Back to Office Report

EuroPython 2013 Conference Florence, Italy

By Simon Griffee, Information Systems Officer, IPPC/AGPMI.
Simon's blog: http://hypertexthero.com/logbook/
Date Submitted: 14 August 2013
Travel Dates: 1–7 July 2013

Background & Purpose

Technology is increasingly a part of everyone's lives especially in the critical field of human communication, and it is moving forward ever more quickly.

To help the International Plant Protection Convention IPPC[1] fulfill its mission of protecting the world's plant resources from pests, and to help its Secretariat meet its obligations to its contracting parties in its core activities of Standards Setting, Capacity Development and National Reporting Obligations, it is important that IPPC's software development activities are modern and efficient.

To this end, a member of IPPC's Information Technology Team attended the EuroPython 2013 conference about the Python[2] programming language and related software projects in Florence during July 1–7 2013.

Results

The following are some of the main knowledge and skills gained at the conference:

1. **Good Enough is Good Enough!** Our culture's default assumption is that everybody should always be striving for perfection – settling for anything less is seen as a regrettable compromise. This is wrong in most software development situations: focus instead on keeping the software simple, just "good enough", launch it early, and iteratively improve, enhance, and re-factor it. This is how software success is achieved!
2. **Hands-on Django tutorial** - Learned to use the latest version of the Django web framework to build a website from the ground up. Covered Python and Django setup, project creation, application creation, models, the Django administration UI, views, data migrations with South, user image uploads, new user registration, forms and internationalization.
3. **How to integrate Python applications to authenticate against already existing identity management systems like LDAP, Kerberos, etc.**
4. **How to streamline web application code deployment.**
5. **Advanced E-Learning** - introducing programming to new coders by creating immediately useful software projects.
6. **The Return of Peer to Peer computing:** Overview of the coming change in computing, with a return to decentralized Peer to Peer systems.
7. **Introducing Python as a Main Programming Language in a Company:** Why Python is a valuable addition as one of the main programming languages to be used by IPPC (and FAO).

For more details please see the appendices beginning on page 4 of this report.

Follow-up

The following plans are recommended to utilize the acquired knowledge and skills:

A. Functional prototype of IPPC version 3.0 website with a new design. A new design of www.ippc.int should be created utilizing the philosophy of agile software development with quick, continuous releases of 'good enough' code and subsequent iteration to create better sofware with less bugs in less time. This can be developed in parallel with existing work. A working prototype has been created during July 2013 and the code can be found at **https://github.com/hypertexthero/ippcdj**

B. Integration of it.ippc.int application authentication framework with IPPC user's database. it.ippc.int currently uses it's own user authentication system. This needs to be integrated to authenticate with the existing www.ippc.int user database. See *Appendix 6: Introduce Django to Your Old Friends.*

C. Streamlining of deployment of IPPC web applications (both Python and PHP applications can co-exist peacefully). A simple (remember that simple does **not** mean easy!) deployment system needs to be put in place where we can easily package, version control, automate, secure, log, deploy, monitor and measure IPPC web applications. See *Appendix 5: Solid Python Application Deployments for Everybody.*

Suggested Next Steps

1. IT Team, lead by Paola Sentinelli, meets with IPPC Coordinator Craig Fedchock, CIOK Information Systems Officer Donatella Mori and CIOH Senior Server System Administrator Johannes Glaser to share information learned and decide on a good workflow to implement the above.
2. Decide on a specific project and set aside time (suggest one day per week) for the IT Team to learn Python and develop the project using the Python language. The suggested project is IPPC version 3.0 using Django web framework and agile development method. This work has begun and documentation to get started with current working prototype code is available at the following URL: **https://github.com/hypertexthero/ippcdj/blob/master/docs/documentation.md**
3. Begin work on integrating it.ippc.int authentication system with www.ippc.int's users database.

Persons Met

- **Alex Martelli** - https://en.wikipedia.org/wiki/Alex_Martelli
  Italian computer engineer and member of the Python Software Foundation.
  Works as "Über Tech Lead" for Google, Inc.
- **Holger Krekel** - http://holgerkrekel.net/
  founder and co-developer of a number of popular Python projects, among them PyPy, py.test, tox and execnet. Regularly gives talks at conferences as well as open courses and in-house trainings on testing topics. His interests also include P2P computing, metaprogramming and politics.
- **Vidmantas Zemleris** - http://zemleris.com/
  Software Engineer at European Organization for Nuclear Research (CERN) - http://home.web.cern.ch/
- **Maarten Kiekje**, sysadmin at University of Amsterdam - http://www.uva.nl/
- **Mike Müller**, Ph.D., Python Trainer and CEO at Python Academy - http://www.python-academy.com - Teaches a wide variety of Python topics including *Introduction to Python*, *Python for Scientists and Engineers*, *Advanced Python*, *Optimization and Extensions of Python Programs* as well as *Software Engineering with Python*.
- **Walter Kummer**, Python software engineer and GIS programmer
- **Christian Bertschy**, Django CMS core contributor & CEO of http://divio.ch
- **Ronald Evers**, Security Expert, https://www.fox-it.com
- **Alessandro Amici**, Physicist, Sofware Engineer, remote sensing, GIS, Earth Observation specialist, CTO of http://www.bopen.eu/
- **Daniel Greenfeld**, Software Developer, co-author of Two Scoops of Django - https://django.2scoops.org/: "*In the spring of 2006, I was working for NASA on a project that implemented a Java-based RESTful web service that was taking weeks to deliver. One evening, when management had left for the day, I reimplemented the service in Python in 90 minutes. I knew then that I wanted to work with Python.*"
- **Audrey Roy**, Software Developer and Designer, co-author of Two Scoops of Django
- **Vasily Kuznetsov**, Software Engineer, United Nations Framework Convention on Climate Change (UNFCCC) - http://unfccc.int/
- **Petr Viktorin** - http://encukou.cz/
  Software Developer for Red Hat. The open source Red Hat Enterprise Linux software runs the machines used to publish ippc.int and phytosanitary.info (and most other FAO websites).
- **Michael Hart**, Software Developer working in the UK

Appendices (conference notes)

1 The Next 20 Years of Python

https://ep2013.europython.eu/conference/talks/next-20-years-python

***V. Van Lindberg**, Chairman of the Python Software Foundation,*
works for Rackspace and OpenStack

- Python is an opinionated language.
- Guido Van Rossum (author of Python) managed to build a **community** with a sense of values.
- *Love, Internet Style* by Clay Shirky: **http://www.youtube.com/watch?v=Xe1TZaElTAs**
- Zen of Python - **http://www.python.org/dev/peps/pep-0020/** == core part of ethos:

  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!

- Standards language for bioinformatics.
- Primary teaching language for most universities, finally replacing Java (thank goodness).
- Community needs to scale.
- Raspberry Pi - **http://www.raspberrypi.org/** - Upton says 'Pi' stands for Python.
- The prompt as first thing you see when booting a computer is a powerful concept. Today computers primarily used for consumption. Consumption-oriented culture.
- Young coders, education, is very important. How to use computers to create. Young coders curriculum will be available in new **http://preview.python.org** (built with Django web framework - **https://www.djangoproject.com/**).

2 Thinking Outside the Box

https://ep2013.europython.eu/conference/talks/thinking-outside-box

***Armin Ronacher**, Creator of the Flask framework - **http://flask.pocoo.org/** and the Werkzeug and Jinja2 libraries among many others*

We all suffer from going with the flow in many cases.

Keynote gave some examples of how questioning some established rules revolutionized the industry and how we then open our mind a bit.

It included interesting examples from the Python and programming community where people tried something new with tremendous success and how they came up with the idea.

- Community influences you.
- Ask right questions, eg. "I don't want my users to wait while I process data."
- Assume you're already wrong.
- Describe the problem, leave room for answers.
- **http://bit.ly/pypush**
- Go on IRC.
- Most things have a design behind them.
- Gets lost as it is copied (Chinese whispers).
- Original design may no longer apply.
- Look up wheel package management.
- Paradigm shift as critical mass of people choose a way to do things.
- **Sometimes all that is necessary is to transport idea from one industry to another.**

3 The Return of Peer-to-Peer (P2P) Computing

https://ep2013.europython.eu/conference/talks/return-peer2peer-computing

***Holger Krekel**, Founder and co-developer of a number of popular Python projects, among them PyPy, py.test, tox and execnet.*

A video of this presentation is now online:
http://www.youtube.com/watch?v=SCgl0xXFVOg

> "Consider myself from global IT community, not from a particular nation."

@gonzohacker:

> "Someday when I'm living in a post-apocalyptic police state, run by autonomous flying drones, I want to tell my kids, 'I helped build that.'"

How much is Google, Facebook, etc. responsible for aiding the Spying Cartel?

Which **Internet Future** do we prefer:

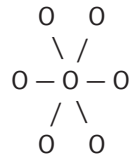
1. Greatest freedom catalyst ever?
2. Greatest surveillance machine ever?

Which one happens depends on:

- Politics & laws.
- Technologies.

IT people are like bishops. We know how programming works and relates to society. **We have a responsibility to make choices that cause no harm**. It's all about communication between devices, mediated by technology.

Centralization (current situation)

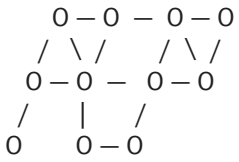


Espionage Cartels are happy:

1. Communication mediated by few companies.
2. Companies can be compelled to co-operate with government.
3. Weak or no end-to-end encryption == **BIG PROBLEM**.

How to disrupt Cartels?

Decentralization



1. End-to-end encryption.
2. Peer-to-Peer routing topologies.
3. Distributed mediation ownership.

Example: **http://crypho.com**, but: crypho as mediator has meta-data == PROBLEM

P2P code needs to be Free and Open Source (FOSS) and run on OWN device.

TAHOE-LAFS (Least-Authority File System) storage
https://tahoe-lafs.org/trac/tahoe-lafs:

- Provider-independent security.
- Decentralized, redundant, distributed.
- Active FOSS (free, open source software) Python project.

Communication is about more than only file storage. **New promising WC3 standard: Web RTC - http://www.webrtc.org** - browser-to-browser data streaming, real-time communications.

- With power comes abuse
- We are good at client/server, not so much P2P
- Global WLAN situation is idiotic. Idiotic means 'only for me' in ancient Greek.

Lookup:

- **http://ozan.io/p/** - Peer-to-peer networking with browsers.
- **http://sharefest.me** - Browser-based P2P filesharing using WebRTC.
- **http://vole.cc** - P2P social networking project.
- **http://nightweb.net** - P2P social networking project with bittorrent.
- WebCrypto API - native encryption in browser.
- Project SPAN: ad-hoc android mesh networks.
- **http://Freifunk.net** - free WLAN network.
- **Bittorrent Sync** - **http://labs.bittorrent.com/experiments/sync.html** Automatically sync files via secure, distributed technology. Great free, open source alternative to Dropbox.
- Bittorrent - **http://bittorrent.com** - written in Python, 10% of total internet traffic.

> "Given enough time, looking at history, countries are in danger of having criminals in charge."

It's about Convenience. Some choose inconvenience (Edward Snowden)
https://pressfreedomfoundation.org/blog/2013/06/snowden-principle

> **Q:** Is there hope for a world without nation states?
>
> **A:** Yes. FOSS (Free Open Source Software) is trans-national. For example, nations had no say on the WebRTC standard. Nations not so important in internet world.

How do we do mediation?

- Need social ID
- TAHOE-LAFS trust model **works** - **https://tahoe-lafs.org/trac/tahoe-lafs**
- Mozilla Persona - **https://www.mozilla.org/en-US/persona/**

> "It would make much more sense to have government communications open and transparent as they have nothing to hide, right?"
>
> One answer to the "*If you have nothing to hide, you shouldn't worry about surveillance*" argument:
> **The damage from a totalitarian society is much higher than from terrorists.**

4 "Good Enough" is Good Enough!

https://ep2013.europython.eu/conference/talks/good-enough-is-good-enough
***Alex Martelli**, Author of Python in a Nutshell. For the last 8+ years has been working (as 'über' tech lead and senior staff engineer) for Google in Mountain View, California.*

"**Worse is Better**" *—Unix, 'New Jersey' approach.*

- Simplicity.
  - Implementation & interface.
  - Most important consideration is design.
- Correctness.
  - Slightly better simple than correct.
- Consistency.
  - Not overly inconsistent.
- Completeness.
  - Can be sacrificed to any of above 3.
  - MUST be, if simplicity is threatened.

Well == Simple.

G.K. Chesterton:

> "Anything worth doing is worth doing badly."

E. Raymond:

> "Given enough eyeballs, all bugs are shallow."

Perfection == Big Design Up Front (not good).

Alex Martelli:

> "The world is changing all the time. Your perfection is a memory of the past."

Use 'Perfect' as VERB, rather than adjective.

Don't cut corners on:

- Light-weight agile process and its steps.
  - Revision control, code reviews, testing.
  - Proper release-engineering practices.
- Code style, clarity, elegance.
- Documentation.

No cowboy coding!

From the start:

- Security.
- Privacy.
- Auditability.

Best at start, but acceptable to add later:

- Modularity, plugins.
- API (application programming interface).
- Scalability.

Okay to incur technical debt, WITH CARE.

Eg: $n \log n$ versus n^2

Focus on potential errors that could cause *irrecoverable losses*:

- As long as you can recover, it's okay.
- In 'beta' at least.
- Is reputational damage to yourself recoverable?
  - Depends, but most usually, yes.
  - Especially courteous, speedy response to issues that get reported == good service.
  - Getting it right the 2nd time is usually OK.
- Hiring: you have to pick the best one *who will have you*.
- No one cares about perfect lines (speaking about a painting) — Am I moving something in people's hearts?
- Keep it simple:
  - **Gettysburg Dedication**
    - the "Oration": the soon-forgotten-one
    - Edward Everett
    - 13,508 words, two hours, reams of paper
    - and then...
  - **Gettysburg Address**
    - Abraham Lincoln
    - 267 words; two minutes, back of envelope
    - "*the world will little note, nor long. remember what we say here...*"
    - But, 150 years later, it certainly still DOES !-)

5 Solid Python Application Deployments for Everybody

https://ep2013.europython.eu/conference/talks/solid-python-application-deployments-for-everybody

***Hynek Schlawack**, Infrastructure and software engineer from Berlin, Germany, PSF fellow and contributor to a wide variety of open source projects.*

How a German web hosting company and domain registrar deploys numerous Python applications on UNIX-based systems. **Simple is not Easy**. Put **effort** into making your deployments **simple**.

Edsger W. Dijkstra:

> "Simplicity is prerequisite for reliability."

Donald Knuth:

> "It is important to find simple solutions instead of stopping as soon as a first solution is found."

- **Package**
  - Use Vagrant - **http://www.vagrantup.com**/
  - Application is tied to server OS **version**.
  - Upgrading servers == updating your app.
  - Packaging such as .rpm, .deb even .pkg.tgz, then:
    1. checkout from VCS.
    2. create virtualenv.
    3. install dependencies.
    4. **do whatever you want.**
    5. package the result.
    6. push to your repo.
  - Packaging is not hard. Use fpm - **https://github.com/jordansissel/fpm**
- **Automate** - use configuration management, test in staging.
- **Security** - Be paranoid, iptables, /bin/false, ssl, fail2ban, file sockets, revoke all, etc.
- **Restart** - upstart, systemd, supervisord, circus, etc.
- **Log** - log to stderr, redirect stderr syslog, use OS tools.
- **Deploy,** using Apache is fine if conscious decision is made to use it, but uWSGI or Gunicorn + Nginx provides better separation of concerns and is simple to set up and has some awesome benefits (uWSGI emperor mode - **http://uwsgi-docs.readthedocs.org/en/latest/Emperor.html**).
- **Monitor** - nagios, pingdom, etc.
- **Measure** - graphite, statsd, etc.

Presentation links:

- **http://hynek.me/talks/python-deployments/**
- **https://speakerdeck.com/hynek/solid-python-application-deployments-for-everybody-ep-edition**

6 Introduce Django to Your Old Friends

https://ep2013.europython.eu/conference/talks/introduce-django-to-your-old-friends

***Lynn Root**, Software engineer at Red Hat on the ID Management team. Founder of PyLadies, mentorship group for women and friends in the Python community.*

How to integrate Django applications to authenticate against already existing identity management systems like LDAP, Kerberos, etc.

- **http://www.roguelynn.com/kerberos/**
- **https://speakerdeck.com/roguelynn/europython-2013-introduce-django-to-your-old-friends**
- **http://chirale.wordpress.com/2013/03/15/unified-login-in-django-and-drupal/**

7 Sink or swim, five jackets to throw to the new coder

https://ep2013.europython.eu/conference/talks/introduce-django-to-your-old-friends

Lynn Root - ***http://newcoder.io***

Advanced e-learning, or introducing programming to new coders.

- Software development moves fast. How to teach programming?
- Concepts.
- Episodic vs. cumulative learning.
- Project-based learning.
- Learn to tread water by playing water polo, not by someone dictating how to move arms & legs.
- Knowledge is actively **constructed** by the student.
- Build things that are used in the real world.

Tutorials:

1. Dataviz - **http://newcoder.io/dataviz/**
2. APIs - **http://newcoder.io/api/**
3. Web scraping - **http://newcoder.io/scrape/**
4. Networks - **http://newcoder.io/networks/**
5. GUI (soon)

8 Intro to Django

https://ep2013.europython.eu/conference/talks/intro-django

Daniel Greenfeld*, software developer, co-author of Two Scoops of Django*

Audrey Roy*, software developer and designer, co-author of Two Scoops of Django*

Hands-on Django workshop. Build a real, working site from the ground up, using Django 1.6 and Python 2.7/3.3. Covered Python and Django setup, project creation, app creation, models, the Django admin UI, views, migrations with South, user image uploads, new user registration, basic forms, and basic internationalization.

Result of training: **https://github.com/hypertexthero/ippcdj**

9 Introducing Python as a Main Programming Language in a Company

Patrick Büchler*, Software Developer*

Car analogy - people who go to BMW will get different quality than Fiat.
Same with **software**, but not appreciated.

Right mix of people important. Pointless to have lots of all-stars because they will all have their own idea of how it **should** be done.

Why Python? Too many languages == uncomfortably hot. Python is platform independent, has big community, good traction, good adoption, good future plans, good learning curve:

1. Simple and readable
2. Versatile and platform agnostic
3. Can be used for web and on client
4. Also a philosophy, i.e. the Zen of Python: http://www.python.org/dev/peps/pep-0020/

How to implement Python? How to manage risks?

1. If you take a decision, implement it quickly.
2. People learn only with real challenges.
3. Innovation not the same as research. As entrepreneur you must innovate, not research.
4. Can't afford to spend much money on not-financed activities.

Choose project and do it with Python

1. Project should not be critical.
2. Not be too small (at least 1 man-year).
3. Should not contain complex technical challenges.
4. Should not contain complex business logic.
5. Should not be in a risky domain.
6. Should use easy to use framework (such as Django - **http://djangoproject.com/** or web2py - **http://www.web2py.com/**)
7. Should be assigned to a specific team, without people who are against the project, with mix of first-movers and conservatives.
8. Primary goal should still be to finish the project with a profit.

Problems

Mostly went surprisingly smoothly! Consider the following:

1. Python is not Java!!
   `def my_function() != public int myFunction()`
   Do code reviews! Especially when devs used Java or C# before. Developers coming from VB have less projects.
2. Allow refactoring: give your team the time to make their first project *good*.
3. Enforce refactoring: force your team to make the first project *good*.
4. Be patient: do not forget that it is new for most developers.
5. Be strict: you decided to risk implementing the project in a new language, stick to your decision.
6. **Python is agile, use this**: with python debug cycle is very quick, use this i.e. in sprint review.
7. Don't interfere too much: let your colleagues work on their own, so they learn!

Factors critical for decision about tech to use:

Technology and organizational factors (reasonable):

1. Surrounding systems enforce specific tech.
2. Base systems used enforce particular tech i.e. Oracle.
3. IT Departments often proficient on one platform only.
4. Contracts with vendors or suppliers enforce given tech.
5. High investments required to change.

Human Factors (personal):

1. IT Departments often reluctant to change their systems even when outdated or try something new (consider the type of personality who becomes sysadmin).
2. Management hear saying (we MUST have SharePoint because it will solve all our problems as my friend told me on golf course).
3. Internal Politics (we decided that XYZ is the best architecture, we cannot change now).
4. Quasi-religious beliefs or important decision makers or influential people ('there is no better sport than to provoke a developer by criticizing his language of choice').

The word 'Python' won't open doors, but the folowing may:

1. Platform independent.
2. **Very fast development cycle** - you can even debug and change the code onsite during the sprint review!!
3. Completely open source - not owned by Oracle unlike Java, MySQL, etc.
4. Mature.
5. Big community.

Fears/questions from customers:

1. Fear of Vendor Lockin: most business people know .NET and Java and believe all else a niche.
2. Fear of Open Source: understandable, many dead open source projects.
3. Lend helping hand to sysadmins: maybe he used SQL Server and IIS since 199x! Suddenly it's PostgreSQL and web2py. Help him and he will be a grateful promoter of your technology.
4. **Most problems are not technical!** Remind your customer and your team about this.

Was it worth it?

:) Yes!

1. Python is well established now
2. At moment four different projects with Python
3. Strategic decision: all individual web projects are done with *web2py*
4. Wherever possible suggest using Python

Q: Have you tried the sales pitch "lets do a prototype in Python first and see if it works"
A: No, but it is a good idea

General notes:

- Education needs to have a practical purpose, i.e. **learn by working on a project**
- Good to point out all the great software already available in Python, i.e. 1 line instead of 25 lines - http requests - **http://docs.python-requests.org/en/latest/**
- If you run into performance issues, do code review, use C and bind to Python, etc.

10 Write a Fault-Tolerant Web Service Using gEvent and uWSGI

https://github.com/ultrabug/ep2013

***@ultrabug**, Software Developer in Paris*

- Nginx (1 process)
- 1 uWSGI gevent process + 1 uWSGI spooler process
- mongoDB

If mongo dies: spool files on disk
https://github.com/ultrabug/ep2013

11 The Agile Movement

https://ep2013.europython.eu/conference/talks/the-agile-movement-and-why-ignoring-it-will-leave-you-stuck-in-a-rut

***Russell Sherwood & David Sale**, Sky Television*

Manifesto for Agile Software Development - http://agilemanifesto.org/

> We are uncovering better ways of developing
> software by doing it and helping others do it.
> Through this work we have come to value:
>
> **Individuals and interactions** over processes and tools
> **Working software** over comprehensive documentation
> **Customer collaboration** over contract negotiation
> **Responding to change** over following a plan
>
> That is, while there is value in the items on the right,
> we value the items on the left more.

Footnotes

1: The International Plant Protection Convention (IPPC) is an international agreement on plant health to which 179 countries currently adhere. It aims to protect cultivated and wild plants by preventing the introduction and spread of pests. The Secretariat of the IPPC is provided by FAO, whose strategic objectives, to which the IPPC contributes, are the following:

a. Help eliminate hunger, food insecurity and malnutrition
b. Make agriculture more productive and sustainable
c. Reduce rural poverty
d. Ensure inclusive and efficient agricultural and food systems
e. Protect livelihoods from disasters

2: Python is paradoxically attractive to both beginners and expert programmers.
See also: http://www.paulgraham.com/pypar.html