60 Minutes of MacPython

Author

Bob Ippolito

Conference

PyCon DC, March 2004

Intended Audience

- Python developers using Mac OS X 10.2 or later
- Spies from the Linux and Win32 camps

Topics

- What is MacPython?
- MacPython IDEs
- Darwin
- Apple Events
- Cocoa
- Cross-Platform
- Community
- What's Next

What is MacPython?

- It's just Python
- Compiled from the same Python source tree

MacPython's Past

- MacPython was for "Classic" MacOS
- Guido used to own a Mac
- ... then Jack took over
- Uses WASTE for text widgets
- ... and GUSI to act more like *NIX
- Still exists, but not as actively maintained
- ... though apparently it does have *some* users

MacPython Now

- Has some rough edges
- ... but we're working on it!
- Inherits the good stuff from BSD
- Has many new APIs of its own
- Ancient MacOS code happens to work
- ... and we still use a lot of it

MacPython Tools

BuildApplet

Turns simple scripts into applications

PackageManager

Installs a few packages from a central db

PythonLauncher

Allows double-clicking of .py files

PythonIDE

Old MacOS IDE, unmaintained

... all of these are being replaced!

How do I get it?

OS X 10.2

- DO NOT USE Apple Python 2.2.0
- Get MacPython-Jaguar 2.3

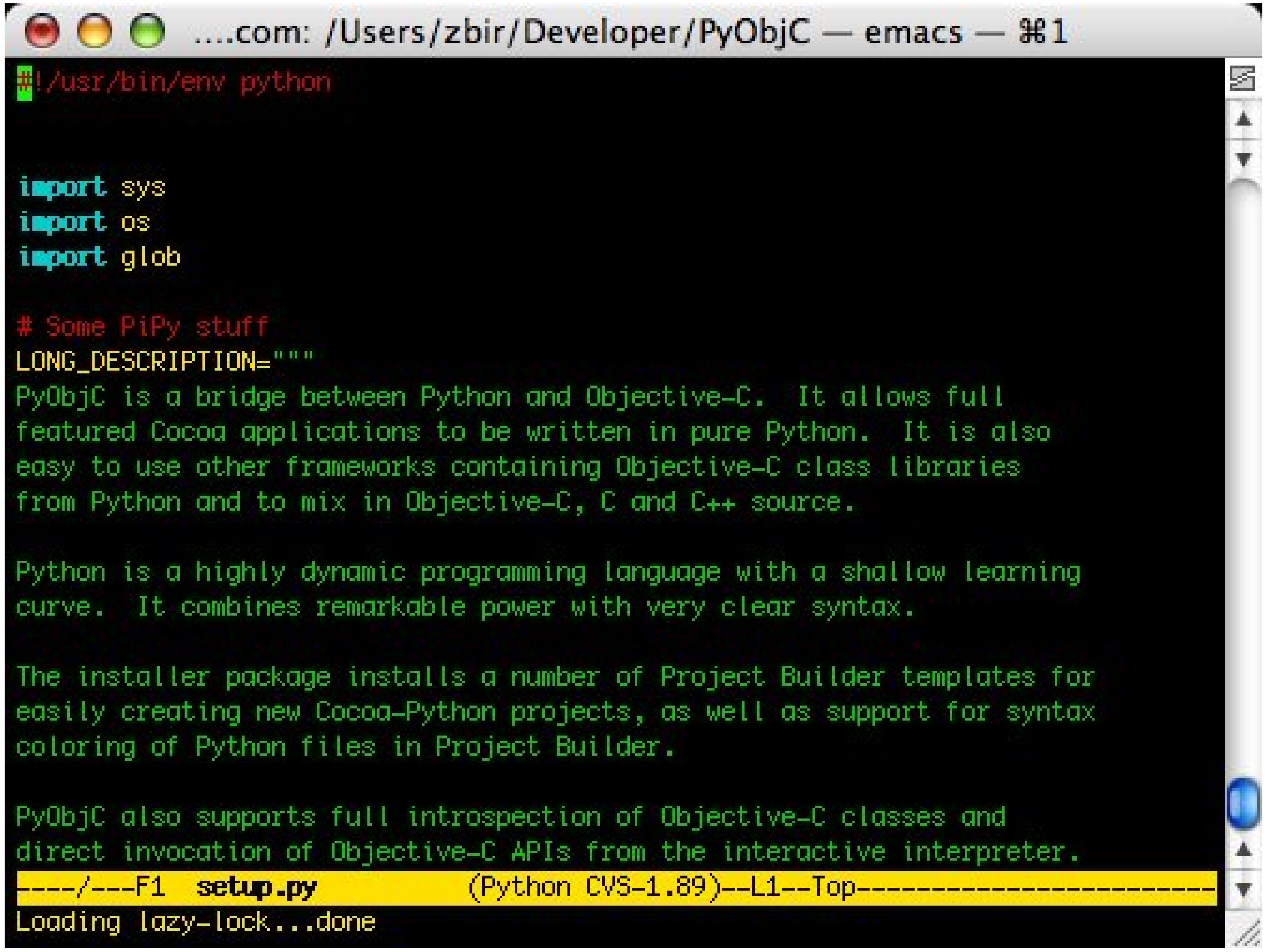
OS X 10.3

- PLEASE USE Apple Python 2.3.0
- ... and don't install any other Python
- Also get the MacPython-Addons

MacPython IDEs

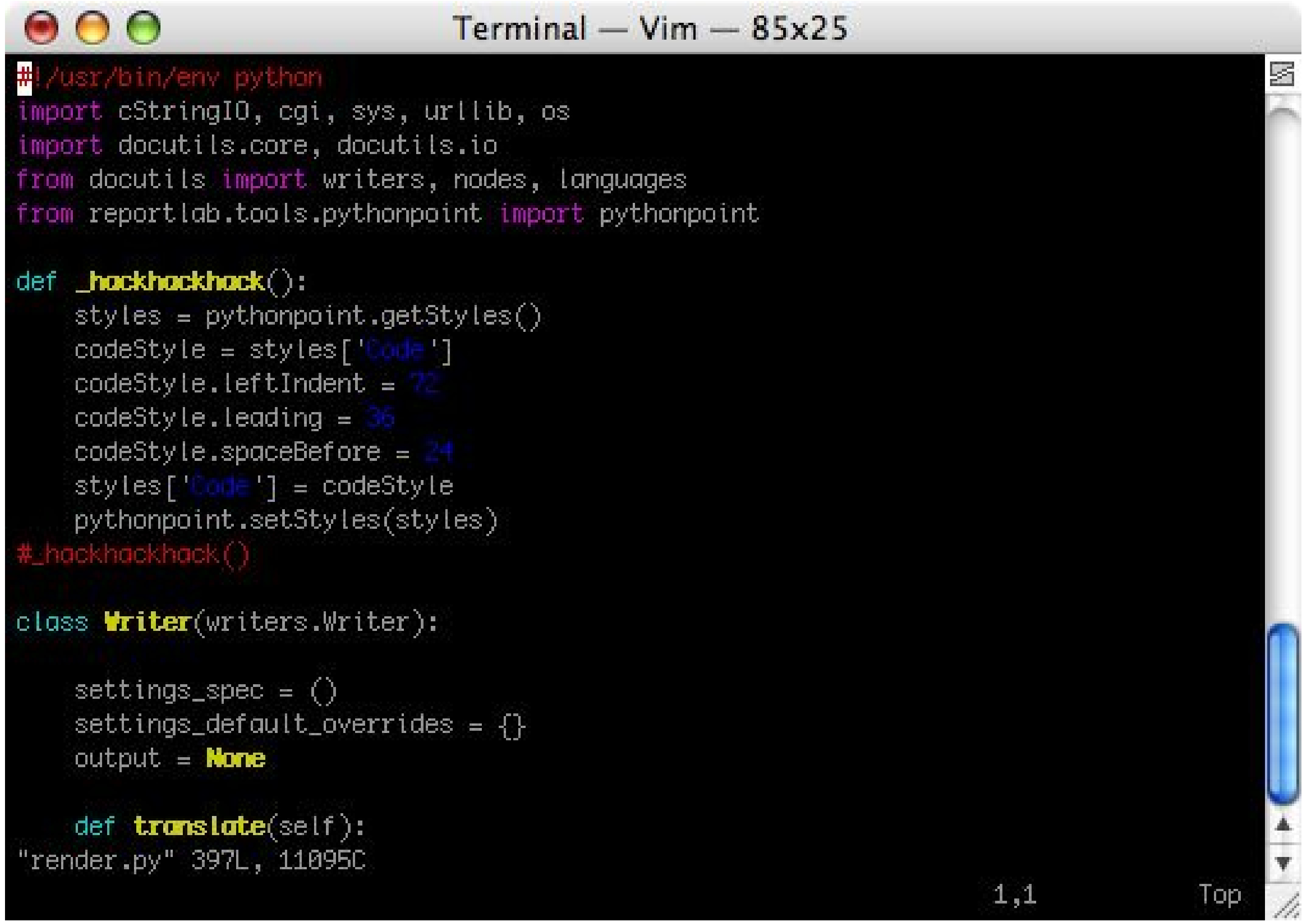
- General Purpose IDEs
- Aqua Python IDEs
- Other Python IDEs

Emacs



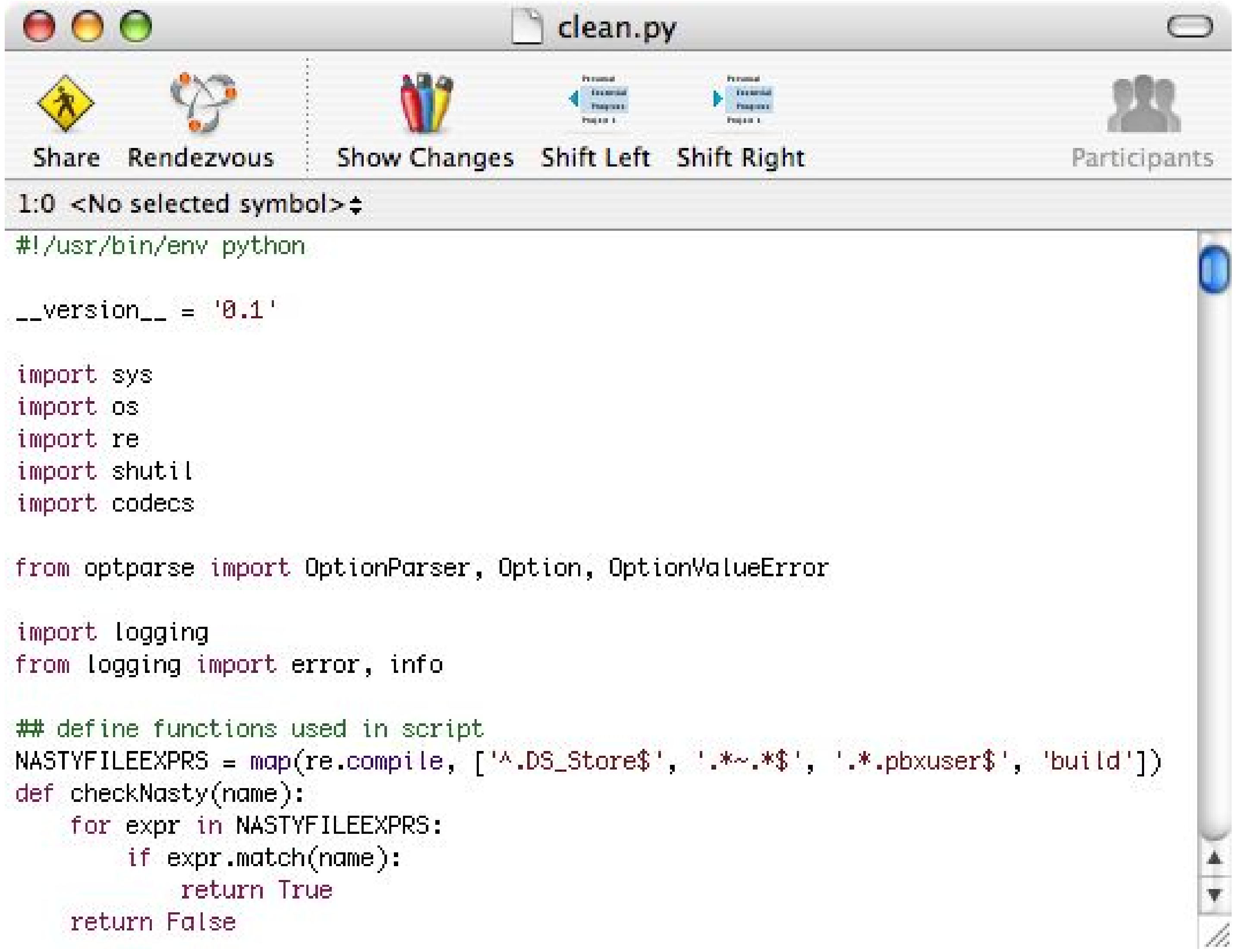
```
#!/usr/bin/env python


import sys
import os
import glob

# Some PiPy stuff
LONG_DESCRIPTION="""
PyObjC is a bridge between Python and Objective-C.  It allows full
featured Cocoa applications to be written in pure Python.  It is also
easy to use other frameworks containing Objective-C class libraries
from Python and to mix in Objective-C, C and C++ source.

Python is a highly dynamic programming language with a shallow learning
curve.  It combines remarkable power with very clear syntax.

The installer package installs a number of Project Builder templates for
easily creating new Cocoa-Python projects, as well as support for syntax
coloring of Python files in Project Builder.

PyObjC also supports full introspection of Objective-C classes and
direct invocation of Objective-C APIs from the interactive interpreter.
----/---F1  setup.py          (Python CVS-1.89)--L1--Top-----------------------
Loading lazy-lock...done
```

Vim



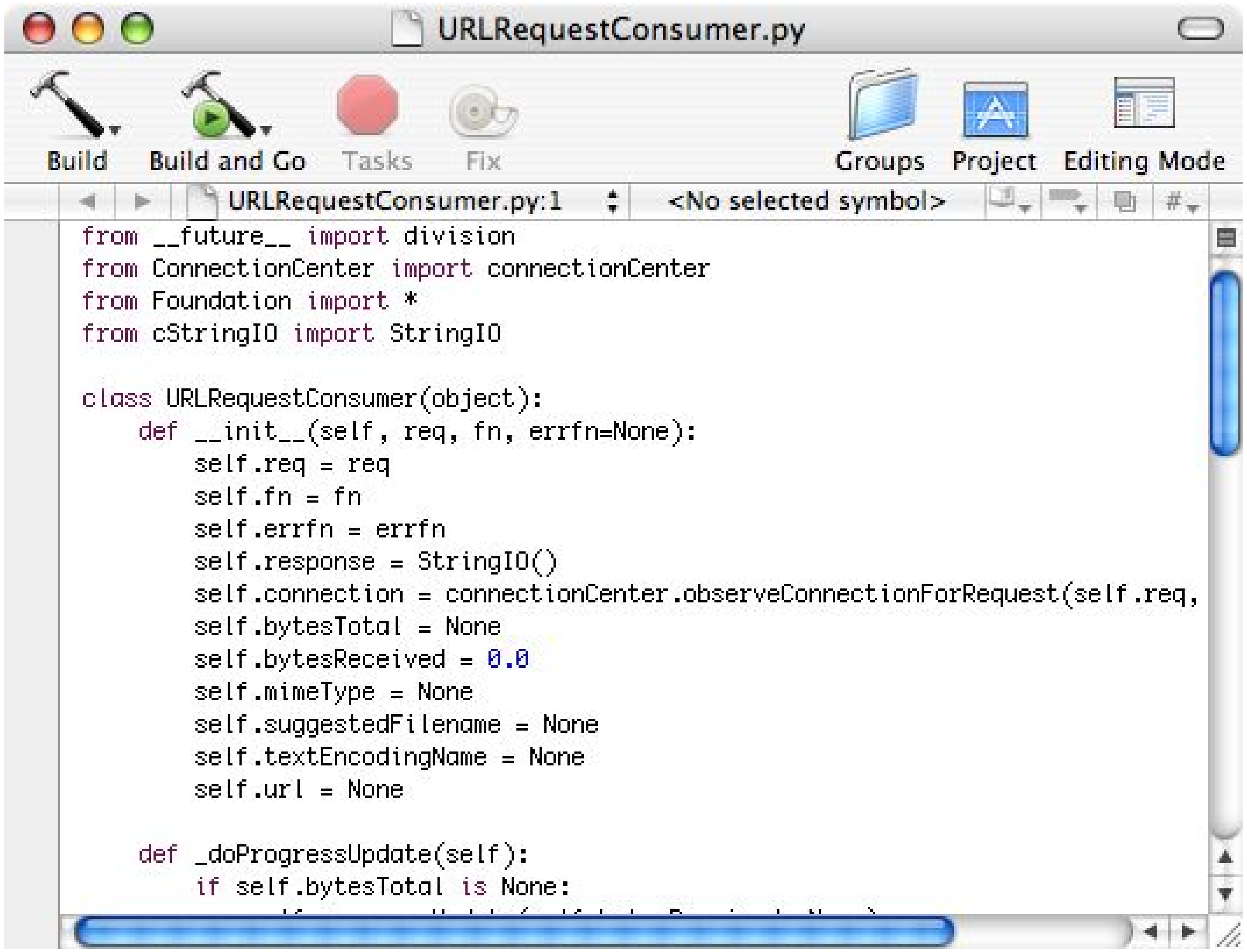
```
#!/usr/bin/env python
import cStringIO, cgi, sys, urllib, os
import docutils.core, docutils.io
from docutils import writers, nodes, languages
from reportlab.tools.pythonpoint import pythonpoint

def _hackhackhack():
    styles = pythonpoint.getStyles()
    codeStyle = styles['Code']
    codeStyle.leftIndent = 72
    codeStyle.leading = 36
    codeStyle.spaceBefore = 24
    styles['Code'] = codeStyle
    pythonpoint.setStyles(styles)
#_hackhackhack()

class Writer(writers.Writer):

    settings_spec = ()
    settings_default_overrides = {}
    output = None

    def translate(self):
"render.py" 397L, 11095C
                                                                  1,1           Top
```

SubEthaEdit



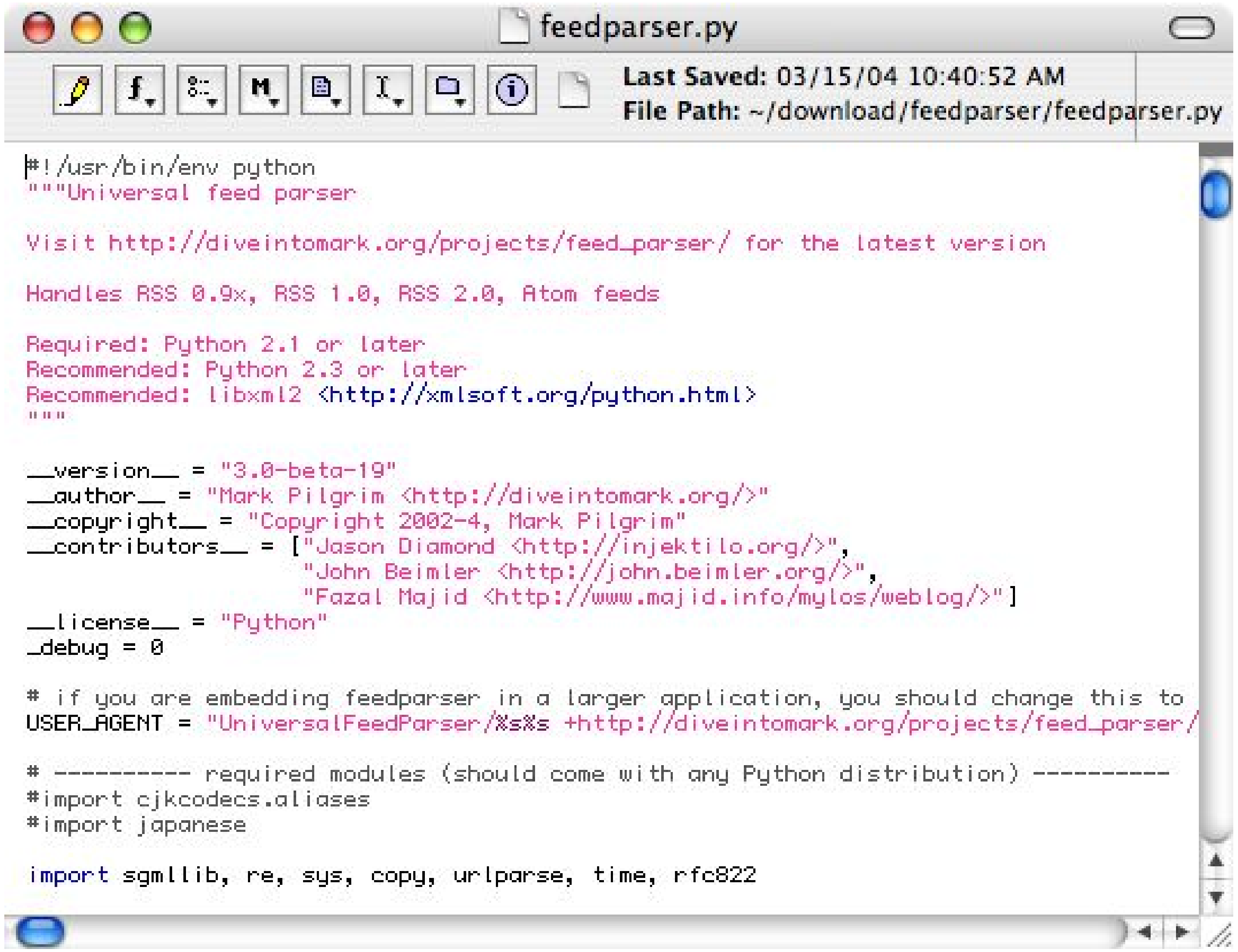
```
#!/usr/bin/env python

__version__ = '0.1'

import sys
import os
import re
import shutil
import codecs

from optparse import OptionParser, Option, OptionValueError

import logging
from logging import error, info

## define functions used in script
NASTYFILEEXPRS = map(re.compile, ['^.DS_Store$', '.*~.*$', '.*.pbxuser$', 'build'])
def checkNasty(name):
    for expr in NASTYFILEEXPRS:
        if expr.match(name):
            return True
    return False
```

Xcode



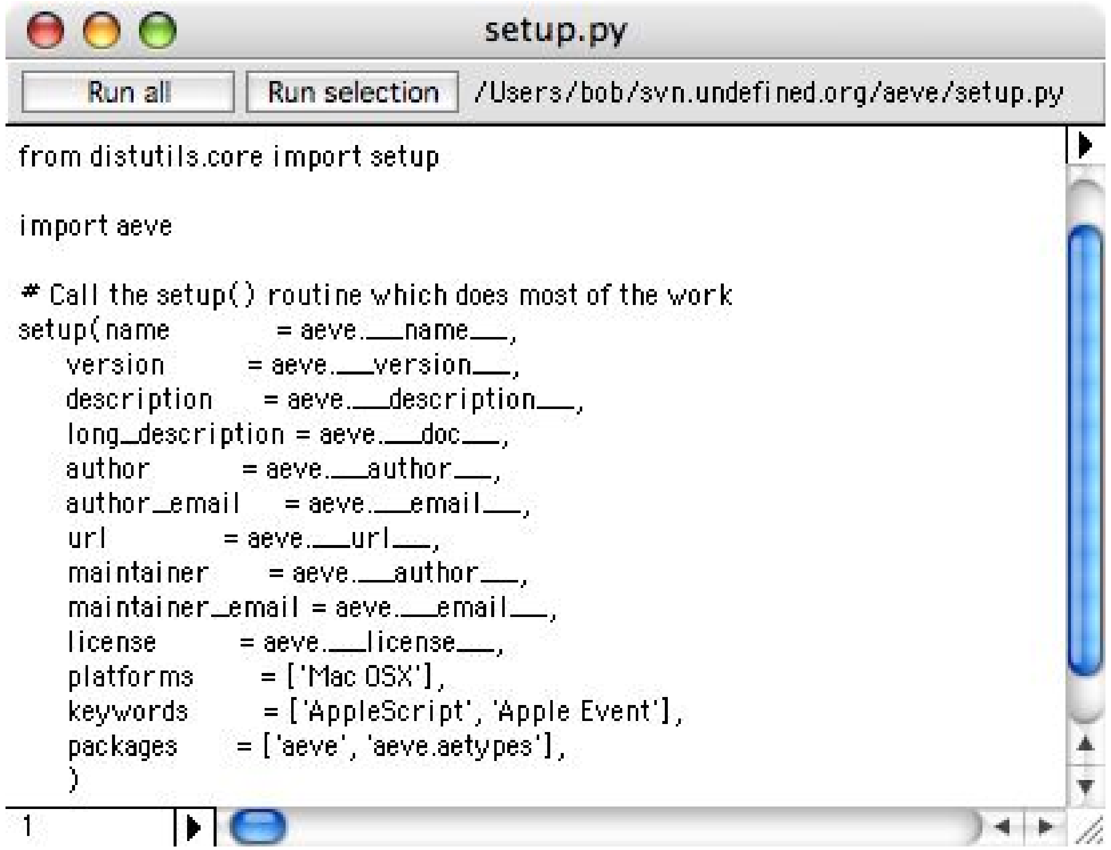
```
from __future__ import division
from ConnectionCenter import connectionCenter
from Foundation import *
from cStringIO import StringIO

class URLRequestConsumer(object):
    def __init__(self, req, fn, errfn=None):
        self.req = req
        self.fn = fn
        self.errfn = errfn
        self.response = StringIO()
        self.connection = connectionCenter.observeConnectionForRequest(self.req,
        self.bytesTotal = None
        self.bytesReceived = 0.0
        self.mimeType = None
        self.suggestedFilename = None
        self.textEncodingName = None
        self.url = None

    def _doProgressUpdate(self):
        if self.bytesTotal is None:
```

BBEdit


feedparser.py
Last Saved: 03/15/04 10:40:52 AM
File Path: ~/download/feedparser/feedparser.py

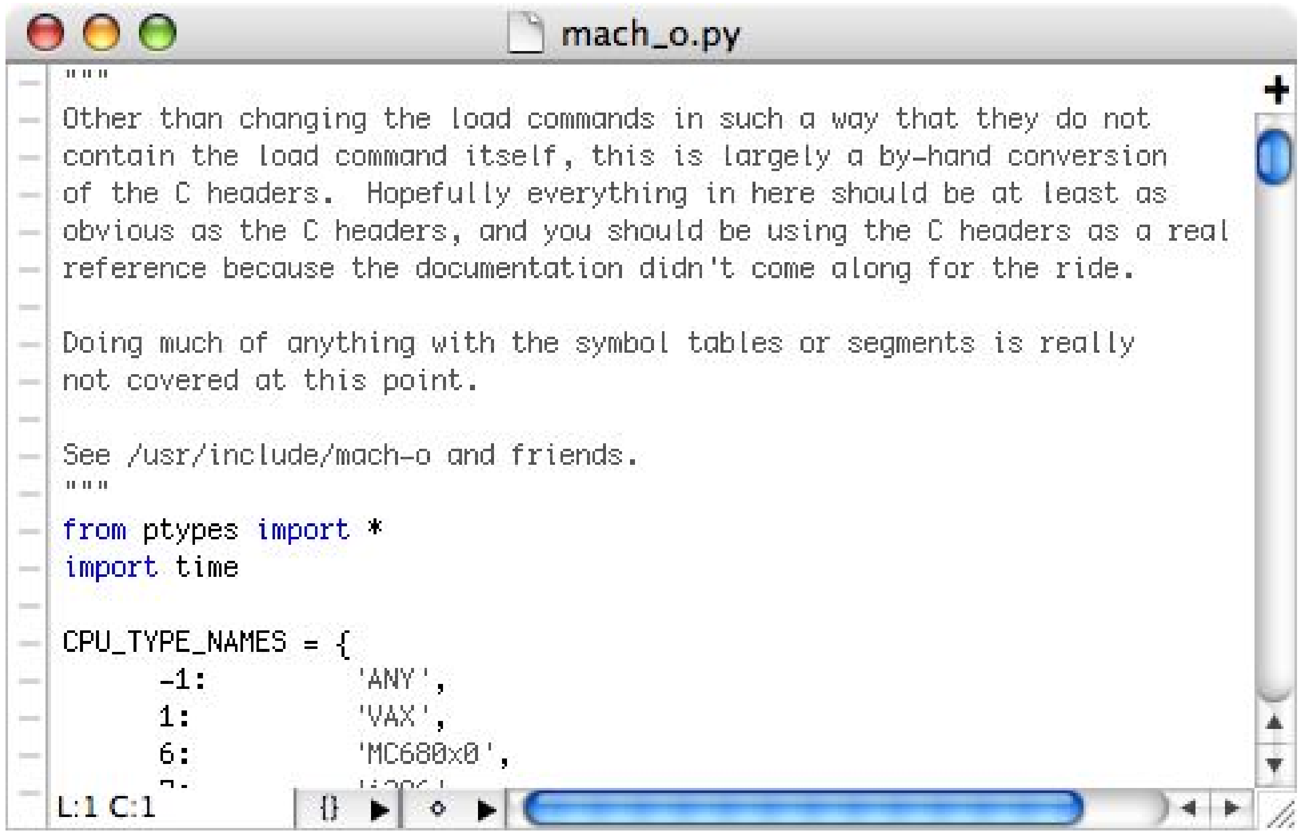
```
#!/usr/bin/env python
"""Universal feed parser

Visit http://diveintomark.org/projects/feed_parser/ for the latest version

Handles RSS 0.9x, RSS 1.0, RSS 2.0, Atom feeds

Required: Python 2.1 or later
Recommended: Python 2.3 or later
Recommended: libxml2 <http://xmlsoft.org/python.html>
"""

__version__ = "3.0-beta-19"
__author__ = "Mark Pilgrim <http://diveintomark.org/>"
__copyright__ = "Copyright 2002-4, Mark Pilgrim"
__contributors__ = ["Jason Diamond <http://injektilo.org/>",
                    "John Beimler <http://john.beimler.org/>",
                    "Fazal Majid <http://www.majid.info/mylos/weblog/>"]
__license__ = "Python"
_debug = 0

# if you are embedding feedparser in a larger application, you should change this to
USER_AGENT = "UniversalFeedParser/%s%s +http://diveintomark.org/projects/feed_parser/

# ---------- required modules (should come with any Python distribution) ----------
#import cjkcodecs.aliases
#import japanese

import sgmllib, re, sys, copy, urlparse, time, rfc822
```

PythonIDE



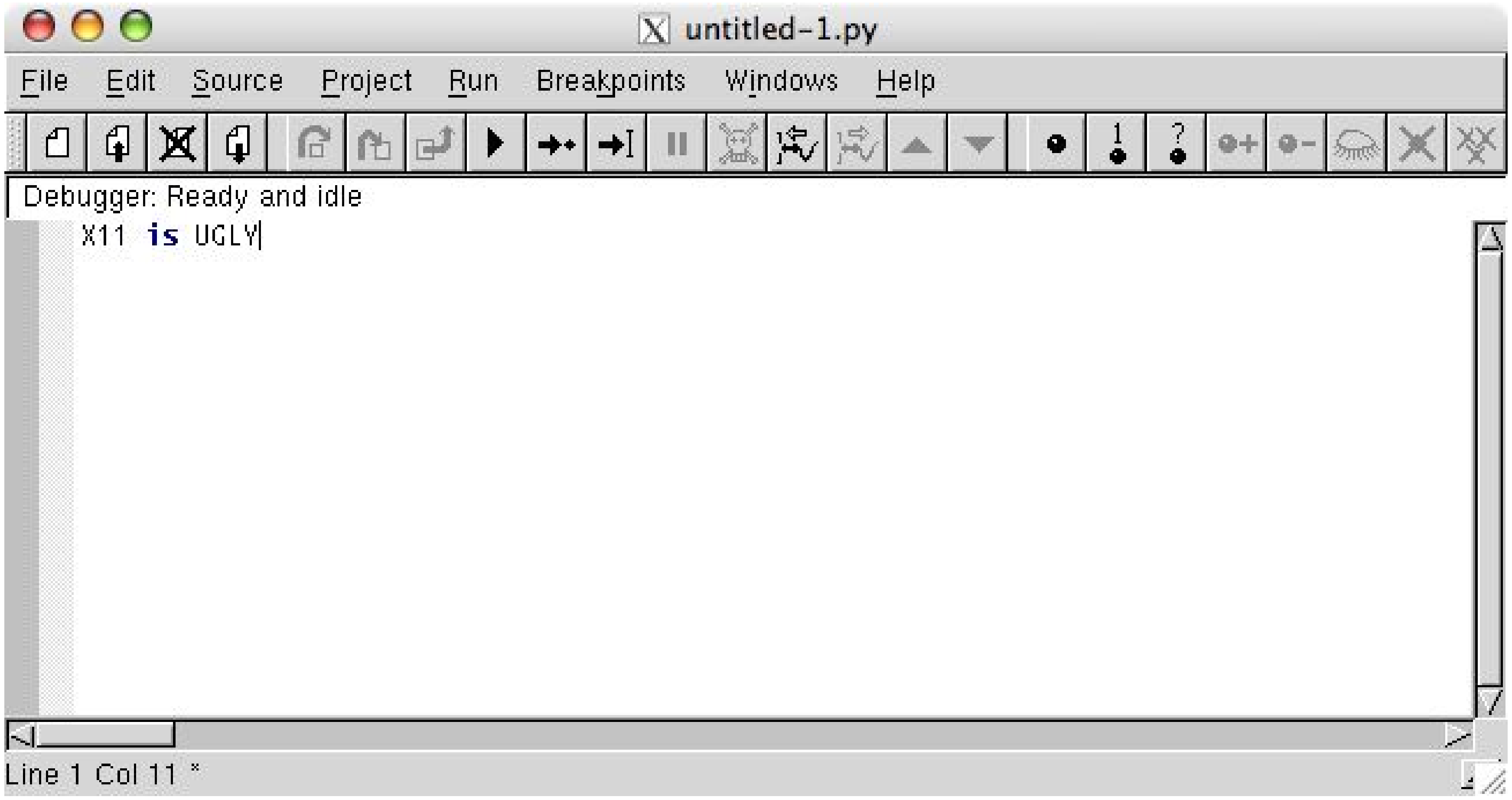
```
from distutils.core import setup

import aeve

# Call the setup() routine which does most of the work
setup(name          = aeve.__name__,
    version         = aeve.__version__,
    description     = aeve.__description__,
    long_description = aeve.__doc__,
    author          = aeve.__author__,
    author_email    = aeve.__email__,
    url             = aeve.__url__,
    maintainer      = aeve.__author__,
    maintainer_email = aeve.__email__,
    license         = aeve.__license__,
    platforms       = ['Mac OSX'],
    keywords        = ['AppleScript', 'Apple Event'],
    packages        = ['aeve', 'aeve.aetypes'],
    )
```

PyOXIDE



```
"""
Other than changing the load commands in such a way that they do not
contain the load command itself, this is largely a by-hand conversion
of the C headers.  Hopefully everything in here should be at least as
obvious as the C headers, and you should be using the C headers as a real
reference because the documentation didn't come along for the ride.

Doing much of anything with the symbol tables or segments is really
not covered at this point.

See /usr/include/mach-o and friends.
"""
from ptypes import *
import time

CPU_TYPE_NAMES = {
        -1:         'ANY',
        1:          'VAX',
        6:          'MC680x0',
```

X11 Python Editors are Ugly

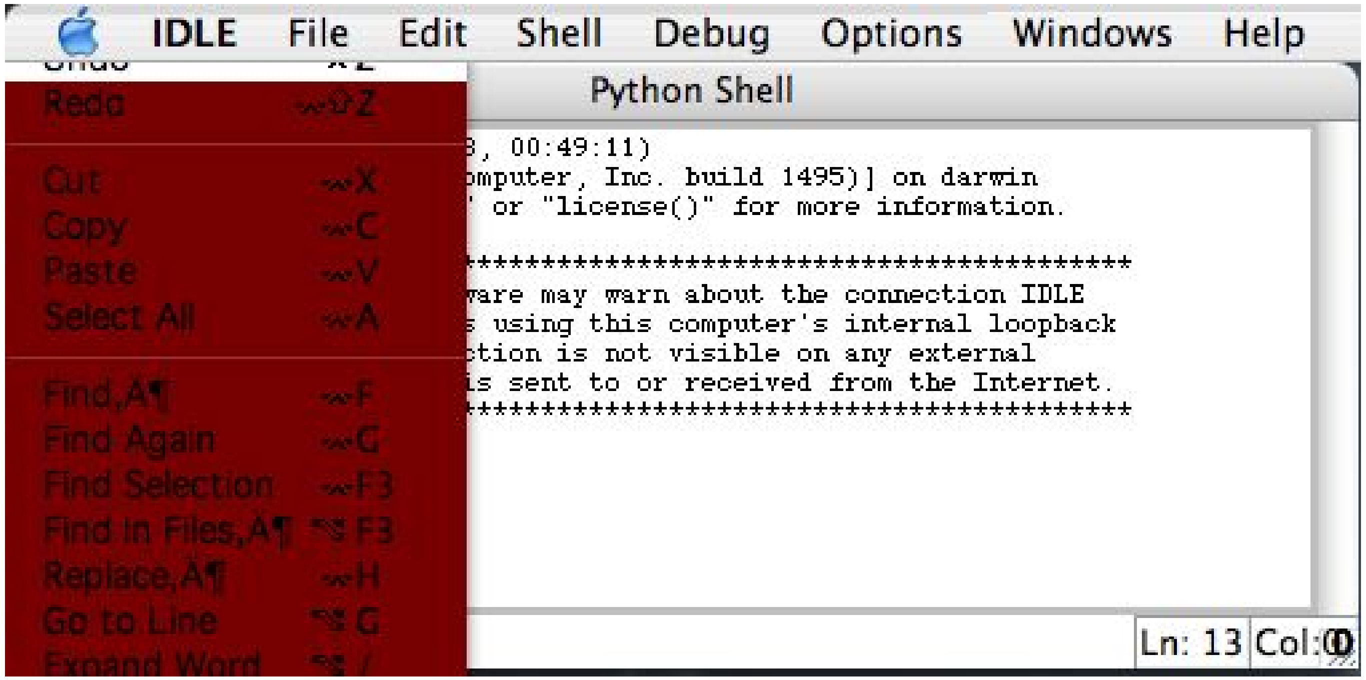


Debugger: Ready and idle

```
X11 is UGLY
```

Line 1 Col 11 *

Cross-Platform Toolkits Are Broken



```
3, 00:49:11)
omputer, Inc. build 1495)] on darwin
' or "license()" for more information.

******************************************
ware may warn about the connection IDLE
s using this computer's internal loopback
ction is not visible on any external
is sent to or received from the Internet.
******************************************
```

Darwin

- os.name == 'posix' ... but it's not quite BSD
- Forking Files
- Three Flavors of Property Lists
- Objective C?
- Bundles Bundles Bundles

os.name == 'posix'

So, it is a *NIX?

- Typically acts like any other BSD
- Especially in modules such as os, sys, socket
- ... but there's a lot more to it

... but it's not quite BSD

HFS+ Thinks Different

- Case insensitive file system
- Files can have multiple forks

Inherited from NeXT

- XML property lists for configuration
- Bundles and Frameworks
- dyld is not your average linker
- Tons of new APIs

Forking Files

Resource forks are Not For POSIX

- Files may have a resource fork
- Not really useful without Carbon APIs
- POSIX layer sees only the data fork!
- Be careful not to lose them when copying files!

But you can get at one if you need to:

```
file(filename + '../namedfork/rsrc')
```

Three Flavors of Property Lists

- Binary
- Text
- XML

Binary Property Lists

- Deprecated, but still used
- Object serialization (pickle)
- Old Nib files (GUI pickle)

```
>>> open('keyedobjects.nib').read()
'bplist00\xd4\x00\x01\x00\x02\x00\x03\x00\x04\x00\x05\x00\x06\x00\x07
\x00\nY$archiverX$versionT$topX$objects_\x10\x0fNSKeyedArchiver\x12\x00
\x01\x86\xa0\xd1\x00\x08\x00\t]IB.objectdata\x80\x01\xaf\x11\x04#\x00
\x0b\x00\x0c\x00+\x00/\x003\x00:\x00=\x00?\x00C\x00G\x00\xa5\x00\xab\x00
\xbb\x00\xc2\x00\xc3\x00\xc4\x00\xc9\x00\xca\x00\xcb\x00\xcf\x00\xd1\x00
\xd2\x00\xd5\x00\xd7\x00\xdb\x00\xde\x00\xe1\x00\xe2\x00\xe3\x00\xe5\x00
\xe8\x00\xec\x00\xef\x00\xf0\x00\xf1\x00\xf3\x00\xf7\x00\xfb\x00\xff\x01
\x00\x01\x01\x01\x03\x01\x06\x01\n\x01\x0b\x01\x0c\x01\r\x01\x10\x01\x13
\x01\x0b\x01\x14\x01\x15\x01\x18\x01\x1a\x01\x1b\x01\x1c\x01 \x01#\x01$
...'
```

Text Property Lists

- Deprecated, but still used
- Looks sort of like YAML
- StartupItems (like /etc/init.d)

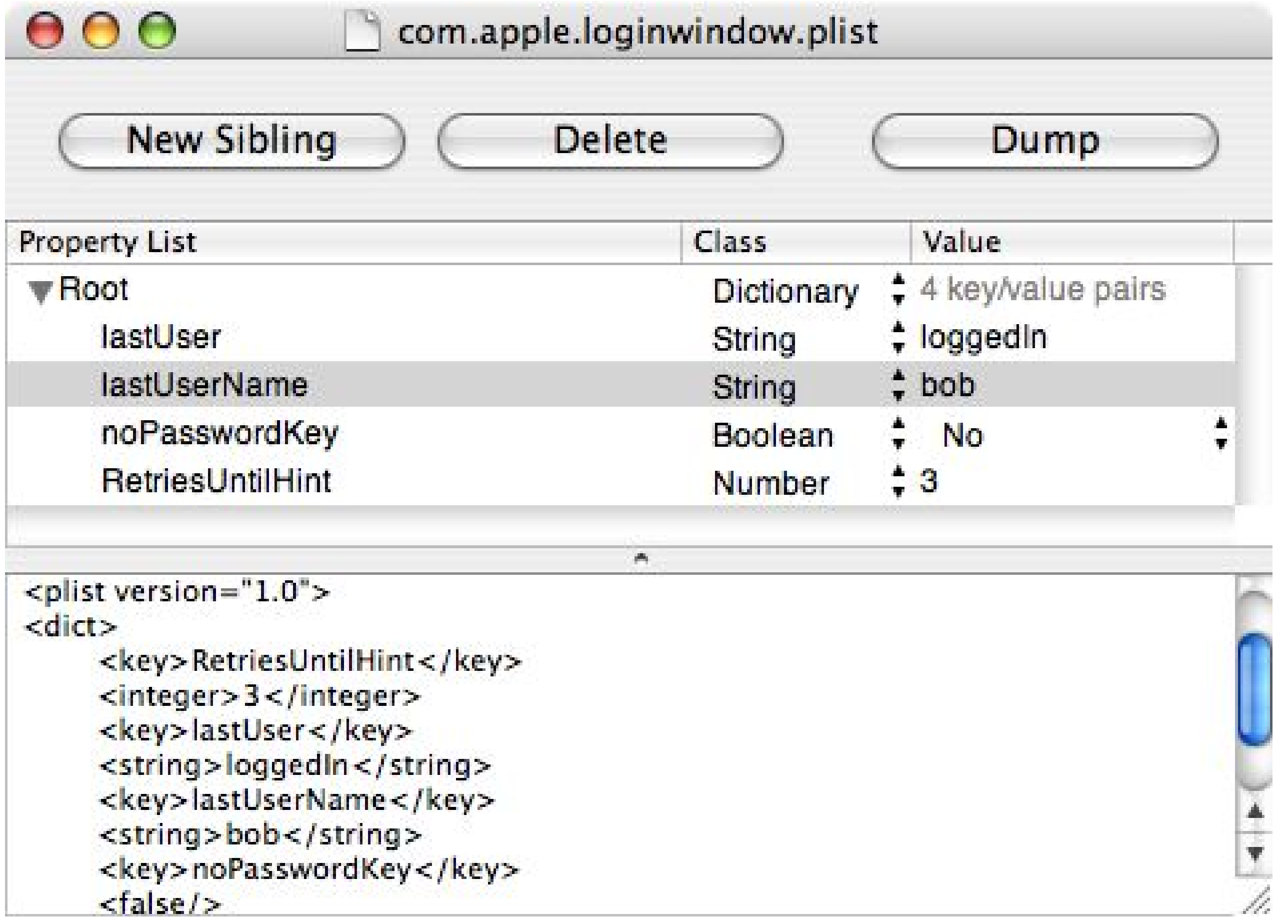
```
{
    Description     = "uControl";
    Provides        = ("uControl");
    /* depend on something so that we don't come too early */
    Requires        = ("Resolver");
    OrderPreference = "None";
    Messages = {
        start = "Starting uControl";
        stop  = "Stopping uControl";
    };
}
```

XML Property Lists

- Preferred
- Dictionary serialization
- Preference files (~/Library/Preferences)
- Info.plist in every application bundle
- New Nib files (GUI pickle)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist SYSTEM "file://localhost/System/Library/DTDs/PropertyLis
<plist version="0.9">
<dict>
    <key>CFBundleDevelopmentRegion</key>
    <string>English</string>
    <key>CFBundleDocumentTypes</key>
    <array>
        <dict>
            <key>CFBundleTypeOSTypes</key>
            <array>
                <string>****</string>
```

Apple's Property List Editor



Property List	Class	Value
▼Root	Dictionary	4 key/value pairs
lastUser	String	loggedIn
lastUserName	String	bob
noPasswordKey	Boolean	No
RetriesUntilHint	Number	3

```
<plist version="1.0">
<dict>
        <key>RetriesUntilHint</key>
        <integer>3</integer>
        <key>lastUser</key>
        <string>loggedIn</string>
        <key>lastUserName</key>
        <string>bob</string>
        <key>noPasswordKey</key>
        <false/>
```

plistlib Example

Using plistlib to find the name of the last logged in user:

```
>>> import plistlib
>>> filename = '/Library/Preferences/com.apple.loginwindow.plist'
>>> plist = plistlib.Plist.fromFile(file(filename))
>>> plist['lastUserName']
'bob'
```

Limitations:

- Only works for XML plist files!
- Not very good at handling dates
- Might not be unicode safe

NSDictionary Example

Using PyObjC to find the name of the last logged in user:

```
>>> from Foundation import NSDictionary
>>> filename = u'/Library/Preferences/com.apple.loginwindow.plist'
>>> plist = NSDictionary.dictionaryWithContentsOfFile_(filename)
>>> plist['lastUserName']
u'bob'
```

Limitations:

- Requires PyObjC to be installed
- ... but PyObjC is awesome, so get it!

Why Objective C

- Comes from NeXT
- Apple has a lot of great frameworks that use it
- ... like Cocoa and Foundation

Objective C Portability

- The GNUStep project uses it too
- ... and PyObjC is portable to GNUStep
- GNUStep works on Win32
- ... but doesn't have a decent backend

What is Objective C

- Mostly just C, but with [some new:syntax]
- ... and a very dynamic runtime (sound familiar?)
- Simpler than C++, similar to Smalltalk

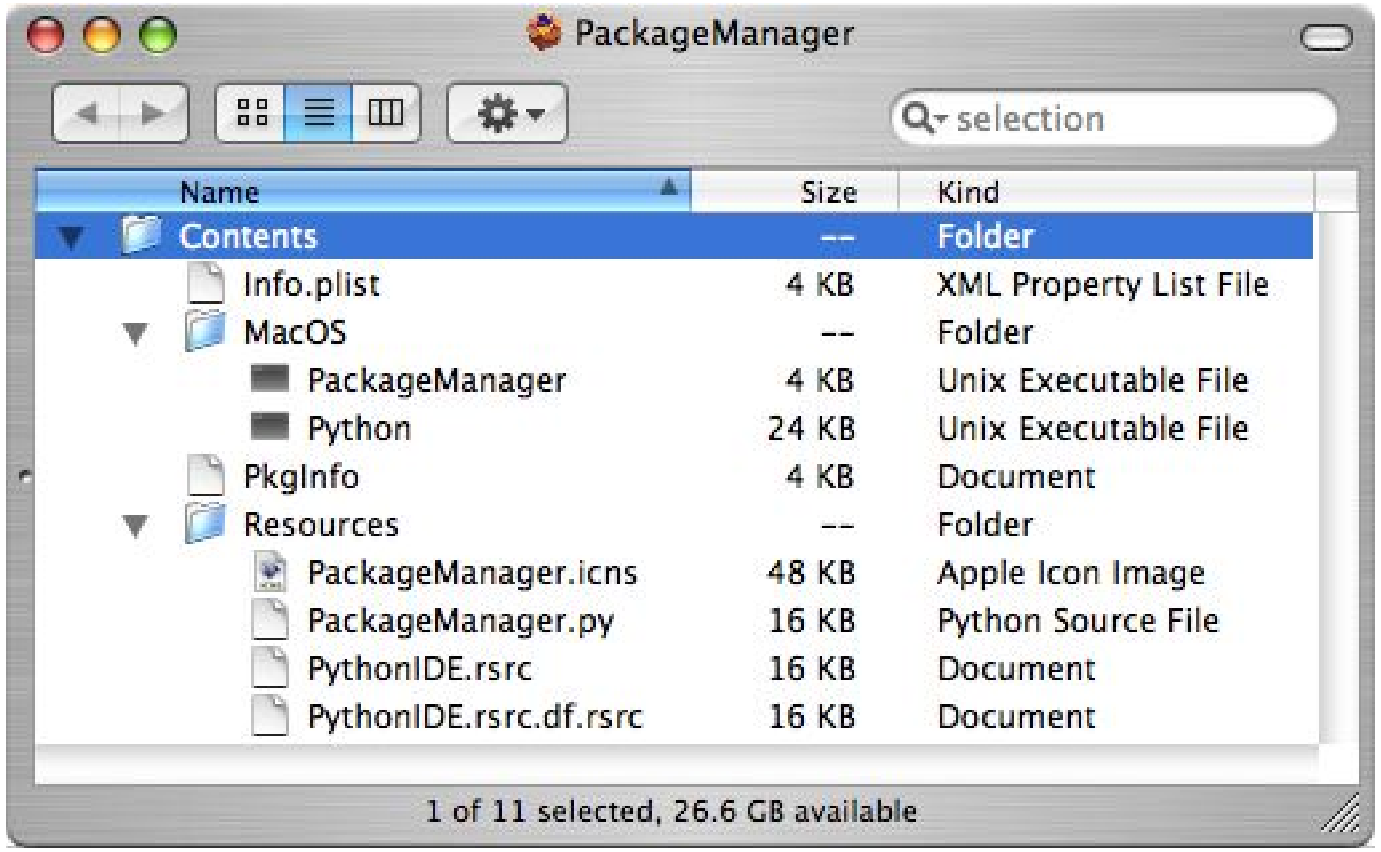
PyObjC

- Plays along VERY nicely with Python
- ... if you have PyObjC, of course

Bundles are like Packages

- They have a fixed directory structure
- General purpose (documents, code, resources)
- Localization features baked in
- Most look like a single file from Finder
- All applications should be bundles
- ... and they need to be to use the GUI

PackageManager.app, dissected


PackageManager

Name	Size	Kind
▼ Contents	--	Folder
Info.plist	4 KB	XML Property List File
▼ MacOS	--	Folder
PackageManager	4 KB	Unix Executable File
Python	24 KB	Unix Executable File
PkgInfo	4 KB	Document
▼ Resources	--	Folder
PackageManager.icns	48 KB	Apple Icon Image
PackageManager.py	16 KB	Python Source File
PythonIDE.rsrc	16 KB	Document
PythonIDE.rsrc.df.rsrc	16 KB	Document

1 of 11 selected, 26.6 GB available

- Bootstrap code in MacOS folder
- Python code and resources in Resources folder
- Info.plist and PkgInfo are required metadata

/usr/bin/pythonw

The Way to run GUI scripts

- GUI-enabled scripts can use the pythonw command
- ... it tricks WindowServer
- ... pretends to be running from the Python application bundle
- Unfortunately, not very useful for Cocoa applications

bundlebuilder

The Way to make Python-based Applications ... for now

- Standard MacPython module
- Similar in purpose to py2exe or freeze
- Similar in API to distutils, but not (yet) integrated
- Documented at the pythonmac.org wiki
- ... but something better is coming soon

Document Bundles

Documents

- Interface Builder nibs
- OmniGraffle
- Keynote
- TextEdit
- ... most other Cocoa applications

Code Bundles

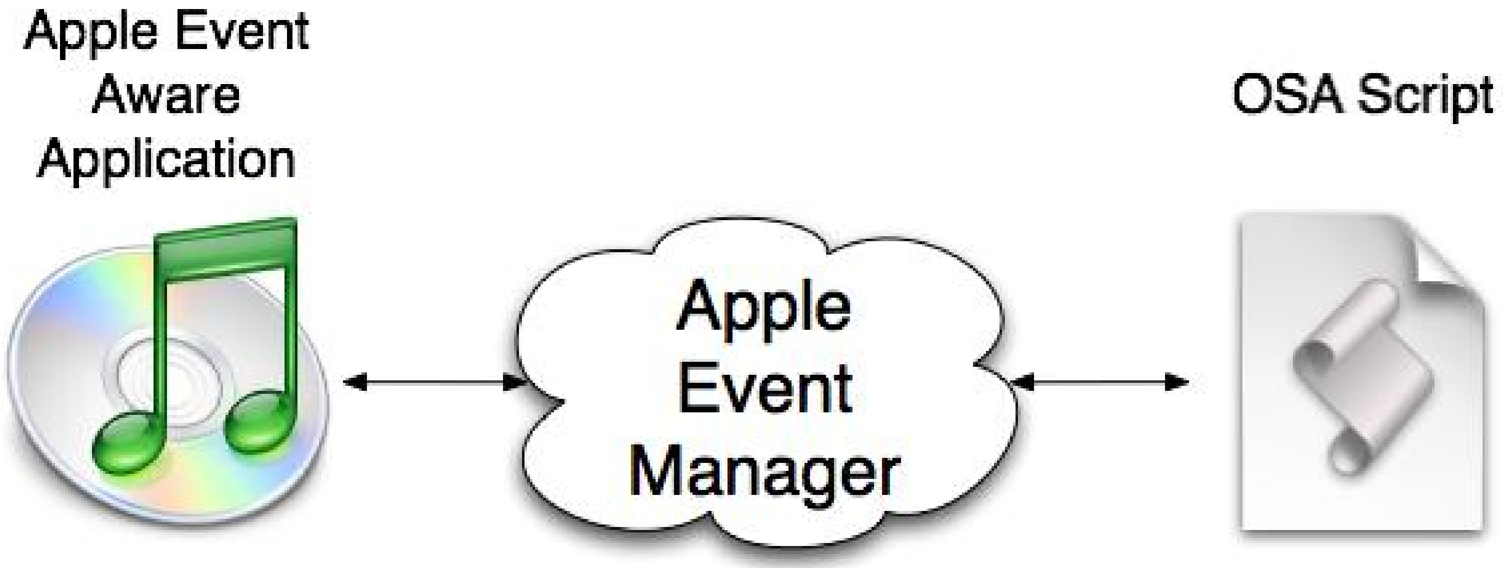
Frameworks

- Versioned
- Shared Libraries
- Headers
- Other data

Other Code

- StartupItems
- Plugins
- Preference Panes

Apple Events

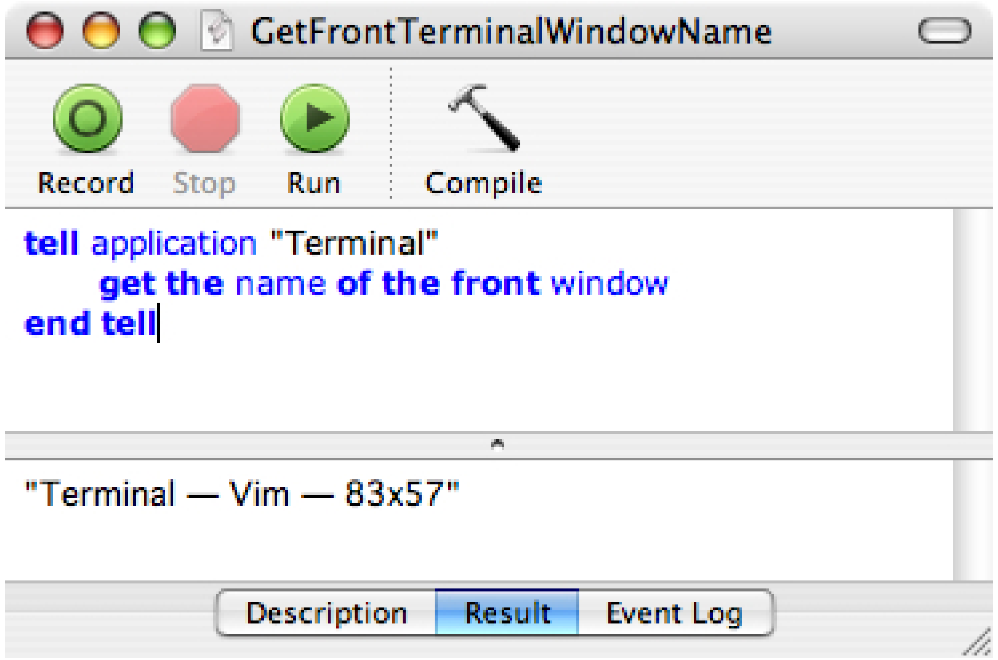


- What are Apple Evevnts?
- Apple Script
- Python and Apple Events

What are Apple Events?

- Interapplication Communication
- Many, but not all, applications support it
- Used primarily for automation tasks
- Very old, low level
- Difficult API
- ... except from Apple Script
- ... or from Cocoa

Apple Script



```
tell application "Terminal"
	get the name of the front window
end tell
```

```
"Terminal — Vim — 83x57"
```

- Looks easy enough, but it's not Python!
- It's crufty, slow, and doesn't scale
- One of the few "read only" languages ;)

Python and Apple Events

Carbon

You may as well write it in assembly!

gensuitemodule

Really old, has limitations, on its way out

aeve

Pythonic bridge, development on hold

appscript

Closer to Apple Script, actively developed

appscript and Finder

The URL of the current user's Desktop:

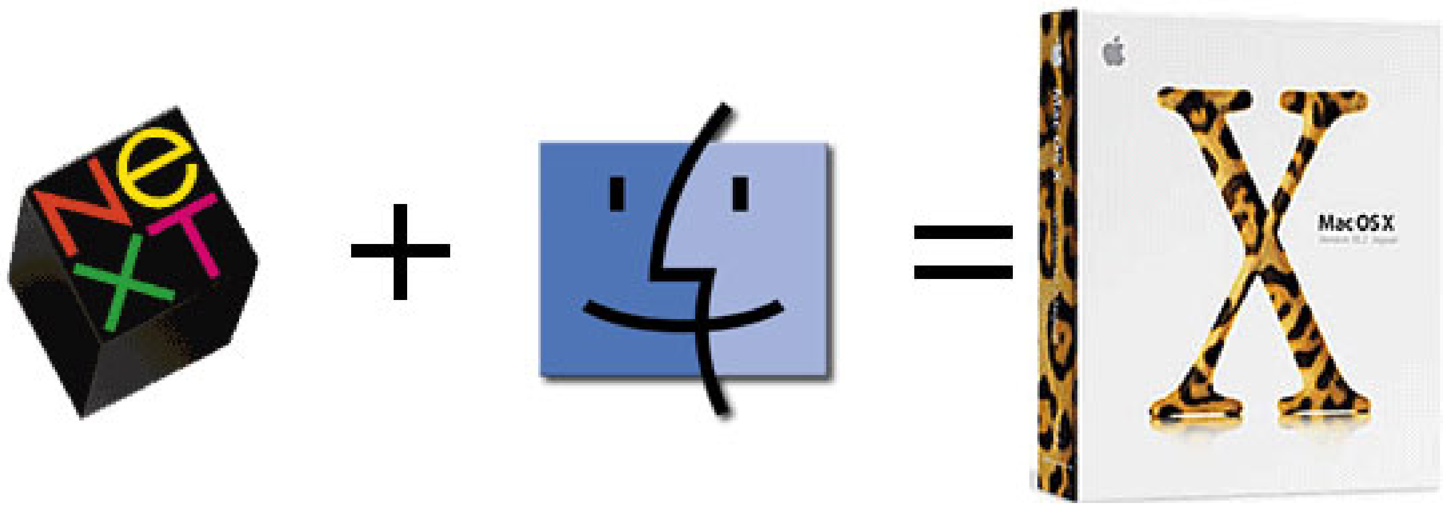
```
>>> from appscript import *
>>> finder = app('Finder.app')
>>> finder.desktop.URL.get()
u'file://localhost/Users/bob/Desktop/'
```

appscript and iTunes

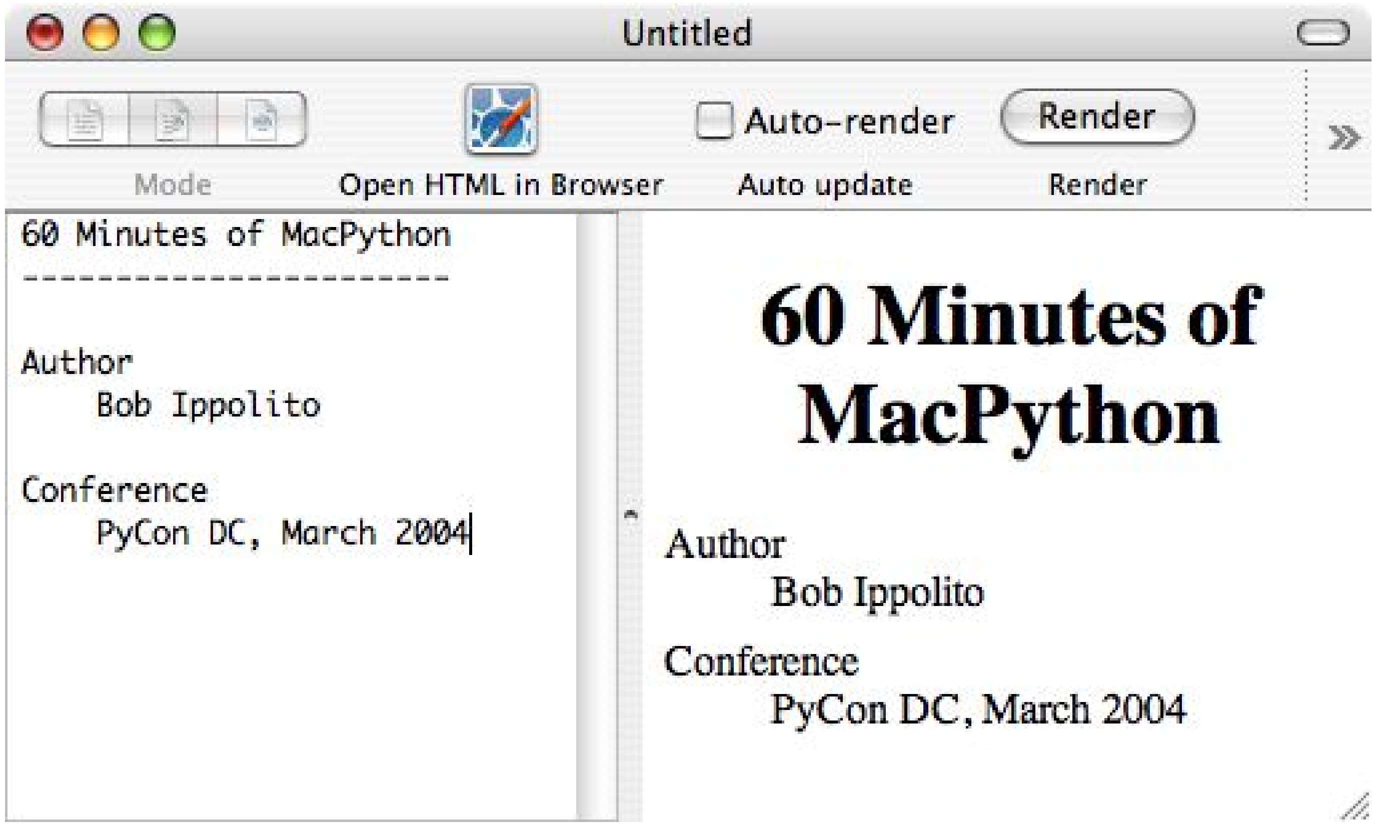
Finding the artist of the song "Python Patrol":

```
>>> from appscript import *
>>> library = app('iTunes.app').sources[1].playlists[1]
>>> mytrack = library.tracks.test(its.name == u'Python Patrol')[1]
>>> mytrack.artist.get()[0]
u'GI Joe Killaz'
```

Cocoa



ReSTedit



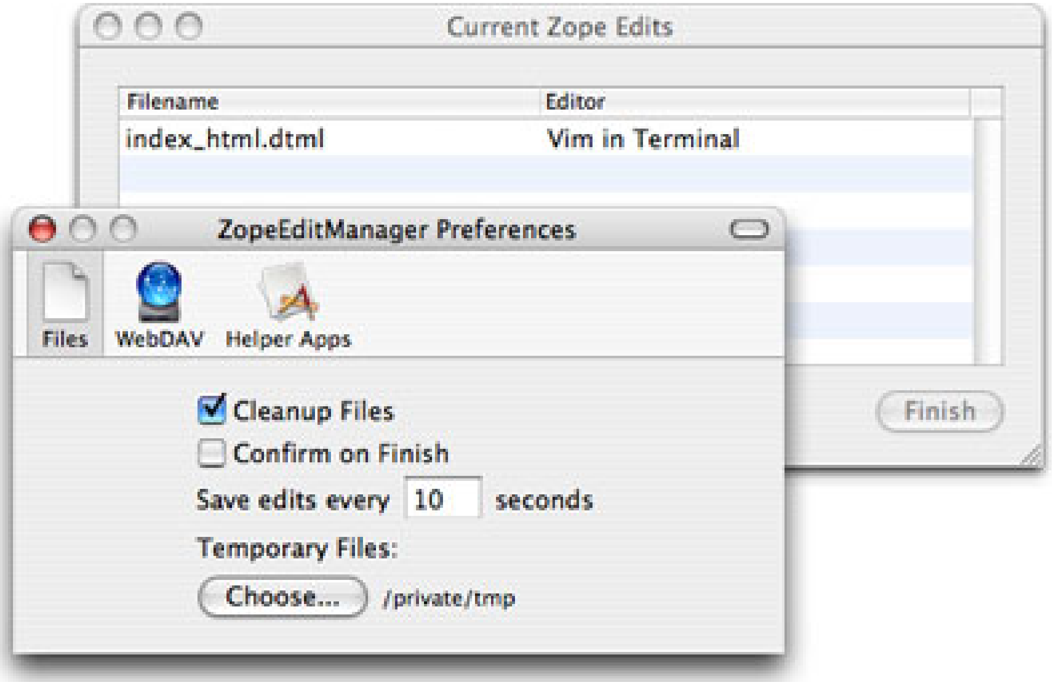
```
60 Minutes of MacPython
-----------------------

Author
    Bob Ippolito

Conference
    PyCon DC, March 2004
```

60 Minutes of MacPython

Author
: Bob Ippolito

Conference
: PyCon DC, March 2004

Zope Edit Manager



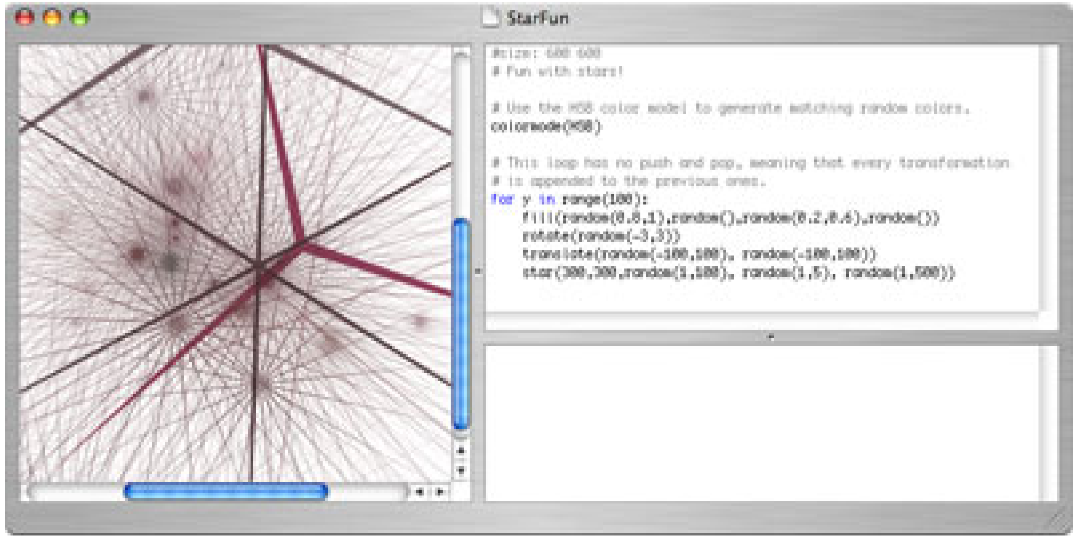
Current Zope Edits

Filename	Editor
index_html.dtml	Vim in Terminal

ZopeEditManager Preferences

[✓] Cleanup Files
[ ] Confirm on Finish
Save edits every 10 seconds
Temporary Files:
/private/tmp

DrawBot



```
#size: 600 600
# Fun with stars!

# Use the HSB color model to generate matching random colors.
colormode(HSB)

# This loop has no push and pop, meaning that every transformation
# is appended to the previous ones.
for y in range(100):
    fill(random(0.8,1),random(),random(0.2,0.6),random())
    rotate(random(-3,3))
    translate(random(-100,100), random(-100,100))
    star(300,300,random(1,100), random(1,5), random(1,500))
```

This Is What Happens

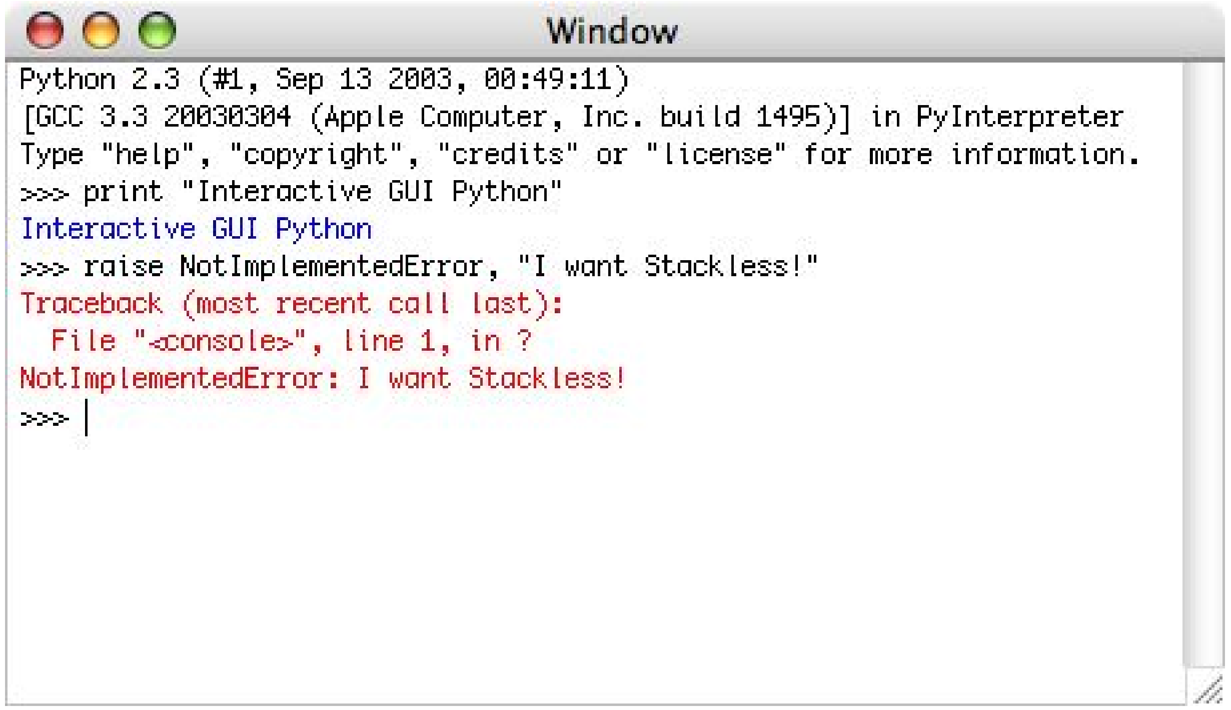
When You Teach Python

To Graphic Designers

In A Simple Way

That They Understand

PyInterpreter



```
Python 2.3 (#1, Sep 13 2003, 00:49:11)
[GCC 3.3 20030304 (Apple Computer, Inc. build 1495)] in PyInterpreter
Type "help", "copyright", "credits" or "license" for more information.
>>> print "Interactive GUI Python"
Interactive GUI Python
>>> raise NotImplementedError, "I want Stackless!"
Traceback (most recent call last):
  File "<console>", line 1, in ?
NotImplementedError: I want Stackless!
>>> 
```

It Really Is Easy

This is where you get to see me use Xcode

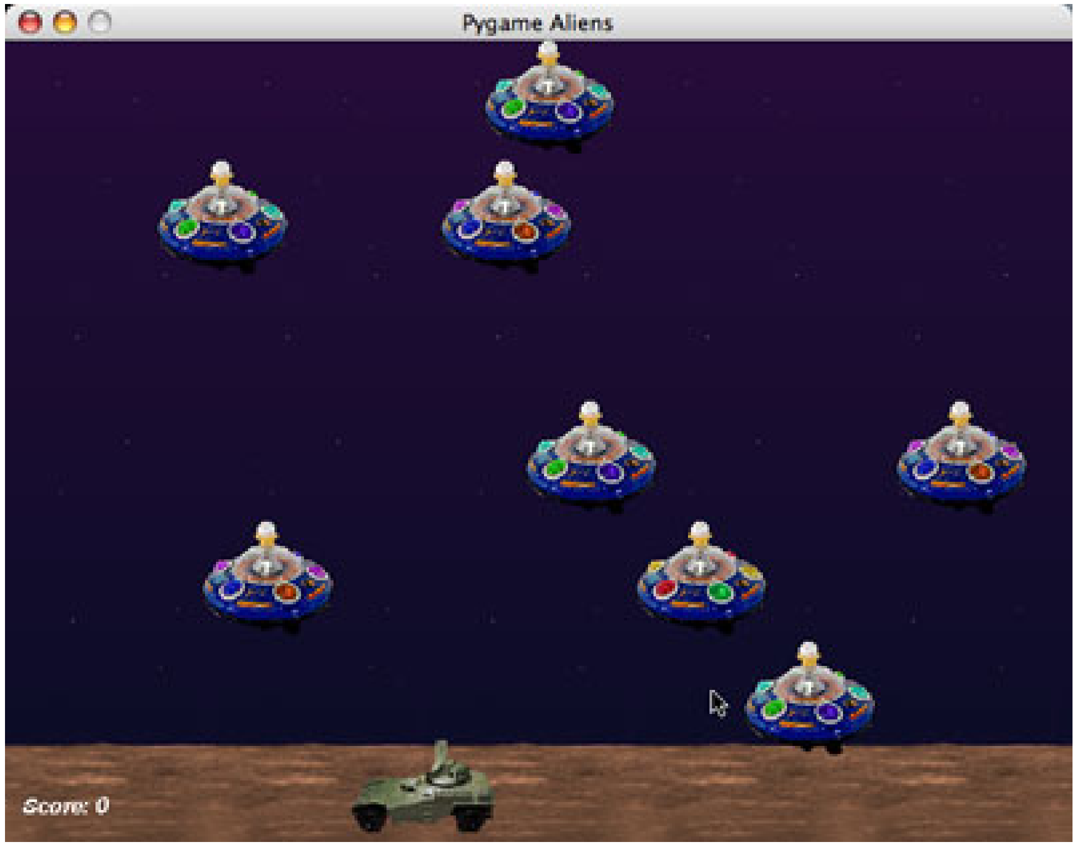
"Cross-Platform" Toolkits

MOST OF THEM ARE NOT READY YET!

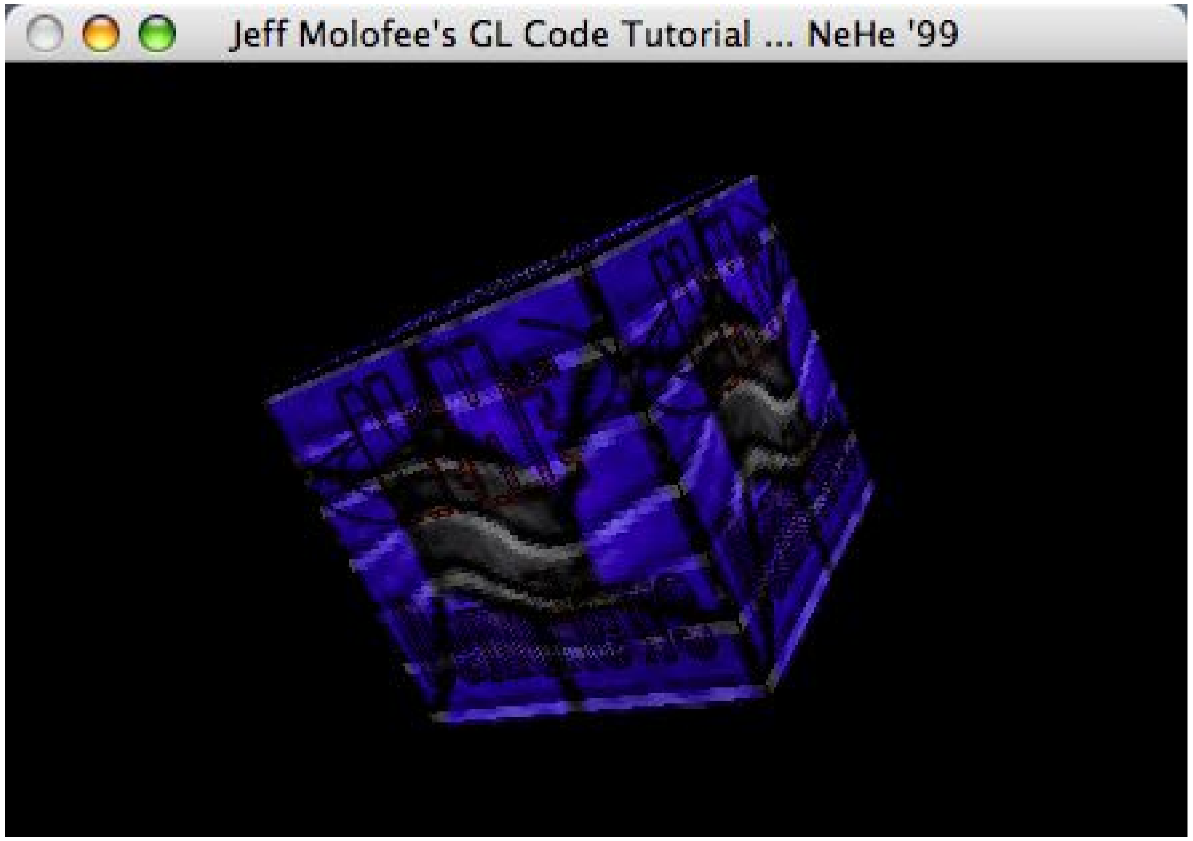
- Tkinter is buggy
- PyQt is buggy
- wxPython is buggy

So why not just use Cocoa, it's easy!

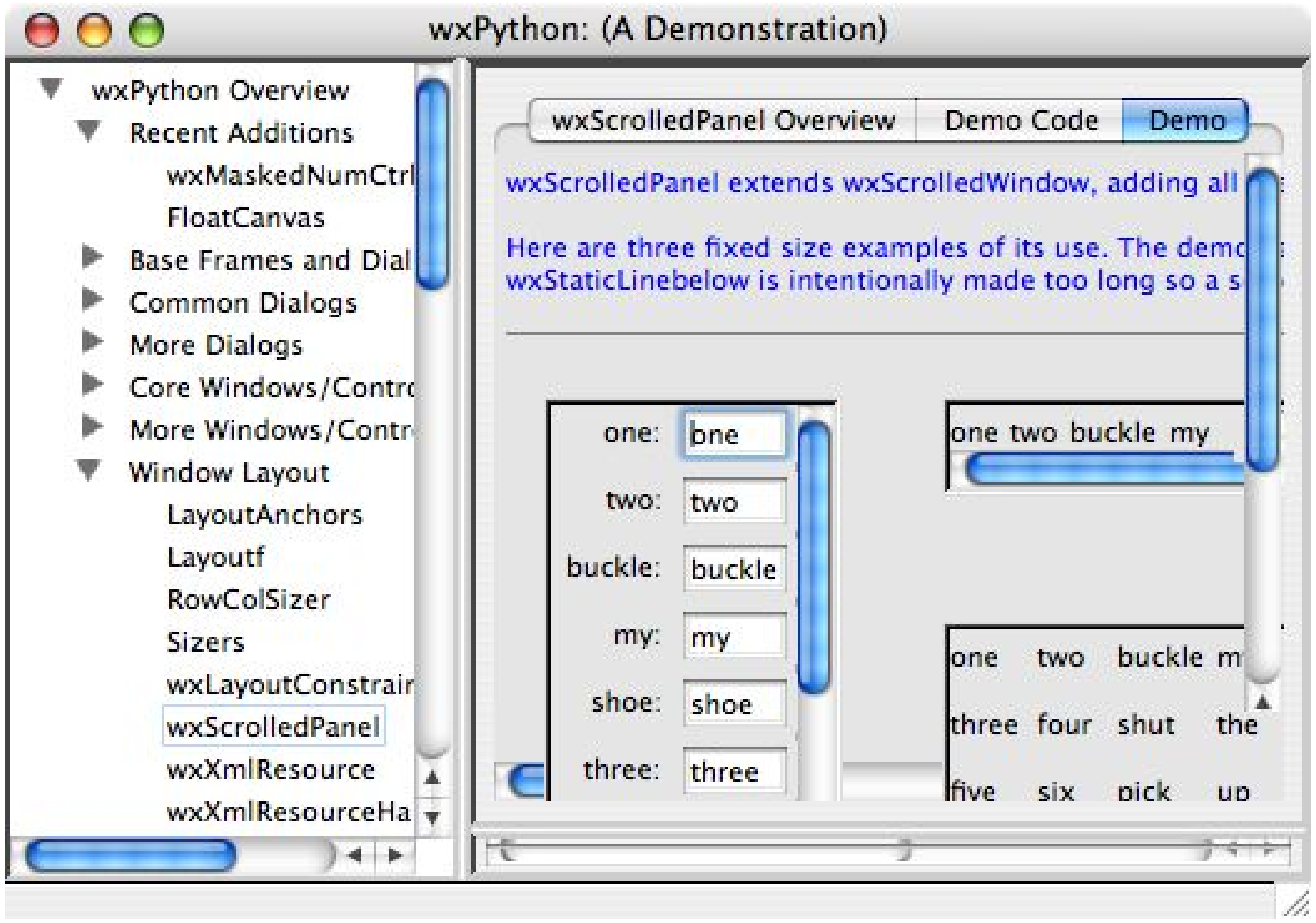
pygame



PyOpenGL



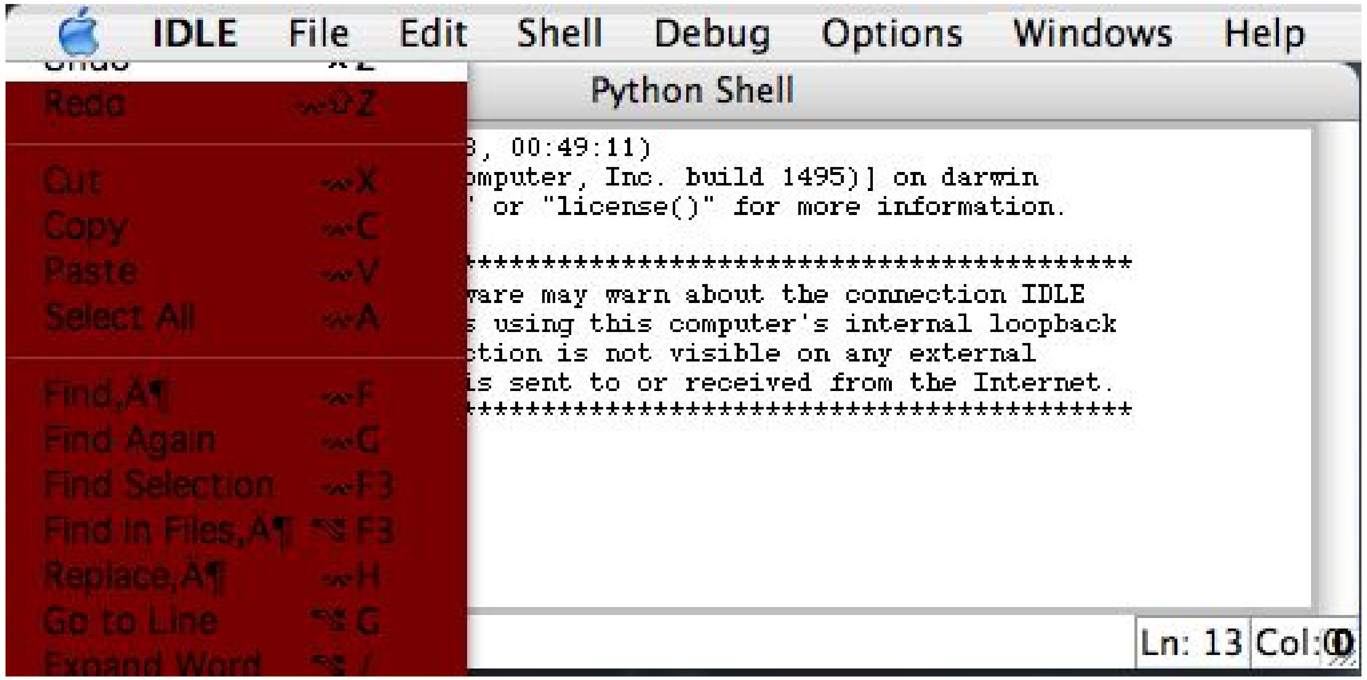
wxPython



wxScrolledPanel extends wxScrolledWindow, adding all

Here are three fixed size examples of its use. The demo
wxStaticLinebelow is intentionally made too long so a s

Tkinter



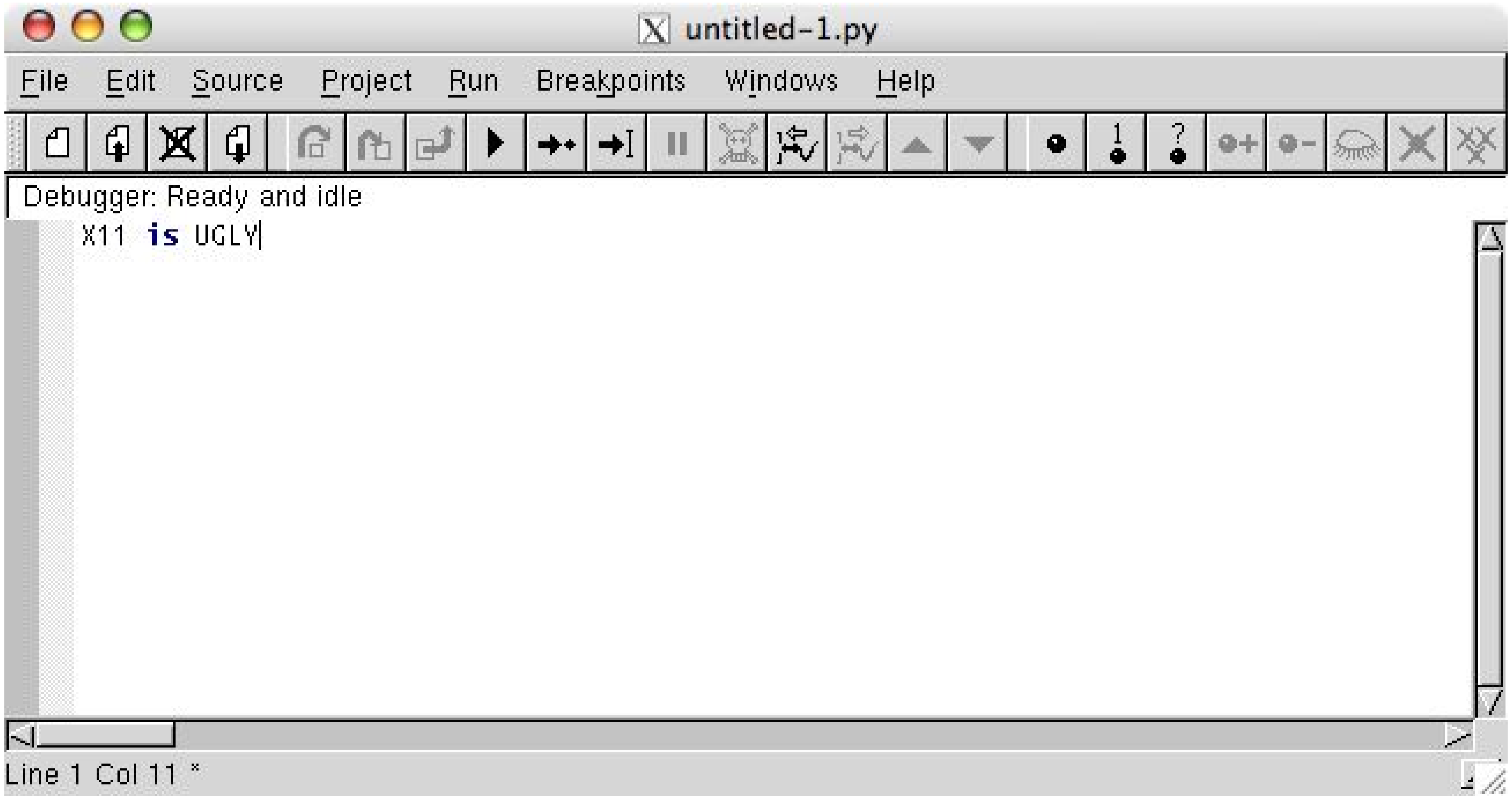
```
3, 00:49:11)
omputer, Inc. build 1495)] on darwin
' or "license()" for more information.

******************************************
ware may warn about the connection IDLE
s using this computer's internal loopback
ction is not visible on any external
is sent to or received from the Internet.
******************************************
```

PyQt

It's probably still building.

... but it does work. somewhat.

And of course, X11 is still ugly



Debugger: Ready and idle

```
X11 is UGLY
```

Line 1 Col 11 *

Community

- pythonmac-sig
- pythonmac.org
- MacPython channel
- Various blogs

The Future

New tools!

Questions?

Go ahead, ask.