python™

BUZZWORD COMPLIANCE

Notes

- Charles Merriam
- Cell: 408.368.6050
- Email: charles.merriam@gmail.com
- Blog: http://charlesmerriam.com/blog
- Copy of this talk: http://charlesmerriam.com/talk
- Yes, you can have this talk given to your group. Email me.

Buzzword Compliance

- 3 Slides Per Buzzword
- High Signal To Noise
- Breadth Over Depth
- About EXPLORING Python

Buzzword Compliance

The Python Language

Library Building Blocks
(Games & Graphics)

Big Honking Frameworks

(Web Application Frameworks)

- All are part of Python

Notes

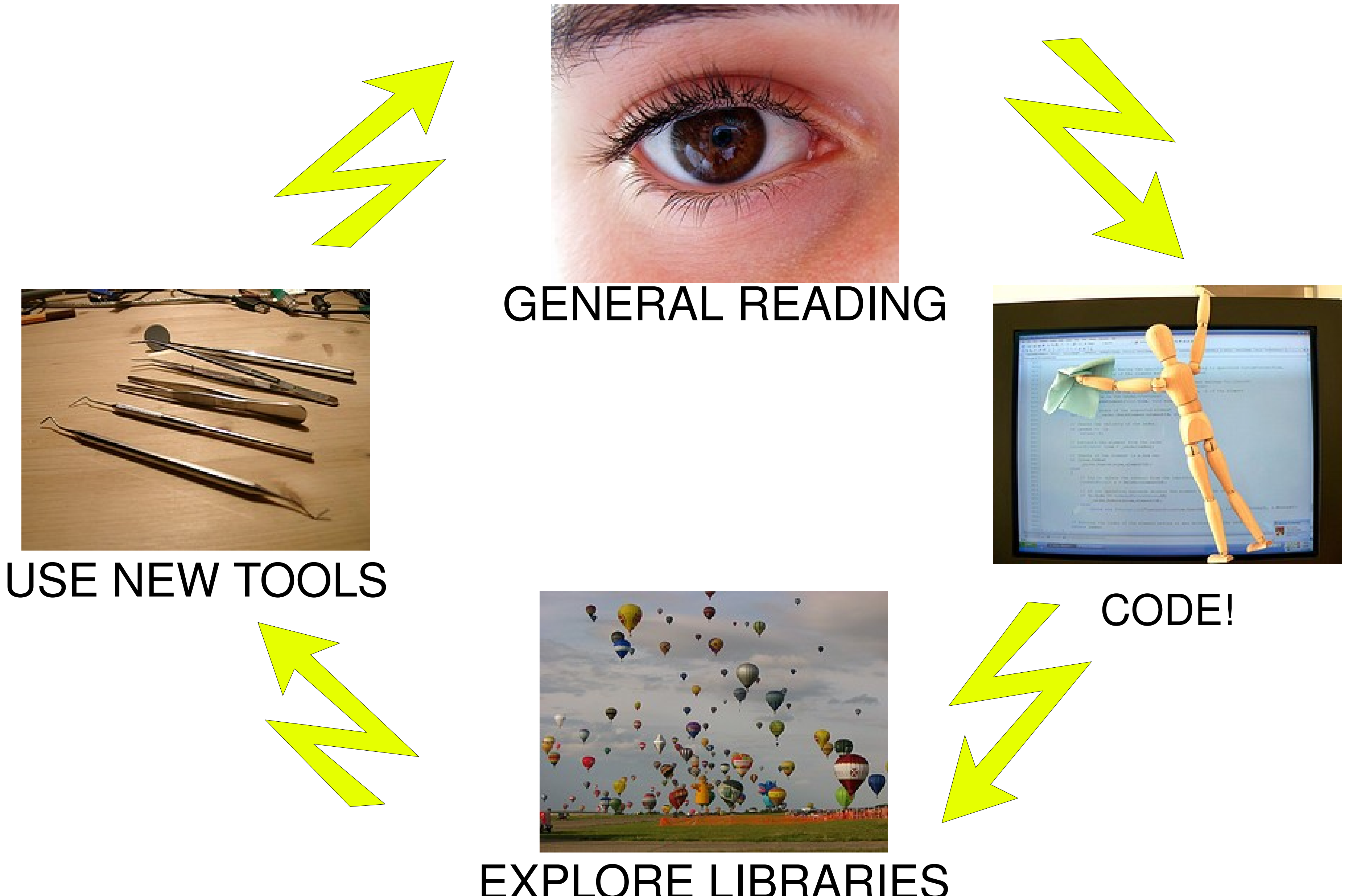
- This talk rapidly covers a lot of buzzwords. For each buzzword covered, many are not covered.
- I hope to provide useful information in the time allocated. That is, “a high signal to noise ratio.”
- I hope to provide some map for how to explore Python, leaving the additional details to your exploration.

LEARNING PYTHON

Learning Python

- The Quick Reference Sheet
- Python Tutorial
- Python Challenge
- A Cycle of Learning

Learning Python



Links

- Yes, this slide has the most links, 10 pages!
- Richard Gruet's Python Quick Reference Sheet
    - http://rgruet.free.fr/PQR25/PQR2.5.html
  - Great reference guide to the language and syntax
  - Online version includes direct links to most of the documentation
  - Read once, and then keep a link to it open

Links

- Bruce Eckel's Why I Love Python Presentation
    - http://www.mindview.net/Books/Python/ThinkingInPython.html
  - A powerpoint of why he loves Python
  - Compares philosophy with other languages
  - Mostly using cartoons

Links

- Python Tutorial
    - http://docs.python.org/tut/
  - Guido van Rossum's basic tutorial
  - Walks through syntax of Python using code examples and text.
  - Covers the language and a brief tour of the standard library.

Links

- Code Like a Pythonista
    - http://python.net/~goodger/projects/pycon/2007/idiomatic/handout.html
  - Covers the Zen of Python, tricks, and idioms
  - Many useful idioms in Python take some practice to get right.
  - A useful guide for those coming to Python from another language.

Links

- Dave Kuhlman's Python 201
    - http://www.rexx.com/~dkuhlman/python_201/python_201.html
  - A talk covering different patches, like this Buzzword Compliance talk, but different topics
  - Looks at Regular Expressions, Testing, Embedding, Parsing and GUIs.
  - Definitely worth a look.

Links

- AwareTek Tutorials Page
    - http://www.awaretek.com/tutorials.html
  - Contains hundreds of tutorials categorized by subject matter
  - Fast way to get more information on some particular topic
  - Not everything is best or relevant, but everything is here

Links

- Advanced Python (or Understanding Python)
    - http://www.charlesmerriam.com/blog/?p=48
  - Video explains how namespaces are built at runtime
  - Goes into iterators, decorators, and metaclasses
  - Covers both the advanced topics and underlying simplicity
  - Write up has time-line by topic

Links

- Python Challenge
    - http://www.pythonchallenge.com/
  - Puzzles for Python Programmers
  - You write code to find the next level
  - Becomes 'guess the buzzword' after six or seven levels.

Links

- Python Debugger
    - http://docs.python.org/lib/module-pdb.html
  - A module in the standard library for debugging
  - Your IDE may give you a better interface
- PyChecker
    - http://pychecker.sourceforge.net/
  - A light, "lint" tool for Python
  - Can find some common errors

Links

- WingWare IDE
    - http://wingware.com/
  - A well recommended closed-source Python IDE
  - There is a minimal free version
- Plug-ins exist for GEdit, Emacs, Vi, Eclipse, and everything else
  - Google is your friend.

Notes

- The key aspect is to not neglect any one of these activities for too long.
- Even when in the midst of a project, explore libraries and tools. It may save you time.

[ ]

LIST COMPREHENSIONS

List Comprehensions

- A Cool Idiom of Python
- Enables Conciseness
- Obviates ***map***, ***filter***, ***reduce***
- ✯ Unrolls into Simple Loops

List Comprehensions

```
lost = sum([c.billed - c.paid for c
 in customers if c.is_deadbeat()])
```

```
l = [ ]
for c in customers:
    if c.is_deadbeat():
        l.append(c.billed - c.paid)
lost = sum(l)
```

Notes

  - Nothing magical, this is just a shorter way of writing common idioms we all have written a million times.
  - Python often uses the idea of “equivalent to longer code” to keep the language simple.
  - After a week, you will never do without them.

- Python Tutorial on List Comprehensions
  - http://docs.python.org/tut/node7.html
  - walks through a few examples in section 5.1.4

Notes

- For extra fun points:
  - list comprehensions use a list
  - list comprehensions return a list
  - so, list comprehensions can be nested.
  - so, you too can make unreadable code

EXECUTING MODULES

Executing Modules

- Import runs code, once.
- ***def*** is just a statement
- Use to precalculate stuff
- Python just runs scripts in namespaces

Executing Modules

```
class C:
    print “Hello from C”
    def help_make_table(size):
        ...
    c_table = help_make_table(64)
    del help_make_table
```

Notes

- The system remembers which classes and modules have been executed.
- .PYC files are simply tokenizers to speed loading. They don't pre-execute.
- The "def **func**():" statement places an entry **func** into the local namespace.
- The namespace is just a dictionary (dict).
- Function names are just variables like any other.

Links

- Advanced Python (or Understanding Python) Video (again)
    - http://video.google.com/videoplay?docid=7760178035196894549
  - At 4:15, talks about Python's execution in namespaces.
  - At 8:45, walks through an equivalent example of executing with a temporary method in a class.

@ DECORATORS

Decorators

- Wraps methods with new functionality
- Useful for logging, security, etc.
- Clean Syntax for use

☆ Unrolls to simple code

Decorators

```
from decorator import decorator
@decorator
def trace(f, *args, **kw):
    print "call %s with args %s, %s"
  % (f.func_name, args, kw)
    return f(*args, **kw)

@trace
def buggy_function(a, b, c)
```

notes

- A decorator does something, then (usually) calls your function.
- This is another Python 'equivalent code' trick:
  - @trace
  - def myFunc(): print “Hello”
- is equivalent to:
  - def myFunc(): print “Hello”
  - myFunc = trace(myFunc())

notes

- Best used when the function and the decorator are solving distinct problems.
- Avoid the Java file types thing where decorators are confused with subclassing.
- Boilerplate code for decorators is required unless you use @decorator.
- Still need boilerplate to pass extra parameters to your decorator, e.g., @check_level(ADMIN)

links

- Michelle Simionato's Decorator Decorator
  - http://www.phyast.pitt.edu/~micheles/python/documentation.html
  - Provides good @decorator syntax for writing decorators
  - Correctly identifies name and number of arguments of decorated function in IDEs and other tools.
  - You will probably never write a decorator without this package again.

links

- David Mertz's Decorator Tutorial
    - http://www-128.ibm.com/developerworks/linux/library/l-cpdecor.html?ca=drs-#resources
  - A good walk through of Decorators, their uses, and their pitfalls
  - Introduces Michelle Simionato's Decorator Decorator

commentary

- This moon introduces the “dark side” of Python. These are areas, in my own opinion, where the Python languages gains opportunities for significant improvement. The characterization of these areas as “dark side” is an opinion, and you may form different opinions.

METACLASSES

Metaclasses

- The superclass '***type***' of classes
- Changes functionality of Python
- Adds complexity to entire project
- ✯ Shiny things can be traps

Metaclasses

```
class Midnight_Hack(type):
  def __new__(cls, name, bases, ats):
    for a,v in ats.items():
          # post-process ats...
    return type.__new__(cls, name,
                            bases,ats)

class Innocent_PEP_3115(P3000):
    __metaclass__ = Midnight_Hack
```

Notes

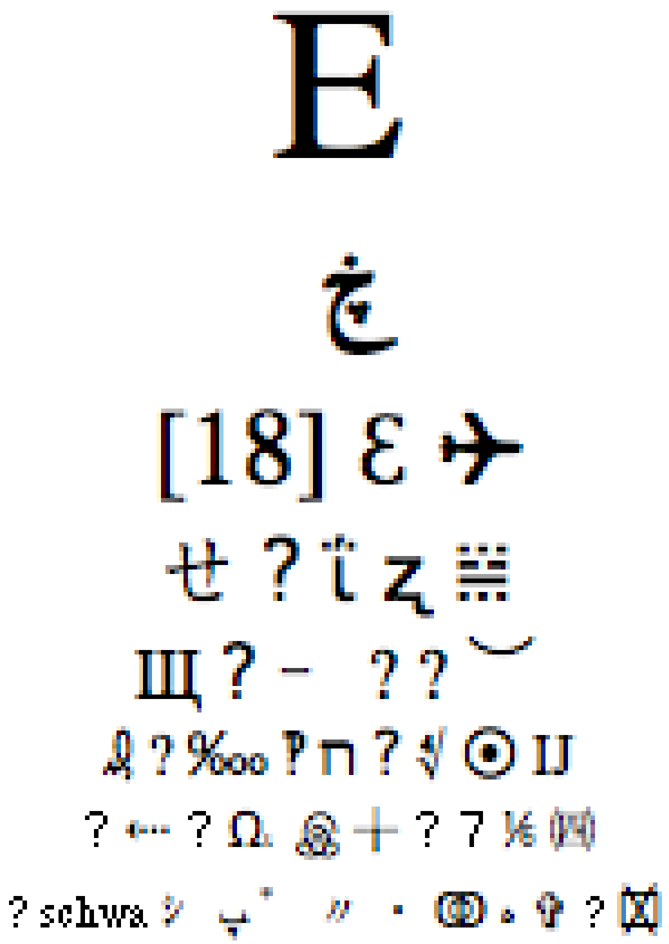
- Mechanically, the 'type' object is a metaclass. You can write your own metaclass to get handle class creation and instantiation. You then specify a class use a particular metaclass.
- You can add in new name swizzling, make Aspect oriented programming
- Changes Python. Code readers and some tools are no longer sure what a line of code does.

Notes

- Using metaclasses adds to the complexity of your project, even though it shouldn't.
- Should be a company level or project level decision.
- Hang rogue programmers by toes if they insist on metaclasses for trivial reasons.

links

- David Mertz's Metaclass Tutorial
    - http://www.ibm.com/developerworks/linux/library/l-pymeta.html
    - http://www.ibm.com/developerworks/linux/library/l-pymeta2/
    - http://www.ibm.com/developerworks/linux/library/l-pymeta3.html
  - A long overview of metaclasses, more than you might want to know.
- Very few uses for metaclasses



UNICODE

Unicode

- Represents **every** human language
- Breaks all ASCII rules
- Designed by Committee
- Real world adds constraints

Unicode

UNICODE Jargon includes: Universal Character Set; The Unicode Standard book; character encodings; enumerate properties; text normalization; decomposition; collation; bidirectional display order; Unicode Consortium; Ligatures; orthographic rules; sidebearing; macron; WGL-4; Multilingual European Subsets MES-1/2/3a/3b; replacement character; LastResort font; UTF-8; codec.open(); ISO 14755;
C0 and C1 control codes; Han unification versus TRON; GB-18030;
Binary Ordered Compression; Basic Multilingual Plane; UnicodeDecoderError on str.encode() contrasted with UnicodeEncoderError on str.decode(); endianness external metadata; PunyCode; graphemes; syllabaries; ConScript Unicode Registry; Universal Transformation Format versus Universal Character Set mappings; Private Use Area; Hangul Jamo; radicals

- Just “make peace” with Unicode

Notes

- All languages:
  - Everything is represented. Umlat's, glyphs, weird directions, different widths. English, French, (Hebrew's right to left with a different alphabet). Sanscrit. Klingon. Elvish.
  - Representation as sequence of bytes is a separate encoding issue. Many use UTF-8, which takes one byte for Latin languages and many for Asian languages.   Encoding is out of band of text.

Notes

- Old programmers must grok Unicode:
  - Used to plan on uniform character widths, heights, and direction for layout.
  - Used to have a unique character for output and be able to glance at screen to determine which character was output.
  - Used to ranges encompassing groups, like all lowercase characters being contiguous.
  - String comparisons must be more careful.

Notes

- Unicode Standards Committees
    - http://unicode.org/
  - High paying members alter standards for corporate strategies and tactics.
  - Official and unofficial sources and variants.
  - Official sources cost money.
  - Scheduled face-to-face meetings and conference calls instead of implementation or chat driven.
  - Very unlike open source standards. It's a career!

Warnings

- Unicode support in Python, and in Python 3.0, is about the best you will find, including codecs classes. Unicode will still be a problem.
- Expect programs to become less reliable as odd Unicode mangles display layouts.
- Unicode support != Localization. You will still use gettext().
- Add some test cases with odd language inputs.

Links

- Wikipedia
    - http://en.wikipedia.org/wiki/Unicode
  - Some general Unicode information
  - Note that even the 'talk' page has 70 entries
  - Primary Wikipedia unicode article is in Russian.
- Oddly, no English language videos explaining Unicode are available on YouTube (as of today).

Links

- Advanced Python (or Understanding Python) Video (again)
    - http://video.google.com/videoplay?docid=7760178035196894549
  - At 1:04:32, it discusses Python 2.x unicode for ten minutes.
  - Recommends not to mix Ascii and Unicode.

Notes

- Python 3.0 uses Unicode strings
  - All “strings” become unicode strings.
  - ByteArray replaces the current strings
  - “Hello” is unicode, b”Hello” is five ASCII bytes.
  - They do not mix implicitly. This is good.

PYTHON 3000

Python 3000

- Incremental not Revolutionary
- Need to read old code
- Available as Alpha (3.02a)
- Guido exercises restraint

Python 3000

Function Annotations PEP-3107

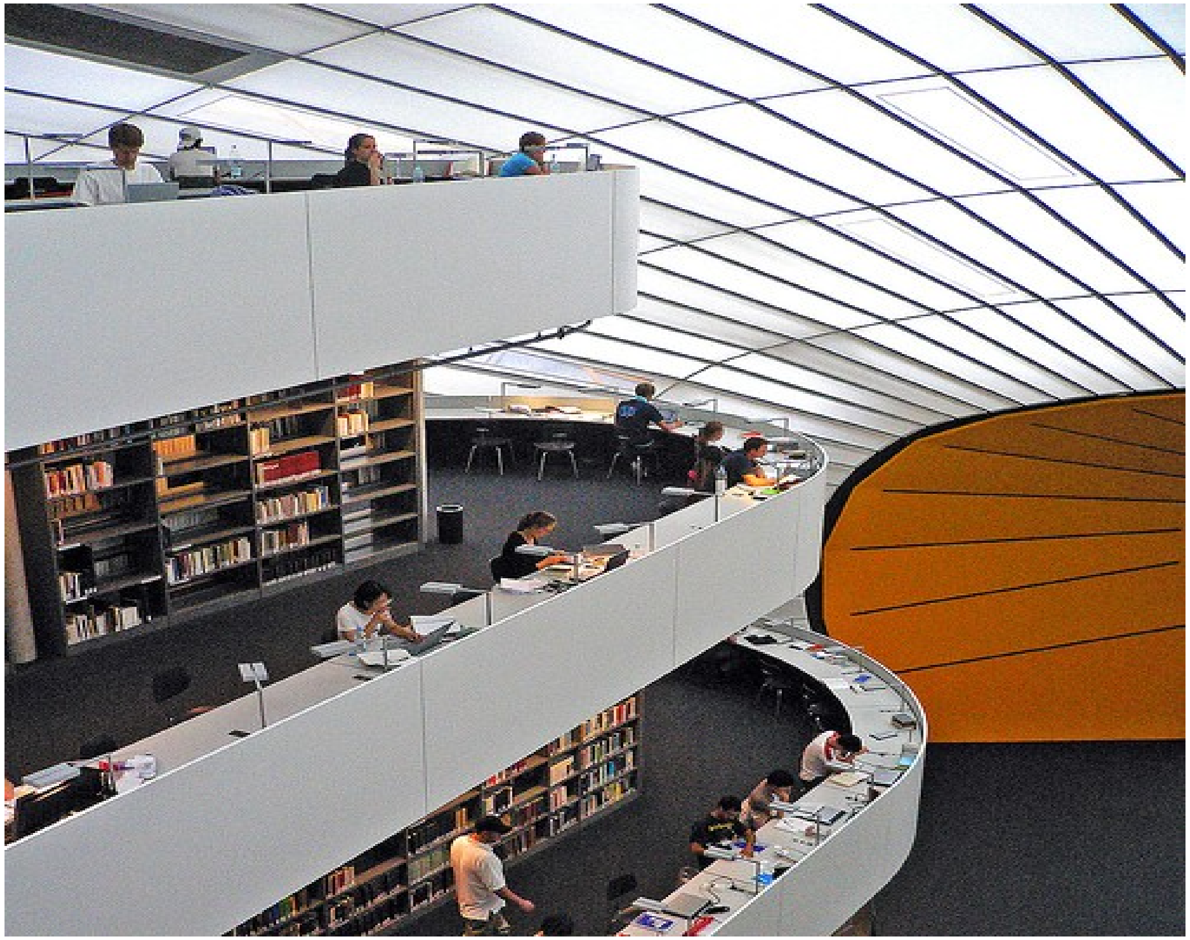
```
def create_map(x: “in map units”,
    y: “in map units”,
    walls: “2D boolean array (x by y)
  with True meaning a wall”,
    pixel_width: “number of pixels per
  map unit”) -> “Graphical PNG map”


def random_map(x: Coord, y: Coord) ->
  Image:
```

notes

- Notice that many features will be of controversial value.
  - Some, like PEP 3107, will allow code bases to implement very different and incompatible styles.
  - Some, like PEP 3129, were added with little discussion.
  - Unclear that Python 3.0 code will be more elegant than Python 2.5.

notes

- Code must be reread and retested
  - Changes, like the printing of long numbers as strings losing the trailing 'L', will require review of all code to avoid subtle bugs.
- 2to3 tool aids conversion, handling mechanical changes
- Download Latest Version (3.02a today)
    - http://www.python.org/download/releases/3.0/
  - It's starting to gel and solidify. Expect 2008.



Libraries

notes

- One can explore any set of libraries, I'm using imaging libraries as an example.
- See Python 3D Software Collection
    - http://www.vrplumber.com/py3d.py
  - A list of various 3D packages and games in Python
  - Good starting place with too few reviews
  - Note how wildly libraries vary in quality

Python Imaging Library (PIL)

Python Imaging Library (PIL)

- Reads and Writes Image Formats
- Rock solid with 250 formats
- Interactive image manipulation
- Not all 'batteries included'.

Python Imaging Library

```
import Image
im = Image.open(“cool.jpg”)
im = im.resize(128,128).rotate(90)
im.save(“cool.png”)

(r,g,b) = im.split()
(x,y) = im.size
im.show()
```

Notes

- The PIL by Pythonware
    - http://www.pythonware.com/products/pil/
  - Tutorial should be sufficient documentation for most users. It's about a page or so.
  - The handbook is the complete documentation.
  - Free is the same version as commercial support
  - Last updated in December 2006, releases about once per year.

pygame

PyGame

- Easy 2D Game Engine
- Aggressively cross platform
- Continuous contests
- Sometimes static != static

FRINKIAC-7 PYBOMBER

next:
new:
2
score:
000001860
next:
new:
0
score:
000004470

Notes

- Complete but small 2D game engine with drawing, sounds, joysticks, sprites, etc.
  - Works on all sorts of smaller devices.
- Last release was August, 2005
  - Various forks for devices, odd integrations, etc.
  - Huge fan base.
  - Very stable core

Links

- PyGame
    - http://www.pygame.org
  - This has samples, tutorials, and all documentation
- "Beginning Game Development with Python and Pygame: From Novice to Professional", 2007
    - http://www.amazon.com/gp/product/1590598725

soya

Soya 3D

- Full 3D Game Engine (almost)
- Uses Pyrex for linking modules
- Slow and forked with PySoy
- Avoid Python centric bias

Notes

- The main Soya site:
    - http://home.gna.org/oomadness/en/soya3d/index.html
- The main PySoy (spin-off) sites:
    - http://www.pysoy.org
    - http://www.soya3d.org/
  - FAQ recommends against PySoy use yet
- Only about a dozen games using either fork

Notes

- Includes graphics, sound, physics, and networking
- Pyrex is a Python variant with C Datatypes
    - www.cosc.canterbury.ac.nz/greg.ewing/python/Pyrex
- Soya is assembled on several libraries from different sources.

OGRE

Python OGRE

- Game Engine From Wrappings
- Uses Py++ to make Python bindings
- OGRE is popular

☆ Sometimes a “Mash-up” is best

Python OGRE

- Rendering (OGRE 3d + Forests/sky/terrain)
- I/O (Object Oriented Input System or OIS)
- GUI (Crazy Eddie GUI & Navi & BetaGUI)
- Sound (OgreAL & Plib + Noise)
- Physics (NxOgre / PhysX, OgreODE / ODE, OgreNewt / Newton, OgreBullet / Bullet)
- Networking (Plib)
- Video (Theora, OgreDshow, ffmpeg)

Notes

- Python Ogre was originally named PyOgre
  - PyOGRE just wrapped OGRE.
  - To find good answers, you may need to check three forums: Python-OGRE, PyOgre, and OGRE.
- OGRE Handles scenes, lighting, object placement. Not game mechanics.
- OGRE spawned many separate projects for game mechanics.

Notes

- OGRE is basis for several commercial games
- Py++ is another wrapping engine for C++ Code
      - http://www.language-binding.net/pyplusplus/pyplusplus.html
  - Py++ wrapping is an automatic process.
  - Handles API changes quickly.

Links

- Python Ogre site
    - http://python-ogre.org/
- The main OGRE site.
    - http://www.ogre3d.org/
- The Two Wikis
    - http://wiki.python-ogre.org/index.php/Main_Page
    - http://www.ogre3d.org/wiki/index.php/PyOgre

Frameworks

notes

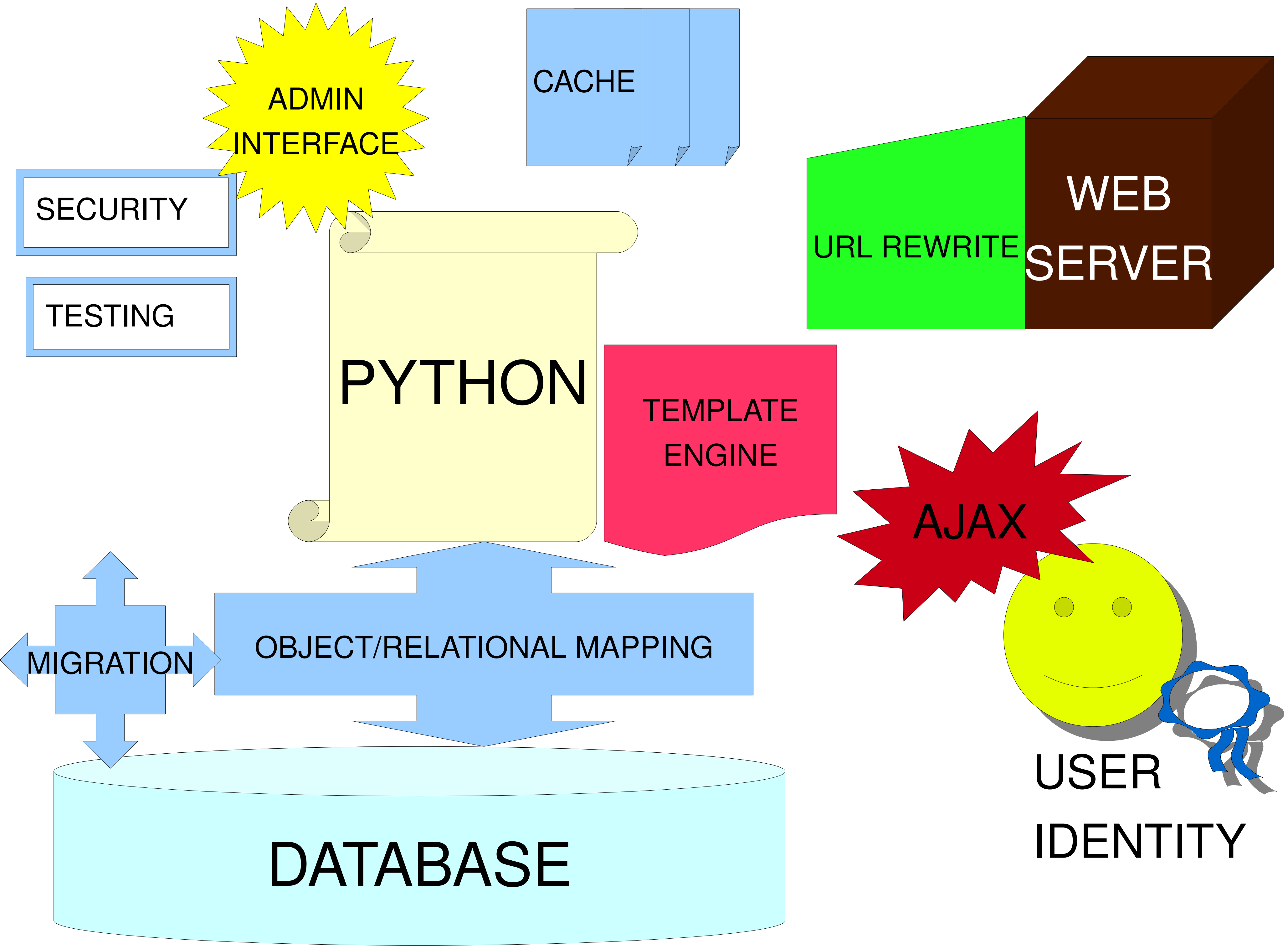
- Frameworks are what libraries morph into when they grow huge.
- Frameworks generally want control. You call a framework and it calls your code. Libraries want you to call them
- Using multiple libraries in an application is easy while using multiple frameworks is significantly harder.

Web Application Frameworks

- Piece of History. Picture is cover of:
  - Phil and Alex's Guide to Web Publishing
    - http://philip.greenspun.com/panda/
    - http://www.amazon.com/Philip-Alexs-Guide-Web-Publishing/dp/1558605347
  - Phil Greenspun wrote this coffee table book full of photos, which also showed how to build a solid database backed web site using TCL and Oracle.

Web Application Servers

- Deploy Web Applications
- Pile of Parts
- Dozens of Choices
- Indecision Breeds Religious Arguments


ADMIN INTERFACE
CACHE
SECURITY
TESTING
URL REWRITE
WEB SERVER
PYTHON
TEMPLATE ENGINE
AJAX
MIGRATION
OBJECT/RELATIONAL MAPPING
USER IDENTITY
DATABASE

notes

- That's a lot of components!
  - Each component represents its own field.
  - Total number of lines of code is immense
- Wikipedia is your friend for overviews
    - http://en.wikipedia.org/wiki/Web_application_framework
    - http://en.wikipedia.org/wiki/Comparison_of_web_applicati on_frameworks

notes

- Note that language has shifted over time.
  - In the 1980's, these were called "Database Backed Websites"
  - In the 1990's, these were called "Application Servers", then "Web Application Servers"
  - In the 2000's, these are called "Web Application Frameworks"
- Still same basic issues, with better parts.

reviews

- Comparison by James Cooley
    - http://james.cooley.ie/2007/03/05/django_turbogears_rails_and_the_feature_curve.html
  - Written back in March, 2007
  - Short comparison of frameworks, along with Ruby on Rails
  - Basically recommends Django when you control the environment and either TurboGears or Pylons when you are integrating existing components.

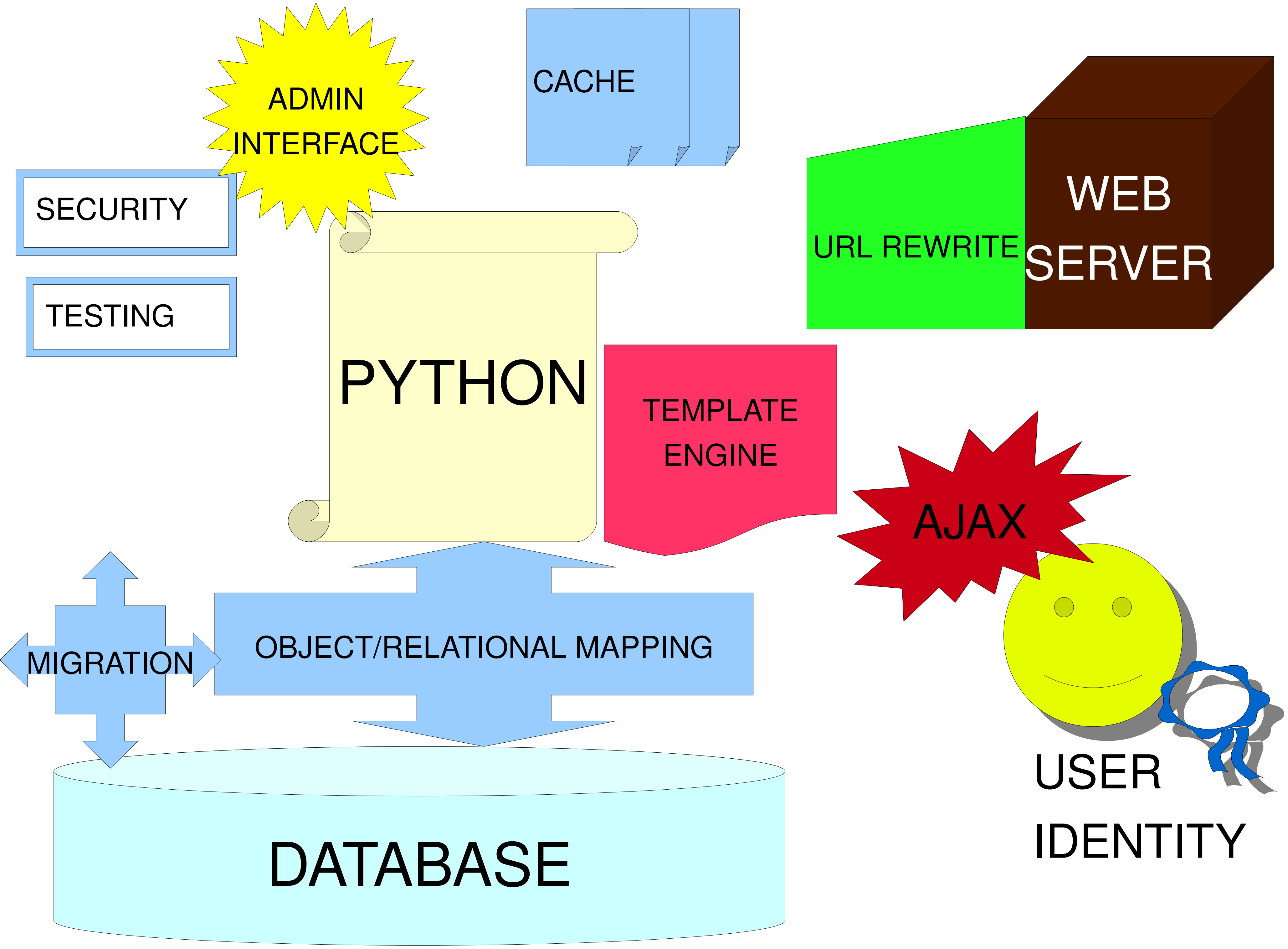
reviews

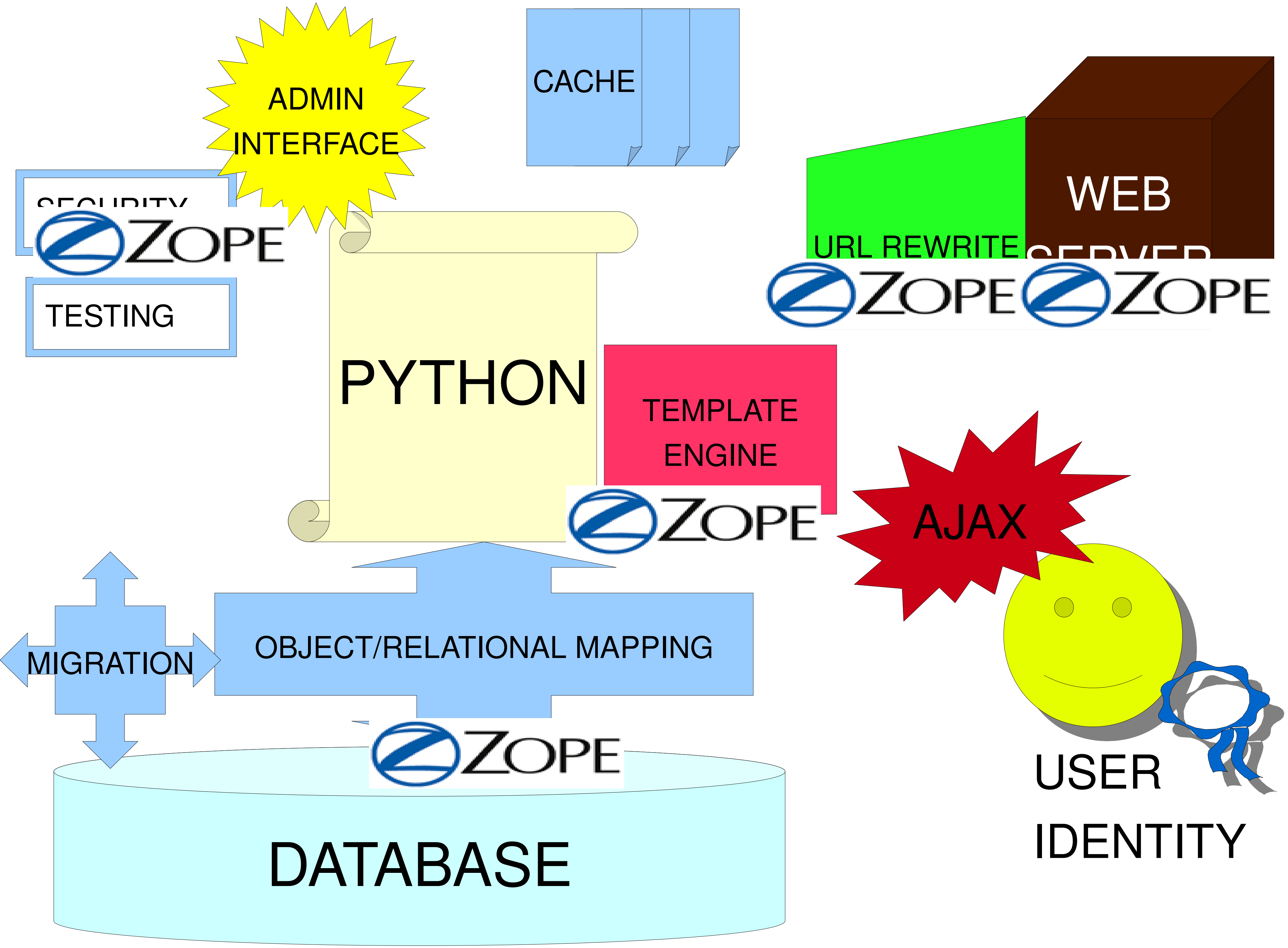
- Neil Blakely-Milner's comparison
    - http://tinyurl.com/2v4p2a
  - Another comparison Written in June, 2007
  - Recommends Django for smaller projects

ZOPE

Zope

- Scalable Since 1995
- Full web based interface
- Zope 2 and Zope 3
- Success is its own reward


ADMIN INTERFACE
CACHE
SECURITY
TESTING
URL REWRITE
WEB SERVER
PYTHON
TEMPLATE ENGINE
AJAX
MIGRATION
OBJECT/RELATIONAL MAPPING
USER IDENTITY
DATABASE


ADMIN INTERFACE
CACHE
SECURITY
ZOPE
TESTING
PYTHON
URL REWRITE
WEB SERVER
ZOPE
ZOPE
TEMPLATE ENGINE
ZOPE
AJAX
MIGRATION
OBJECT/RELATIONAL MAPPING
ZOPE
DATABASE
USER IDENTITY

notes

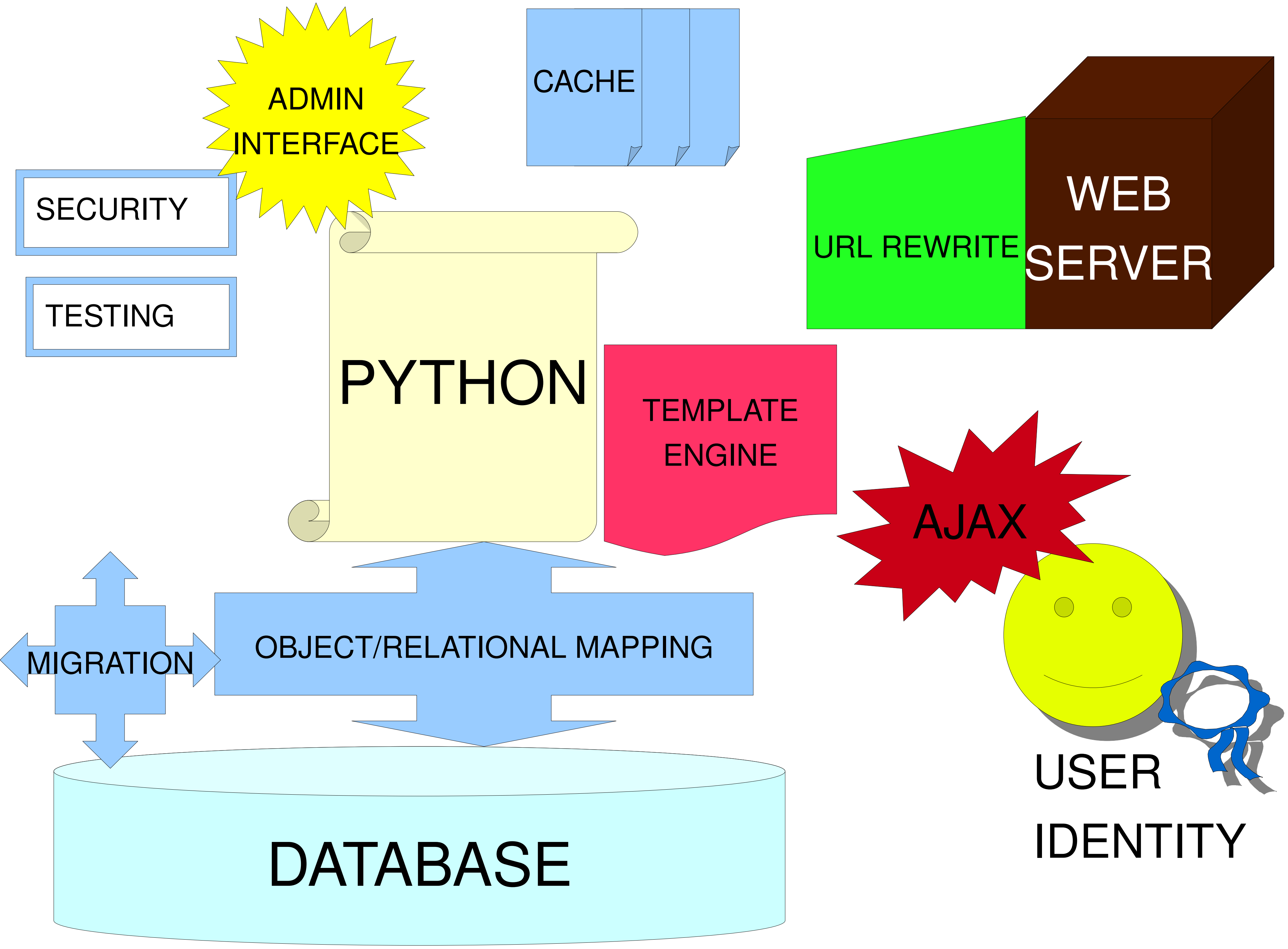
- See also:
    - http://en.wikipedia.org/wiki/Zope
    - http://www.zope.org/
    - http://www.zopelabs.com/
    - http://wiki.zope.org/WikiZopeOrg
  - – Note the major disparate sets of Zope 2 and Zope 3 pages
- Many Zope Books published, all over two years ago.

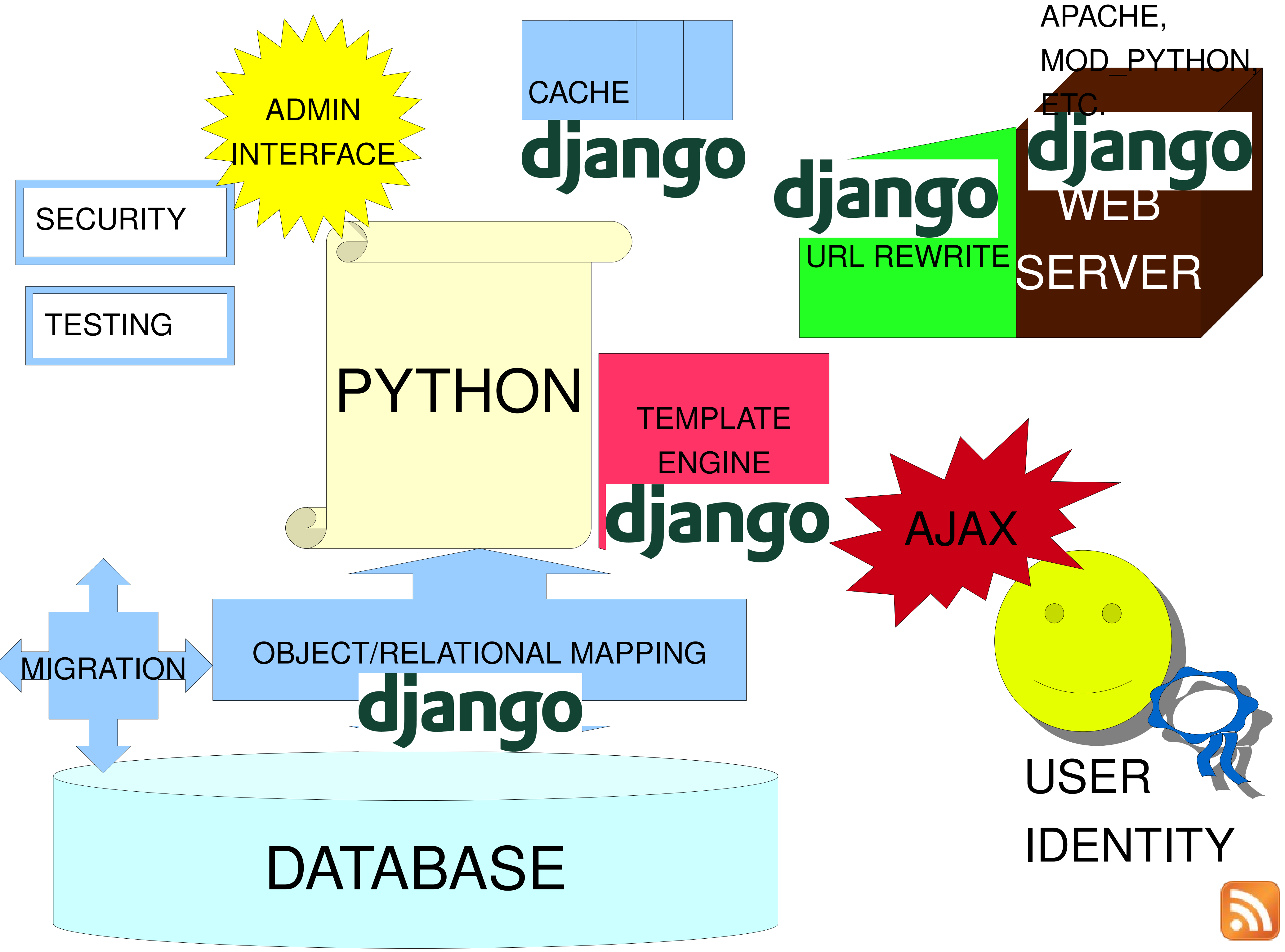
django

Django

- Emphasis on *Don't Repeat Yourself* (***DRY***)
- Interactive console
- Unfortunate naming, e.g., MVC

☆ Confusion Comes From Names


ADMIN INTERFACE
CACHE
SECURITY
TESTING
URL REWRITE
WEB SERVER
PYTHON
TEMPLATE ENGINE
AJAX
MIGRATION
OBJECT/RELATIONAL MAPPING
DATABASE
USER IDENTITY


APACHE,
MOD_PYTHON,
ETC.
CACHE
django
ADMIN INTERFACE
django
URL REWRITE
django
WEB SERVER
SECURITY
TESTING
PYTHON
TEMPLATE ENGINE
django
AJAX
MIGRATION
OBJECT/RELATIONAL MAPPING
django
USER IDENTITY
DATABASE

Notes

- Naming
  - MVC language in Django is "Model-Template-View"
  - The 'D' is silent in Django
  - Most top Google hits will be to a musician
- Informal extra reviews
  - Great, fast development if you stay within the lines
  - Extending Django feels like “struggling in straightjacket”

links

- Wikipedia Overview
    - http://en.wikipedia.org/wiki/Django_(web_framework)
- Main Django Site
    - http://www.djangoproject.com/

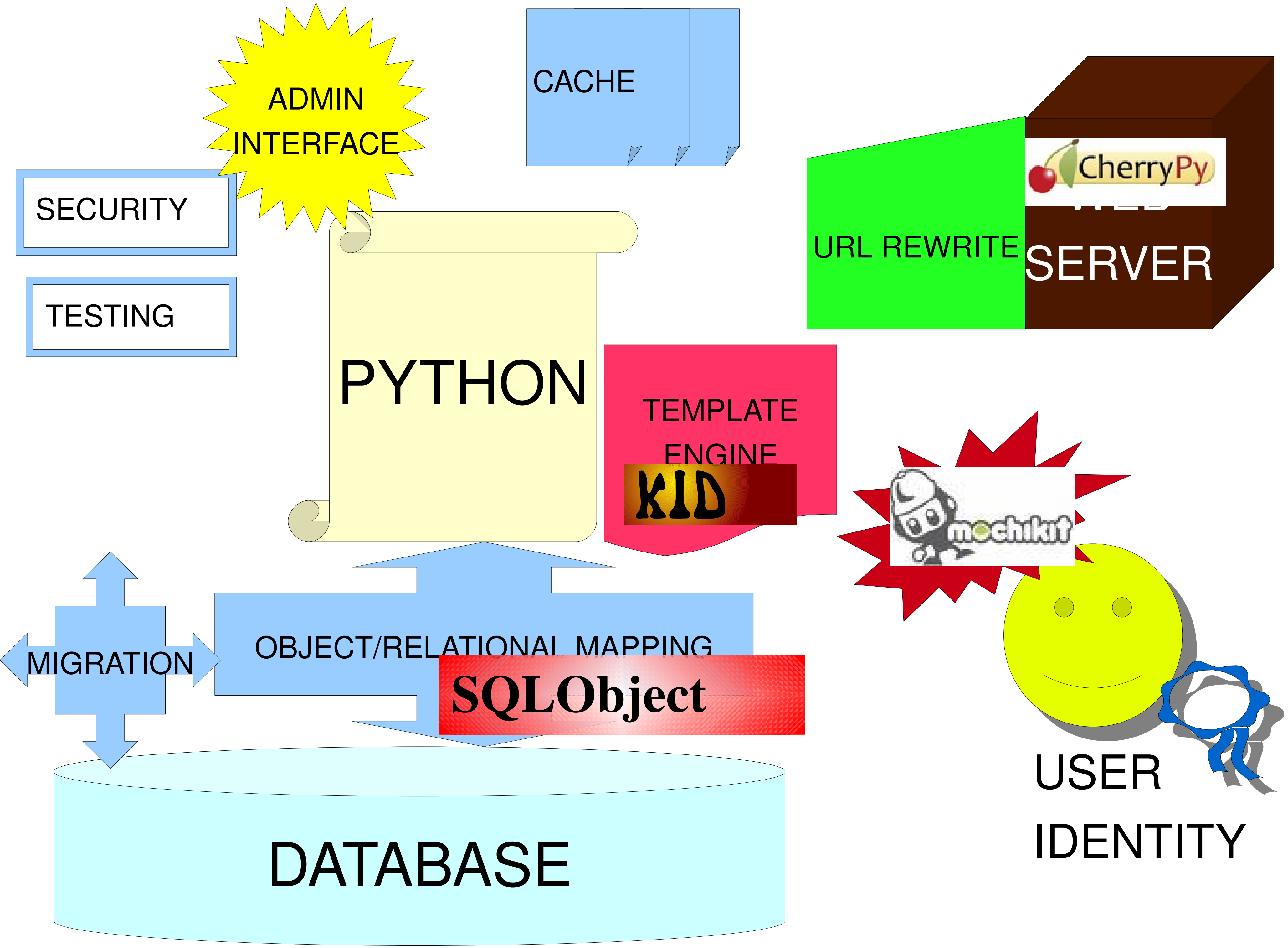
books

- "The Definitive Guide to Django: Web Development Done Right", December 2007
  – http://www.amazon.com/Definitive-Guide-Django-Development-Right/dp/1590597257/
- "Professional Python Frameworks: Web 2.0 Programming with Django and Turbogears (Programmer to Programmer)", 2007
  – http://www.amazon.com/Professional-Python-Frameworks-Programming-Turbogears/dp/0470138092
- “Sams Teach Yourself Django in 24 Hours", expected March 2008
  – http://www.amazon.com/Sams-Teach-Yourself-Django-Hours/dp/067232959X

TURBO GEARS

TurboGears

- Assembled from more parts
- More screencasts
- Still fairly young
- ✯ Outreach Makes People Happy

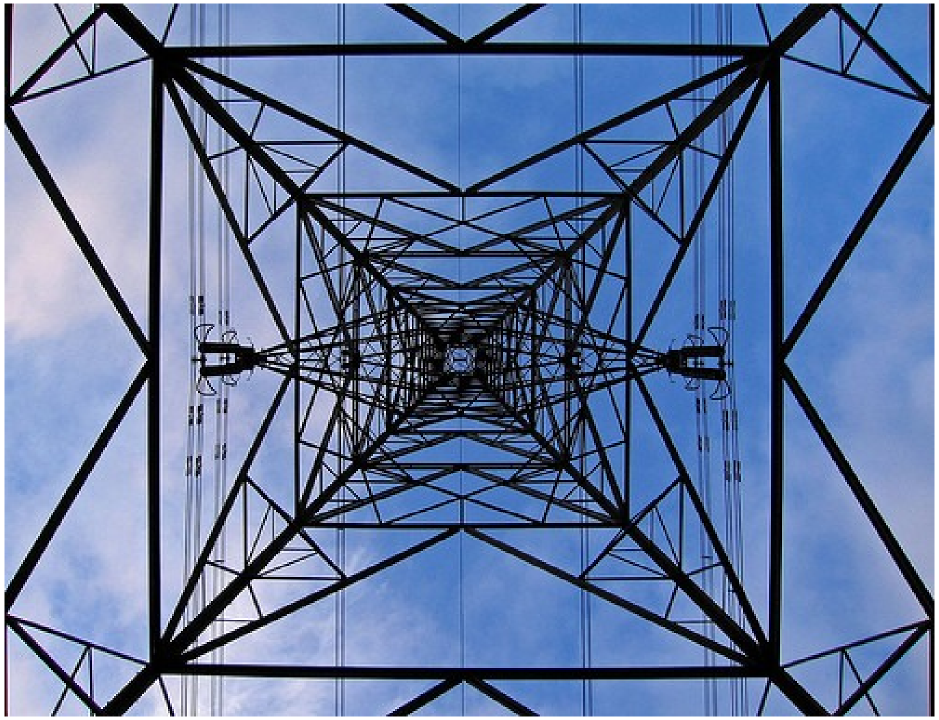

ADMIN INTERFACE
CACHE
CherryPy
WEB SERVER
URL REWRITE
SECURITY
TESTING
PYTHON
TEMPLATE ENGINE
KID
mochikit
OBJECT/RELATIONAL MAPPING
SQLObject
MIGRATION
DATABASE
USER IDENTITY

notes

- TurboGears is rumored to be being reimplented using Pylons as a foundation.
- Wikipedia Overview
  - http://en.wikipedia.org/wiki/TurboGears
- Main Site
  - http://turbogears.org/

books

- "Rapid Web Applications with TurboGears: Using Python to Create Ajax-Powered Sites", November 2006
  - http://www.amazon.com/Rapid-Web-Applications-TurboGears-Ajax-Powered/dp/0132433885
- "Professional Python Frameworks: Web 2.0 Programming with Django and Turbogears (Programmer to Programmer)", 2007
  - http://www.amazon.com/Professional-Python-Frameworks-Programming-Turbogears/dp/0470138092



Pylons

Pylons

- Reimplements Ruby on Rails
- Lots of Code Generators
- Flexibly Assembled with Python Paste

★ Flexibility has its limits

notes

- So flexible, I can't even tell you what modules are used or included.
- Really a paste for assembling custom web application servers.
- Many audience members suggest I'm unfairly harsh about its maturity.

links

- Wikipedia Overview
  - http://en.wikipedia.org/wiki/Pylons
- Main Site
  - http://pylonshq.com/
- Walk-through of building first Pylons site
  - http://www.rexx.com/~dkuhlman/pylons_quick_site.html

Content Management Systems

Content Management Systems

- Web Site Publishing
- On top of Web Application Servers
- One Serious Choice

☆ Great When It Just Works!

notes

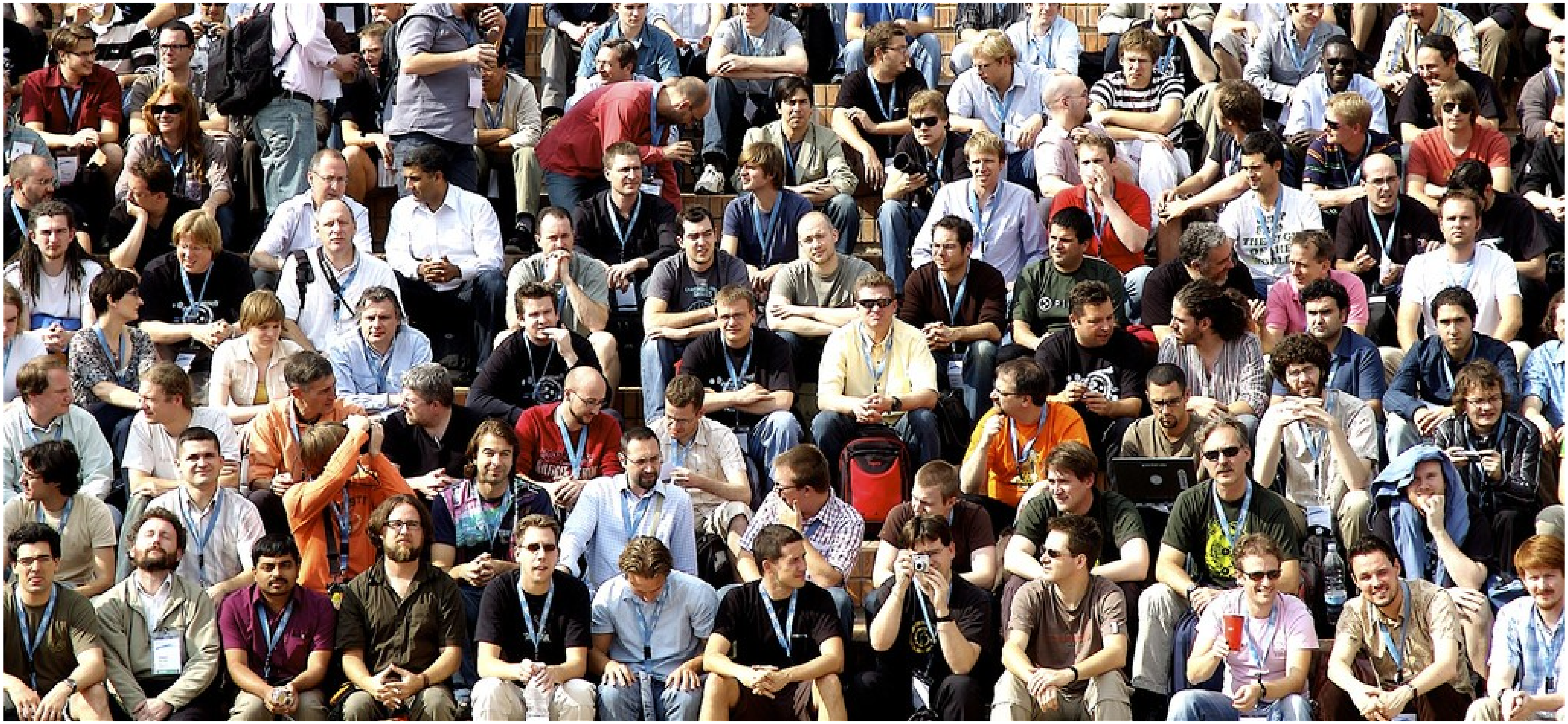
- Extends application server to include workflow, more user management, versioning, and testing sample audiences.
- Used for those who need to keep a web site running with fresh content.
- Some web frameworks try to add “CMS” modules.

Plone

Plone

- Built on Zope
- Strong Support Base
- Archetypes and Extendable Content Types
- Right Design Helps a Lot

Plone



notes

- Wikipedia Overview
  - http://en.wikipedia.org/wiki/Plone_(content_management_system)
- Main Site
  - http://plone.org/
- The Plone Network advocacy/referral site
  - http://plone.net/
- San Francisco Plone Users Group (Plone Lounge)
  - http://www.plonelounge.com/

Finished!

Summary

- Keep Exploring Python
- Wikipedia is Your Friend
- Ignore Version Numbers
- Learn until you are dead and buried.

notes

- This talk covered some of the squares of the large quilt that is Python.
- The “Cheese Shop” global Python modules list
  - http://pypi.python.org
- The CheeseRater module ratings
  - http://www.cheeserater.com

Summary

- Charles Merriam
- Phone: 408.368.6050
- Email: charles.merriam@gmail.com
- Blog: charlesmerriam.com/blog
- Slides and Notes: charlesmerriam.com/talk

Photo credits

Snakes on my head by O Pish Post
http://flickr.com/photos/opishposh/221395779/

Running Away by Ozyman
http://flickr.com/photos/ozyman/148793655/

Meta Mini Moo by Darkmatter
http://flickr.com/photos/cdm/1410539606/

Jagwangsa 3000 Buddha by Bebouchard
http://flickr.com/photos/bebouchard/404576063/

Unicode Eyechart in Firefox 2 by sillygwalio
http://flickr.com/photos/sillygwailo/543388337/

Vitamin E Redux by Selva
http://flickr.com/photos/selva/7737130/

Carry on up the Pylon by Arkadyevna
http://flickr.com/photos/arkadyevna/246047088/

Trains and Messes by Martin Kelley
http://flickr.com/photos/martin_kelley/408474778/

My Eye by OrangeAcid
http://flickr.com/photos/orangeacid/234358923/

Code Washing by Ezu
http://flickr.com/photos/ezu/277341190/

My New Mupal Tools by animaux
http://flickr.com/photos/animaux/103725084/

A "Few" Balloons by Mortimer
http://flickr.com/photos/mortimer/127194972/

Finished Quilt by Annette Pedrosian
http://flickr.com/photos/annettepedrosian/437261669/

Library by Swenwerk
http://flickr.com/photos/svenwerk/248594239/

behind... by Thomas23
http://flickr.com/photos/livenow/185186715/

Phil & Alex by Phil Greenspun

Lots of Plone Folk by MrTopF
http://flickr.com/photos/mrtopf/1543890810/

Dark Side of the Moon by Callumalden
http://flickr.com/photos/callumalden/48953166/