Writing Books Using Open Source Software

Wesley J. Chun
wescpy@gmail.com
http://corepython.com
Winter 2010

About The Talk

- We can write software using open source tools...
- Can the same be said of writing books?
- Hypothesis
  - What do *you* think?!? :-)
- Proof?
  - Give examples of open source tools used for software
  - Show some of those same tools used for book-writing
  - Discuss some authors and their tools

Software on the Shelf

Books on the Shelf

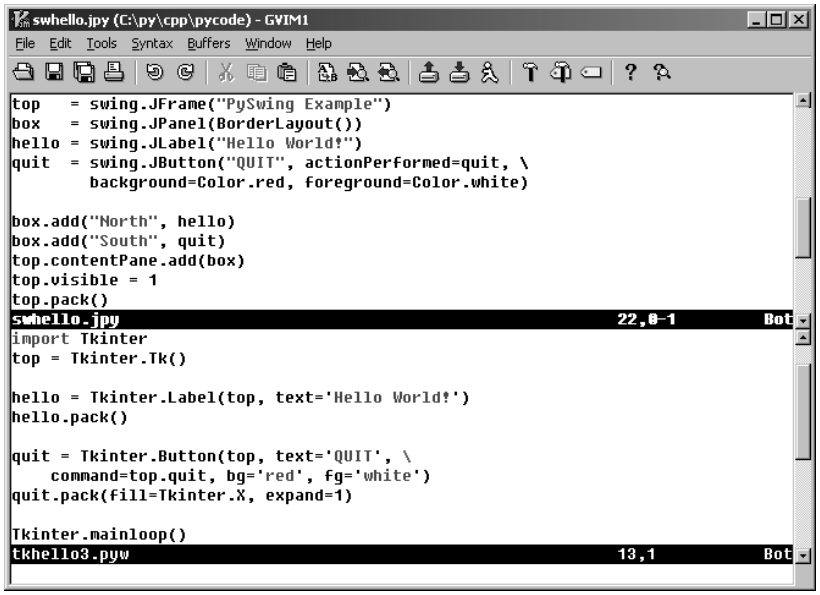
About The Speaker

- Software engineer by profession, also book author
- Skipped generations of writing tools
  - `troff/nroff` user but not LaTeX
  - FrameMaker user but not Word
  - Markdown, *wiki user but not DocBook nor Textile
  - reST is next...
- `vi` (now vim) user since 1985
  - Not totally naive; know some `emacs` too ;)
  - Prefer plain text editing (wonder why?)
    - XML doesn't count... :P

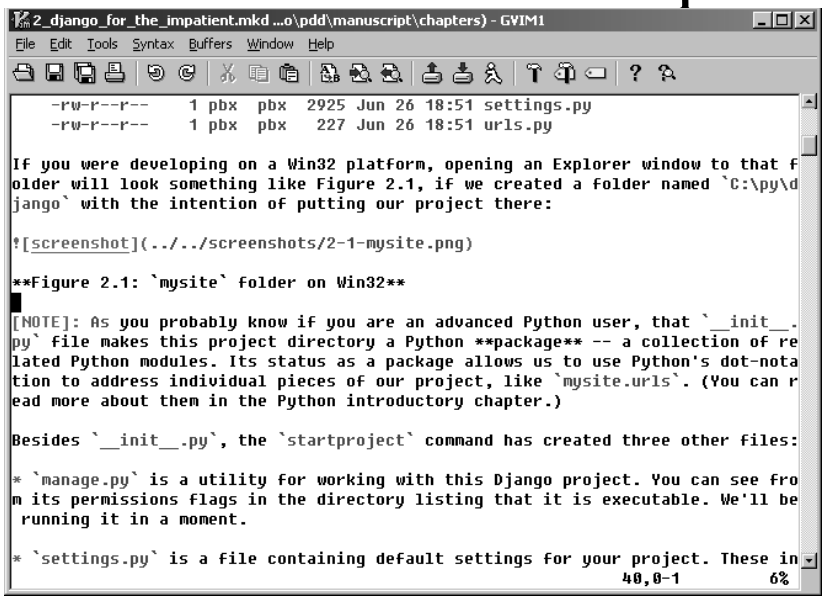
About the Development Tools

- Text editor(s)
- Syntax, formatting, and layout
- Versioning and backup
- Running tests
- Team communication
- Issue-tracking
- Generate for production

Text Editors: Source Code



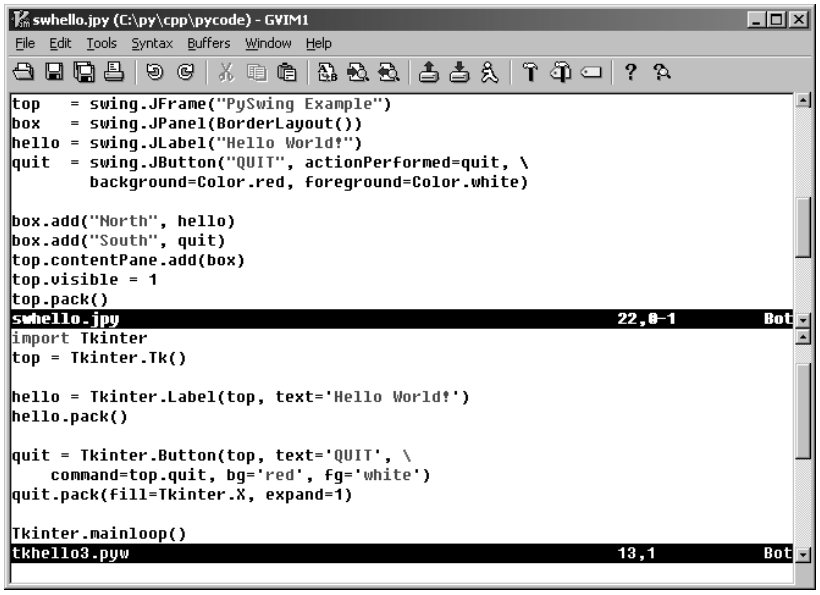
Text Editors: Manuscript



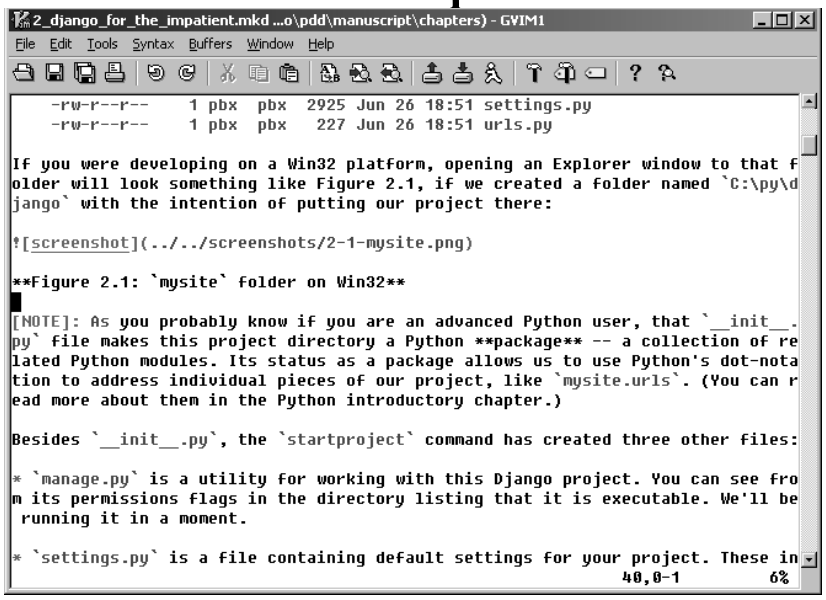
Syntax, Formatting, Layout (SFL)

- Writing software
  - Programming language syntax
    - Python, Perl, Ruby, VB
    - Java, C, C++; PHP
- Writing books
  - Markup Syntax
    - reST, Markdown, *wiki, Textile
    - DocBook-XML, SGML, LaTeX, *roff

SFL: Source Code



SFL: Manuscript "Source"



Versioning and Backup (V&B)

- Repository system
  - SCCS, RCS, CVS, SVN (Subversion)
  - git, hg (Mercurial), bzr (Bazaar)
- Backup and recovery
  - cp, rcp, scp/rsync, tar, cpio, dump
  - Amanda, Bacula, Mondo; DirSync, Unison; FlyBack, TimeVault
- Code vs. Manuscript
  - Both sources are in like formats (plain text)
    - Identical: managing files from repository
    - Backup & recover: works the same for both

Running Tests

- Code: Unit and regression tests
- Question: can you "test" a document?
- Sure! (to a certain extent)
  - For software, you're testing the code itself
  - For a book, you can test...
    - The code in your book if it is a technical book
    - "Validity" if markup system is "tagged" (XML, etc.)
- Python "docstrings" really help

Python Docstrings & `doctest` Module

```
"doctestDemo.py - demo doctest module"

def foo(x):
    """foo(x): display argument 'x'

       >>> foo(123)
       123
    """
    print x
```

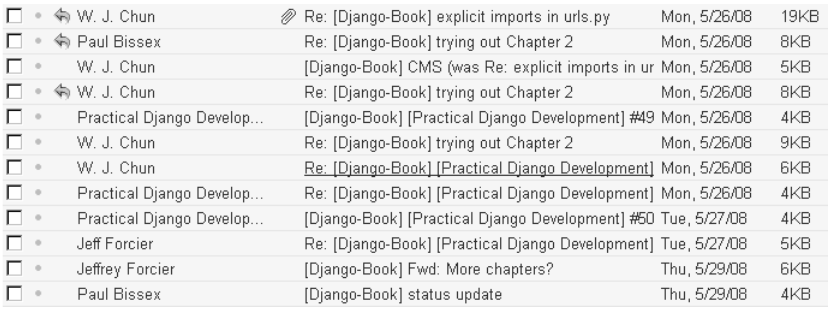
"Testing" a Manuscript

```
myMac$ make test
../../tools/test_snippets.py introduction.txt 1_practical_python...

No tests in introduction.txt

Testing 1_practical_python_for_django.txt

**********************************************************************
File "1_practical_python_for_django.txt", line 1288
Failed example:
    if m is not None: print m.group()
Expected:
    'foo'
Got:
    foo

**********************************************************************
1 items had failures:
   1 of 153 in 1_practical_python_for_django.txt
153 tests in 1 items.
152 passed and 1 failed.
***Test Failed*** 1 failures.

No tests in 2_django_for_the_impatient.txt

No tests in 3_starting_out.txt

Testing 4_model.txt

**********************************************************************
File "4_model.txt", line 531, in 4_model.txt
Failed example:
    from myapp.models import Person
Exception raised:
    Traceback (most recent call last):
      File "/Library/Frameworks/Python.framework/Versions/2.5/lib/python2.5/doct
est.py", line 1228, in __run
        compileflags, 1) in test.globs
      File "<doctest 4_model.txt[0]>", line 1, in <module>
        from myapp.models import Person
    ImportError: No module named myapp.models
          :
```

Team Communication

- Similar for software or book development
- Email
  - *various*
- Mailing List and Archive
  - Listserv, Majordomo, Mailman
  - Listproc, Lyris, SmartList

Email, Mailing List, Archive



W. J. Chun	Re: [Django-Book] explicit imports in urls.py	Mon, 5/26/08	19KB
Paul Bissex	Re: [Django-Book] trying out Chapter 2	Mon, 5/26/08	8KB
W. J. Chun	[Django-Book] CMS (was Re: explicit imports in ur	Mon, 5/26/08	5KB
W. J. Chun	Re: [Django-Book] trying out Chapter 2	Mon, 5/26/08	8KB
Practical Django Develop...	[Django-Book] [Practical Django Development] #49	Mon, 5/26/08	4KB
W. J. Chun	Re: [Django-Book] trying out Chapter 2	Mon, 5/26/08	9KB
W. J. Chun	Re: [Django-Book] [Practical Django Development]	Mon, 5/26/08	6KB
Practical Django Develop...	Re: [Django-Book] [Practical Django Development]	Mon, 5/26/08	4KB
Practical Django Develop...	[Django-Book] [Practical Django Development] #50	Tue, 5/27/08	4KB
Jeff Forcier	Re: [Django-Book] [Practical Django Development]	Tue, 5/27/08	5KB
Jeffrey Forcier	[Django-Book] Fwd: More chapters?	Thu, 5/29/08	6KB
Paul Bissex	[Django-Book] status update	Thu, 5/29/08	4KB

Issue-Tracking

- trac, RT, Bugzilla, Mantis, Redmine
- Code issues
  - New features
  - Bugs
  - Feature enhancements (or removals)
  - Milestones
- Manuscript "issues"
  - New material
  - "Bugs:" typos, errata
  - Future or outdated material
  - Milestones

Generate for Production

- For end-user consumption; software & books...
  - Undergo some production/release process
  - More similar than you would think
- Code
  - (Possible compilation process)
  - Install on production server
  - Shrinkwrapped software package
  - (Possible printing, boxing, distributing, media)
- Manuscript
  - HTML, PDF®, XML or other formatting "compilation"
  - Proofreading/editing(*) and layout
  - Printing, binding, distributing, perhaps media

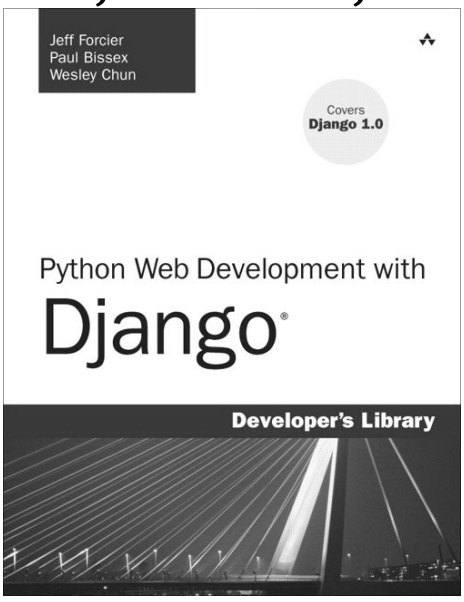
Just run `make` (!)

```
myMac$ make allzip
rm -f *0706.zip
markdown introduction.txt -x wtables >> introduction.html
markdown 1_practical_python_for_django.txt -x wtables >> 1_practical_python_for_
django.html
markdown 2_django_for_the_impatient.txt -x wtables >> 2_django_for_the_impatient
.html
            :
cd .. && zip chapters/html0706.zip chapters/introduction.html chapters/1_practic
al_python_for_django.html chapters/2_django_for_the_impatient.html [...]
  adding: chapters/introduction.html (deflated 54%)
  adding: chapters/1_practical_python_for_django.html (deflated 67%)
            :
html2ps introduction.html | ps2pdf - introduction.pdf
html2ps 1_practical_python_for_django.html | ps2pdf - 1_practical_python_for_dja
ngo.pdf
            :
cd .. && zip chapters/pdf0706.zip chapters/introduction.pdf chapters/1_practical
_python_for_django.pdf chapters/2_django_for_the_impatient.pdf [...]
  adding: chapters/introduction.pdf (deflated 34%)
  adding: chapters/1_practical_python_for_django.pdf (deflated 26%)
            :
  adding: chapters/matter/colophon.pdf (deflated 42%)
myMac$
```

Case Studies

- Forcier, Bissex, Chun
- Ramm, Dangoor, Sayfan
- Mertz
- Beazley
- Martelli
- Summerfield
- Chun

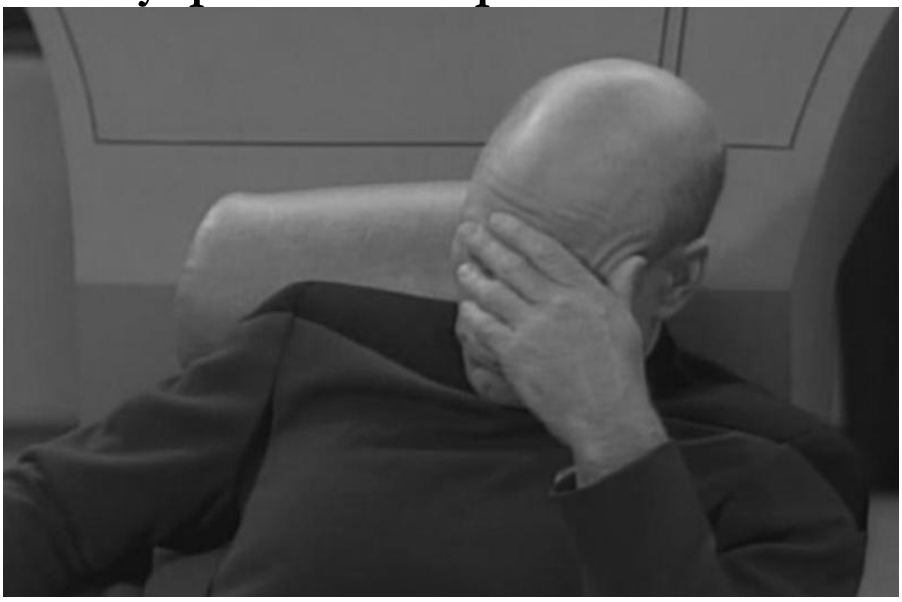
J. Forcier, P. Bissex, W. Chun



Python Web Development w/Django

- Editing: vim, emacs, TextMate
- Format: Markdown
- Repository: svn
- Issue-tracking: trac
- Mailing list: Mailman
- Conversion: HTML (Markdown) => PS (html2ps) => PDF (ps2pdf)
- Build: make
- Miscellaneous: "try_excerpt" tool cuts-n-pastes code snippets; "test_snippets" tool tests code execution in manuscript
- FUTURE: reST (format), Sphinx (conversion), Redmine (issue-tracking), hg (repository)

Irony: publisher imported into Word



Our `makefile`

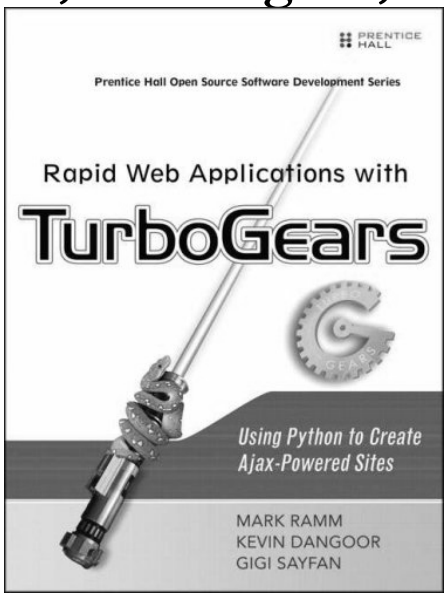
Printed by wesc

Jul 20, 08 21:24 **makefile** Page 1/1

```
# Python Web Development with Django makefile
#
# created by wesc on 2007 oct 20
#
# $Id: makefile 728 2008-07-21 03:24:10Z wchun $

MANUSCRIPT = manuscript.html
FOLDER = chapters/
NAMES = $(shell cat order)
SRCS = $(addsuffix .txt,$(NAMES))
MKDS = $(addsuffix .mkd,$(NAMES))
OBJS = $(addsuffix .html,$(NAMES))
PDFS = $(addsuffix .pdf,$(NAMES))
TSTAMP = $(shell date +'%m%d')

%.html: %.txt
        @echo "<html><head><title>$(basename $<)</title></head><body>" > $@
        python -m markdown $< -x wtables >> $@
        @echo "</body></html>" >> $@

html: $(OBJS)

manuscript: html
        cat $(OBJS) > $(MANUSCRIPT)

%.pdf: %.html
        html2ps $< | ps2pdf - $@

pdf: $(PDFS)

%.mkd: %.txt
        ln -s $(shell basename $<) $@

mkd: $(MKDS)

zip:
        @echo "zip" deprecated, try "pdfzip", "htmlzip", or "allzip" instead

allzip: htmlzip pdfzip

pdfzip: clean_zip pdf
        cd .. && zip $(FOLDER)pdf$(TSTAMP).zip $(addprefix $(FOLDER),$(PDFS))

htmlzip: clean_zip html
        cd .. && zip $(FOLDER)html$(TSTAMP).zip $(addprefix $(FOLDER),$(OBJS))

test: $(SRCS)
        ../../tools/test_snippets.py $(SRCS)

browse: html
        for file in $(OBJS); do open $$file; done

clean: clean_zip
        $(RM) $(OBJS) $(PDFS) $(MKDS) $(MANUSCRIPT)

clean_zip:
        $(RM) *$(TSTAMP).zip
```

Friday July 03, 2009 1/1

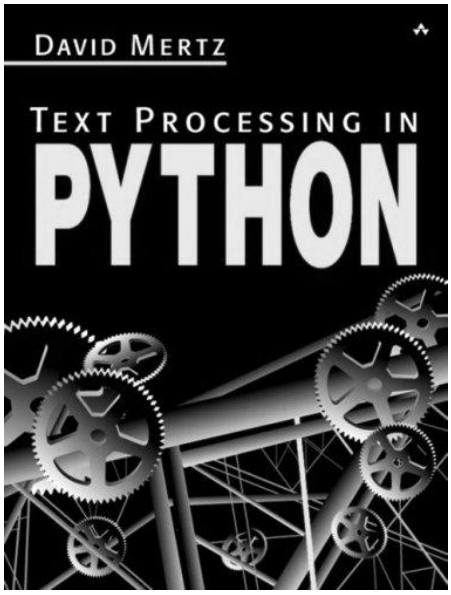
M. Ramm, K. Dangoor, G. Sayfan



Rapid Web Apps with TurboGears

- Format: Markdown
- Conversion: HTML (Markdown) => PDF (Adobe InDesign) -- all subsequent edits in InDesign
- Repository: svn
- Issue-tracking: trac
- FUTURE: reST (format), Sphinx (conversion), hg (repository)

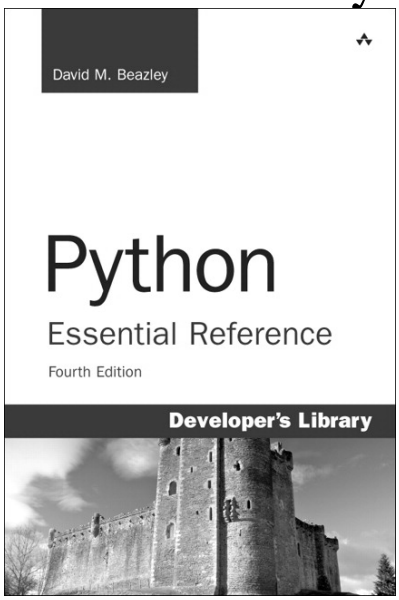
David Mertz



Text Processing in Python

- Preferences
  - Using a plain text editor
  - Extremely minimal markup language
- Editing: Boxer (OS/2), jEdit (plain text editors)
- Markup: "Smart ASCII"
- Conversion: LaTeX (Python+shell scripts) => PDF (LaTeX tools)
- FUTURE: same as above but add repository

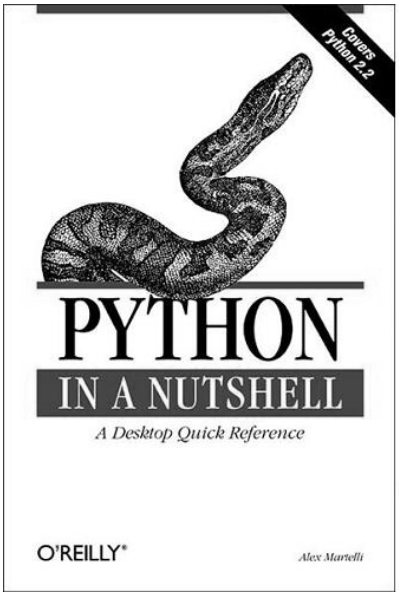
David Beazley

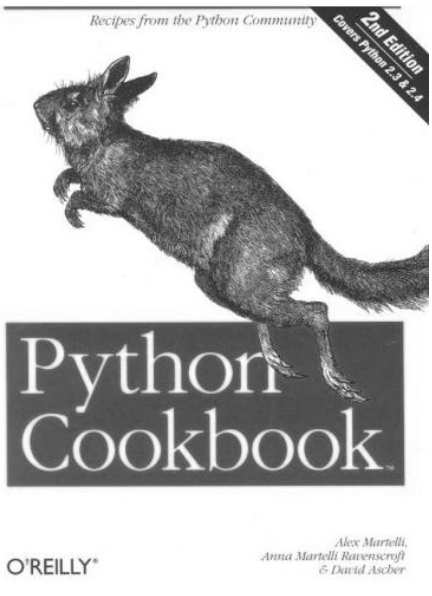


Python Essential Reference

- Editing: emacs
- Format: plain text (1st edition only)
- Conversion: imported into Word
- Miscellaneous
  - Created own indexer in Python
  - Subsequent edits stay in Word
  - Next editions use final master from previous
- FUTURE: not sure but better be plain-text editable

Alex Martelli





Alex Martelli

- *Python in a Nutshell*
  - Editing: Word
    - "EEK, what a nightmare: never EVER again"
- *Python Cookbook*
  - Markup: Docbook (XML)
  - Editing: Oxygen (XML editor), vim and gvim
  - Conversion: lots of Python scripts
  - Repository: svn
- FUTURE: XMLmind (editing), hg (repository)

Mark Summerfield

Mark Summerfield

- *Rapid GUI Programming with Python and Qt*
- *Programming in Python 3*
  - Markup: lout
    - Similar to LaTeX but only does PS... no HTML, etc.
  - Editing: vim and gvim
  - Conversion: PS (lout) => PDF (ps2pdf)
  - Miscellaneous: "snip" tool that cuts-n-pastes code snippets

Wesley Chun

Wesley Chun

- *Core Python Programming*
- Masters Thesis
- (Employers') technical manuals
  - Editing: Adobe FrameMaker
  - Conversion: PS and PDF (FrameMaker)
- *Python Fundamentals*
  - Markup: text w/minimal markup for **bold** & *italics*
  - Editing: vim and gvim
  - Conversion: imported into Word

Conclusion?

- Introduced our hypothesis
- Demonstrated tools, usage for writing software & books
- Gave numerous examples of what authors use today
- Major outstanding issue
  - Publisher still want .doc
    - They will import your book to Word
  - Manuscript files edited by publisher

Disconnect

- Edits/corrections not propagated back to source files
  - Generated HTML imported elsewhere (e.g., into Word)
    - All further editing with new master
  - Authors may do it themselves only...
    - If they insist and there is time
    - If they *have* time
- Not applicable for developing software

The Future and You

- Using and/or supporting open source?
- Considering writing a book?
- Afraid you can't stick with the tools you're familiar with?
- Think you can't publish unless you submit files in a proprietary word processor format?
- **Think again**

Conclusion

- Industry Changes
  - Hardware: proprietary *ix OSs to Linux, *BSD, etc.
  - Software: proprietary tools to open source
- Book writing seems to follow the same trend
- For coders, writing a book in plain text makes you a happier person
  - Spend less time as a fish out of water
- Problem to solve: publishers still want Word... solutions?
  - reST to doc/x converter? (need both ways though)
  - May have to force publishers to be more flexible
    - Convince publishers to accept other formats?
    - Find editors who are more open to open source tools?
  - Else produce camera-ready PDFs on your own

FINIS