EuroPython 2007

Monday 09 July 2007 - Wednesday 11 July 2007

Vilnius, Lithuania

Programme

Table of contents

Monday 09 July 2007

Registration (09 July 08:00-09:00)

Open Space - Lambda (09 July 09:00-10:30)

time	[id] title
09:00	**[76] Open Space (01h30')**

Python Language and Libraries - Theta (09 July 09:00-10:30)

time	[id] title
09:00	**[21] Extending Python with EasyExtend (00h30')** *Speaker: Mr. SCHLUEHR, Kay* Extending Python as a language has always been among the main objectives of the Python core development team. Enhancing Python and developing the CPython runtime have become interchangeable activities. Those who tried to extend Python had to be core developers of CPython or at least experimenting with one of the alternative runtime environments like PyPy, IronPython or Jython. For those who had research interests in the language or writing domain specific languages on top of Python this has always been an obstacle. EasyExtend is a grammar based code generator and Python preprocessor framework in the tradition of Lex/Yacc and ANTLR but written in pure Python and seamlessly integrated with Python. EasyExtend is also inspired by the languages-as-components metaphor which is reified in its so called fiber-space architecture. This talk will give a short introduction into EasyExtend, its design and prospectives.
09:30	**[3] Parsing Languages with mxTextTools (00h30')** *Speaker: Mr. LEMBURG, Marc-Andre* mxTextTools comes with a high performance tagging engine for text and Unicode data which can be used to tokenize and parse (little) languages. The resulting abstract syntax tree can then be hooked up to a generator to build a complete and fast compiler in pure Python. The talk will give a short introduction to the way the tagging engine works and how it can be used to build compilers. mxTextTools is an eGenix Open Source product available as part of the eGenix mx Base Distribution.
10:00	**[44] Pythonic Interfaces (00h30')** *Speaker: Mr. HAWKER, Michael* With the evolution of computer systems, software development has become increasingly more complex. One way to deal with this increased complexity is through the use of software libraries. Many object-oriented languages provide special constructs such as abstract classes and interfaces which ensure that components are properly extended and executed. Unfortunately, the Python programming language is devoid of such features. In this article, we present a library extension for Python to include these features into the language and allow for explicit class interfaces and abstract classes. While it has been attempted before, our library provides a simple, elegant, and Pythonic solution to the problem via a pure-Python stand-alone library. By extending the Python language in this manner, we allow developers to define concrete models for libraries and create modular code, while ensuring that software system designs are enforced at run time. We also argue that this provides the Python language another degree of flexibility in a formalized mechanic, as opposed to error-prone traditional “hand-shake” contracts. The usability of our proposed Python extension is demonstrated in a case study of an original game engine framework.

Web Related Technologies - Zeta (09 July 09:00-09:30)

time	[id] title
09:00	**[16] Object databases for Python (00h30')** *Speaker: Mr. FRANZ, Markus* After years of decline, object databases are now becoming more and more important again because of new open source systems. Their data objects directly appear as programming language objects without any mapping - avoiding the impedance mismatch between objects and relational tables. Most object databases are designed to work well with languages such as Java, C# or C++. This talk covers object database systems for Python with market overview, evalution of products and case studies from enterprise environments. Not only systems with direct access from within Python, but also complex ones with a Java bridge, for example, are shown.

Web Related Technologies - Alpha (09 July 09:00-10:30)

time	[id] title
10:00	**[53] Googled Python (00h30')** *Speaker: Mr. FRANZ, Markus* Google offers a variety of services for web developers, including local search and maps. This talk shows how to use these services. The best libraries are shown including case studies for enhancing own applications.

Python Language and Libraries - Zeta (09 July 09:30-10:30)

time	[id] title
09:30	**[34] Seamless object persistence with ZODB (00h30')** *Speaker: Mr. THEUNE, Christian* Object DBMS' have not been wildly successful as a generic database. Persistency frameworks however have popped up in many places (like Hibernate) and most of them use a relational database in the backend. This always involves an impedance mismatch and also a performance penalty. The ZODB is a pure object database written in Python (and a little help from C) that has been around for about 10 years and provides reliable object persistence in Python. It is an offspring of the Zope project and can be used in any Python program. The ZODB includes is designed with all the properties of an enterprise level DBMS including logical/physical data representation, multiple backend types, in-memory snapshots, transparent persistence for Python objects, ACID compatible transactions and more. The talk will show the architecture of the ZODB, how to write a standalone application including the use of the cluster mode and some advanced features.
10:00	**[61] The Storm Object-Relational Mapper (00h30')** *Speaker: NIEMEYER, Gustavo* This talk will present Storm, a new Python ORM developed at Canonical which permits mapping of objects against multiple relational databases with ease. Topics covered include the project history, the high-level architecture, and examples. This EuroPython talk will also be the first public announcement of Storm, and will mark the release of the project under an open source license for general use and contributions.

Open Space - Lambda (09 July 11:00-12:30)

time	[id] title
11:00	**[77] Open Space (01h30')**

Python Language and Libraries - Alpha (09 July 11:00-11:30)

time	[id] title
11:00	**[35] Snakes on a Phone (00h30')** *Speaker: Mr. LAURILA, Jukka* Past, present and a bit of the future of Nokia's Python porting work. How do you squeeze the interpreter into a small device? How do you provide access to complex C++ APIs in a way that makes sense? Also, what does the brave new world of Trusted Computing mean for the Python coder?

Science - Theta (09 July 11:00-12:30)

time [id] title

11:00	**[23] OpenAlea - Visual Programming and Component based software for plant modeling (00h30')** *Speaker: Mr. DUFOUR-KOWALSKI, Samuel* Building plant models at different scales requires integrating tools from various scientific domains such as biology, computer science, and mathematics. The open source OpenAlea project's goal is to define a framework to share and reuse heterogeneous models from the plant modeling community. A visual environment is made available to researchers to ease the building of high-level computational tasks in a simple and flexible way. The core of OpenAlea, implemented in Python, consists of a component framework that allows for the dynamic management and composition of software components. A component is a Python callable object with input and output ports. Components communicate through their ports, and can be interconnected to form complex processing graphs. A package is a deployment unit that provides components, data, and widgets as well as meta-information. The package manager is able to find and load dynamically packages installed on the computer without specific configuration. A series of packaging tools, SconsX and Distx, based on SCons and Distutils, have been developed to ease the distribution of complex C++ extensions, and to share dynamic libraries between different OpenAlea packages. A special application, Visualea, has been created as a PyQt4 visual programming environment to edit and run dataflow. For each component, graphical widgets can be either provided by user packages or generated automatically, based on their input port interfaces. This makes it possible to reuse widgets in different packages. More information on OpenAlea is available at http://openalea.gforge.inria.fr.
11:30	**[54] PLASMAKIN – A Python package to handle chemical kinetics in plasma physics modelling (00h30')** *Speaker: PINHãO, Nuno* PLASMAKIN is a package for handling physical and chemical data used in plasma physics modelling and for computing kinetics data from the reactions taking place in the gas or at the surfaces: particle production and loss rates, photon emission rates and energy exchange rates. It has no limitation on the number of chemical species and reactions that can be handled, is independent of problem dimensions and can be used in both steady-state and time-dependent problems. A large number of species properties and reaction types are supported, namely: gas or electron temperature dependent collision rate coefficients, vibrational and cascade levels, evaluation of branching ratios, superelastic and other reverse processes, three-body collisions, radiation imprisonment and photoelectric emission. Non-standard rate coefficient functions can be handled by a user supplied shared library. PLASMAKIN is based on a shared library with data reading and computational functions and a Python module based on the ctypes foreign function library and providing python function interfaces and classes. Together with other Python numerical and data plotting libraries such as SciPy and matplotlib, PLASMAKIN allows a fast and efficient analysis of plasma physics problems.
12:00	**[0] ThanCad: 2dimensional cad for engineers (00h30')** *Speaker: Prof. STAMOS, athanassios* ThanCad is 2dimensional CAD aimed to meet the, ever growing, specific needs of civil and surveing engineers. It is largely command compatible with the leading commercial CAD, but it differtiantes to a few concepts such as hierarhcical layers, and lack of elements attributes, which is the CAD equivalent to structured programming. ThanCad adds some productivity tools such as line continuation, layer selection and cross-save/read undo mechanism. ThanCad harnesses the power of Python to shrink the development time and the volume of code; to implement and test new ideas in virtually no time; to make ThanCad programmable without the need of separate libraries, plug-ins, special languages, or special OSes; to make 32bit/64bit processor, OS and OS version irrelevant. ThanCad uses Tkinter, the defacto GUI/drawing standard for python, in order not to reinvent the wheel and to achieve platform independence. Several concepts were addressed such as compound elements, text in arbitrary directions, cursor/crosshair, zoom, image zoom, coordinate systems tracking, image resolution, draworder, object snap, different element intersection, mouse wheel windows/linux differences, input from command window and/or GUI, hierarchical modal windows. Finally, ThanCad uses Python's object oriented programming, but sometimes it follows the Zen of Python and the Linux kenel's philosophy, practicality beats purity.

Web Related Technologies - Zeta (09 July 11:00-12:30)

time	[id] title
11:00	**[4] Case study of a Pylons project (00h30')** *Speaker: ISCHENKO, Max* The talk is a case study for my experience rebuilding a PHP-powered site in Python. Primarily focus is Pylons which is a modern, cool and immature web framework, currently heading for 1.0 release. Topics covered include: Pylons itself, Paste, deployment, i18n, Mako templating and SQLAlchemy. I'll share our experience about what worked and what didn't and what to watch for. The talk will fit in a 30-minutes timeslot. The site studied is www.developers.org.ua -- Ukrainian software developers' community site. Presenter (Max Ischenko) is its creator and primarily developer, his blog can be found at http://maxischenko.in.ua/blog/.
11:30	**[18] KSS, Ajax development with style (00h30')** *Speaker: Mr. VLOOTHUIS, Jeroen* Writing Ajax application is normally associated with writing Javascript. Not any more, KSS will let you use a familiar CSS like syntax combined with highlevel commands to write your applications in style. In this presentation you will learn what KSS is and what it can do for you. Because KSS is easy to integrate with any server platform everyone is invited. During the presentation this will be proven by a demonstration of KSS integration in both Plone and Pylons.

12:00	**[56] Using FormEncode for web data validation (00h30')** *Speaker: Mr. STRATTON, Graham* FormEncode is the form data validation library included by default with both TurboGears and Pylons. It is a powerful and flexible tool for both validation of form data and conversion of raw data to Python objects. This talk will cover the basic principles of FormEncode and give examples of how to achieve a number of common tasks. The use of htmlfill to render customised form pages with input errors will be demonstrated. Finally, there will be a brief demonstration of the integration between Pylons and FormEncode.

Web Related Technologies - Alpha (09 July 11:30-12:00)

Education - Alpha (09 July 12:00-12:30)

time	[id] title
12:00	**[58] Tux Droid, a python-fueled robot (00h30')** *Speaker: Mr. BOURGEOIS, David* Tux Droid is a tux-shaped robot wirelessly controlled from a computer running Linux. It can talk and move, but also listen and react to events. The wireless link is the key here, it provides Tux Droid with access to all the power of Python. Python's interpreter is of great help if you don't have any clue about programming languages and would like to make your first steps into robotics and programming. Simple scripts to control your robot are really at your fingertips. You can easily stream music and sounds to and from Tux Droid, blinking the eyes, wave the flippers, spin, send and receive IR codes or detect if the head button is pressed. Tux Droid's embedded processing is only meant to interface sensors and actuators. But by developing a complete python API, we can leverage the embedded basic set of features to get a complex robot which can bring life to your applications and act as a desktop companion. Something advanced python users will also have fun with.

Education - Zeta (09 July 14:00-14:30)

time	[id] title
14:00	**[64] Pythonic Math (00h30')** My paper for EuroPython 2005 explored what I call Pythonic Mathematics, a way of presenting pre-computer analytical content within the OO paradigm, including pre-college.[1] This thinking informed my participation in Shuttleworth Foundation planning meetings and presentation to the London Knowledge Lab in the following year.[2] This year, I'm delving yet more deeply into Pythonic Math, while also weaving in some more cultural threads, especially the "design science" thread with its geodesic spheres and other graphical content, the theme of my OSCON 2005 presentation.[3] I've been field testing these combinations in my home town of Portland, through a school called Saturday Academy.[4] Whereas Guido named Python for Monty Python, begetting allusions which aren't going to go away, there's more we might do to make our snake come across as charming and smooth, not too slimy or oily (negative attributes customarily associated with snakes by the more snake-unfriendly).[5] [1] http://www.4dsolutions.net/presentations/urner_europython4.pdf [2] http://www.bfi.org/bfi_community/pythonic_mathematics_talk_by_kirby_u rner [3] http://worldgame.blogspot.com/2007/01/reviewing-my-oscon-2005-talk.ht ml [4] http://www.saturdayacademy.org/ [5] http://worldgame.blogspot.com/2007/06/pro-python-propaganda.html

Open Space - Lambda (09 July 14:00-15:30)

time	[id] title
14:00	**[78] Open Space (01h30')**

Python Language and Libraries - Alpha (09 July 14:00-15:30)

time	[id] title
14:00	**[37] PyPy 1.0 and Beyond (01h00')** *Speakers: Mr. PEDRONI, Samuele, Mr. RIGO, Armin* PyPy released 1.0 in March of this year. PyPy contains a very compliant Python interpreter, and with 1.0 the first incarnation of a Just-In-Time compiler which is generated from the interpreter automatically with novel techniques. In this talk we are going to give a brief introduction to PyPy and its motivation. After recapitulating PyPy architecture we are going to give an overview of how the JIT generation is achieved, through examples and introducing the general ideas. In the last part we are going to reflect on the current status of PyPy's Python interpreter and our thoughts about its future and further progress on the JIT compiler.

15:00	**[39] PyPy Python Interpreter(s) Features (00h30')** *Speakers: FIJALKOWSKI, Maciek, KREKEL, Holger* We quickly recap the basic architecture of PyPy Python interpreter(s) and then demo and discuss the following unique features: * transparent proxy: a way to customize behaviour of builtin objects, enabling new models of persistence and distribution * distribution prototype: have objects from remote places appear as local ones, including frames (PDB!), file objects etc. * object tainting: automatically track and control propagation of sensitive data through an application * lazy computations: defer computation until result is needed * ... and more, as talk topics/features might appear just-in-time :) All above features might be translated to any supported backend, which is by now C, LLVM and CLI, soon JVM. We'll also see to discuss future efforts and remaining obstacles to reach wider adoption.

Web Related Technologies - Theta (09 July 14:00-15:30)

time	[id] title
14:00	**[19] An introduction to working with relational databases from Zope (00h30')** *Speaker: Mr. CLARK, Charlie* Although Zope has been around for quite a while arguably contains some fairly outdated code, it continues to find new users particularly amongst non- programmers who are looking for a way to work with existing data which is usually in some relational database (PostgreSQL, MySQL, MS SQL, Oracle, DB2, etc.). One of the reasons for this is at Zope provides an extremely powerful yet secure through the web environment. The presentation is directed towards those users and will provide a brief introduction by example into the Zope way of doing things and at the same time highlighting how working within Zope (within its limitations) is automatic training in good programming methodology: data management is delegated to ZSQL methods, PythonScripts act as controllers and Zope Page Templates provide the views. Together they encourage modularity and reusability. The sample application and database will be available for download.
14:30	**[7] Zope on a Paste (00h30')** *Speaker: Mr. VON WEITERSHAUSEN, Philipp* This talk, not starring Samuel L. Jackson, explores how to deploy a Zope application using Paste. For a while now, Zope has had support for WSGI and has been using it internally to connect to the two officially supported server frameworks, zope.server and twisted. Other Python web frameworks, on the other hand, have been using PasteDeploy to connect their web applications to any WSGI gateway or WSGI middleware using nothing but a configuration file. In this talk we will see how this approach is brought to Zope, opening the possibility of using various server gateways and middlewares in Zope without much or any code. Zopistas will learn about the world of WSGI, Paste and middlewares. Non-Zopistas will see how to talk to Zope from their WSGI-capable webserver or middleware.
15:00	**[51] High performance Zope3 (00h30')** *Speaker: Mr. BATLOGG, Jodok* in this talk we'd like talk about using zope3 for "web2.0" community portals. we're serving more than 100mio requests per month with a peak traffic of 250mbit/s at more than 1000 concurrent requests. we were fighting hard to "tune" zope3 to that scale. we'll talk about: - the hardware architecture when running these applications: nginx reverse proxies, varnish caches, memcached caches, ipvs load balancers, XEN virtualization,...), - strategies to monitor and improve zope3 settings (z3monitor, zservertracelog, cache tuning,...) with python plugins for cacti - deployment of multiple servers with buildout/eggs - software quality (buildbot, ftests/unit testing, selenium) - do and dont's to consider when building highly personalized web2.0 portals we'll give a "hand's on" how to use our performance and caching python packages to improve the speed of zope3.

Python Language and Libraries - Zeta (09 July 14:30-15:00)

time	[id] title
14:30	**[89] Optimizing MySQL (00h30')** *Speaker: Mr. AXMARK, David* How to make MySQL go fast, from someone who was involved with the project before it had a name.

Open Space - Zeta (09 July 15:00-15:30)

time	[id] title
15:00	**[79] Open Space (00h30')**

Agile Experiences and Testing - Alpha (09 July 16:00-17:30)

time	[id] title
16:00	**[42] py.test: towards interactive, distributed and rapid testing (00h30')** *Speakers: KREKEL, holger, FIJALKOWSKI, Maciej* We'll talk about py.test, an advanced and easy-to-use Python-based testing tool, aiming to speed up and integrate testing, development and documentation efforts. py.test is a mature tool and used in many projects. We will briefly present highlights of both long-standing and new features (since ep2006): * cross-project external tool for collecting and running application tests * minimal boilerplate approach, write and deploy first test in 60 seconds * ad-hoc distribution of tests on many computers (e.g. by ssh invocations) * generated AJAX application to report multi-host test events * document Python functions with type information tracked at test-run-time * project specific test configurations allow extensions such as: - documentation syntax and referential integrity checks - ad-hoc driving of Windows GUI acceptance tests from Linux - JavaScript Regression tests on PyPy's emerging JavaScript interpreter There are several more features and ideas being considered currently ... all aiming to create a more effective, interactive and joyful test-driven development process ... which we'd like to discuss with the audience and test-tool developers and users.
16:30	**[45] unittest is Broken (00h30')** *Speaker: WINTER, Collin* In this paper I examine the shortcomings and core design flaws of Python's standard unittest module, focusing specifically on the programmer's ability to extend unittest. I then discuss the requirements for an extensible testing framework and introduce test_harness, an alternative framework designed from the ground up to address these fundamental issues. Finally, examples drawn from real-world, unittest-based test suites are reformulated using test_harness to demonstrate the power of the new framework.
17:00	**[49] A practical example of Test Driven Development for a GUI using wxPython (00h30')** *Speaker: Mr. MASINI, Stefano* I will show a step by step example of building a small user interface using wxPython. The example will only be marginally related to wxPython itself and, even though previous background won't be necessary, it'll be explained only as much as necessary to understand the rest of the talk. The goal is to show the practice of Test Driven Development. The code will follow the Model View Controller pattern and testing will be making use of mock objects, in order to simulate the View.

Open Space - Lambda (09 July 16:00-17:30)

time	[id] title
16:00	**[80] Open Space (01h30')**

Open Space - Zeta (09 July 16:00-17:30)

time	[id] title
16:00	**[81] Open Space (01h30')**

Web Related Technologies - Theta (09 July 16:00-17:30)

time	[id] title
16:00	**[30] z3c.dav – an implementation of WebDAV for Zope3 (00h30')** *Speaker: Mr. KERRIN, Michael* z3c.dav is an implementation of the WebDAV protocol for Zope3. It contains a number of components that help developers provide WebDAV support for their application. z3c.dav works by parsing all the WebDAV requests. It then uses the Zope component architecture to lookup a WebDAV component that can handle the data it just received. This component may get or set the value of a property, or it can call on other components from the Zope3 framework to perform the requested action. Locking, copy and move are implemented this way. These WebDAV components do not deal with parsing of the request data nor do they deal with generating any possible response, they just handle the data according to the WebDAV components contract and z3c.dav will handle the rest of the protocol details. This design as meant that z3c.dav can support multiple content types to varying degrees within the one application. In the future I am planning to build on z3c.dav to support extensions to the WebDAV protocol like search, access control lists, and calendaring. This talk will introduce the project and the features it supports. I will then cover all the common components implemented in z3c.dav that a developer will need in order to provide WebDAV support for their application. I will then demonstrate this with a sample application using OpenOffice.org as a client. Finally I will give a outline for the future of this project.

16:30	**[20] Silva 2.0 (00h30')** *Speaker: Mr. BLAKE, Kit* Since it's launch at EuroPython 2002, Silva has grown into a powerful CMS for organizations that manage complex websites. The first block of the talk will describe Silva's feature set, covering its versioning, workflow, content reuse, access control, multi-site management, virtual host awareness, import/export facilities, templating, and image manipulation. The second section will explain Silva 2.0's jump into Zope3 technology, how it uses Five for layout, view, i18n'ing, and extension development. The last block will trace Silva's road map for the foreseeable future.
17:00	**[60] A dynamic Learning Content Management System (dLCMS) (00h30')** *Speaker: Mr. BLAKE, Kit* The dynamic Learning Content Management System (dLCMS) is a content management system for web-based learning materials and supports easy editing and user-friendly compilation of learning contents, enhanced scalability, and flexible use of the materials in various didactic contexts. An adaptable learning content component model defines different levels of learning components, the properties of these components, such as granularity, and how the components can be aggregated into larger learning units. The dLCMS functional architecture consists of four primary components: authoring, repository, assembly and linking, and publishing and export. The talk will explain the basic learning concepts behind the dLCMS and demonstrate these within the application.

Keynotes - Alpha (09 July 17:45-18:45)

time	[id] title
17:45	**[67] Keynote by Simon Willison (01h00')** *Speaker: Mr. WILLISON, Simon*

Tuesday 10 July 2007

Business and Applications - Theta (10 July 09:00-10:30)

time	[id] title
09:00	**[57] 3D Geo-portal Visualization Software for Control Rooms in the Oil and Gas Industry using Python (00h30')** *Speakers: Mr. FAHMY, Tamer, Mr. SEEHUUS, Rolv* Off-shore operations in the oil and gas industry are becoming increasingly expensive and complex. One of the strategies to overcome this development is to create operation centers on-shore. Centralizing the control of oil-fields allows field-managers to make optimal decisions in a global scale. However, the amount of information they must process increases dramatically. We present the application Vispo3D Operations. Vispo3D Operations ties together information from various data sources and provide an overview of the current situation in the North Sea. The decision making is facilitated by providing 3D visualization for various aspects in oil-rig management, such as the simulation of the current weather conditions. The application is implemented almost exclusively in Python. Due to the novel and challenging requirements of the project for both developers and customers, Python proved to be a valuable tool for Rapid Application Prototyping and more importantly for explorative development. The original plan was to re-implement the whole solution in C++. However, the Python version proved to be sufficiently responsive and adequate, very flexible and especially adaptable with regards to the changing requirements. Vispo3D Operations makes use of Kongsberg SIM's mature 3D visualization libraries and toolkits collectively named Coin3D. Coin3D is a high-level, retained-mode visualization suite for effective 3D graphics development, which is cross-platform, open source and without any royalties or run-time fees for professional licensees. Consequently, Python bindings for Coin3D, called Pivy, were created to interface with the Coin3D C++ libraries. PyQt from Riverbank and Qt from Trolltech are used for the graphical user interface. CherryPy, a pythonic, object-oriented HTTP framework, is used for client server communication, as well as serving a web-interface for off-shore users. The data itself is stored in SQLite databases, which is fetched through XML-RPC from the clients. The decision to use Python proved to be very rewarding and allowed us to cope with the changing and demanding requirements from the customer with ease. The successful usage of Python within this project convinced upper management to leverage the advantages and benefits of Python for further upcoming new projects.
09:30	**[48] Python for Industry and the ORDID Project (00h30')** *Speakers: PLIGER, Fabio, BURATO, Elisa* Sia and S3 focus their work on supplying both low level and high level systems solutions for industrial partners. This year SIA started a research project with the university of Verona to improve it's applications and solutions named ORDID: Upper-level Ontology-Driven Interpretation of Raw Data. The project aims at looking for new and inovative construction of high level symbol grounding techniques to replace the traditional techniques used before, executing ontologic schemas to guarantee interoperability. The talk will look into the role played by Python into SIA, S3 and the ORDID project itself.
10:00	**[9] Twisted and Zope in real time monitoring for oil and gas industries (00h30')** *Speaker: Mr. STRICKLAND-CLARK, Dale* An innovative company in the South of England has produced the first safe way to remotely monitor oil and gas wells wirelessly. They'd built the hardware and now urgently needed software so they could demonstrate their product at a trade show. Riverhall Systems used Twisted, Zope and a home-grown graphics server to produce a useful demonstration which was used successfully at the event. Twisted handled the real-time communication with the wireless monitors and also provided an RPC interface to a data logger. The control and reporting front-end was written in Zope. Our graphics server, also written in Python, created the dials and charts in real-time.

Python Language and Libraries - Lambda (10 July 09:00-10:30)

time	[id] title
09:00	**[40] RPython: Need for speed aka C and C# considered harmful (00h30')** *Speakers: FIJALKOWSKI, Maciek, CUNI, Antonio* Restricted Python (RPython) is a subset of a Python language designed to be compiled into lower-level languages, suitable for direct compilation into C, CLI, JVM or others. RPython cuts some of python dynamism (allows for full type inference and creation of flow graphs), but doesn't require explicit type annotations. Results might be up to 300 times faster than the original python program. Although it was born as an implementation detail of the PyPy interpreter, experience showed that it can be useful by itself. This talk will present some typical RPython's use cases, including: * how to write a standalone executable in RPython. * how to write an extension module for CPython. * how to produce .NET libraries/executables as fast as C# * how to write AJAX applications without a single line of Javascript. (and with tests!) * how to create bub-n-bros client in just few lines of code running on a browser :-)

09:30	**[33] How our Python trading platform got 40 times faster by switching to RPython (00h30')** *Speakers: Mr. BURTON, Simon, Mr. EMSLIE, Richard* At EWT we develop systems that trade stocks on the electronic exchanges. The idea is that a computer can augment a human trader's reflexes by responding to market movements on the millisecond timeframe. This year we made the transition from python to rpython. We found that not only is our turnaround much faster but the code itself is able to be cleaned up as performance critical hacks are not needed anymore. RPython: it's here, it's useable in real world situations, and it really rocks. RPython feels like writing C (C++/C#/Java) code in python. Since it is almost entirely a subset of python it's like being able to run dangerous C code in the safe confines of a python interpreter. RPython is also a powerful lint checker: many bugs are now caught at compile time. As well as some info on EWT and the general rpython experience I will talk about * the transition process, ideas for how to migrate a python system to rpython, when not to, and debugging. * Also, we will mention the to-be-released RIO: a high performance rpython buffer/networking library based on Java's NIO. * finally, we present some examples of using ctypes to interface to external libraries, including embedding python itself, and also using cairo and libsdl for graphics.
10:00	**[62] The Essentials of Stackless Python (00h30')** *Speaker: TISMER, Christian* This is a re-worked, actualized and improved version of my talk at PyCon 2007. Repeating the abstract: As a surprise for people who think they know Stackless, we present the new Stackless implementation For PyPy, which has led to a significant amount of new insight about parallel programming and its possible implementations. We will isolate the known Stackless as a special case of a general concept. This is a Stackless, not a PyPy talk. But the insights presented here would not exist without PyPy's existance.

Web Related Technologies - Zeta (10 July 09:00-10:30)

time	[id] title
09:00	**[6] Grok: an introduction (ME GROK SAY HI) (00h30')** *Speaker: FAASSEN, Martijn* This talk gives an introduction to the Grok web application framework. The framework will be introduced and we will go into the design concepts behind it. We will then set upon a difficult task: to try to convince the audience that Grok is interesting, even among all the other Python-based web frameworks already out there. Grok combines ease of use with the power of Zope 3. Grok explicitly aims at a beginner audience, and no previous experience with Zope is required. A smooth growth curve then leads the developer towards untapping the full power of Zope 3. This way, Grok also should please the experts. Grok: now even cavemen can use Zope 3. And everybody, after all, descends from cavemen.
09:30	**[38] Bebop Protocols (00h30')** *Speaker: Dr. OESTERMEIER, Uwe* Zope3 has been criticized as an overly complex framework with a steep learning curve. Especially the ZCML configuration language and the missing Python API for configuration actions seems to be an obstacle for Python programmers. The talk introduces bebop.protocol, an experimental package that tries to combine the conciseness of Python with the explicitness, fine-grained configurability, and conflict management of ZCML. A protocol is a Python class that defines how a component is configured, registered, called, and unregistered. Protocols are used and extended by declarations, i.e. class advisors and decorators that correspond to existing ZCML directives. All declarations within a package can be activated with a single line of ZCML. The equivalent ZCML configuration can be recorded for documentary purposes and used as a basis for more selective configurations and overrides. In comparison to Grok, which tries to simplify Zope3 by using conventions instead of ZCML, Bebop favors a less radical approach. Grok smashes ZCML, Bebop generates ZCML. Since the protocol package mimics the ZCML directives as closely as possible it provides no extra learning curve for the experienced Zope3 programmer. Predefined protocols are available for adapters, utilities, subscribers, pages, and menus. Since protocols are extensible, they can also be used to define generic functions and extend the component architecture with special forms of utilities and adapter lookup. The zope.fssync package is used as an example that illustrates how existing code could benefit from protocols. Relationships to PEP 3124 and other proposals are also discussed. Bebop Protocols: http://svn.kmrc.de/projects/devel/bebop.protocol/trunk/src/bebop/protocol/README.txt

Web Related Technologies - Alpha (10 July 09:00-10:00)

time	[id] title
09:00	**[5] Python + .NET = IronPython (00h30')** *Speaker: Mr. BECHYNSKY, Stepan* IronPython is a new implementation of the Python programming language running on .NET. It supports an interactive console with fully dynamic compilation. It is well integrated with the rest of the .NET Framework and makes all .NET libraries easily available to Python programmers, while maintaining full compatibility with the Python language. You will see samples, how to use and run Python in ASP.NET world. You will not see PowerPoint presentation :-)

09:30	**[10] Silverlight and Python (00h30')** *Speaker: Mr. BECHYNSKY, Stepan* Microsoft® SilverlightTM is a cross-browser, cross-platform plug-in for delivering the next generation of .NET based media experiences and rich interactive applications for the Web. Hmmm, nice, but can I use Python as language for application logic? Yes, you can. Silverlight 1.1 brings support for dynamic languages (DLR) includes Python and Ruby. You will see demo how to write video player using Silverlight and Python and you will learn basics of XAML.

Python Language and Libraries - Alpha (10 July 10:00-10:30)

time	[id] title
10:00	**[32] Applications of python win32console module - a merging of character-mode applications and graphical user interfaces. (00h30')** *Speaker: Mr. GRAZ, Michael* With the recent addition of the win32console module in python it is now possible to create GUI applications that have embedded console sessions. The win32console module is a wrapper around the Microsoft Windows console API which is the operating system component that enables character-mode applications to display data to a windows console. The most typical console mode application on Windows is the cmd.exe command line interpreter. By utilizing the capabilities of the win32console module, it is now possible to embed and enhance routine command line/shell interactions by the use of GUI applications. Examples of embedding cmd.exe sessions would be: - wxPython based shell acting as a console replacement - Divmod Nevow web pages using "server-push/comet" technologies as an interface around a remote console session - cmd.exe session running in a Vim buffer It is not only cmd.exe that can be embedded but any other character-mode applications such as a Cygwin shell or even the python or iPython interpreters. The basic process flow is described as: - Client process (1) such as wxPython or Vim editor starts an intermediary control process (2). - The intermediary control process (2) is a child python process which imports win32console and allocates its own private console. - This control process (2) then starts the target character-mode application (3) such as cmd.exe or python.exe. - The control process (2) installs WinEventHook functions which are triggered whenever the target process (3) or any child processes of the target process write text to the console of the control process. - The control process (2) traps the console output and then relays it the client process (1) by means of shared memory and event synchronization. - The client process (1) can also generate command line input via shared memory to the control process (2) which then writes it into the input buffer of the target process (3). From the perspective of the wrapped console process (cmd.exe or python.exe interpreter for example) there is nothing to distinguish its input as coming from an interprocess communication mechanism versus a person typing on a keyboard. Likewise there is nothing different in the output statements of the wrapped console process since it is just doing standard console output. The use of the win32console module wrapping the Windows console API simplifies interaction with the console processes since they will be running in their native command-line mode with no modifications required. Technical discussion with slides and demonstrations will be presented.

Business and Applications - Lambda (10 July 11:00-12:00)

time	[id] title
11:00	**[11] Python Development Case Study: Enso Autocomplete (00h30')** *Speaker: Mr. DICARLO, Jonathan* Enso is a user-interface enhancement product created by Humanized (http://www.humanized.com) and is somewhat unusual in that it's an application written almost entirely in Python meant to be downloaded and installed locally by non-technical users. This spring, driven by user feedback, Humanized began upgrading Enso with a new auto-completion algorithm. This project presented several major challenges in UI design, algorithm design, performance, and testing. In this talk, I will present a case study of these challenges and how we overcame them. Specific topics of interest to Python developers will include: UI design: how to create an auto-completion algorithm (i.e, one that correctly guesses the user's intended command from a minimum number of keystrokes) that is efficient, learnable, and respects the user's habits. Development and deployment methodology: How Python enabled an agile cycle of quick prototyping and integration, limited beta release, and rapid user feedback, and how user input drove our process. Performance: algorithmic efficiency, how to achieve acceptable response times in Python, the profiling tools we used to identify hot-spots, and the techniques we used to defeat them.
11:30	**[46] Python in a large commercial application (00h30')** *Speaker: Dr. GRISBY, Duncan* In this presentation, I will talk about our experience with using Python as the main programming language in the development of Tideway Foundation, a product that helps large enterprises understand and manage the truth about their complex IT environments. Foundation automatically generates dependency information about hardware and software within an environment, involving accessing target machines with a wide range of techniques, working out what the information retrieved means, and storing it in a highly-interconnected object database. The presentation will cover areas in which Python has served us well, and areas where we have encountered both technical and non-technical limitations with Python.

Business and Applications - Alpha (10 July 11:00-11:30)

time	[id] title
11:00	**[41] How to do an EU open source research project (00h30')** *Speakers: KREKEL, holger, WAGNER, Lene* We'll look at how to do an EU research project, based on experiences obtained through the PyPy project. We quickly walk through the basics of the initial proposal, negotiations, the EU funding contract, cost/funding models and how we modified and amended the contract afterwards. The PyPy project has been one of the first bigger open source research projects that received funding from the EU - a liaison of diverging cultures, as it turned out. We had to learn how to manage this project: track/report results, budget, 'resources' to match our contractual obligations, but also make the EU contract and circumstances fit our needs and actual project developments. We'll highlight how the finalisation process with the EU (still pending for 31st May and following) worked out, summarize our findings and give some recommendations maybe helpful for other projects and organisations applying for EU Research funding.

Games - Theta (10 July 11:00-11:30)

time	[id] title
11:00	**[14] Taking advantage of multiple CPUs for games - simply. (00h30')** *Speaker: Mr. DUDFIELD, Rene* Taking advantage of multiple CPUs for games --- simply, is the topic of this paper. Using a simple interface that many are already familiar with --- Pythons 'map' function. It talks about how to avoid the Global Interpreter Lock (GIL) limitation in Pythons threading libraries. As well as how to choose which parts of a game to thread. Finally it shows how easy it can be, by converting a full pygame to make use of multiple CPUs --- and benchmarking the game.

Open Space - Zeta (10 July 11:00-12:30)

time	[id] title
11:00	**[82] Open Space (01h30')**

Agile Experiences and Testing - Alpha (10 July 11:30-12:30)

time	[id] title
11:30	**[43] PyPy: Why and how did it (not) work? (00h30')** *Speakers: Mrs. DURING, Beatric, Mr. KREKEL, Holger* Some people have said that they have hardly seen a large project succeed with delivering to the original goals as much as the PyPy project did. Others consider it a failure because it did not take over the world yet or is not usable for mainstream purposes. Anyway, we'll talk about the mix of development processes, methods and infrastructure of the PyPy project. We will summarize our experiences regarding the evolving development environment (version control, automated test-driven development, sprints, synchronization, "conceptual integrity", communication channels). We will also try to share our "learning by burnings" so that other projects can explore other creative problems and mistakes than maybe repeating the ones we explored in depth already during the last 4 years of PyPy development.
12:00	**[90] Sprinting offline with bzr, dbus and avahi (00h30')** *Speaker: Mr. COLLINS, Robert*

Python Language and Libraries - Theta (10 July 11:30-12:30)

time	[id] title
11:30	**[28] Integrating Python and TeX: MathTran and beyond (00h30')** *Speaker: Dr. FINE, Jonathan* This talk describes the MathTran system for translating mathematics from TeX to MathML and vice versa, and its use of TeX as a daemon. It surveys related Python and TeX software, and calls for the creation of standard Python library modules as a means of unifying and simplifying these projects. Finally, it shows how Python can be used to script TeX typesetting and use TeX as a callable function.

time	[id] title
12:00	**[26] Python Data-Driven Parsing For The Real World (00h30')** *Speaker: Dr. STOVALL, Johnny* Data-driven parsers have been used in AI and inductive reasoning in ways beyond the abilities of rule-driven parsers. This paper presents an annotated bibliography and guidelines for developing multipurpose data-driven parsers in hopes that developers will assist the author in disaster recovery and OLPC development, will push Free Open Software to new heights, and will benefit from commercial opportunities. Data-driven parsers of the highest types don't require a predetermined set of rules like rule-driven parsers do. This is most like discovery learning that maybe helps children to learn their first language so easily and fluently. The author contends this same type of learning is necessary for natural responses to a wide variety of real world situations. The data drives the progress and in the higher types can change goals just like it real life. Rule driven parsers are primarily useful in closed languages such as computer programs where the parser developer can know all of the rules before starting. Declarative programming will also be briefly described. This is an underused part of Python's flexibility but the basis for some data-driven parsers.

Games - Lambda (10 July 12:00-12:30)

time	[id] title
12:00	**[52] Pyweek: Making games in 7 days (00h30')** *Speakers: CURA, Alejandro J., TORRE, Lucio* The objective of this talk is to encourage people to participate in pyweek. We explain what the contest is, when it takes place, who can and does participate and why everybody should. A quick review of the most unique games is given, plus an account of the fun experience we had developing our entries. We also talk about how python is the perfect match for this kind of tight schedules, and about what the contest brings back into the python community.

Agile Workshops - Alpha (10 July 14:00-15:30)

time	[id] title
14:00	**[68] Arlo Belshee: Safe Is For Weenies (00h30')** *Speaker: Mr. BELSHEE, Arlo*
14:30	**[69] Arlo Belshee: Agile Adoption (00h30')** *Speaker: Mr. BELSHEE, Arlo*
15:00	**[70] Arlo Belshee: Appraising The Loot (00h30')** *Speaker: Mr. BELSHEE, Arlo*

Business and Applications - Lambda (10 July 14:00-15:30)

time	[id] title
14:00	**[22] If you can't beat them ... Pythonic explorations of Microsoft Sharepoint Portal Services (00h30')** *Speaker: Mr. MURRE, Jan* The majority of office workers use ... Microsoft Office! Especially in intranet enviroments. Things, especially versioning and tracking of changes, can go horribly wrong when people start working together, sending eachother Office documents. To accomodate this problem, Microsoft came up with Sharepoint. A full-blow Sharepoint implementation can get horribly expensive. However, the basic functionality of Sharepoint is a free addition to Windows Server 2003 under the name Sharepoint Portal Services. During this talk a general introduction on Sharepoint will be given, followed by an exploration of the way the Sharepoint Services can be leveraged from the python world.
14:30	**[50] py2exe, dbimport - ways of Python distribution on Windows (00h30')** *Speaker: Mr. MASSA, Harald Armin* Windows-Users are generally accustomed to "point, click, destroy" installation. Distributing .py files and installing the interpreter is often no-go. We will look into some samples of the great tool py2exe. dbimport is some lines of code to update Python applications within closed environments.
15:00	**[27] Managing Source Code with Bazaar (00h30')** *Speaker: Mr. HUDSON, Michael* Bazaar is a safe, friendly, free and fast distributed version control system. (will add more here soon)

Open Space - Zeta (10 July 14:00-15:30)

time	[id] title
14:00	**[83] Open Space (01h30')**

Python Language and Libraries - Theta (10 July 14:00-14:30)

time	[id] title
14:00	**[63] Streaming with Python, Twisted and GStreamer (00h30')** *Speaker: Mr. VANDER STICHELE, Thomas* Flumotion is a GPL streaming media server written in Python. It is distributed and component-based: every step in the streaming process (production, conversion, consumption) can be run inside a separate process on separate machines. Flumotion uses Twisted and GStreamer. Twisted enables the high-level functionality, distributing components over the network. GStreamer, through the Python bindings, enables the high-speed low-level functionality: actual media processing. Flumotion uses a central manager process to control the complete network; one or more worker processes distributed over machines to run actual streaming components; and one or more admin clients connecting to the manager to control it. Flumotion is under very active development. In its latest stable release (0.4.2), it already supports the following features: - various sources: webcams, soundcards, TV cards, Firewire cameras, looper - various codecs: Vorbis, Theora, mulaw, JPEG, smoke - various containers: Multipart, Ogg - synchronized capturing across machines - username/password authentication - overlaying, colorbalance - HTTP streaming, disk archiving - administration GUI with a wizard for the most basic scenario - ncurses-based administration - code distribution from a central location - local caching of the distributed code space - strong focus on ease of use and usability - improved support for network failure and reconnection - multiplex all component feeds through only one port on the worker - sharing HTTP port among streamers - support for GStreamer 0.10 The current design allows for the following future features: - any number of sources/containers/codecs/effects/protocols to be added - completely centralized code upgrades - complete code upgrades with minimal downtime - any kind of authentication mechanism (key exchange, challenge/response, ...) - any number of possible scenarios for actual content production and distribution - manager failover and state replication - loadbalancing streams over different servers or from different locations - internal stresstests Some of the features are only possible or easily implementable thanks to using a high-level language like Python: - sending GUI code for the wizard and component administration from the manager to the admin client, making the admin GUI a lightweight shell - sending component code to any of the workers over the wire at startup - rebuilding modules and reloading code on the fly while running; allowing for a distributed code upgrade without losing clients - rapid development of new components, allowing to catch up with and eventually to keep one step ahead of the competition - easy networking code thanks to Twisted In this project, we came up with solutions to specific problems presented to us that would be interesting to share with others. - all code is stored centrally and partitioned into "bundles" which are cached by clients who need them. Versioning and dependencies are correctly handled, and to the code being run this is handled transparently. Code can still import as if it were one big file tree. - the manager sends GUI code to an admin client and component code to a worker. The GUI code running on the admin machine then controls the behaviour of code running on the worker machine by going through the manager machine - state of components is automatically replicated one level deep to the manager, and two levels deep to all connected admin clients - authentication of any service in the network is handled by creating keycards which can be exchanged between all processes, implementing any type of challenge/response authentication as securely as possible. - Twisted's Perspective Broker was extended to use this generic keycard concept instead of the current (limited) username/password credentials. - the open-ended nature of Twisted and GStreamer is difficult to harness into a usable GUI. The wizard provides a good way of crystallizing all possibilities into a sensible task-based presentation. The design of the wizard also incorporates the flexibility of the network distribution and the dynamic code distribution by pulling in the necessary GUI code for the next step based on previous choices. - moving to run-time checks of functionality as opposed to compile/configure-time. Since the server can be distributed over any number of machines, and actual components have different run-time needs, all checks for features (devices, required libraries, versions, permissions) Python also presents some specific challenges when used in a large project as compared to more low-level languages. When handled right, these can actually be turned into project engineering advantages. Python's weak argument typing forces developers to document the API correctly. Python's dynamic nature (running code from received chunks, extensive subclassing) and Flumotion's design (componentized functionality, lots of small pieces of code sent back and forth) forces us to aggressively write unit testing for all functionality.

Web Related Technologies - Theta (10 July 14:30-15:00)

time [id] title

14:30	**[8] Nevow Developer (00h30')** *Speaker: Mr. VOLONGHI, Valentino* I'd like to show how I wrote my starter kit [1], how it works and what you can do with it. Nevow is sometimes a bit hard when it comes to starting a new application, especially because some details like authentication, pages organization, configuration management and deployment are not really part of what nevow provides. In my experience with many different applications like stiq [2], wirc [3] and many others that are closed source, I found that a framework architecture for a full web stack was emerging. I extracted that structure into yet another full web stack. My main objective is to show how each piece is architected and how it is connected with the rest of the system. I'll also show how I (and anyone could) wrote something cool like wirc (irc from the web) using this starter kit. [1]: http://hg.stiq.it/starter [2]: http://hg.stiq.it/stiq [3]: http://hg.stiq.it/wirc

Science - Theta (10 July 15:00-15:30)

time	[id] title
15:00	**[2] Managing and displaying user track data with Python (00h30')** *Speaker: Mr. APRILE, Walter* User studies that feature user movement in the real world or in simulated environment generate datasets, usually in the form of logfiles, that need to be stored, summarized, processed and represented. Datasets must additionally include metadata that accounts for experimental conditions. We have developed a class that produces graphical displays of user travels over a regularly-spaced grid, and a set of web-controllable database management tools that allow incremental data exploration as the user experiments progress.

Lightning Talks - Alpha (10 July 16:00-17:00)

time	[id] title
16:00	**[86] Lightning Talks (01h00')**

Keynotes - Alpha (10 July 17:15-18:15)

time	[id] title
17:15	**[71] Keynote by Guido van Rossum (01h00')** *Speaker: Mr. VAN ROSSUM, Guido*

Social events (10 July 20:00-23:59)

time	[id] title
20:00	**[91] Conference dinner (03h59')**

Wednesday 11 July 2007

Agile Workshops - Alpha (11 July 09:00-10:30)

time	[id] title
09:00	**[73] Arlo Belshee: Promiscuous Pairing (00h30')**
09:30	**[74] Arlo Belshee: The XP Sprint (01h00')**

Web Related Technologies - Lambda (11 July 09:00-10:30)

time	[id] title
09:00	**[24] The Zope Foundation (00h30')** *Speaker: FAASSEN, Martijn* The Zope Foundation was incorporated last year. Its aim is to take over the whole intellectual property "Zope" from Zope Corporation and lead Zope's further development as OpenSource software, but now owned by a non-profit organisation. This talk is about the future of Zope from the point of view of the Zope Foundation. Two members of the ZF Board of Directors, Martijn Faassen and Aroldo Souza-Leite, will be present.
09:30	**[31] What Zope did wrong (and what to do instead) (01h00')** *Speaker: Mr. REGEBRO, Lennart* Zope was an early web framework, and one of the first complete frameworks to include such things as persistance, security management and extensibility. It was an early open source adopter and was ar ahead of it's time. So why didn't it take over the world? Why didn't it even take over the Python world? This presentation will look at the mistakes of Zope 2 and primarily Zope 3, and where we should head forward with Python web frameworks in the future. Zope 2: The monolithic lock-in * Why is Zope 2 so monolithic? * No forward path from TTW development * Complexities galore, and the ever increasing learning-curve Zope 3: Principles before practicality * How the momentum was lost * Unclear signals * Backwards, forwards or no compatibility? * Death by abstraction * Try to be all things to all people, and you will be nothing at all The future: * How should the "perfect" web framework look Talk length 30-60 minutes Lennart Regebro has been using Python and Zope since 1999, and been a full time Zope developer since 2001, and a Zope contributor since 2002, and been one of the developers of Five, the piece that brings Zope3 technology into Zope2. He has been a part of the design and development of three different zope-based content management systems, lastly at Nuxeo, where he helped design and develop the very successful CPS3 ECMS. He is now an independent developer in Paris, France.

Web Related Technologies - Zeta (11 July 09:00-10:30)

time	[id] title
09:00	**[25] Introduction to Web Programming with WSGI (00h30')** *Speaker: Dr. SIMIONATO, Michele* As the old saying goes, Python is the only language with more Web frameworks than keywords. This is sometimes an advantage, but more often than not, it is an issue. In order to improve the interoperability between different frameworks, Phillip J. Eby proposed in 2003 a specification, the WSGI or Web Server Gateway Interface, a.k.a Whiskey. In my talk I will discuss how you can achieve portability of your application by following the WSGI specification. I will give practical examples of how to use the WSGI reference implementation which is part of the standard library (wsgiref), of how to supplement it with other WSGI-compliant libraries (notably Paste by Ian Bicking) and of how to integrate your WSGI application in different frameworks including Zope, Twisted, TurboGears et al. The talk is intended as a tutorial and requires only elementary knowledge of Web programming, at the level of simple CGI.
09:30	**[55] Using Genshi to Produce Markup for the Web (01h00')** *Speaker: Mr. LENZ, Christopher* Genshi (http://genshi.edgewall.org/) is a relatively new toolkit aimed at producing output for the web. It's focus is the generation of markup, in particular X/HTML, which includes a template language inspired by Kid. In this talk I will present the advantages of using Genshi for templating, such as automatic escaping, solid error handling, and different serialization formats. Also, I'll discuss the various features Genshi provides for working with templates and markup in general, such as HTML tag soup parsing and sanitization, form filling, programmatic markup generation, and internationalization support.

Web Related Technologies - Theta (11 July 09:00-10:30)

time [id] title

09:00	**[1] Developing an Internationalized Application in Python: Chandler a case study (00h30')** *Speaker: Mr. KIRSCH, Brian* Intended Audience == This talk is intended for experienced Python programmers interested in developing Internationalized Applications using Python and Open Source libraries. The audience is expected to be familiar with some basic Internationalization concepts, as the presentation is not a full Unicode / i18n tutorial. Overview == Internationalization is the most often overlooked aspect of Application development. It is a mistaken belief that Internationalization can easily be added at anytime. This mistake ultimately results in developers frantically scrambling to patch together a solution for an architecture which was never designed for it. Too many products end up in a rewrite when the team finally discovers just how fundamental a role Internationalization plays. This talk will cover general concepts on how to design Internationalized Applications, as well as focus in depth on the specific choices made for Chandler including leveraging Open Source libraries and designing for multiple Operating Systems. About Chandler =============== Chandler is an innovative Open Source Personal Information Manager (PIM). In addition to being written in Python, Chandler uses the following Open Source libraries: BerkeleyDB, M2Crypto, Twisted, pyLucene, PyICU, and wxPython/wxWidgets. Chandler is designed to be an extensible PIM. Chandler's unit of extensiblity is called a parcel, and Chandler's "built-in" functionality is itself composed of parcels. Internally, Chandler is designed as layers of frameworks which provide applications functionality to parcels. Parcels communicate with each other via the data in the Chandler repository. For more information about the product please visit: http://wiki.osafoundation.org/Projects/ChandlerHome
09:30	**[12] Managing a multilingual decentralised internet network (00h30')** *Speaker: Mr. MORAL, Gorka* The European Agency for Safety and Health at Work has developed, in cooperation with Syslab.com, an open source based Content Management System, to manage its international network of websites, composed of more than 50 websites, hosted centrally at its premises in Bilbao, Spain. This CMS allows the Agency to manage the workflow of hundreds of editors; to manage, through the web, all the network websites; to personalise the information showed, according to user's interests; to syndicate contents in an easy way; to tag all the information through an integrated thesaurus; to publish hundreds of publications in PDF format; to share experiences and information with its network partners through 2.0 technologies, such as wikies and blogs; or to translate its pages into the EU official languages. The Agency was awarded in 2005, in the category of Innovation in Content Management Systems, in the International Information Industry Awards, and plans to release this software solution before the end of the year, for public use.
10:00	**[13] Technical issues of a multilingual decentralised internet network (00h30')** *Speaker: Mr. PILZ, Alexander* The European Agency for Safety and Health at Work has developed, in cooperation with Syslab.com, an open source based Content Management System, to manage its international network of websites, composed of more than 50 websites, hosted centrally at its premises in Bilbao, Spain. Besides the usual tasks to operate a CMS, this system has some specific challenges. It is bad to change content because you need to do it in more than 22 languages. It is bad to rely on folders to structure the site, because one piece of information may be needed in several places. It is bad to add another portal for another campaign because that duplicates a lot. Current efforts include thoughts about radical reduction of manual created pages, utilizsation of specialized structuring objects like subsites and multi-tenant-features and shared resources. Personalisation is used in a way that information can be selected based on a centrally managed list of "areas of interest" which consist of a combination of several metadata each.

Agile Workshops - Alpha (11 July 11:00-12:30)

time	[id] title
11:00	**[75] Arlo Belshee: The XP Sprint (01h00')**

Open Space - Lambda (11 July 11:00-12:30)

time	[id] title
11:00	**[84] Open Space (01h30')**

Open Space - Zeta (11 July 11:00-12:30)

time	[id] title
11:00	**[85] Open Space (01h30')**

Web Related Technologies - Theta (11 July 11:00-12:00)

time	[id] title
11:00	**[15] Batching APIs, as applied to web applications. (00h30')** *Speaker: Mr. DUDFIELD, Rene* Batching APIs, as applied to web applications, is what this paper is about. It explains what a batching API is, with it's advantages, and limitations. Then it shows how to reduce latency for large amounts of content --- by combining things together. As well as explaining the reasons for the latency, it also discusses what advantages presenting 10x more content at the same speed can offer. One technique this paper discusses is packing images. Packing all of the images on a page into one big image - with efficient algorithms, then using CSS to place the image onto the page correctly.
11:30	**[29] Measuring Web Services (00h30')** *Speaker: Mr. QUINLAN, Brian* Web 2.0 (whatever that means) is here and along with it has come an increased need for high-performance web services. In this talk, I'd to discuss some ideas on how to measure the utilization and performance of Python-implemented web-services in order to: - plan capacity - find performance hotspots - decide when to run around like a decapitated chicken

Python Language and Libraries - Theta (11 July 12:00-12:30)

time	[id] title
12:00	**[59] Measuring Python Performance (00h30')** *Speaker: DALKE, Andrew Dalke* "Make it work, ... then make it fast". "If you can't measure it, it doesn't exist." Both useful adages, but how do you measure the performance of a Python program and identify bottlenecks? In my talk I'll start with generating simple timing numbers and how to interpret the results. I'll show how to use Python's profiler and convert the results for kcachegrind, a KDE profile visualization tool. Sometimes function-call level information is too detailed or otherwise inappropriate so I'll show how to instrument and create higher level traces for kcachegrind. I used all of these techniques while consulting for AstraZeneca's R group. I'll base many of my examples on that experience, and describe some of ways to improve overall performance.

Keynotes - Alpha (11 July 14:00-15:00)

time	[id] title
14:00	**[72] Keynote by David Axmark (01h00')** *Speaker: Mr. AXMARK, David*

Future Of EuroPython Plenary Session - Alpha (11 July 15:15-15:45)

time	[id] title
15:15	**[92] Future of EuroPython Plenary Session (00h30')**

Lightning Talks - Alpha (11 July 16:00-17:45)

time	[id] title
16:00	**[88] Lightning Talks (01h45')**