



Introduction to PyObjC

Author

Bob Ippolito

Conference

PyCon DC, March 2005



Intended Audience

- Python developers using Mac OS X 10.3 or later
- Spies from the Linux and Win32 camps
- Hopefully a GNUstep porter/maintainer



Topics

- Installing PyObjC
- Why Bother?
- Objective-C Primer
- Crossing the Bridge
- Interface Builder
- Your First Application
- Help!
- Who's Using This Stuff?



Installing PyObjC

Install Xcode:

<http://developer.apple.com/>

Install PyObjC:

<http://pyobjc.sourceforge.net/>



Why Bother?

- You paid for that Mac
- The tools kick ass
- Apple (often) writes good code
- The tools kick ass
- Objective-C and Python are friends



Objective-C

- True superset of C
- Everything is not an object
- Looks kinda like Smalltalk



Classes

- Flat Namespace
- Single Inheritance
- ... with Categories and Protocols
- Classes are objects
- Instance Variables



Objective-C Interface

```
@interface MyClass : NSObject
{
    int myInt;
}
+(id)myClassWithInt:(int)anInt;
-(int)myInt;
@end
```




Objective-C Implementation

```
@implementation MyClass

+(id)myClassWithInt:(int)anInt;
{
    self = [[self alloc] init];
    intInstanceVariable = anInt;
    return self;
}

-(int)myInt
{
    return myInt;
}

@end
```



Objects

- Separate alloc/init
- Everything is an accessor
- ... except when using Key-Value Coding
- Reference counted
- ... but we take care of that
- ... except where Apple doesn't



Messages

- Target
- ... can be nil
- Selector
- Arguments



Exceptions

- Exceptions are exceptional
- Expect bad code to just crash
- ... even from Python



Crossing the Bridge

- unicode, int, long, float work magically
- ... str is not safely bridged!
- None is just like nil
- ... except you can't send messages to it!



Objective-C Messages

Objective-C Message:

```
[NSMutableArray addObject:@"someObject"]
```

Target:

```
NSMutableArray
```

Selector:

```
addObject:
```

Arguments:

```
@"someObject"
```



PyObjC Messages

Python Message:

```
aMutableArray.addObject_(u'someObject')
```

Target:

```
aMutableArray
```

Selector:

```
addObject: (with colons replaced by underscores!)
```

Arguments:

```
u'someObject' (unicode is equivalent to @"string")
```



Key-Value Coding

- Kinda like getattr protocol
- ... but it calls accessors for you (like property)
- ... or it will fetch an ivar and convert to an object
- valueForKey: (like __getattr__)
- valueForKeyPath: looks like a Python expression
- ... except it will also "map" over arrays
- ... and can do cool things like sum



Interface Builder

- Design your interface
- ... using a well designed interface
- Don't write so much code
- Plug objects together
- Manages an *object graph*
- ... think pickle

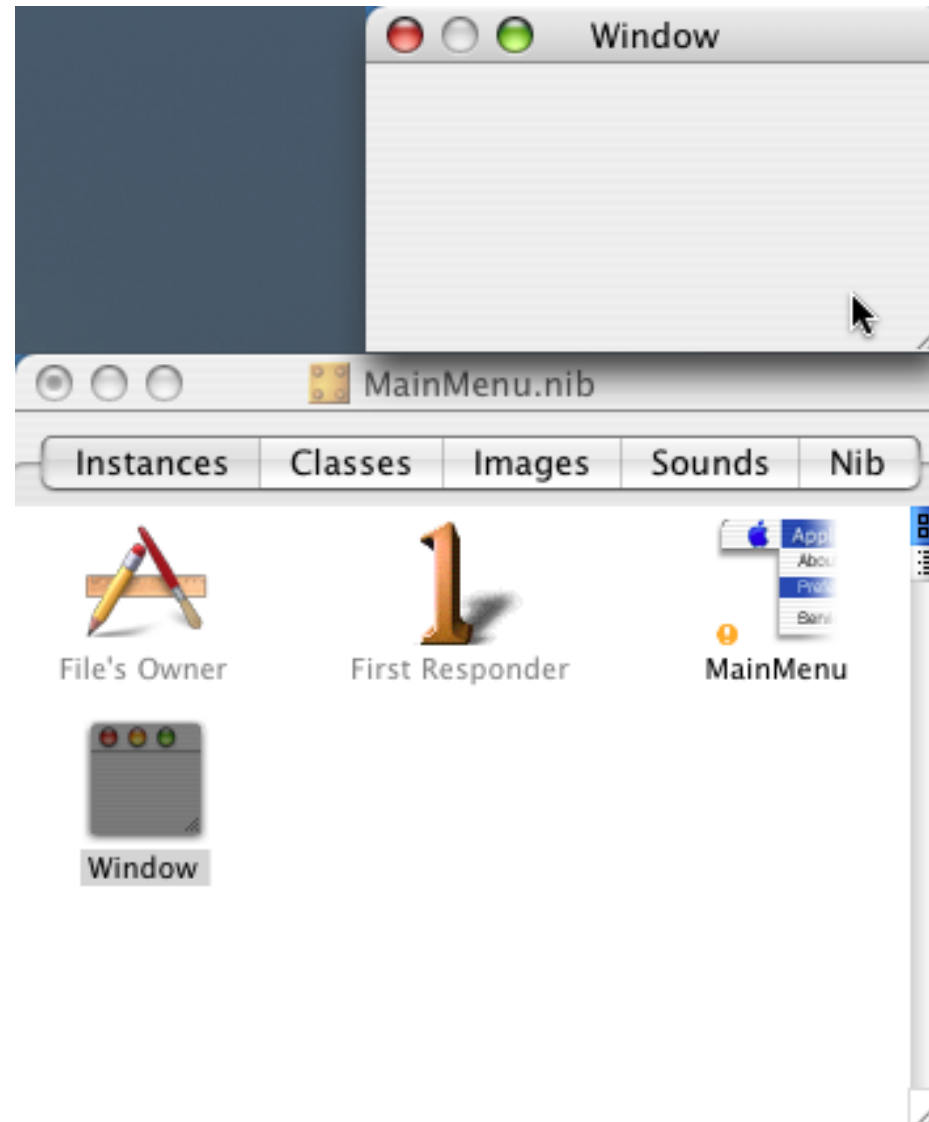


Making Money

- Currency Converter
- Using Cocoa Bindings
- Almost entirely in Interface Builder

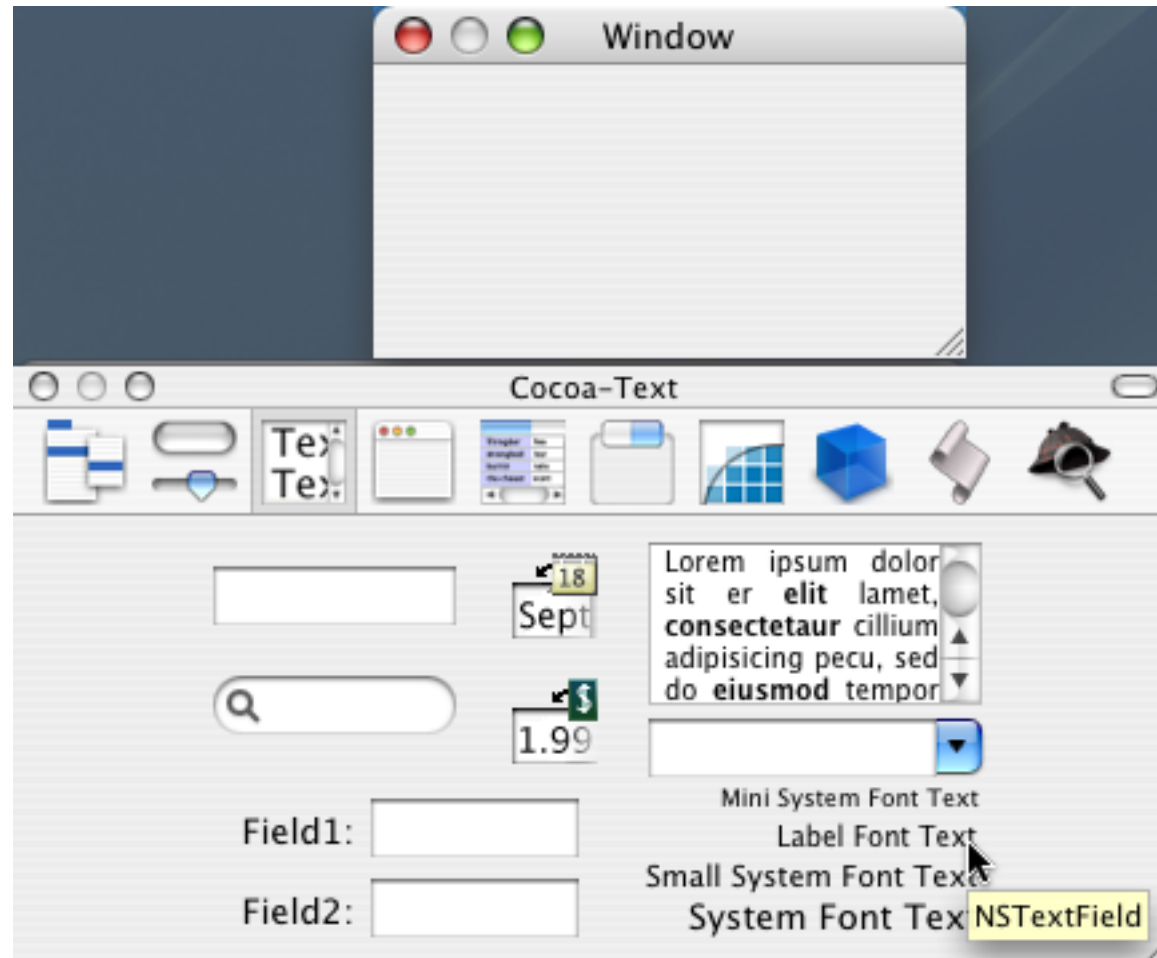


New Application in IB



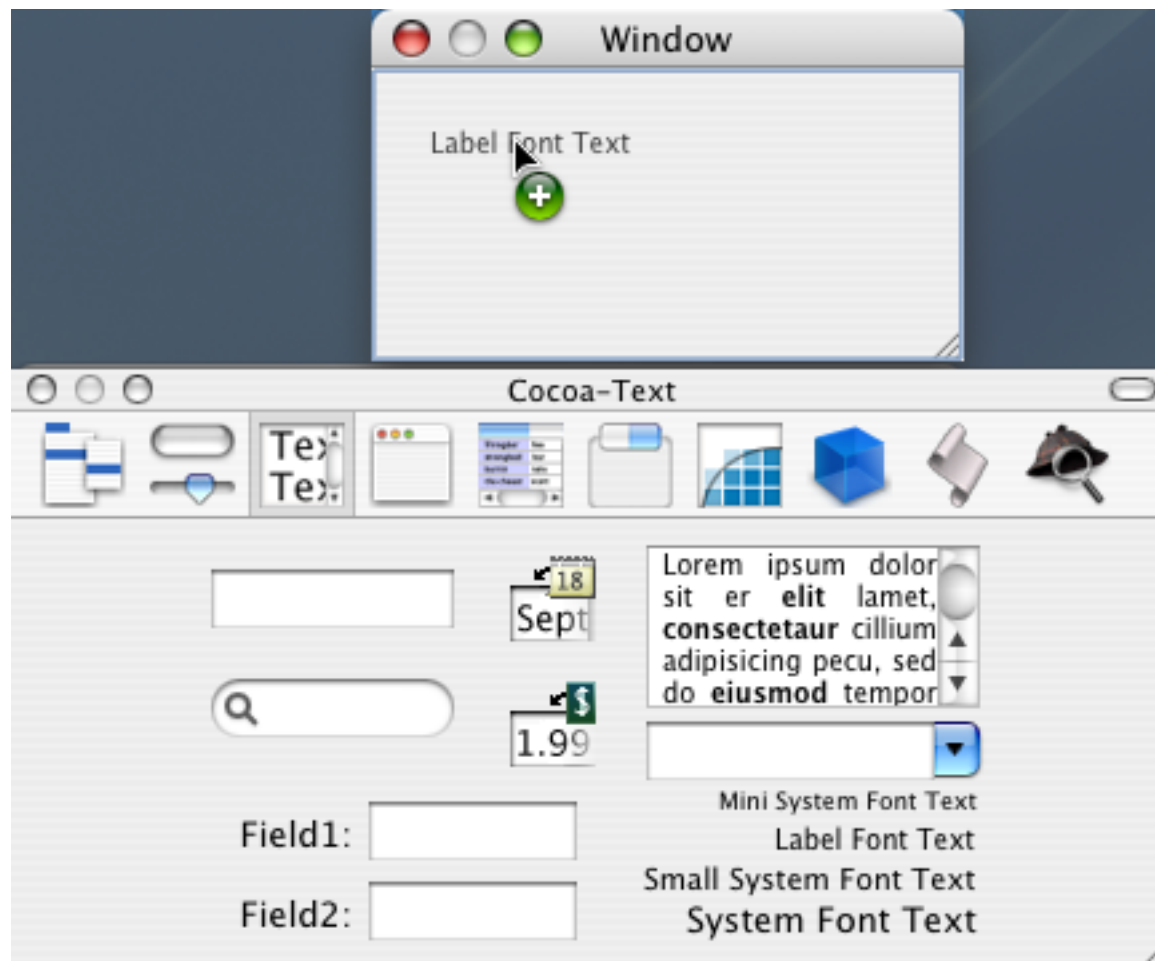


Create an NSTextField



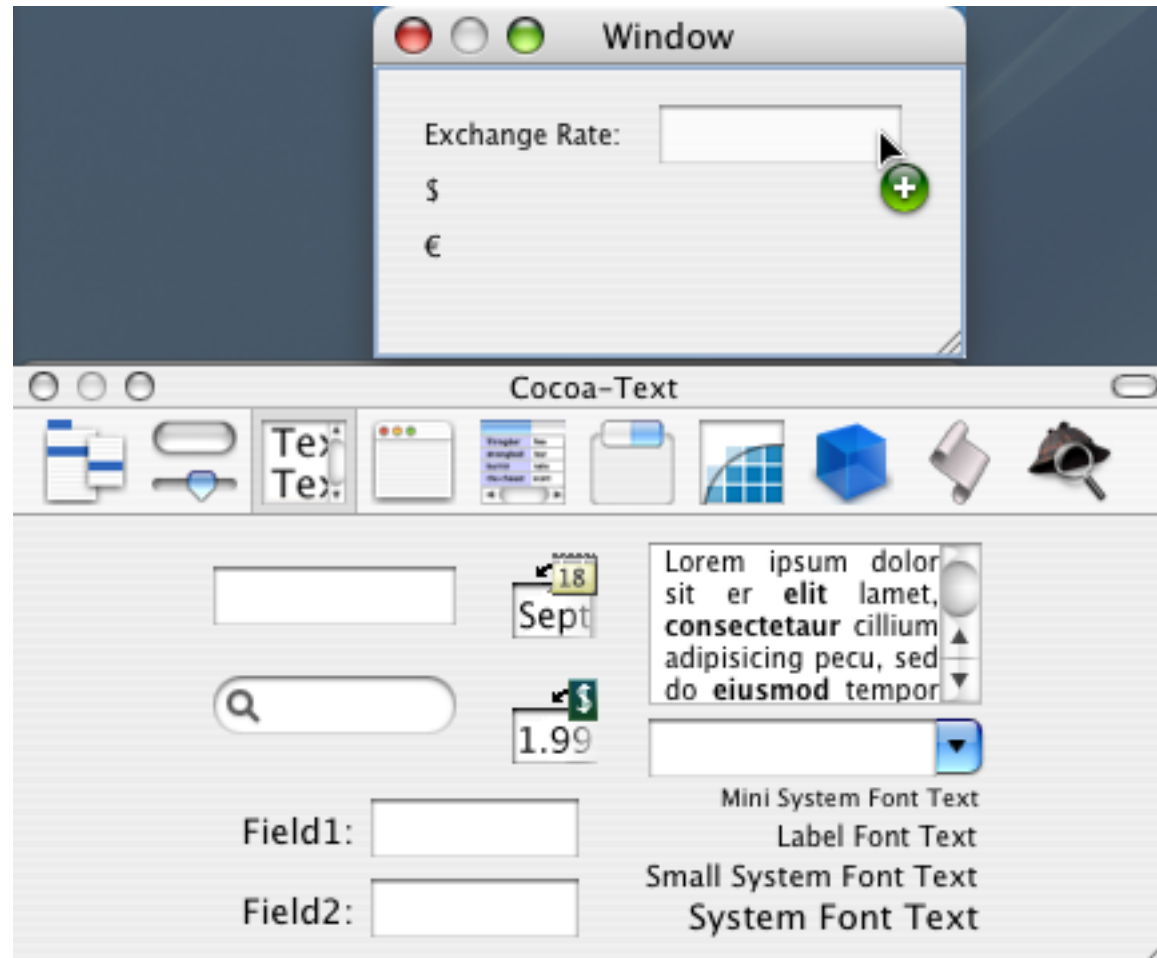


Drag to the NSWindow



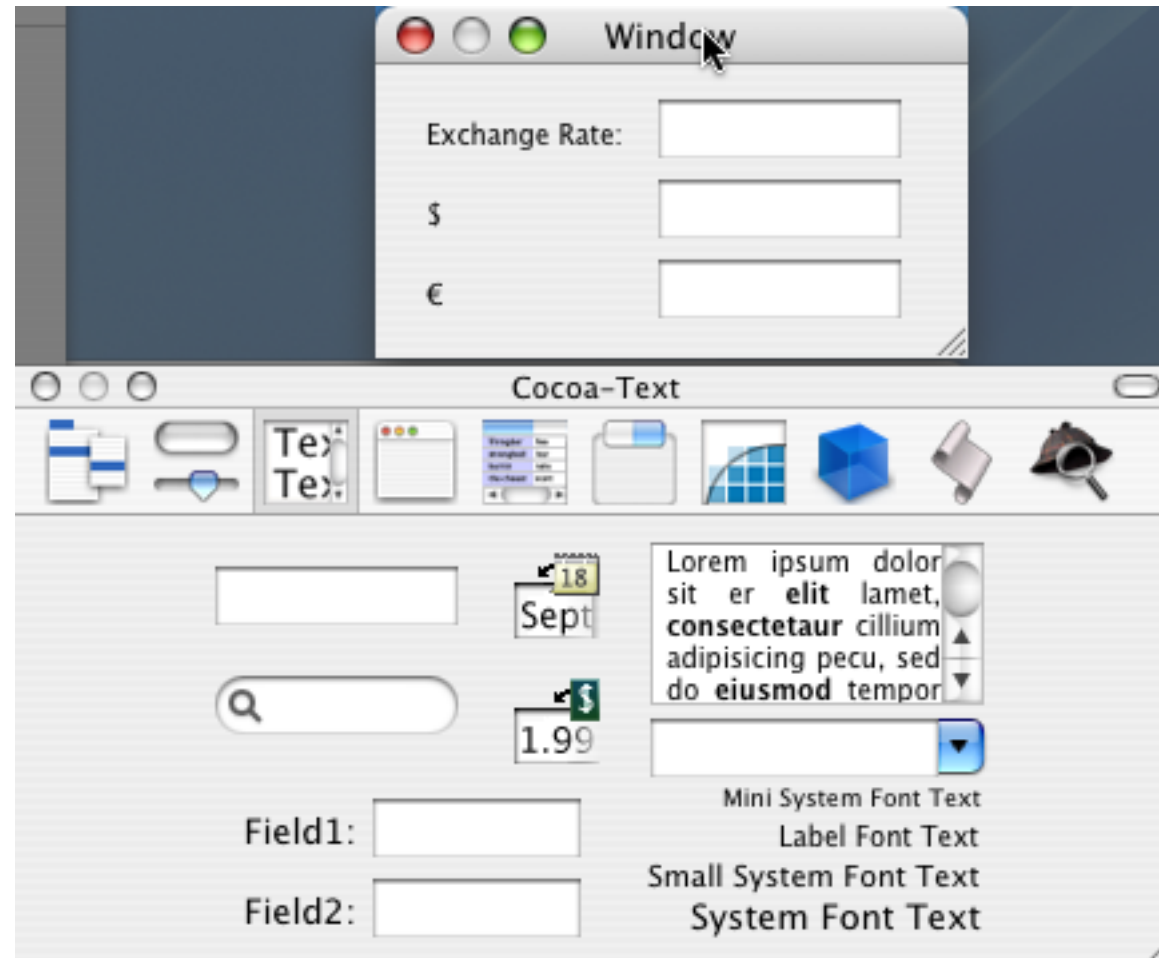


Create the input NSTextFields





Almost finished UI Layout





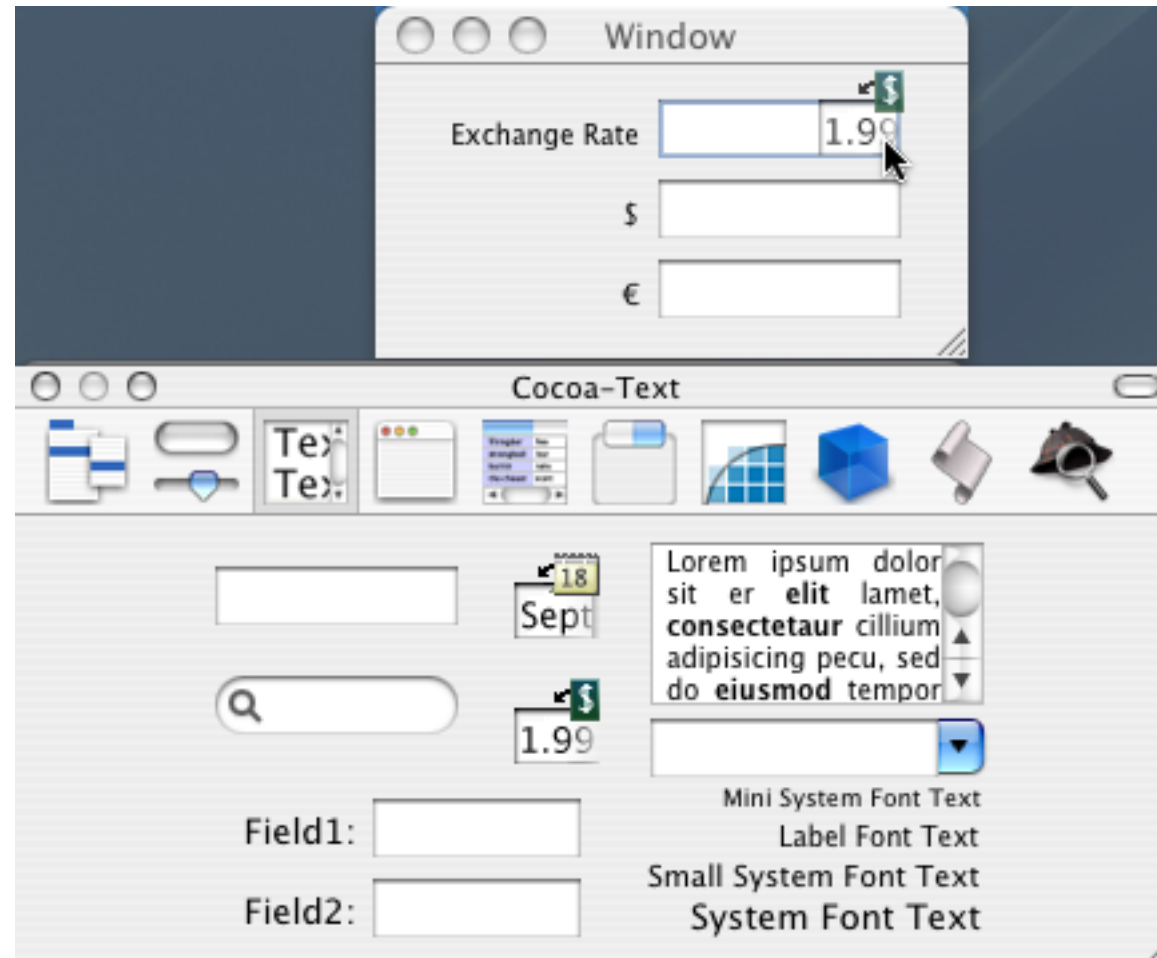
Align the labels

The image shows a screenshot of Xcode with two windows. The top window, titled "Window", contains three text fields. The first field has the text "Exchange Rate:" followed by a small blue selection handle. Below it are two empty text fields, one with a "\$" symbol and one with a "€" symbol. The bottom window is the "NSTextField Info" inspector, which is currently displaying the attributes for the selected "Exchange Rate:" text field. The inspector shows the following settings:

- Attributes: [Dropdown menu]
- Title: Exchange Rate:
- Placeholder: [Empty text field]
- Text Color: [Black color swatch]
- Backgrnd Color: [White color swatch] Draw
- Tag: 0
- Alignment: [Left] [Center] [Right] [Justified] [Left-Right] [Right-Left] (The Right alignment button is selected)
- Text Border: [None] [Solid] [Inset] (The None button is selected)
- Size: Regular
- Layout: Scrollable Wraps
- Send Action On: Enter only End editing
- Options: Editable Enabled Selectable Rounded Hidden



Use currency NSNumberFormatters





Set up the Bindings

The screenshot displays the Xcode interface. On the left, a window titled "Window" contains a slider for "Exchange Rate" and two text input fields labeled with "\$" and "€". Below this, the "MainMenu.nib" window is open, showing a palette with icons for "File's Owner", "First Responder", "MainMenu", and "Window". On the right, the "NSTextField Info" panel is visible, with the "Bindings" section selected. A tooltip points to the "value" property, stating: "Specifies the value of the bound object." The "Value" section also includes "Value With Pattern" (with a sub-property "displayPatternvalue1") and "Availability" (with sub-properties "editable", "enabled", and "hidden"). The "Font" section lists properties like "font", "fontBold", "fontFamilyName", "fontItalic", "fontName", and "fontSize". The "Text Color" section lists "textColor".



To point to your delegate

The image shows a screenshot of Xcode's interface. On the left, a window titled "Window" displays a UI element with an "Exchange Rate" slider and two text input fields labeled with "\$" and "€". Below this, the "MainMenu.nib" window is open, showing a palette with icons for "File's Owner", "First Responder", "MainMenu", and "Window". On the right, the "NSTextField Info" inspector is open, showing the "Bindings" section. The "value" property is set to "File's Owner (NSApplication)". The "Model Key Path" is set to "delegate.exchangeRate", which is highlighted by a mouse cursor. The "Controller Key" is empty, and the "Value Transformer" is also empty. Below these fields, there are several checked and unchecked options: "Allows Editing Multiple Values Selection" (checked), "Conditionally Sets Editable" (checked), "Conditionally Sets Enabled" (unchecked), "Conditionally Sets Hidden" (unchecked), "Continuously Updates Value" (unchecked), "Raises For Not Applicable Keys" (checked), and "Validates Immediately" (unchecked). At the bottom, there are fields for "Multiple Values Placeholder:" and "No Selection Placeholder:".



Dollars binding...

The image shows a screenshot of Xcode with two windows. The left window, titled "Window", displays a UI element with an "Exchange Rate" label, a text field, a dollar sign (\$) with a slider, and a euro sign (€) with a text field. The right window, titled "NSTextField Info", shows the "Bindings" panel. The "Value" section is expanded, showing the following configuration:

- value** (checked): Bind
- Bind to:** File's Owner (NSApplication)
- Controller Key:** (empty)
- Model Key Path:** delegate.dollarsToConvert
- Value Transformer:** (empty)
- Allows Editing Multiple Values Selection
- Conditionally Sets Editable
- Conditionally Sets Enabled
- Conditionally Sets Hidden
- Continuously Updates Value
- Raises For Not Applicable Keys
- Validates Immediately
- Multiple Values Placeholder:** (empty)
- No Selection Placeholder:** (empty)

The bottom of the screenshot shows the "MainMenu.nib" window with a palette containing icons for "File's Owner", "First Responder", "MainMenu", and "Window".



Other Currency Binding...

The image shows a screenshot of Xcode's interface. On the left, a window titled "Window" contains three text fields: "Exchange Rate", "\$", and "€". Below this is a "MainMenu.nib" window with tabs for "Instances", "Classes", "Images", "Sounds", and "Nib". The "Instances" tab is active, showing icons for "File's Owner", "First Responder", "MainMenu", and "Window". On the right, the "NSTextField Info" inspector is open, showing the "Bindings" section. The "value" property is bound to "File's Owner (NSApplication)" with the "Model Key Path" set to "delegate.amountInOtherCurrency". The "Value Transformer" is empty. Several checkboxes are checked, including "Allows Editing Multiple Values Selection", "Conditionally Sets Editable", "Raises For Not Applicable Keys", and "Validates Immediately".

Window

Exchange Rate

\$

€

MainMenu.nib

Instances Classes Images Sounds Nib

File's Owner First Responder MainMenu

Window

NSTextField Info

Bindings

value Bind

Bind to: File's Owner (NSApplication)

Controller Key:

Model Key Path: delegate.amountInOtherCurrency

Value Transformer:

- Allows Editing Multiple Values Selection
- Conditionally Sets Editable
- Conditionally Sets Enabled
- Conditionally Sets Hidden
- Continuously Updates Value
- Raises For Not Applicable Keys
- Validates Immediately

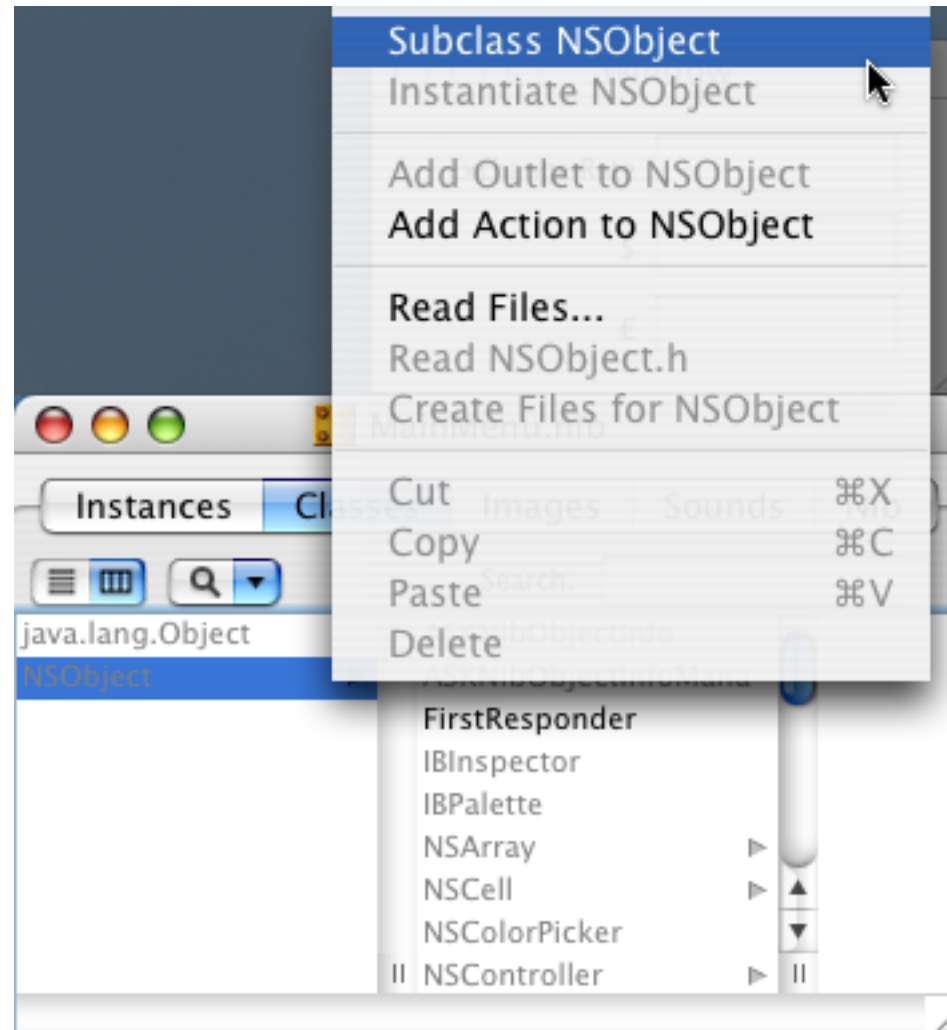
Multiple Values Placeholder:

No Selection Placeholder:

Not Applicable Placeholder:

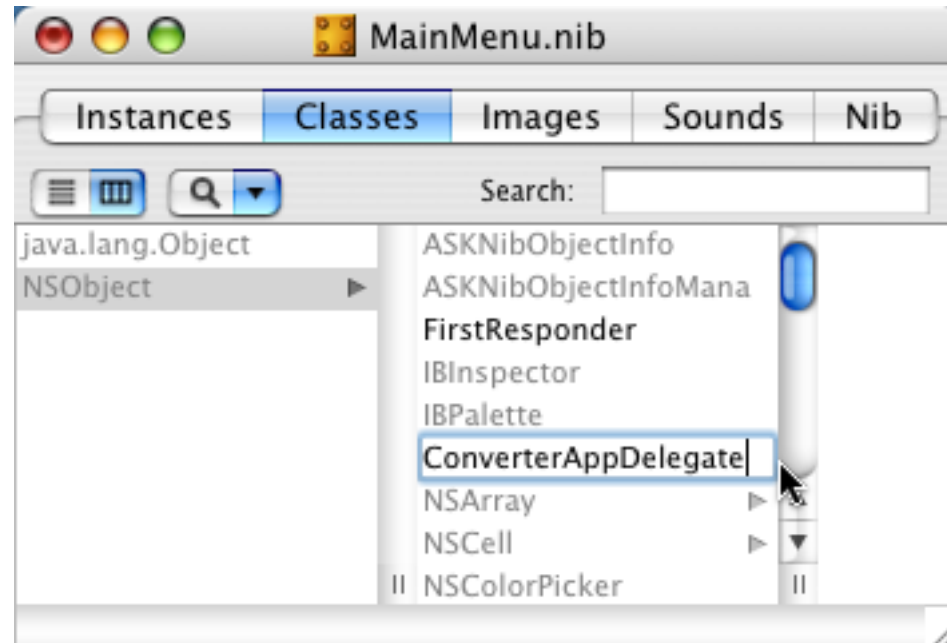


Subclass NSObject



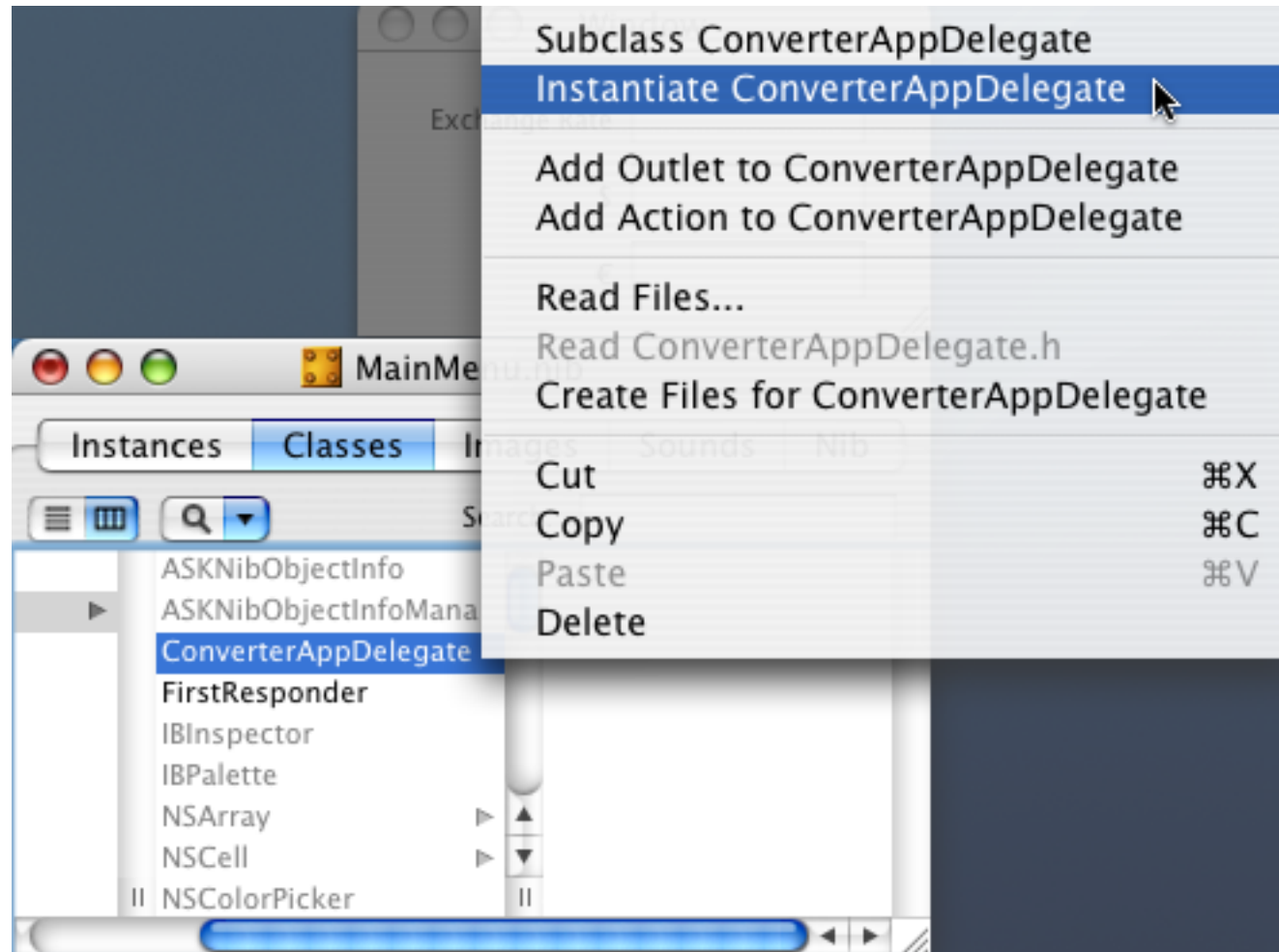


To create your delegate class





Instantiate it in your nib



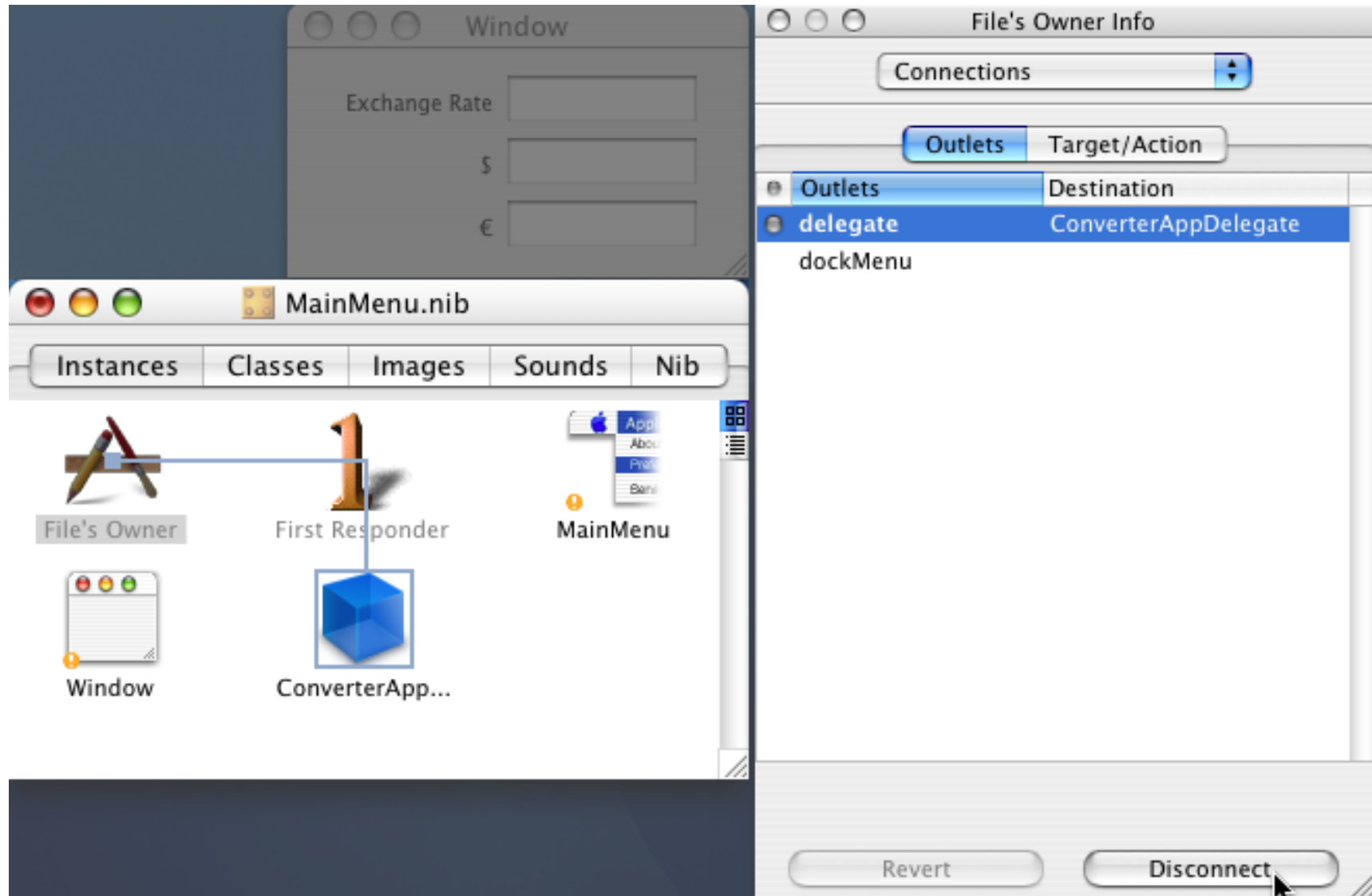


Create a connection

The screenshot displays the Xcode interface during the process of creating a connection in a nib file. On the left, the 'MainMenu.nib' window is open, showing a palette of objects. A blue line connects the 'File's Owner' object to the 'ConverterApp...' object. The 'Connections Inspector' on the right is open, showing the 'Outlets' tab. The 'Connections' dropdown is set to 'Outlets', and the 'Destination' is set to 'delegate'. The list of outlets includes 'delegate' and 'dockMenu'. At the bottom of the inspector, there are 'Revert' and 'Connect' buttons.



To the NSApplication





ConverterAppDelegate.py Class

```
from Foundation import *
from AppKit import *
import objc

class ConverterAppDelegate(NSObject):
    def init(self):
        self = super(ConverterAppDelegate, self).init()
        self.exchangeRate = 3
        self.dollarsToConvert = 4
        return self

    def amountInOtherCurrency(self):
        return self.dollarsToConvert * self.exchangeRate

    def setAmountInOtherCurrency_(self, amt):
        self.dollarsToConvert = amt / self.exchangeRate

# shamelessly preventing line wrapping
cls = ConverterAppDelegate
cls.setKeys_triggerChangeNotificationsForDependentKey_(
    [u'dollarsToConvert', u'exchangeRate'],
    u'amountInOtherCurrency',
)
```



Converter.py script

```
from PyObjCTools import AppHelper
import ConverterAppDelegate
if __name__ == '__main__':
    AppHelper.runEventLoop()
```



Converter setup.py script

```
from distutils.core import setup
import py2app
setup(
    app = ['Converter.py'],
    data_files = ['MainMenu.nib'],
)
```



Build and Run

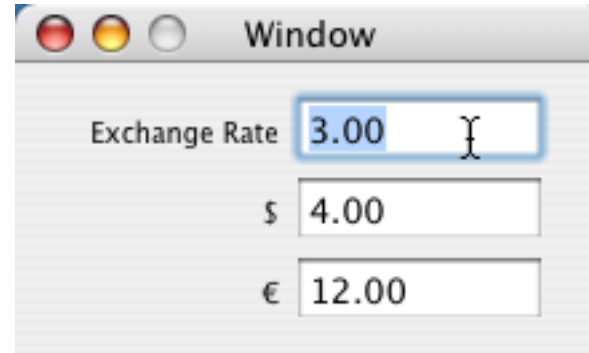
Build:

```
% python setup.py py2app --alias
```

Run:

```
% open dist/Converter.app
```

Done:



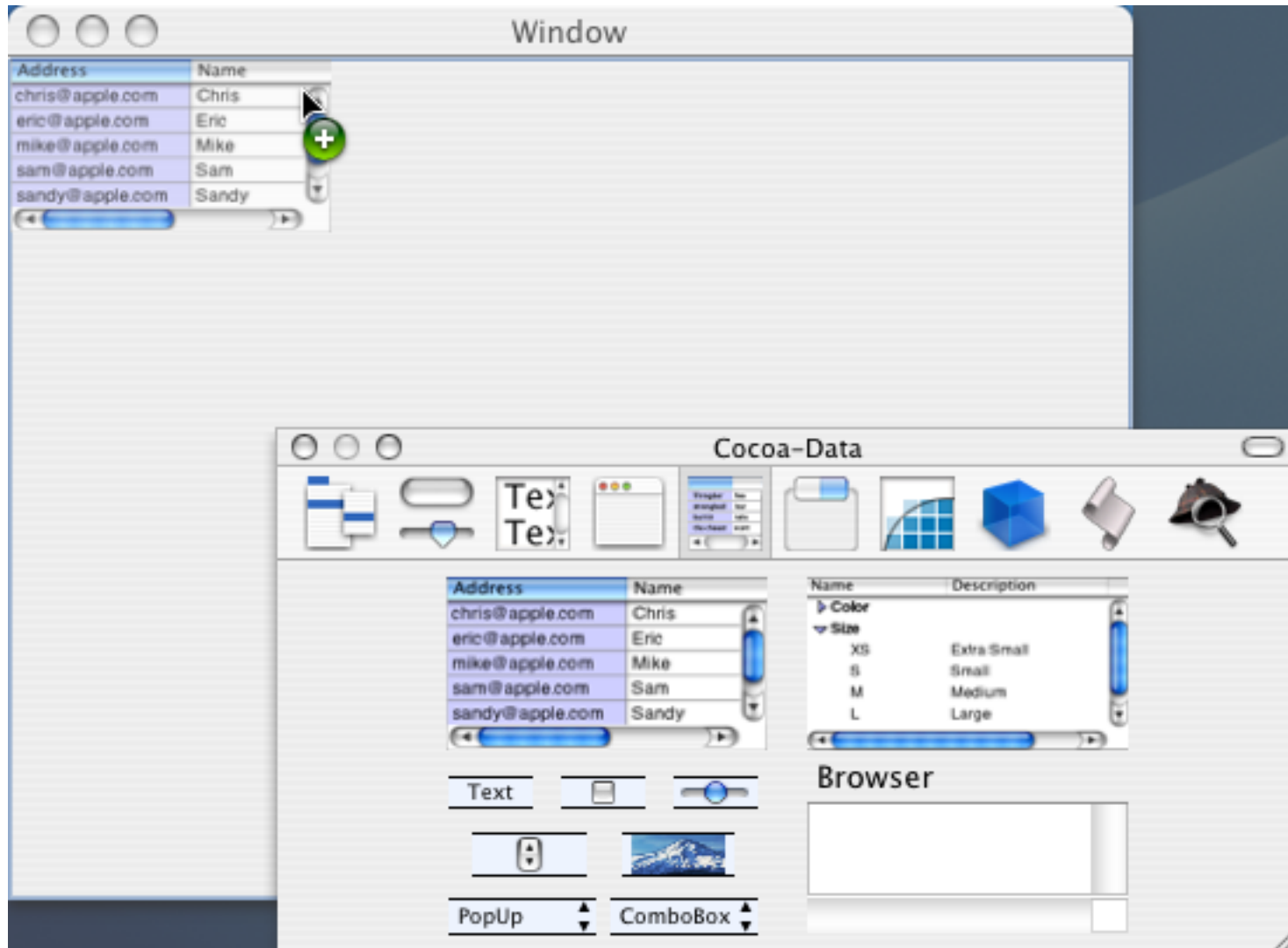


Hack the Gibson

- Views password file
- ... using nidump utility
- In a table view



New NSTableView



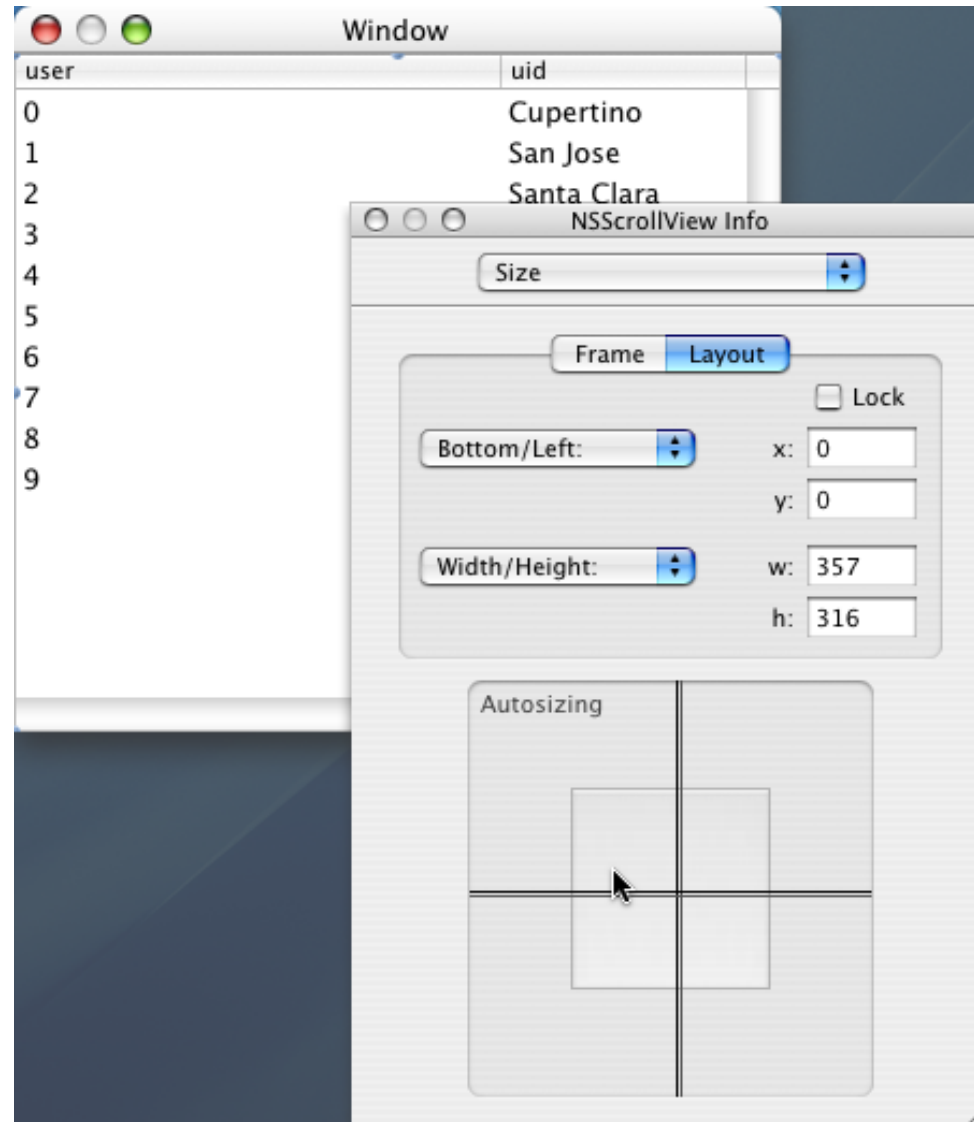


Name the columns

Window	
user	
0	Cupertino
1	San Jose
2	Santa Clara
3	San Francisco
4	Palo Alto
5	San Carlos
6	Los Gatos
7	Sunnyvale
8	Mountain View
9	Redwood City

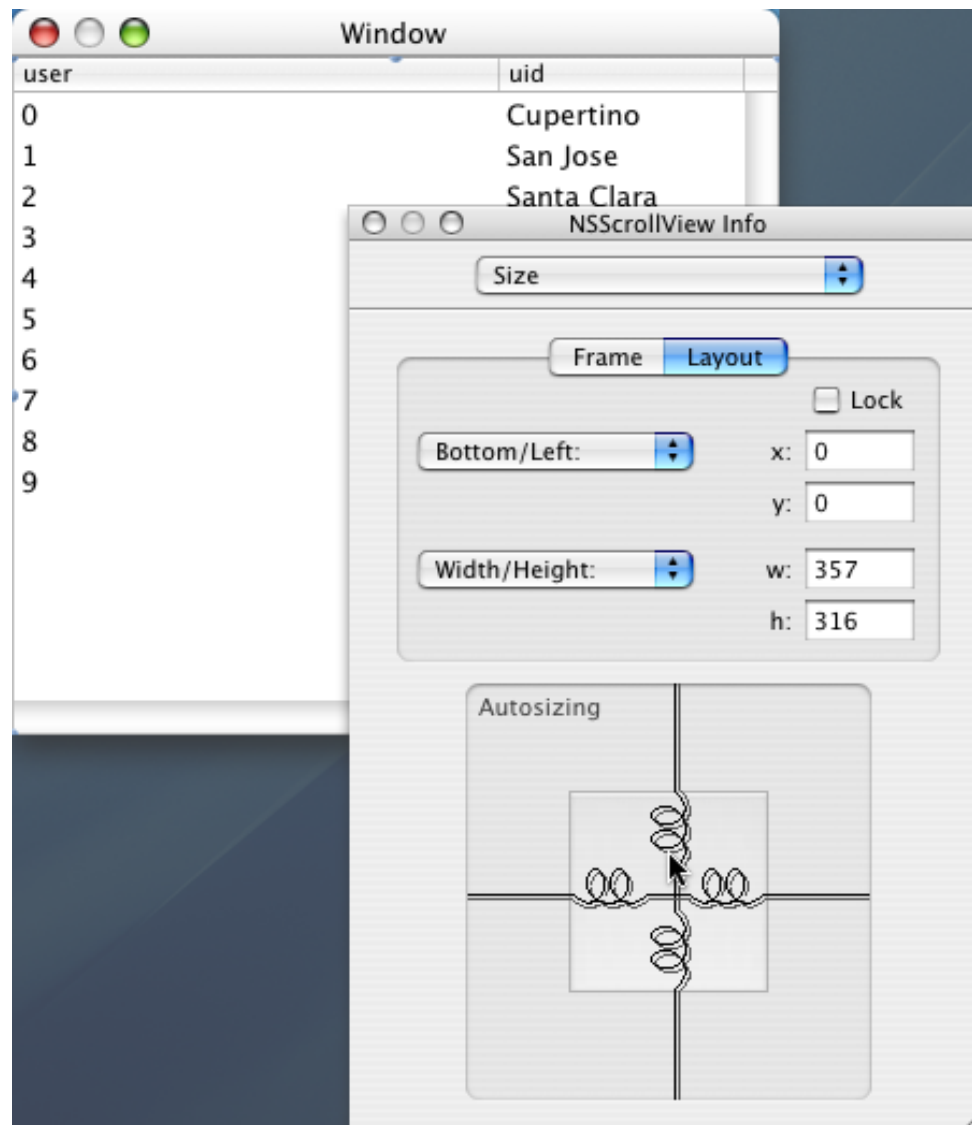


Change the resize behavior



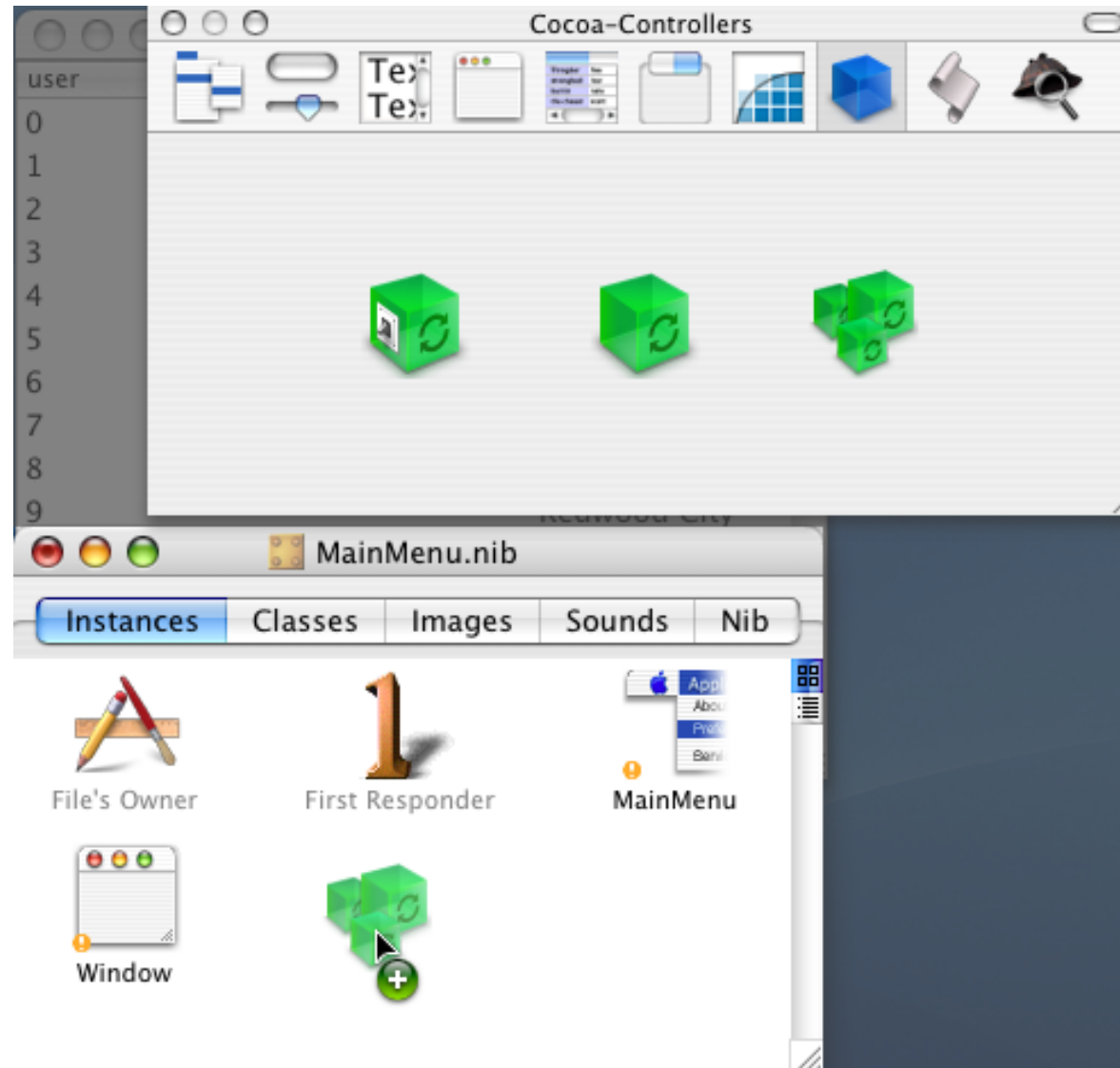


To expand with the NSWindow



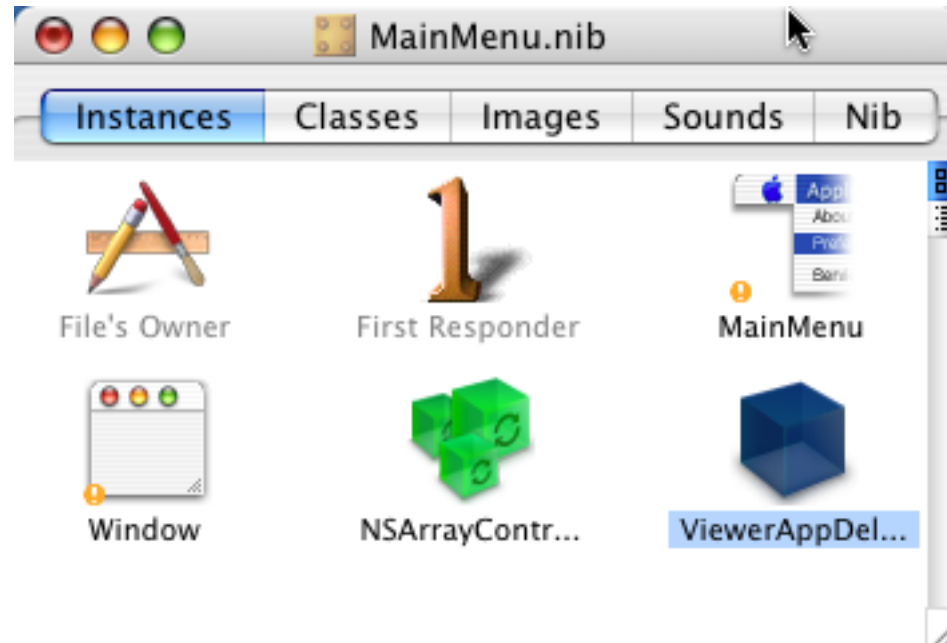


Create an NSArrayController





Create the `ViewerAppDelegate`





Bind the NSArrayController

The screenshot shows the Xcode interface with the 'NSArrayController Info' panel open. The 'Bindings' section is selected, and the 'contentArray' property is being configured. The 'Bind to' field is set to 'File's Owner (NSApplication)', the 'Controller Key' is empty, and the 'Model Key Path' is 'delegate.passwords'. The 'Value Transformer' field is empty. The 'Conditionally Sets Editable' checkbox is checked, while 'Handles Content As Compound Value', 'Raises For Not Applicable Keys', 'Selects All When Setting Content', and 'Validates Immediately' are unchecked. The 'Instances' tab is active, showing 'File's Owner', 'First Responder', 'Window', and 'NSArrayContr...'. The 'MainMenu.nib' window is also visible in the background.

Window

user

0

1

2

3

4

5

6

7

MainMenu.nib

Instances Classes Images

File's Owner

First Responder

Window

NSArrayContr...

NSArrayController Info

Bindings

Availability

- ▶ editable

Controller Content

- ▼ contentArray Bind
- Bind to: File's Owner (NSApplication)
- Controller Key:
- Model Key Path: delegate.passwords
- Value Transformer:
- Conditionally Sets Editable
- Handles Content As Compound Value
- Raises For Not Applicable Keys
- Selects All When Setting Content
- Validates Immediately

- ▶ contentArrayForMultipleSelection
- ▶ contentObject



Bind the user column

The screenshot displays the Xcode interface with a table column selected. The table has a column named 'user' and rows indexed 0 through 9. The 'NSTableColumn Info' panel is open, showing the following configuration:

- value** (with a 'Bind' checkbox):
 - Bind to: NSArrayController
 - Controller Key: arrangedObjects
 - Model Key Path: user
- Value Transformer: (empty)
- Allows Editing Multiple Values Selection
- Conditionally Sets Editable
- Conditionally Sets Enabled
- Continuously Updates Value
- Raises For Not Applicable Keys
- Validates Immediately
- Multiple Values Placeholder: (empty)
- No Selection Placeholder: (empty)
- Not Applicable Placeholder: (empty)

At the bottom, the 'File's Owner' and 'First Responder' sections are visible, showing a 'Window' object and an 'NSArrayContr...' object.



Bind the uid column

The screenshot shows a window titled "Window" containing a table view with 10 rows (0-9) and a column named "uid". The "NSTableColumn Info" panel is open, showing the binding configuration for the "uid" column. The "value" section is expanded, and the "Bind" checkbox is checked. The binding configuration is as follows:

- Bind to: NSArrayController
- Controller Key: arrangedObjects
- Model Key Path: uid

Below the binding configuration, there are several checkboxes for additional options:

- Allows Editing Multiple Values Selection
- Conditionally Sets Editable
- Conditionally Sets Enabled
- Continuously Updates Value
- Raises For Not Applicable Keys
- Validates Immediately

At the bottom of the panel, there are three text fields for placeholders:

- Multiple Values Placeholder:
- No Selection Placeholder:
- Not Applicable Placeholder:

The "File's Owner" and "First Responder" sections at the bottom of the panel show the "Window" and "NSArrayContr..." objects respectively.



Viewer.py

```
from PyObjCTools import AppHelper
from Foundation import *
from AppKit import *
import os

# another shameless anti-line-wrapping hack
FIELDS = """
user password uid gid class change
expire gecost home_dir shell
""".split()

class ViewerAppDelegate(NSObject):
    def init(self):
        self = super(ViewerAppDelegate, self).init()
        self.passwords = [
            dict(zip(FIELDS, line.rstrip().split(':')))
            for line in os.popen('/usr/bin/nidump passwd .')
            if line and not line.startswith('#')
        ]
        return self

if __name__ == '__main__':
    AppHelper.runEventLoop()
```



Build and Run Viewer

Build (redistributable!):

```
% py2applet Viewer.py MainMenu.nib
```

Run:

```
% open Viewer.app
```

Done:

user	uid
nobody	-2
root	0
daemon	1
unknown	99
smmsp	25
lp	26
postfix	27
www	70
eppc	71



Bindings give you sorting for free!

user	uid
appserver	79
bob	501
cyrus	77
daemon	1
eppc	71
lp	26
mailman	78
mysql	74
nobody	-2
postfix	27
postgres	401
qtss	76
root	0
smmsp	25
sshd	75



Help!

Documentation:

[/Developer/Python/PyObjC/Documentation](#)

Examples:

[/Developer/Python/PyObjC/Examples](#)

Wiki:

<http://pythonmac.org/wiki>

IRC:

#macpython (on freenode)

Mailing Lists:

- pyobjc-dev@lists.sourceforge.net
- pythonmac-sig@python.org



Help! (Objective-C)

Documentation:

<http://developer.apple.com/>

Examples:

[/Developer/Examples/AppKit](#)

Wiki:

<http://cocoadev.com/>

Mailing List:

cocoa-dev@lists.apple.com



ReSTedit

The screenshot shows a window titled "slides.rst" with a menu bar containing "Mode" and "Open HTML in Browser". The window is split into two panes. The left pane shows the raw ReST source code, and the right pane shows the rendered HTML output.

Left Pane (Source Code):

```
Introduction to PyObjC
-----

Author
  Bob Ippolito

Conference
  PyCon DC, March 2005

Intended Audience
-----

- Python developers using Mac OS X 10.3 or
  later
- Spies from the Linux and Win32 camps
- Hopefully a GNUstep porter/maintainer

Topics
-----

- Installing PyObjC
- Why Bother?
- Interface Builder
- Objective-C Primer
- Crossing the Bridge
- Your First Application
Help
```

Right Pane (Rendered HTML):

- Introduction to PyObjC**
- Author
Bob Ippolito
- Conference
PyCon DC, March 2005
- Intended Audience**

 - Python developers using Mac OS X 10.3 or later
 - Spies from the Linux and Win32 camps
 - Hopefully a GNUstep porter/maintainer

- Topics**

- Installing PyObjC

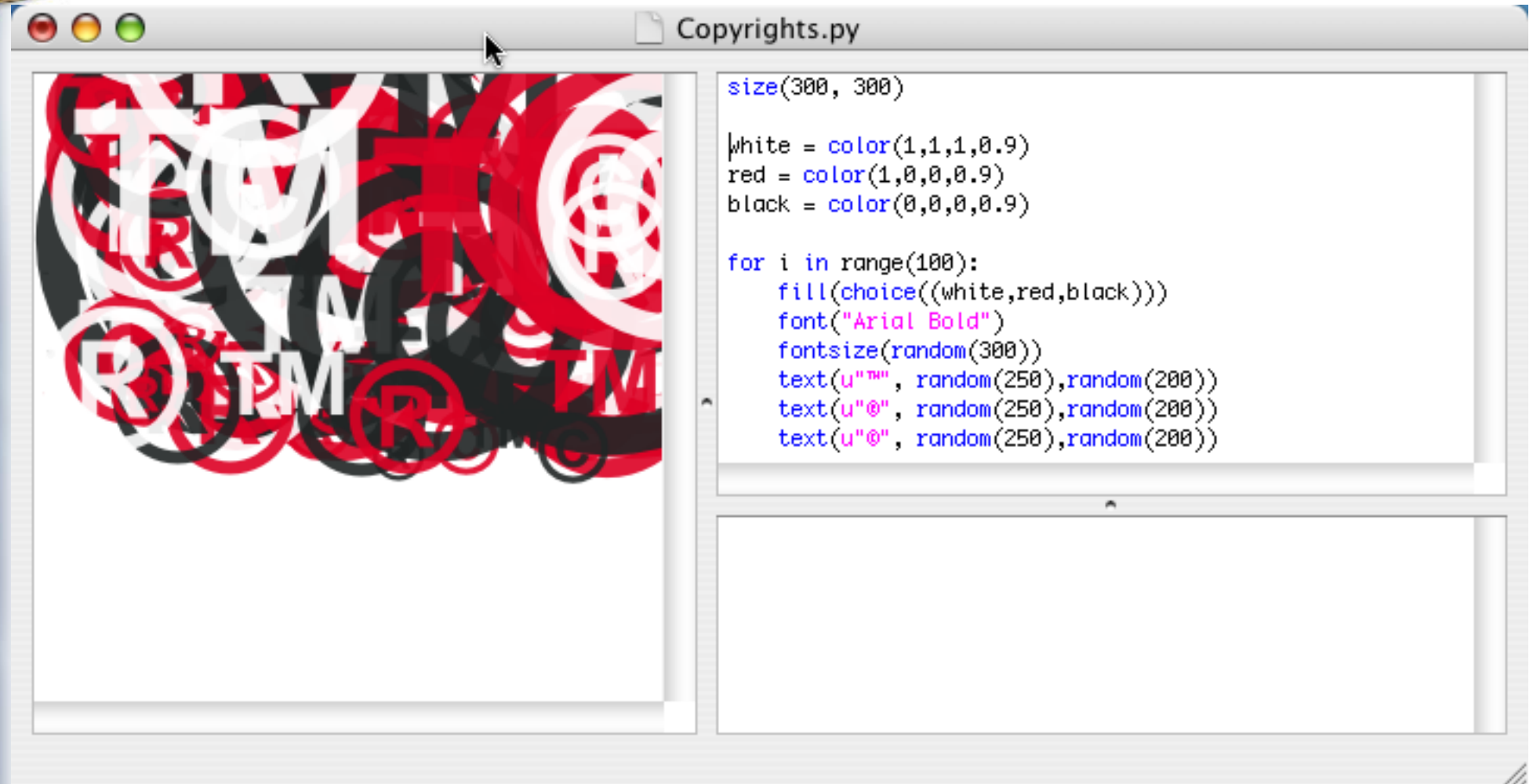


Flame

Host	Service
▶ Andrew Dalke's Computer [00:0a:95:68:26:c8] (1	
▼ Andrew Gross (10.0.43.224)	
iChat 2 presence	Andrew Gross
Remote login	mitya
Personal file sharing	mitya
Workgroup Manager	mitya [00:11:24:73:74:88
iTunes shared music	arg
iTunes remote control	iTunes_Ctrl_9DF57C44AD
▶ Bob Ippolito (10.0.40.155)	
▶ Daniel Krech (10.0.43.214)	
▶ David Goodger's Computer (10.0.40.157)	
▶ Drifty's Computer [00:0d:93:c5:a0:b6] (10.0.40.1	
▼ Ian Bicking's Computer (10.0.42.153)	
Remote login	Ian Bicking's Computer
Personal file sharing	Ian Bicking's Computer
Workgroup Manager	Ian Bicking's Computer [0
FTP server	Ian Bicking's Computer
Web server	Emily Murphy
_MacOSXDupSuppress._tcp.	-366817258;-36681725
▶ James Knight's Computer [00:0a:95:a5:0f:b2] (1C	
▶ Linden Wright (10.0.43.159)	
▶ MailMaster [00:0a:95:ca:1e:cc] (10.0.41.184)	
▶ Nicholas Bastin's Computer [00:0d:93:29:27:fe] (



NodeBox





Go ahead, ask.

Questions?