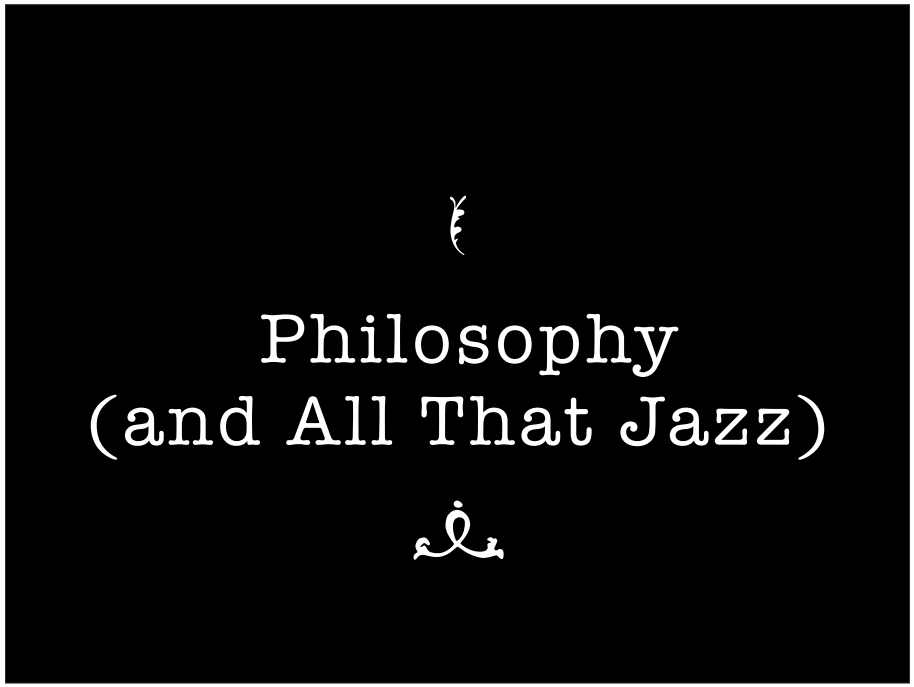


Still not started your thesis work?

- SPIDER - The Swedish Program for Information and Communication Technology in Developing Regions
- Drupal
- For more information contact Paula Uimonen paula@dsv.su.se

New Lecture

- Erlang by Patrik Nyblom
- March 13, 13-15(?)
- Bonus: get the exam earlier

Literature Seminars

- Lists are available on ...
- Sign up for one slot for each of the books you have read
- Seminars will be added if needed

Presentations

- All groups send me an email with the name of group members and the number of the assignment
- Please include information about times when you're unavailable
- I will schedule you for a presentation time slot on March 12-16 (times will be extended as needed)

The Background Material for Today

- Worse Is Better [14]
- Objects Have Failed [15]
- Asher's paper [2]
- Wilmott's paper [3]

Worse is Better

MIT/Stanford Design

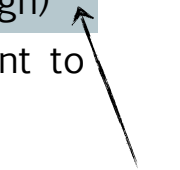
- *Simple* -- design of both implementation and interface (but interface before impl.)
- *Correct* -- incorrectness not allowed
- *Consistent* -- design slightly less simple if inconsistency can be avoided
- *Complete* -- all reasonably expected cases must be covered

  Ex.: CLOS and Scheme

Worse-is-better

- Simple -- implementation and design, (but implementation before design)
- Correct -- slightly more important to be simple than correct
- "Not too much" inconsistency
- "Reasonable" completeness

Ex.: (Early) C and UNIX



[...] worse-is-better [...] has better survival characteristics than the-right-thing [and] when used for software is a better approach than the MIT approach

—Richard P. Gabriel [14]

The Next Lisp

I think there will be a next Lisp. This Lisp must be carefully designed, using the principles for success we saw in worse-is-better

—Richard P. Gabriel [14]

Objects Have Failed

Objects, as envisioned by the designers of languages like Smalltalk [...] before C++ and Java [...] were for modelling and building complex, dynamic worlds

—Richard P. Gabriel [15]

[...] as a result we find that [OOPLs] have succumb to static thinkers who worship perfect planning over runtime adaptability, early decisions over late ones, and the wisdom of compilers over the cleverness of failure detection and repair

—Richard P. Gabriel [15]

Small languages like Tcl, Perl, and Python are gathering adherents, but are making no progress in language and system design at all

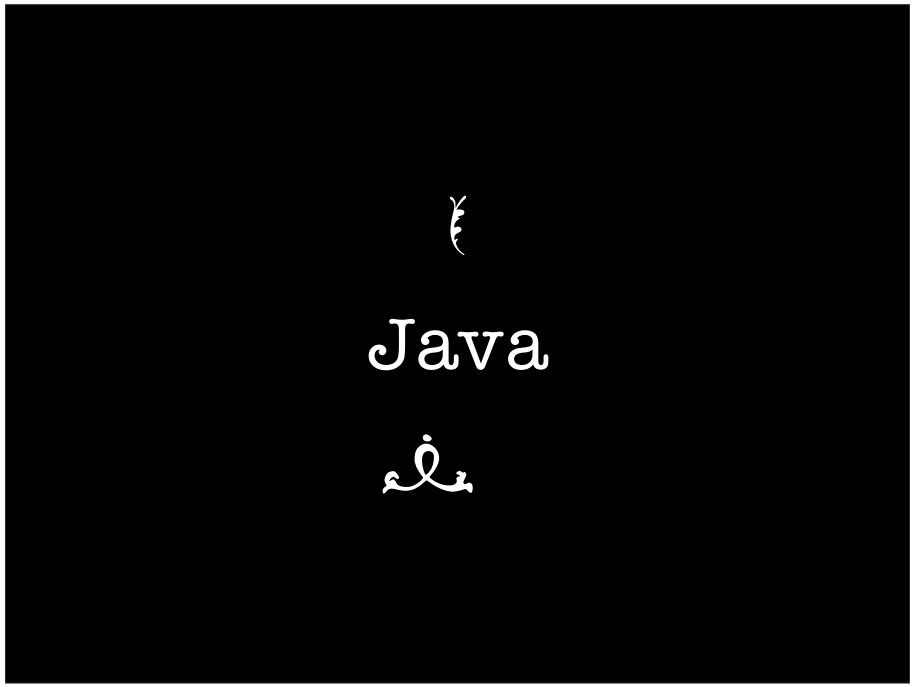
—Richard P. Gabriel [15]

David Asher [2]

- Open-source is like Darwin's law for programming languages
  - Technocracy and meritocracy vs. democracy
  - Development driven by actual programming—not business—needs

Sam Wilmott [3]

- "Dynamic languages work because they minimize the barriers placed in the way of the programmers"
- "There's something fundamentally wrong with a programming language that's continually developing."



I didn't really care so much about how long it took to get the first demo to run, but how long it took you to get to something you could actually release and call a solid piece of software

—James Gosling

Design Philosophy

- Be object-oriented and "familiar"
- Promote robustness
- Write once—run anywhere
- Good network support
- Execute code remotely
- Smalltalk, Simula and Objective-C

Community

- Collaborate on establishing a binary software standard
- Generate Java Specification Requests
- Republic—two executive committees

ME Community

- Aplix, AT&T, CableLabs, IBM, Nokia, Oracle, Orange France, Research In Motion, Samsung, Sean Sheedy, Siemens, SK Telecom, Sony Ericsson, T-Mobile, Vodafone ...

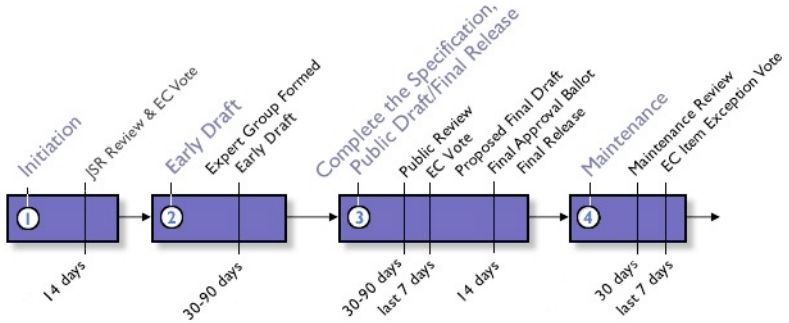
SE/EE Community

- Apache Software Foundation, Eclipse Foundation, Ericsson, Fujitsu Limited, Google, Hewlett-Packard, IBM, Inte, Werner Keil, Doug Lea, Oracle, Tim Peierls, Red Hat middleware, SAP, Vmware ...

Decision Making

- Members: ~350 companies, ~50 institutions and ~450 individuals
- Spec Lead and Expert Group may choose their own model
- Executive Committee uses formal voting to approve JSRs

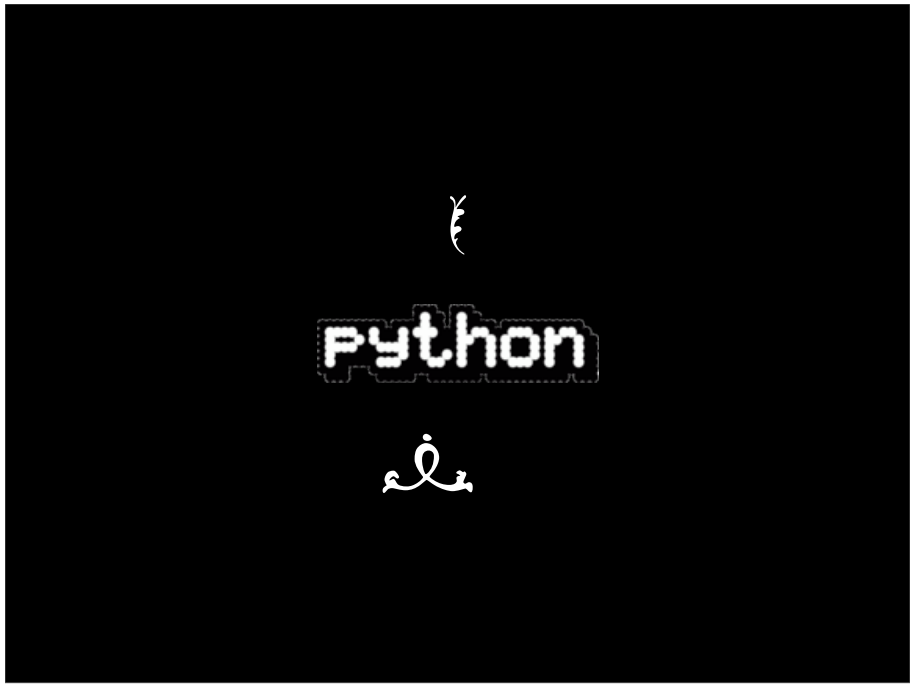
Development Process



- Must be a member to submit a JSR

Becoming a Member

- You must sign the Java Specification Participation Agreement
- $5.000/$2.000/$0 annual membership fee



Origins of Python

- ABC—programming for “intelligent computer users who were not computer programmers or software developers”
- Guido worked with ABC from 1983 to 1986/7

ABCs Users

- Scientists that needed to program as part of their research
- They were “surprised at certain limitations, restrictions, and arbitrary rules that programming languages had traditionally set out”

.

Amoeba

- Guido’s next project at CWI
- Distributed OS that needed a scripting language
- Guido rehashes what he liked of ABC into Python
- No need or desire to be backwards compatible

I think my most innovative contribution to Python’s success was making it easy to extend

—Guido van Rossum [19]

[M]y initial goal for Python was to serve as a second language for people who were C or C++ programmers, but who had work where writing a C program was just not effective

—Guido van Rossum [20]

I always feel that the interesting programming jobs are the ones in which you don't know exactly where you'll end up. Implementing another spreadsheet is boring.

—Guido van Rossum [20]

Python Software Foundation

- Nominated members -- candidates nominated and elected by existing members
- Sponsor members -- like nominated members, except that they must pay a yearly fee
- Emeritus members -- former nominated members who are no longer active

BDFL

- **B**enevolent **D**ictator **F**or **L**ife
- Guido has final say regarding Python's development
  - Is often delegated to the “master of the area in point”

Python Dev Process

- Informal votes using Apache voting scheme
  - +1 (pro), –1 (con), +0 (don’t care, but ok), –0 (don’t care so don’t)
  - The BDFL is free to ignore the result of the vote

Python Culture

- Python tries to keep things simple, to be orthogonal but not too much so, and to assist the programmer as much as possible
- Be friendly, especially to people that disagree with you

Python Culture, cont'd

- Have a sense of humour, and don't take things too seriously
- Know when to give up

Design Principles

- Beautiful is better than ugly
- Explicit is better than implicit
- Simple is better than complex
- Complex is better than complicated
- Flat is better than nested

- Sparse is better than dense
- Readability counts
- Special cases aren't special enough to break the rules
- Although practicality beats purity
- Errors should never pass silently
- Unless explicitly silenced

- In the face of ambiguity, refuse the temptation to guess
- There should be one—and preferably only one—obvious way to do it
- Although that way may not be obvious at first unless you're Dutch
- Now is better than never

- Although never is often better than right now
- If the implementation is hard to explain, it's a bad idea
- If the implementation is easy to explain, it may be a good idea
- Namespaces are one honking great idea—let's do more of those!

Other Principles

- Do the simplest thing that can possibly work
- Correctness and clarity before speed

PEP

- **P**ython **E**nhancement **P**roposals
- Significant changes must be described in a PEP
  - Proposals, descriptions, design rationales, and explanations for language features
  - Community consensus

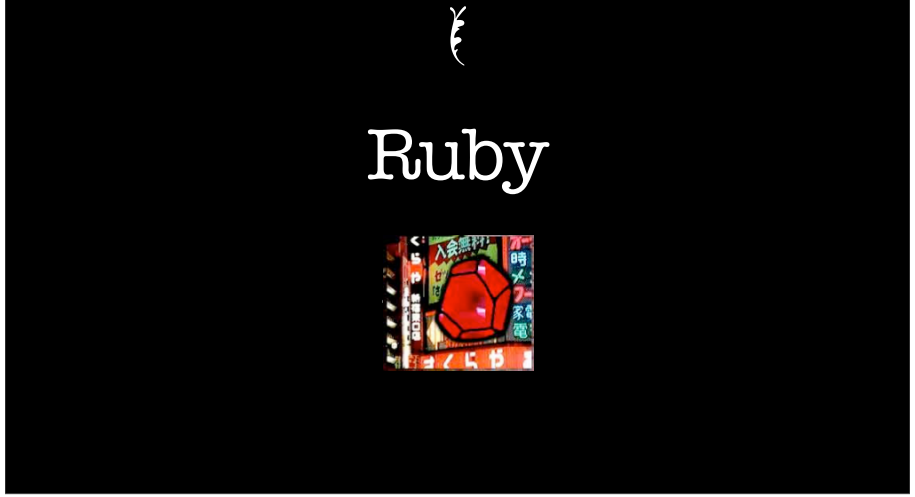
PEP (cont'd)

- PEPs are important for rejections
  - Many suggestions come back againEx.: `with`, interfaces, not using indentation as grouping syntax
  - PEP collection works like a newsgroup FAQ

PEP 343—with

- Describes adding a `with` statement to Python
  - Motivation
  - Use cases and examples
  - Transition plan
  - Rejected options

http://www.python.org/dev/peps/pep-0343/



I tried to make Ruby perfect for me, but maybe it's not perfect for you. The perfect language for Guido van Rossum is probably Python.

—Yukihiru Matsumoto [21]

Origins of Ruby

- A one-man endeavour: "Matz"
- Became impressed by DYPLs in –93
  - "Scripting was the way to go"
  - Liked object-orientation
  - Looked at Python and Perl

.

Ruby Culture

- More unorganised than Python's
  - Exactly how things work is not externally visible
- Enjoys being "in the hip" currently
  - Largely due to Rails and the Pragmatic Programmers

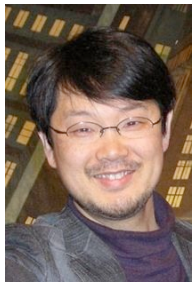
Ruby Change Req.

- Remember the purpose of the RCR is to convince Matz and other Rubyists that they would benefit from your proposed language change
- This isn't a democracy, Matz still makes the final call

Matz's RC2003 Talk

- "How Ruby Sucks"
  - Start from scratch
  - Consider performance
  - Will allow incompatibilities to arise in Ruby 2.0

- Slow
- Complex
- Surprising
- Inconsistent
- Bad embedding
- No native threads
- No Multilingualisation



According to Matz

- Languages affect the way we think
- Good programmers think about programming in a programming language
  - Natural languages are too verbose, ambiguous or indirect

Purpose of PLs

- Instruct computers
- Describe problem
- Express programmer's thought in the form of program (Most important!)

A Good Language

- Helps humans think and solve problems
- Gives a better programming experience

How Ruby Does It

- Ruby helps solve problems by being object-oriented (influences the programmer's design strategy)
- Providing good usability
- Ex.: !- and ?-methods, $-variables

Usability

- Easily accessible to beginners
- Efficiency is more important, since beginners wont stay that way long
  - Performance
  - Programmer productivity

Prog. Productivity

- Returning to a design after a period of inactivity
- We program to have fun (and $)
- Principle of least (bad) surprise (from the designer's point of view)

Succinctness is Power

- Less code implies less bugs
- Less bugs, consider yourself smart
- Productivity in LOC is invariant of programming language used [18]
  - Thus, succinctness is power

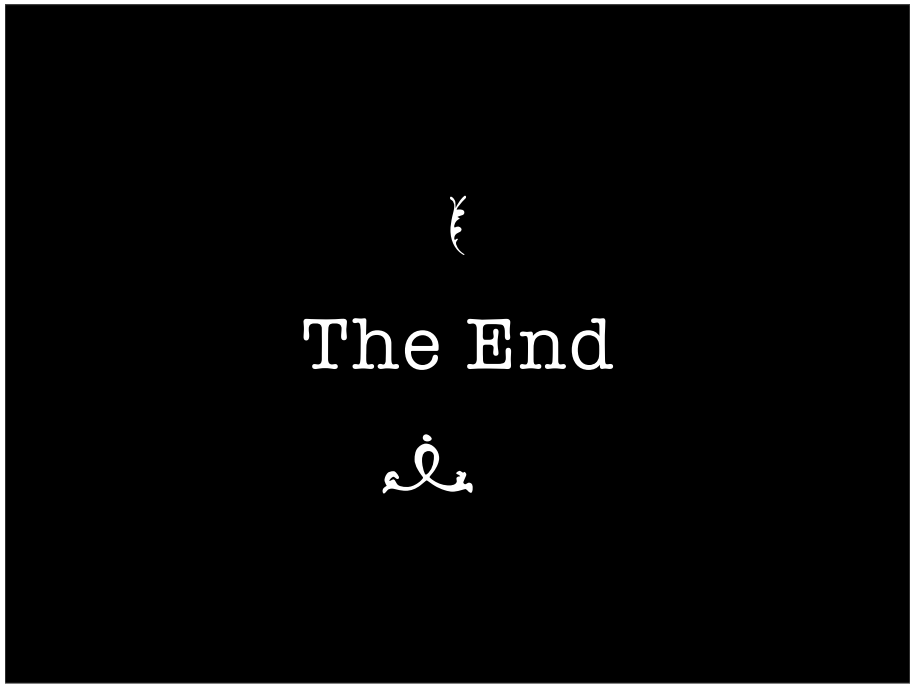
Wrapping it All Up

What's the Philosophy?

- Possible answers
  - This questions is wrongly put
  - There is none
  - There are many
  - Worse *is* better

Our Philosophy?

- Clearly postmodern
- Worse is (seemingly) better
- We ask you to make up your own minds



References

- Richard Gabriel, Worse Is Better [14]
- Richard Gabriel, Objects Have Failed [15]
- David Ascher, Dynamic Languages, Ready for the Next Challenges, by Design [2]
- Sam Wilmott, When Is A Dynamic Programming Language Not Dynamic [3]

References

- Bill Venners interviews Guido van Rossum on The Making of Python [19,20] and Yukihiro Matsumoto on the Philosophy of Ruby [21]
- python.org
- jpc.org