Python, Zope and Plone

or: "How I Learned to Stop Worrying and Love the Bomb"

Georg Gogo BERNHARD
gogo@bluedynamics.com

What is Python

Python appeared in 1991
Designed by Guido van Rossum

Python is an interpreted, general-purpose high-level programming language whose design philosophy emphasizes code readability.

- Supports multiple programming paradigms
- Object oriented
  Imperative
- Functional
- Fully dynamic type system
- Automatic memory management (similar to Scheme, Ruby, Perl, and Tcl)
- Use of indentation for block delimiters
- Capable of exception handling
- Reference implementation (CPython) is free and open source software, community-based development model (managed by the non-profit Python Software Foundation)
- Interpreters available for many operating systems
- very clear, readable syntax
- strong introspection capabilities
- intuitive object orientation
- natural expression of procedural code
- full modularity, supporting hierarchical packages
- exception-based error handling
- very high level dynamic data types
- extensive standard libraries and third party modules for virtually every task
- extensions and modules easily written in C, C++ (or Java for Jython, or .NET languages for IronPython)
- embeddable within applications as a scripting interface

Python is powerful... and fast

- Standard Library (everything from asynchronous processing to zip files, web server)
- Powerful and dynamic introspection capabilities
- Meta-Classes, Duck Typing and Decorators
- Easy to write
- Byte-Compiled, fast enough

Python plays well with others

- Python can integrate with COM, .NET, and CORBA objects
- Jython, an implementation of Python for the Java Virtual Machine
- IronPython , Microsoft's new implementation of Python for .NET
- Internet Communications Engine (ICE) and many other integration technologies.

If you find something that Python cannot do, or if you need the performance advantage of low-level code, you can write extension modules in C or C++, or wrap existing code with SWIG or Boost.Python. Wrapped modules appear to your program exactly like native Python code. That's language integration made easy. You can also go the opposite route and embed Python in your own application, providing your users with a language they'll enjoy using.

Python runs everywhere

Python is available for all major operating systems: Windows, Linux/Unix, OS/2, Mac, Amiga, among others. There are even versions that run on .NET, the Java virtual machine, and Nokia Series 60 cell phones. You'll be pleased to know that the same source code will run unchanged across all implementations.

Your favorite system isn't listed here? It may still support Python if there's a C compiler for it. Ask around on news:comp.lang.python - or just try compiling Python yourself.
Python is friendly... and easy to learn

The Python newsgroup is known as one of the friendliest around. The avid developer and user community maintains a wiki, hosts international and local conferences, runs development sprints, and contributes to online code repositories.

Python also comes with complete documentation, both integrated into the language and as separate web pages. Online tutorials target both the seasoned programmer and the newcomer. All are designed to make you productive quickly. The availability of first-rate books completes the learning package.

Python is Open

The Python implementation is under an open source license that makes it freely usable and distributable, even for commercial use. The Python license is administered by the Python Software Foundation.

Take a look at application domains where Python is used, or try the current download for yourself.

History

- Late 1980s: Conceived by Guido Van Rossum, CWI, Netherlands as successor to the ABC programming language, ABC inspired by SETL)
- 1989, December: Implementation started

- 2000 October 16: Python 2.0 released (full garbage collector , Unicode.), development process shifted to a more transparent and community-backed process
- 2008 December 3: Python 3.0 released (backwards-incompatible) , long testing, some features backported to Python 2.6 and 2.7.
- 2010: Winner of the TIOBE Programming Language Award for having the greatest programming language market share gain over the course of that year (+1.81%).

Guido van Rossum is Benevolent Dictator for Life (BDFL).

Programming philosophy

Python is a multi-paradigm programming language, it permits several styles:

- Object-Oriented Programming
- Structured Programming
- Functional Programming
- Aspect-Oriented Programming (Metaprogramming, Magic Methods)
- Extensions for Design by Contract

- Sparse, less-cluttered grammar
- "beautiful", "explicit" and "simple"
- Dynamic typing
- Garbage collector (reference counting and cycle-detecting)
- Dynamic name resolution (late binding) during execution.
- Small core

- Large standard library
- Highly extensible
- New built-in modules can be written in C, C++ or Cython
- Python can also be used as an programmable interface for applications
- "there should be one - and preferably only one - obvious way to do it".

Reject patches to non-critical parts of CPython which would offer a marginal increase in speed at the cost of clarity

- Python is sometimes described as "slow"
- Most problems and sections of programs are not speed critical
- Try using a JIT compiler such as Psyco
- Rewrite the time-critical functions in "closer to the metal" languages such as C
- Translatie (a dialect of) Python code to C code using tools like Cython.

The core philosophy of the language is summarized by the document "PEP 20 (The Zen of Python)".

```
>>> import this
The Zen of Python, by Tim Peters

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!
```

Name and neologisms

- Python’s name is based on the television series Monty Python's Flying Circus
- Metasyntactic variables “foo” and “bar” often replaced by “spam” and “eggs” in tutorials

- The word “pythonic” has a wide range of meanings related to program styles, it means that Python idioms are used well and natural to show the fluency in the language, or that code meshes well with the language.
- The word “unpythonic” describes attempts to write C++ (or Lisp, Perl, or Java) code in Python - that code is providing a rough transcription rather than an idiomatic translation of forms from another language
- The concept of “pythonicity” is tightly bound to Python's minimalist philosophy of readability and avoiding the "there's more than one way to do it" approach. Unreadable code or incomprehensible idioms are “unpythonic”.
- Users and admirers of Python - are often referred to as “Pythonists”, “Pythonistas,” and “Pythoneers”
- The prefix “Py” can be used to show that something is related to Python. Examples of the use of this prefix in names of Python applications or libraries include Pygame, PyQt, PyGTK, or PyPy. The prefix is also used outside of naming software packages: the major Python conference is named PyCon.

Usage

Scripting language for web applications

- Apache / mod_wsgi (Web Server Gateway Interface)
- Django
- Pylons
- TurboGears
- web2py
- Flask

- Zope

Scientific Computing

- NumPy
- SciPy
- Matplotlib

Embedded scripting language

- Finite element analysis (Abaqus)
- 3D rendering (Maya, MotionBuilder, Softimage, Cinema 4D, BodyPaint 3D, Modo, Blender, …
- 2D imaging programs (GIMP, Inkscape, Scribus, Paint Shop Pro, …)
- GNU GDB uses Python as a pretty printer to show complex structures such as C++ containers.
- ESRI is now promoting Python as the best choice for writing scripts in ArcGIS
- Python has even been used in several video games and has been adopted as one of the two available scripting languages in Google Docs

Operating systems

- Most Linux distributions (Ubuntu Ubiquity installer, Red Hat Linux/Fedora Anaconda installer, Gentoo Linux "emerge" package management system)
- Mac OS X

Information security industry (exploit development.)

Companies

- YouTube
- Original BitTorrent client
- Google
- Yahoo!
- CERN
- NASA
- Sugar software for the One Laptop per Child XO (Sugar Labs)

Syntax and semantics

- Highly readable language
- Uncluttered visual layout
- English keywords
- Less boilerplate
- Small number of syntactic exceptions and special cases
- Whitespace indentation

Statements and control flow

Python's statements include (among others)

- The **`if`** statement, which conditionally executes a block of code, along with **`else`** and **`elif`** (a contraction of else-if).
- The **`for`** statement, which iterates over an iterable object, capturing each element to a local variable for use by the attached block.
- The **`while`** statement, which executes a block of code as long as its condition is true.
- The **`try`** statement, which allows exceptions raised in its attached code block to be caught and handled by except clauses; it also ensures that clean-up code in a finally block will always be run regardless of how the block exits.
- The **`class`** statement, which executes a block of code and attaches its local namespace to a class, for use in object-oriented programming.
- The **`def`** statement, which defines a function or method.
- The **`with`** statement (from Python 2.6), which encloses a code block within a context manager (for example, acquiring a lock before the block of code is run, and releasing the lock afterwards).
- The **`pass`** statement, which serves as a NOP and can be used in place of a code block.
- The **`assert`** statement, used during debugging to check for conditions that ought to apply.
- The **`yield`** statement, which returns a value from a generator function. (From Python 2.5, **`yield`** is also an operator. This form is used to implement coroutines -- see below.)

Each statement has its own semantics: for example, the **`def`** statement does not execute its block immediately, unlike most other statements.

CPython does not support first-class continuations (and it never will). Support for coroutine-like functionality is provided in 2.5 (generators, it is possible to pass information back into a generator as of Python 2.5)

Expressions

Python expressions are similar to languages such as C and Java. Some important notes:

- In Python 2, the **`/`** operator on integers does integer division, i.e. it truncates the result to an integer. In Python 3, however, the result of **`/`** is always a floating-point value, and a new operator **`//`** is introduced to do integer division. In Python 2, this behavior can be enabled using the statement **`from __future__ import division`**.
- In Python, **`==`** compares by value, in contrast to Java, where it compares by reference (value comparisons in Java use the equals method). To compare by reference in Python, use the **`is`** operator.
- Python uses the words **`and`**, **`or`**, **`not`** for its boolean operators rather than the symbolic **`&&`**, **`||`**, **`!`**.
- Python has an important type of expression known as a list comprehension. Recent versions of Python have extended list comprehensions into a more general expression known as a generator expression.
- Anonymous functions are implemented using **`lambda`** expressions; however, these are limited in that the body can only be a single expression.
- Conditional expressions in Python are written as **`y if x else z`** (different in order of operands from the **`?:`** operator common to many other languages).
- Python makes a distinction between lists and tuples. Lists are written as **`[1, 2, 3]`**, are mutable, and cannot be used as the keys of dictionaries (dictionary keys must be

immutable in Python). Tuples are written as `(1, 2, 3)`, are immutable and thus can be used as the keys of dictionaries. The parentheses around the tuple are optional in some contexts. Tuples can appear on the left side of an equal sign; hence an expression like `x, y = y, x` can be used to swap two variables.

- Python 2 has a "string format" operator `%`. This functions analogous to printf expressions in C, e.g. `"foo=%s bar=%d" % ("blah", 2)` evaluates to `"foo=blah bar=2"`. In Python 3 this has been replaced by the `format` method in the string class, e.g. `"foo={0} bar={1}".format("blah", 2)`.
- Python has various kinds of strings.
- Either single or double quotes can be used to quote strings. Unlike in Unix shell languages, Perl or Perl-influenced languages such as Ruby or Groovy, single quotes and double quotes function identically, i.e. there is no string interpolation of `$foo` expressions.
- There are also multi-line strings, which begin and end with a series of three single or double quotes and function like here documents in shell languages, Perl and Ruby.
- Finally, all of the previously-mentioned string types come in "raw" varieties (denoted by placing a literal `r` before the opening quote), which do no backslash-interpolation and hence are very useful for regular expressions or Windows-style paths; compare "@-quoting" in C#.
- Python has slice expressions on lists, denoted as ...`[left:right]` or ...`[left:right:stride]`. For example, if the variable nums is assigned the list `[1, 3, 5, 7, 8, 13, 20]`, then the following expressions will evaluate `True`:
  `nums[2:5] == [5, 7, 8]`, i.e. the slice goes up to, but not including, the right index.
  `nums[1:] == [3, 5, 7, 8, 13, 20]`, i.e. all elements but the first, because an omitted right index means "the end".
  `nums[:-3] == [1, 3, 5, 7]`, i.e. an omitted left index means "the start", and a negative index (either left or right) counts from the end.
  `nums[:]` makes a copy of the list. This means `nums == nums[:]` is true but nums is `nums[:]` is false. Changes to the copy will not affect the original.
  `nums[1:5:2] == [3, 7]`, i.e. if three numbers are given, the third is called the "stride", indicating in this case that every second element will be selected.

In Python, a distinction between expressions and statements is rigidly enforced, in contrast to languages such as Common Lisp, Scheme or Ruby. This leads to some duplication of functionality, e.g.

- list comprehensions vs. "`for`" loops
- conditional expressions vs. "`if`" blocks
- The `eval()` vs. `exec()` builtins (in Python 2, `exec` is a statement declarator); `eval()` is for expressions, `exec()` is for statements.

Statements cannot be a part of an expression and so list and other comprehensions or `lambda` expressions, all being expressions, cannot contain statements. A particular case of this is that an assignment statement such as '`a = 1`' cannot form part of the conditional expression of a conditional statement; this has the advantage of avoiding a classic C error of mistaking an assignment token '`=`', for an equality operator '`==`' which would remain valid C in `if (c = 1) { ... }` but `if c = 1:` ... is invalid Python code.

Methods

Methods on objects are functions attached to the object's class; the syntax **`instance.method(argument)`** is, for normal methods and functions, syntactic sugar for **`Class.method(instance, argument)`**. Python methods have an explicit **`self`** parameter to access instance data, in contrast to the implicit self in some other object-oriented programming languages (for example, Java, C++ or Ruby).

Typing

Python uses duck typing and has typed objects but untyped variable names. Type constraints are not checked at compile time; rather, operations on an object may fail, signifying that the given object is not of a suitable type. Despite being dynamically typed, Python is strongly typed, forbidding operations that are not well-defined (for example, adding a number to a string) rather than silently attempting to make sense of them.

Python allows programmers to define their own types using classes, which are most often used for object-oriented programming. New instances of classes are constructed by calling the class (for example, **`SpamClass()`** or **`EggsClass()`**), and the classes themselves are instances of the metaclass type (itself an instance of itself), allowing metaprogramming and reflection.

Prior to version 3.0, Python had two kinds of classes: "old-style" and "new-style". Old-style classes were eliminated in Python 3.0, making all classes new-style. In versions between 2.2 and 3.0, both kinds of classes could be used. The syntax of both styles is the same, the difference being whether the class object is inherited from, directly or indirectly (all new-style classes inherit from **`object`** and are instances of type).

Here is a summary of Python's built-in types:

Type	Description	Syntax example
bool	An immutable truth value	True False
complex	An immutable complex number with real number and imaginary parts	3+2.7j
float	An immutable floating point number (system-defined precision)	3.1415927
int	An immutable fixed precision number of unlimited magnitude	42
dict	A mutable group of key and value pairs	{'key1': 1.0, 3: False}
tuple	Immutable, can contain mixed types	(4.0, 'string', True)
list	Mutable, can contain mixed types	[4.0, 'string', True]
bytes	An immutable sequence of bytes	b'Some ASCII'
str	An immutable sequence of characters. In Python 2, strings are a sequence of	'Wikipedia' "Wikipedia" """Spanning

	characters. Unicode strings need to be declared by prefixing with the letter u. In Python 3 strings are Unicode by default.	multiple lines"""

Mathematics

Python defines the modulus operator so that the result of a `% b` is in the open interval `[0,b)`, where `b` is a positive integer. (When `b` is negative, the result lies in the interval `(b,0]`). This is the usual way of defining the modulus operation in mathematics. However, this consequently affects how integer division is defined. To maintain the validity of the equation `b * (a // b) + a % b = a`, Integer division is defined to round towards minus infinity. Therefore `7 // 3` is `2`, but `(-7) // 3` is `-3`. This is different from many programming languages, where the result of integer division rounds towards zero and the modulus operator is consequently defined in a way which can return negative numbers.

Python provides a round function for rounding floats to integers. Version 2.6.1 and lower use round-away-from-zero: `round(0.5)` is `1.0`, `round(-0.5)` is `-1.0`. Version 3.0 and higher use round-to-even: `round(1.5)` is `2.0`, `round(2.5)` is `2.0`. The Decimal type/class in module decimal (since version 2.4) provides exact numerical representation and several rounding modes.

Python allows boolean expressions with multiple equality relations in a manner that is consistent with general usage in mathematics. For example, the expression `a < b < c` tests whether a is less than b and b is less than c. C-derived languages interpret this expression differently: in C, the expression would first evaluate `a < b`, resulting in `0` or `1`, and that result would then be compared with c.

Interpretational semantics

- Python acts as a shell
- IDLE and IPython (auto-completion, retention of session state, and syntax highlighting)
- PyPy can turn restricted subsets of Python code (RPython) into machine code
- Psyco is a specialising just in time compiler that integrates with CPython, transforms bytecode to machine code at runtime, Psyco is compatible with all Python code.

Development

- Python Enhancement Proposal (PEP, standardized design documents, proposals, descriptions, design rationales, and explanations for language features, reviewed by Van Rossum)
- python-dev CPython mailing list
- bug tracker python.org
- svn.python.org.

CPython's public releases come in three types, distinguished by which part of the version number is incremented:

- backwards-incompatible versions, where code is expected to break and must be manually ported. The first part of the version number is incremented. These releases happen infrequently - for example, version 3.0 was released 8 years after 2.0.
- major or 'feature' releases, which are largely compatible but introduce new features. The second part of the version number is incremented. These releases are scheduled to occur roughly every 18 months, and each major version is supported by bugfixes for several years after its release.
- bugfix releases, which introduce no new features but fix bugs. The third and final part of the version number is incremented. These releases are made whenever a sufficient number of bugs have been fixed upstream since the last release, or roughly every 3 months. Security vulnerabilities are also patched in bugfix releases.

Libraries

- Standard library providing pre-written tools, "batteries included", of standard formats and protocols (such as MIME and HTTP), graphical user interfaces, relational databases, arithmetic with arbitrary precision decimals, regular expressions, unit testing, doctest
- Boost.Python
- Powerful glue language between languages and tools.

Influences on other languages

Python's design and philosophy have influenced several programming languages, including:

- Pyrex and its derivative Cython are code translators that are targeted at writing fast C extensions for the CPython interpreter. The language is mostly Python with syntax extensions for C and C++ features. Both languages produce compilable C code as output
- Boo uses indentation, a similar syntax, and a similar object model. However, Boo uses static typing and is closely integrated with the .NET framework
- Cobra uses indentation and a similar syntax. Cobra's "Acknowledgements" document lists Python first among languages that influenced it. However, Cobra directly supports design-by-contract, unit tests and optional static typing
- ECMAScript borrowed iterators, generators, and list comprehensions from Python.
- Go is described as incorporating the "development speed of working in a dynamic language like Python"
- Groovy was motivated by the desire to bring the Python design philosophy to Java.
- OCaml has an optional syntax, called twt (The Whitespace Thing), inspired by Python and Haskell

What is Zope

For the fish known as "Zope", see Abramis ballerus.

Zope is a free and open source object-oriented web application server primarily written in the Python programming language.

- "Z Object Publishing Environment"
- First Web Object Publishing System for the web
- Recognized as a Python killer app
- Objects in a Zope Object Database, not files on a file system
- Zope maps URLs to objects using the containment hierarchy or traversal
- Methods are considered to be contained in their objects as well
- Data is stored in the ZODB (or in other databases or the file system)
- Object technologies, such as encapsulation
- Dynamic Template Markup Language “DTML” (tag-based simple scripting, variable inclusion, conditions, and loops; not well-formed XML)
- Zope Page Templates “ZPT”: TAL (Template Attribute Language), TALES (Template Attribute Language Expression Syntax), and METAL (Macro Expansion Template Attribute Language), namespaces, most logic implemented in Python code, can be edited in most graphical HTML editors, direct support for internationalization
- Zope underlies Plone and Silva content management systems, as well as the ERP5 open source enterprise resource planning system.

Zope 2 was releasd 1998

Zope 3 was released 2004

- BlueBream (earlier called Zope 3) is less widespread but underlies several large sites, including Launchpad.
- Grok was started as a more programmer-friendly framework, "Zope 3 for cavemen", and in 2009
- BFG gained popularity in the Zope community as a minimalistic framework based on Zope principles.

History

- 1995: Zope Corporation was formed in Fredericksburg, Virginia “Digital Creations” as joint venture with InfiNet (a joint newspaper chain venture).
- 1997: Zope Corp became an independently owned private company, CTO Jim Fulton

- 1998: Initial release of Zope
- 2000: PythonLabs (creators of Python) become part of the company
- 2003: Python founder Guido van Rossum leaves Zope Corp, Zope 2 began as “Principia application server” as a merge of Bobo, Document Templates and BoboPOS
- 2004, November 6: Zope 3 released (complete rewrite, not compatible with Zope 2), Backwards-compatibility layer “Five” for Zope 2 followed
- 2010, January: Zope 3 was renamed to “BlueBream”

Zope 3, BlueBream, Zope Toolkit, Grok

The existence of two incompatible web frameworks called Zope has caused a lot of confusion. In response Zope 3 was renamed to “BlueBream”.

Component Architecture that makes it easy to mix software components of various origins written in Python.

To make it clearer which developments are independently usable, and also to improve the re-usability of the packages the Zope Toolkit (ZTK) project was started.

Thus the Zope Toolkit works as the base where each of the Zope-world frameworks builds on. Zope 2.12 is the first release of a web framework that builds on Zope Toolkit, and Grok and BlueBream are set to have releases based on the ZTK during 2010.

In 2006 the Grok project was started by a number of Zope 3 developers who wanted to make Zope 3 technology more agile in use and more accessible to newcomers. Grok has since then seen regular releases and its core technology (Martian, grokcore.component) is also finding uptake in other Zope 3 and Zope 2 based projects.

Zope Page Templates

- Zope Page Templates are XHTML documents
- They can be viewed and edited using normal HTML editors or XHTML compliant tools (a big advantage compared to other template languages used for web applications)
- Templates can be checked for XHTML compliance
- Zope Page Templates are marked up with additional elements and attributes in special XML namespaces (see below) to describe how the page template should be processed.

Examples

To conditionally include a particular element, like a div element, simply add the tal:condition attribute to the element as follows:

```
<div tal:condition="...">
  ...
</div>

To control what appears inside an element, use the tal:content attribute
like this:

<h1><span tal:content="..."/></h1>
...

Finally, to introduce or replace values of attributes use the
tal:attributes attribute as follows: The power of python could also be
utilised to dynamically alter the href at runtime.

<a href="" tal:attributes="href
python:'http://someurl.com/%s'%someobject">...</a>
```

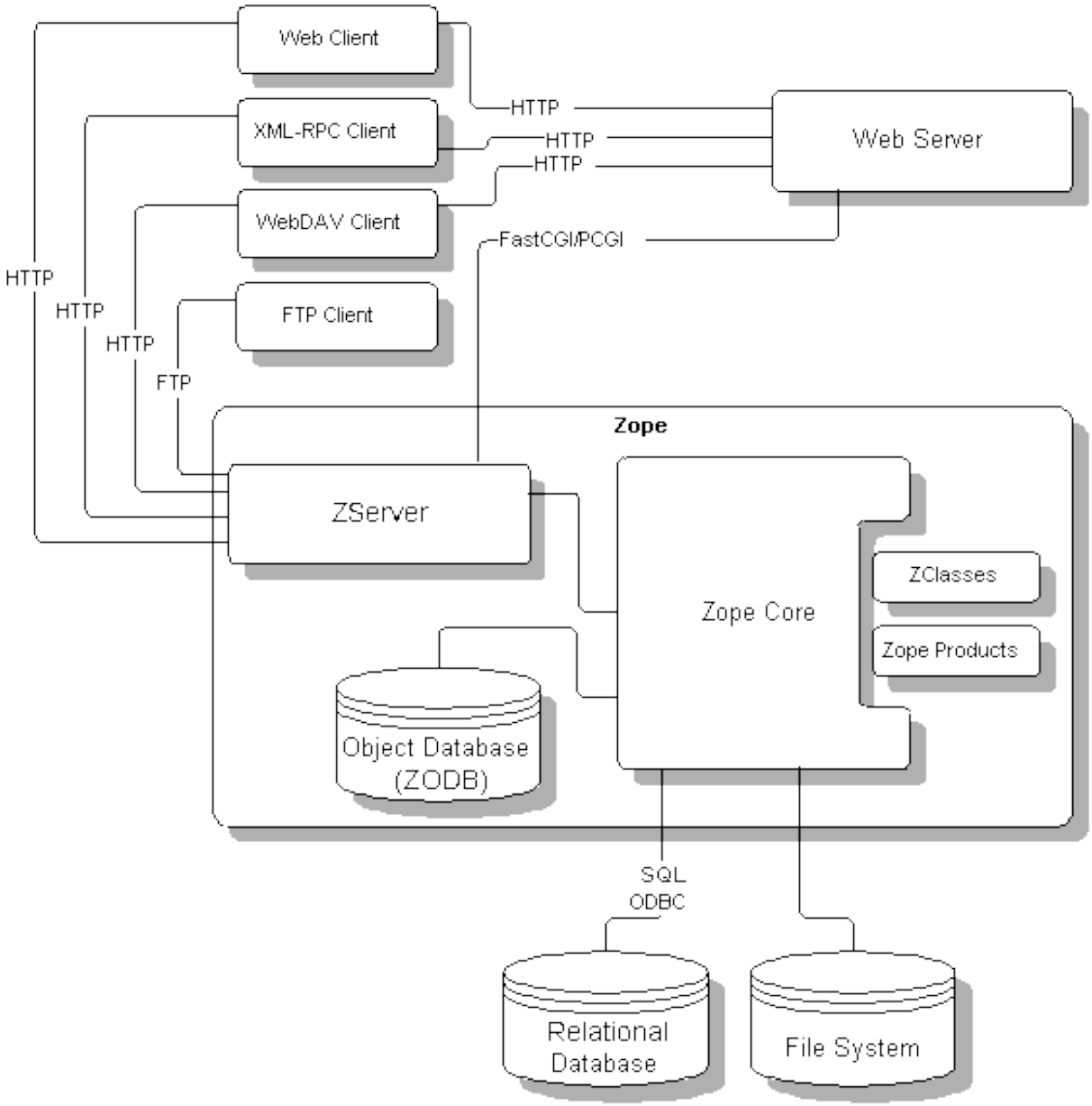
Features

Transactional object database (not only to Zope's object database, but to many relational database connectors as well, allowing for strong data integrity)

Transaction model happens automatically, ensuring that all data is successfully stored in connected data sources by the time a response is returned to a web browser or other client.
ZODB stores Content and Custom Data, Dynamic HTML Templates, Scripts
search engine, relational database (RDBMS) connections, code
Strong through-the-web development model
Tightly integrated security model
Concept of "safe delegation of control" allows you to turn control over parts of a web site to other organizations or individuals

Numerous products (plug-in Zope components): New content objects; advanced content management tools; and full applications for e-commerce, content and document management, or bug and issue tracking.
Zope includes its own HTTP, FTP, WebDAV, and XML-RPC serving capabilities
Integrates with Apache or other web servers.

Zope Architecture



ZServer
- provides flexible internet connectivity supporting many network protocols including HTTP, FTP, XML-RPC, FastCGI, and PCGI. ZServer can operate in tandem with existing web servers.

Zope Core
- includes a web ORB, search engine, security layer, membership, and dynamic information sharing.

Zope Object Database (ZODB)
- supports transactions, undo, private versions, and scales to gigabytes of data. There is also an optional enterprise option available which provides scalability and failover.

RDBMS integration
- offers easy to use and powerful connection to industry leading databases including: Oracle, Sybase, MySQL, and PostgreSQL, as well as ODBC drivers. (!SQLAlchemy)

Zope Products
- extend the Zope core by adding new object types and custom facilities written in Python.

ZClasses
- extend the Zope core, but by now they are outdated.

Who Uses Zope?

- Viacom
- Boston.com
- SGI
- AARP
- Bell Atlantic Mobile (now Verizon Wireless)
- Red Hat
- NASA
- US Navy
- IDG (Brazil)
- GE
- Digital Garage
- Verio
- Park City Ski Area
- Storm Linux.

Who Maintains Zope?

- Originally authored by Zope Corporation
- Parts of Zope have been open source as far back as 1996
- The whole application server is open source since 1998
- Zope Corporation still drives the primary vision and development of Zope
- More and more community members
- The Zope Development Site contains ongoing projects and proposals for Zope's past and future. <http://dev.zope.org>
- Zope Solution Providers (internet providers and developers worldwide)
- Numerous Mailing Lists (Zope users available to answer questions about Zope development, deployment, or any other topic.)
  IRC (irc.freenode.net #zope)

What Can Zope Do For You?

- Create dynamic web applications (portals or intranet sites)
- Provides top-notch access to databases and other legacy data.
- Zope's open support for web standards such as XML-RPC, DOM, and WebDAV allows you unparalleled flexibility and interoperability.

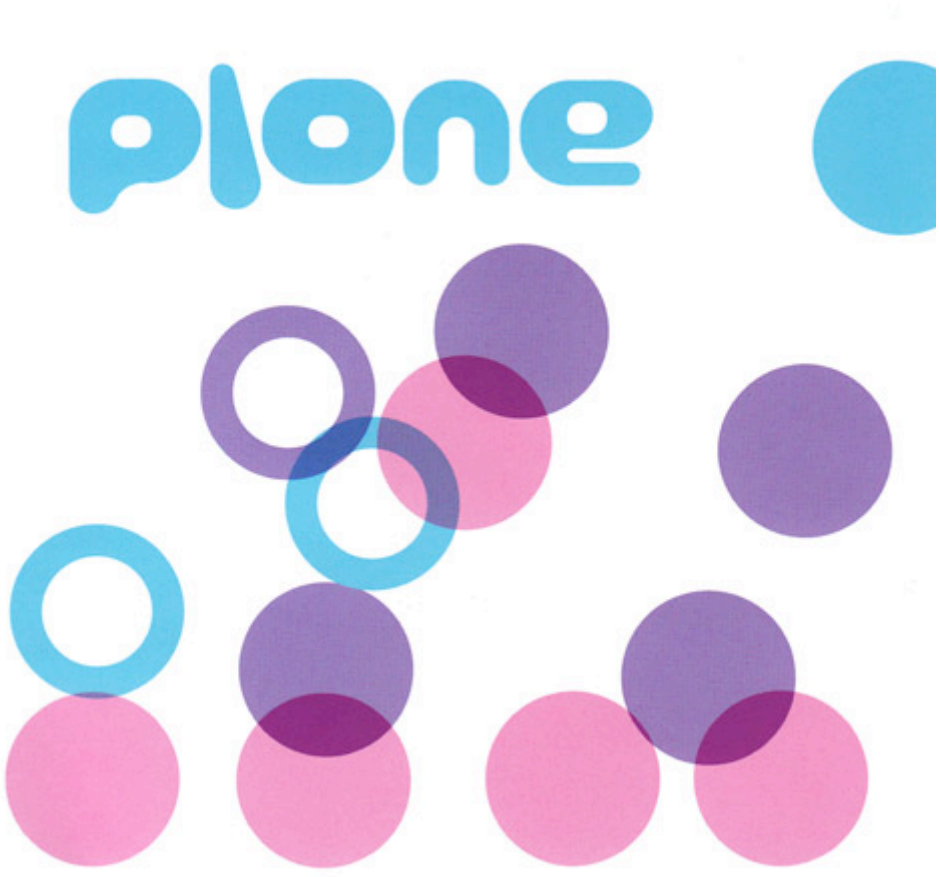
Applications available right away

- The CMF (Content Management Framework): Tools and Services to Zope to allow community or organization based content management. Workflow System, Customization Framework, Review and Publishing.
- Plone: Ready to use Content Management System, polished interface, content such as News Items, Documents, and Events. Delivers the powerful set of tools of the CMF while augmenting behavior with helpful content entry forms and validation.
- Silva: Browser based publication system for different media, including web and paper. Focuses on authoring, versioning, and publishing of complex documents.
- Bizar Shop: Full-featured e-commerce application built on Zope.
- "Helper Products" area: Numerous tools available, like form building and validation, LDAP connectivity, HTML converters, and more.

What is Plone?

Plone is a Powerful, flexible Content Management Solution, easy to install, use and extend.

- Lets non-technical people create and maintain information for a public website or an intranet using only a web browser
- The name Plone comes from a band by that name and “Plone should look and feel like the band sounds”



- Free Open Source Content Management System GNU General Public License (GPL)
- Built on top of Zope
- High usability, flexibility, adaptable workflow, very good security, extensibility

- Over 35 languages
- Conforms to WCAG-AA and U.S. section 508
- Updates on Plone's website
- Development is conducted periodically and continually
- Special meetings called “Sprints”
- Additional functionality with Products (Blogs, Internet Sites, Webshops and Internal Websites, Document Publishing System and Groupware Collaboration Tool)
- Thousands of websites (large and small businesses, education, government, non-profits, sciences, media and publishing)
- Supported by a global network of over 300 solution providers in more than 50 countries

History of Plone

- 1999: Alexander Limi, Alan Runyan and Vidar Andersen started a “Usability Layer” for Zope Content Management Framework (CMF)
- 2001: Plone 1.0
- 2002, June 26-28: Presentation at the Euro Python (community stared)
- 2003, February 10-12: Plone Sprint in Berne

- 2003, October 15-17: First of the annual Plone conference in in New Orleans

- 2004. March: Plone 2.0
- 2004, May: Plone Foundation (development, marketing, codebase, trademarks)

- 2007. March 12: Plone 3 (inline editing, upgraded visual editor, security)
- 2007, September: Over 200 developers contributing, over one million downloads
- 2009: Two Packt Open Source CMS Awards

Design

Plone < Zope < Python
ZODB (**Z**ope **O**bject **D**ata**B**ase)
Installers for Windows, Mac OS X, and Linux (and other operating *systems)*

- *Templates can be used to customize a website's look.*
- Cascading Style Sheets
- User management system “Pluggable Auth(entication) Service” (Users and Groups, Security, Authentication, User Data)

Programming Languages

- Python 77%
- XML 16%
- JavaScript (including the JQuery Javascript framework) 5%
- Other 2% (CSS, XSLT)

Community

The Plone community is an incredibly diverse group that bridges many types and sizes of organizations, many countries and languages, and everything from technical novices to hardcore programmers. Out of that diversity comes an attention to detail in code, function, user interface and ease of use that makes Plone one of the top 2% of open source projects worldwide. (Source: Ohloh)

- Plone “Sprints” (Members of the Community coming together) helping improve Plone
- Annual Plone Conferences
- IRC channel (`irc.freenode.net #plone`)

Strengths

- Linux, Windows, Mac OS X, FreeBSD, Solaris and more
- Standards conformance

- Access control
- Excellent security record compared to other popular content management systems
- Internationalization (i18n)
- Aggregation
- Active user groups
- Different operating systems
- Works with most common web browsers
- Accessibility standards (disabilities)
- Customizable
- Free add-ons are available
- Easy to understand and use for the Content Provider
- Integrates with Active Directory, Salesforce, LDAP, SQL, Web Services. LDAP or Oracle

Weaknesses

- Hard to learn for developers
- Over-Engineered?
- Lagging in repository services when compared to other major CMSs.

Features (Plone 3.0)

- Inline editing
- Working Copy support
- Automatic locking and unlocking
- Link and reference integrity checking
- Collaboration and sharing
- Versioning, history and reverting content
- Upgraded visual HTML editor
- Workflow capabilities
- Authentication back-end
- Full-text indexing of Word and PDF documents
- Collections
- Presentation mode for content
- Support for the search engine Sitemap protocol
- Support for multiple mark-up formats
- Wiki support
- Automatic previous/next navigation
- Rules engine for content
- Auto-generated tables of contents
- Portlets engine
- LiveSearch
- Multilingual content management
- Time-based publishing
- Human-readable URLs
- Powerful graphical page editor
- Navigation and updated site maps
- Resource compression
- Caching proxy integration
- Drag and drop reordering of content

- XML exports of site configurations
- Localized workflow configuration
- Adjustable templates on content
- Standard content types
- Content is automatically formatted for printing
- Standards-compliant XHTML and CSS
- Accessibility compliant
- RSS feed support
- Automatic image scaling and thumbnail generation
- Free add-on products
- Cross-platform
- Comment capabilities on any content
- Microformat support
- Installer packages for multiple platforms
- WebDAV and FTP support
- In-context editing
- Backup support
- Cut/copy/paste operations on content

Renowned Users

- NASA
- Oxfam
- Amnesty International
- Nokia
- eBay
- Novell
- State Universities of Pennsylvania and Utah,
- Brazilian and New Zealand governments
- CIA
- Burning Man

Hands on installation of Plone-4.0.2:

This is an example on how to install (and run) a Plone4 instance:

```
$ wget http://launchpad.net/plone/4.0/4.0.2/+download/Plone-4.0.2-
UnifiedInstaller.tgz
$ tar -xzvf Plone-4.0.2-UnifiedInstaller.tgz
$ cd Plone-4.0.2-UnifiedInstaller
$ ./install.sh --target=/Users/georg/Documents/Work/tmp/metalab/Plone-4.0.2
--user=gogo --password=test zeo
ZEO Cluster Install selected

Detailed installation log being written to
/Users/georg/Documents/Work/tmp/metalab/Plone-4.0.2-
UnifiedInstaller/install.log

Rootless install method chosen. Will install for use by system user georg

Installing Plone 4.0.2 at /Users/georg/Documents/Work/tmp/metalab/Plone-
4.0.2
```

```
Compiling and installing jpeg local libraries ...
Skipping zlib build
Compiling and installing readline local libraries ...
Installing Python 2.6.5. This takes a while...
Installing distribute...
Unpacking buildout cache to /Users/georg/Documents/Work/tmp/metalab/Plone-
4.0.2/buildout-cache
Compiling .py files in egg cache
Copying Plone-docs
Copying buildout skeleton
Fixing up bin/buildout
Running buildout; this takes a while...

######################################################################
######################  Installation Complete  ######################

Plone successfully installed at
/Users/georg/Documents/Work/tmp/metalab/Plone-4.0.2
See /Users/georg/Documents/Work/tmp/metalab/Plone-
4.0.2/zeocluster/README.html
for startup instructions

Use the account information below to log into the Zope Management Interface
The account has full 'Manager' privileges.

  Username: admin
  Password: test

This account is created when the object database is initialized. If you
change the password later, you'll need to use the new password.

- If you need help, ask the mailing lists or #plone on irc.freenode.net.
- The live support channel also exists at http://plone.org/chat
- You can read/post to the lists via http://plone.org/forums

- Submit feedback and report errors at http://dev.plone.org/plone
(For install problems, specify component "Installer (Unified)")

$ cd Plone-4.0.2
$ ./zeocluster/bin/startcluster.sh
$
```

Open a browser and point it to http://127.0.0.1:8080, then click “Create Plone Site”.

Putting it together

High performance sites, cluster

When Zope sites are too slow you can kill the dragon by throwing more hardware at it: