
- That will happen, don't worry
- Better tools and techniques now available
- More from dependency graph will be available
  - "Waiting for pkg X to be available"
  - Game of chicken but someone is going to jump

Since PyPy so much faster, let's abandon CPython

- How many using PyPy in production?
  - How many using CPython in production? "no hands?"
  - Difficult to break into ecosystem
- Think PyPy should be tinkered with
- See many exciting possibilities
- See many years of living together

Dynamically-typed languages aren't safe

- If you trust your compiler to find all bugs in your code...
  - Haven't been doing software development very long
- Programmers make mistakes
  - Sometimes trivial, like variable misspelling
  - Sometimes deep, like...
    - Not understanding physics/math in a process/calculation
- Should test and test again
  - Think about it, test again
- Hire smart people, but not too smart nor too cocky
  - May think they don't need to test
  - Work with team with different capabilities
  - Work together; help each other

Dynamic languages aren't safe (cont.)

- Look at your code; think about it
  - How can you make it better?
  - Don't focus on a narrow thing like a typechecking compiler
- Dynamic typing very important & very useful in many ways
  - Not saying there's no room for static typing
  - Instead: dynamic typing not inferior to static typing
- New statically-typed langs seem to have "dynamic features"

Speed: Python too slow for real work

- Some seem to have some proof of that
- Some reject Python outright
- It's fast enough... absolutely fast enough
- If you do it right, it won't make a difference
- Extreme situation: Python replacing Java apps

Python slow; can't you just write a compiler for it?

- An interesting question... turns out it's not so easy
- Compiler must...
  - Maintain dynamic nature
  - Semantic correctness of code
  - Run faster

Global Interpreter Lock

- "Python is useless because of the GIL"
- "You can't use it on a multicore computer"
- Parents warn children using Python because of the GIL
- Makes me sad it's become out-of-propotion caracature
- Makes common case Python & C interaction faster, cleaner, safer
- Some people say we don't have real threads
  - Use them for what they're created for...
  - Parallel I/O not parallel computation
- If you want to run something on lots of CPUs, use a diff arch
  - Some ppl run Python on 64k cores
- Don't get so hung up on it
- Sure Dave Beazley came up w/counterexample
  - But you have to be Dave Beazley to do that example

Async I/O & Concurrency

- When will you integrate Stackless?
  - Changes implementation too much
  - Will make PyPy, IronPython, other impls more complicated or impossible
- When you add a proper event loop?
  - Depends what your job is; may or may not be useful
  - Not everything that everybody does requires an event loop
- Why don't you add...
  - Channels, msg-passing, actors, Erlang-style procs, X (fave abstraction)?
  - Not fan of callbacks; like other approaches
  - Would probly prefer libs built on top of gevent
    - Would let you write sync code that does things async
  - Many will/should be investigated to see if can be integrated
  - May wish to consider to target for Python 4

In the Browser

- "Please make Python work in the browser...
- It would solve all the world's problems."
- Introducing a new lang to replace JS... not likely
- Biggest browser vendors can't even do it

Mobile

- You work for Google. Why isn't Python lang of choice for Android?
  - If it were possible, I'd try it.
- At some point, did get involved with some other Googlers
  - Told to not work on it any more
- Nothing technologically from stopping anyone from doing this
- I'm sure it can be done...
  - But hope someone else can do it because I can't

Functional Programming

- Why did you kill reduce?
- Please fix lambda
- Please add more functional features
- Don't get hung up on doing everthing in a functional style
  - Or high-order programming
  - Or something else that emulates Haskell's utility functions
  - Then write lambdas to glue it altogether
- When moment is right for functional style, use it

Functional Programming (cont)

- Python not a func lang for good reason
  - Very pragmatic
  - Lets you fiddle with state
- Haskell is...
  - Incredibly beautiful language
  - Great when you follow examples
  - But when you try to write filesystem code or a web server...
  - Messy code becomes messier

Standard Library

- Why hasn't X been added to the stdlib yet?
  - Hesistant to add anything to stdlib
  - Probably better off maintained by 3rd party
  - If you have a broken API, it will be broken forever
  - Stuck w/a release schedule
    - Have to wait 1.5-2 years for next Python version
    - Then convince your users to upgrade
  - On the other hand, if you put it in teh Cheeseshop
    - Can release whenever
    - Move at the speed around you
- Purpose of stdlib: to help support everything
- Joke: packages move to stdlib to die

Garbage Collection

- Python doesn't have garbage collection
- "Reference counting too slow"
- Yes there are better ways of doing this
- PyPy is where to experient w/better memory allocators
- Personally happy w/CPython's memory allocator
- Refcounting a legit algorithm for GC

Language evolution

- On my left: "Stop changing the language already"
- On my right: "Please add my favorite feature"
- Never going to resolve satisfactorily for everyone
- As user base grows, more conservative with changes

Advanced Python

Raymond Hettinger
PyCon 2012 Tutorials
0900, Th 2012 8 Mar
http://pyvideo.org/video/879 (The Art of Subclassing)
Tutorial videos not online

Example code

- http://tinyurl.com/py2adv
- http://tinyurl.com/py3adv

Use -m option. Why?

- Uses
  - python -m test.pystone
  - python -m pdb MODULE
  - python -m turtle
  - python -m test.regrtest
- Previously hard to run __main__ of stdlib modules:
  - python /usr/lib/python2.X/lib/timeit.py MODULE
  - vs. python -mtimeit MODULE

Use IDLE

- Open module (Alt-M) - pull up stdlib code
- Class browser (Alt-C) - reduce time to analyze large codebases

Optimizations

- Replace global lookups (builtins, modules, globals) w/locals
- Use (local variable) bound methods

```
bm = g.foo
bm()   # instead of g.foo()
```

- Minimize pure-python func calls inside loop
  - New stack frame on each call; recursion expensive

Vectorization

- Replace CPython's eval loop w/C function that does work
  - `[ord(c) for c in long_string]` ->

```
itertools.imap(ord, long_string) # 2.x
map(ord, long_string) # 3.x
```

  - `map()` allocates once vs. [ ]
- Use pow, itertools.count, itertools.repeat
  - `[i**2 for i in range(100)]` ->

```
list(map(pow, count(0), repeat(2, 100))
list(map(pow, xrange(100), repeat(2))
```

    - `count()` more flexible

Don't waste memory

```
def getvar(name, cmdline_args, env,
  def_vals):
   d = def_vals.copy()
   d.update(env)
   d.update(cmdline_args)
   return d[name]
```

- Instead, link multiple dicts

```
def getvar(name, *dicts):
   for d in dicts:
      if name in d:
         return d[name]
   else:
      raise KeyError('cannot find %r' % name)
```

- Look for chainmaps in 3.3

Use namedtuples

```
>>> x =
 collections.namedtuple('XXX', 'a b
 c d')
>>> y = x(1, 2, 3, 4)
>>> y
XXX(a=1, b=2, c=3, d=4)
>>> y.a, y.b, y.c, y.d
(1, 2, 3, 4)
```

Use sort keys powerfully

- Old way: Schwarzian transform

```
dec = [f(rec), rec) for rec in recs]
dec.sort()
res = [rec for key, rec in recs]
```

- With sort key
  - Sort stability and multiple passes

```
s.sort(key=attrgetter('lastname')) #
  2ndary
s.sort(key=attrgetter('age'),
  reverse=True) # primary
```

  - Still partially sorted by last name!!

Collections

- Deque
  - pop(0) or insert(0) problems? Use collections.deque
  - Fast O(1) appends and pops from both ends
- Defaultdict
  - Regular dict but gets a factory func for missing vals

```
d = defaultdict(list)
d[k].append(v) # new keys create new lists
```

- Counter
  - Dict that knows how to count

```
c[k] += 2 # zero value assumed for new keys
```

- OrderedDict
  - dict that remembers insertion order
  - OrderedDict()

Use __missing__()

```
class ZeroDict(dict):
  def __missing__(self, key):
    return 0
>>> z = ZeroDict()
>>> z['red'] += 1
>>> z['blue'] += 2
>>> z['red'] += 10
>>> z
{'blue': 2, 'red': 11}
```

Investigate __getattribute__()

- Check instance
- Check MRO/class tree
  - Check if descriptor -> invoke it
- Check if __getattr__ -> invoke it
- raise AttributeError()

Descriptors

- Tutorial http://tinyurl.com/d63d
- Object with "binding behavior"
- Object whose attribute access overriden by descriptor protocol methods
  - Those methods are __get__, __set__, __delete__
- If any of these definied for an object, it's a descriptor

How Python Works

- Type metaclass controls how classes created
  - Supplies them with __getattribute__()
- Dotted-attribute access like A.x or a.x calls __getattribute__()
- __getattribute__ does a dict lookup
  - Either returns result or invokes if a descriptor
- Everything else derived from these 3 precepts
  - Properties
  - super()
  - bound & unbound methods
  - classmethods
  - staticmethods
  - slots

Properties: descriptor examples

```
class Demo(object):
   def__init__(self, a, b):
       self.a = a
       self.b = b
   def add_parts(self):
       return self.a+self.b
   total = property(add_parts)
>>> d = Demo(10, 20)
>>> d.a, d.b
(10, 20)
>>> d.total
30
>>> vars(d)
{'a': x, 'b': y}
>>>>
```

FINIS