

wxPython Recipes

A Problem - Solution Approach

—

Mike Driscoll



Apress®

wxPython Recipes

A Problem - Solution Approach

Mike Driscoll

Apress®

uxPython Recipes

Mike Driscoll
Ankeny, New York, USA

ISBN-13 (pbk): 978-1-4842-3236-1
<https://doi.org/10.1007/978-1-4842-3237-8>

ISBN-13 (electronic): 978-1-4842-3237-8

Library of Congress Control Number: 2017963132

Copyright © 2018 by Mike Driscoll

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logo, or image we use the names, logos, and images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Cover image designed by Freepik

Managing Director: Welmoed Spahr
Editorial Director: Todd Green
Acquisitions Editor: Todd Green
Development Editor: James Markham
Technical Reviewer: Kevin Ollivier and Andrea Gavana
Coordinating Editor: Jill Balzano
Copy Editor: Lori Jacobs
Compositor: SPi Global
Indexer: SPi Global
Artist: SPi Global

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail rights@apress.com, or visit <http://www.apress.com/rights-permissions>.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at <http://www.apress.com/bulk-sales>.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/9781484232361. For more detailed information, please visit <http://www.apress.com/source-code>.

Printed on acid-free paper

This book is dedicated to the wxPython community

Table of Contents

About the Author	xvii
About the Technical Reviewers	xix
Acknowledgments	xxi
Chapter 1: Introduction.....	1
Who Should Read This Book	1
About the Author	2
Conventions	2
Requirements.....	3
Book Source Code.....	4
Reader Feedback	4
Errata	5
Chapter 2: Working with Images	7
Recipe 2-1. How to Take a Screenshot of Your wxPython App	7
Problem	7
Solution	8
How It Works	10
The Snapshot Printer Script	11
Recipe 2-2. How to Embed an Image in the Title Bar	14
Problem	14
Solution	15
How It Works	15
Using Your Own Image.....	17
Create the Image in Python Code	18

TABLE OF CONTENTS

- Recipe 2-3. How to Put a Background Image on a Panel 20
 - Problem 20
 - A Bad Example 20
 - Solution 22
 - A Better Example 22
- Chapter 3: Special Effects 27**
- Recipe 3-1. Resetting the Background Color 27
 - Problem 27
 - Solution 27
 - How It Works 29
- Recipe 3-2. How to Create a “Dark Mode” 30
 - Problem 30
 - Solution 30
 - How It Works 33
 - Trying Out Dark Mode 33
- Recipe 3-3. How to Fade-in a Frame/Dialog 37
 - Problem 37
 - Solution 37
 - How It Works 38
- Recipe 3-4. Making Your Text Flash 39
 - Problem 39
 - Solution 39
 - How It Works 40
 - Creating Changing Text 41
- Chapter 4: The Publish–Subscribe Pattern 43**
- Recipe 4-1. An Intro to Pubsub 43
 - Problem 43
 - Solution 43
 - How It Works 45

Recipe 4-2. Using PyDispatcher Instead of PubSub.....	46
Problem	46
Solution	47
How It Works	49
Chapter 5: Wizard Recipes	51
Recipe 5-1. Creating a Simple Wizard	51
Problem	51
Solution	52
How It Works	53
Using PyWizardPage.....	54
Recipe 5-2. How to Disable a Wizard's Next Button	58
Problem	58
Solution	59
How It Works	60
Getting It to Work with wxPython 4/Phoenix	63
Recipe 5-3. How to Create a Generic Wizard	66
Problem	66
Solution	66
How It Works	67
Chapter 6: Creating Simple Widgets.....	71
Recipe 6-1. Creating an About Box	71
Problem	71
Solution	71
How It Works	71
Using HtmlWindow for an About Box.....	73
Updating the Code for wxPython 4/Phoenix	78
Recipe 6-2. Creating Graphs with PyPlot.....	79
Problem	79
Solution	80
How It Works	82

TABLE OF CONTENTS

- Graphing Using Saved Data 83
- Point Plot with Thousands of Points 86
- Creating a Sine/Cosine Graph..... 89
- Recipe 6-3. Creating a Simple Notebook 91
 - Problem 91
 - Solution 92
 - How It Works 93
 - The Refactored Notebook..... 94
- Chapter 7: Using Config Files..... 97**
- Recipe 7-1. Generating a Dialog from a Config File 97
 - Problem 97
 - Solution 98
 - How It Works 101
- Recipe 7-2. Saving Data to a Config File..... 102
 - Problem 102
 - Solution 102
 - Creating a Controller..... 103
 - How It Works 104
 - Creating the View 104
- Chapter 8: Working with Events 111**
- Recipe 8-1. Binding Multiple Widgets to the Same Handler 111
 - Problem 111
 - Solution 112
 - How It Works 113
- Recipe 8-2. How to Fire Multiple Event Handlers 114
 - Problem 114
 - Solution 114
 - How It Works 115

Recipe 8-3. Get the Event Name Instead of an Integer	116
Problem	116
Solution	116
How It Works	118
Recipe 8-4. Catching Key and Char Events.....	119
Problem	119
Solution	119
How It Works	120
Catching Char Events	124
Recipe 8-5. Learning About Focus Events	125
Problem	125
Solution	126
How It Works	127
Losing Focus	130
Chapter 9: Drag and Drop	133
Recipe 9-1. How to Use Drag and Drop	133
Problem	133
Solution	133
Creating a FileDropTarget.....	134
How It Works	136
Creating a TextDropTarget	137
Custom DnD with PyDropTarget	139
Creating a Custom Drag-and-Drop App.....	142
Recipe 9-2. How to Drag and Drop a File from Your App to the OS.....	146
Problem	146
Solution	146
How It Works	149

Chapter 10: Working with Frames 151

- Recipe 10-1. Using wx.Frame Styles 151
 - Problem 151
 - Solution(s) 152
 - How It Works 153
- Create a Frame Without a Caption 154
- Create a Frame with a Disabled Close Button..... 156
- Create a Frame Without Maximize/Minimize 157
- Create a Un-Resizable Frame 158
- Create a Frame Without a System Menu..... 159
- Create a Frame That Stays on Top 160
- Recipe 10-2. Making Your Frame Maximize or Full Screen 162
 - Problem 162
 - Solution 162
 - How It Works 163
- Making Your Application Full Screen..... 164
- Recipe 10-3. Ensuring Only One Instance per Frame 165
 - Problem 165
 - Solution 165

Chapter 11: wxPython and the System Tray 169

- Recipe 11-1. Creating Taskbar Icons 169
 - Problem 169
 - Solution 169
- Creating the TaskBarIcon in Classic..... 169
 - How It Works 171
- Creating the TaskBarIcon in wxPython 4 172
- Recipe 11-2. Minimizing to the System Tray..... 175
 - Problem 175
 - Solution 175
 - How It Works 176
- Making the Application Minimize to Tray 178

Chapter 12: Fun with Panels.....	181
Recipe 12-1. Making a Panel Self-Destruct.....	181
Problem	181
Solution	181
How It Works	183
Recipe 12-2. How to Switch Between Panels.....	184
Problem	184
Solution	185
How It Works	188
Chapter 13: Using Objects in Widgets	189
Recipe 13-1. Using ObjectListView Instead of ListCtrl	189
Problem	189
Solution	190
How It Works	192
Recipe 13-2. Storing Objects in ComboBox or ListBox	194
Problem	194
Solution	195
How It Works	197
Adding Objects to the wx.ComboBox	197
Chapter 14: XML and XRC.....	201
Recipe 14-1. Extracting XML from the RichTextCtrl	201
Problem	201
Solution	201
How It Works	202
Updating for wxPython 4.....	203
Recipe 14-2. An Introduction to XRC.....	204
Problem	204
Solution	205
How It Works	207

TABLE OF CONTENTS

- Creating a Notebook with XRC 208
- Adding Controls Outside XRC 212
- Recipe 14-3. An Introduction to XRCed 214
 - Problem 214
 - Solution 214
 - How It Works 216
- Creating Something More Complex 218
- Using XRCed to Generate Python Code 220
- Recipe 14-4. How to Create a Grid in XRC 222
 - Problem 222
 - Solution 224
 - How It Works 225
- Chapter 15: Working with Sizers 227**
- Recipe 15-1. How to Get Children Widgets from a Sizer 227
 - Problem 227
 - Solution 228
 - How It Works 229
- Recipe 15-2. How to Center a Widget 229
 - Problem 229
 - Solution #1—Using Faux Spacers 230
 - How It Works 231
 - Solution #2—Using an AddStretchSpacer 231
 - How It Works 232
 - Solution #3—Centering Without Nested Sizers 232
- Recipe 15-3. How to Make Widgets Wrap 233
 - Problem 233
 - Solution 234
 - How It Works 235
- Recipe 15-4. Adding/Removing Widgets Dynamically 236
 - Problem 236
 - Solution 237

Chapter 16: Threads and Timers	241
Recipe 16-1. How to Update a Progress Bar from a Thread	241
Problem	241
Solution for wxPython 2.8.12 and Earlier	242
How It Works	244
Solution for wxPython 3 and Newer	246
How It Works	248
wx.PostEvent and Threads.....	248
Recipe 16-2. How to Update a Progress Bar from a Thread	252
Problem	252
Solution	252
Updating the Code for wxPython 3.0.2.0 and Newer	257
Recipe 16-3. A wx.Timer Tutorial	260
Problem	260
Solution	261
How It Works	262
Using Multiple Timers	264
Chapter 17: Redirecting Text	269
Recipe 17-1. Redirect Python’s Logging Module to a TextCtrl	269
Problem	269
Solution	269
How It Works	272
Recipe 17-2. Redirecting stdout/stderr.....	273
Problem	273
Solution—The Thread-Safe Method.....	274
How It Works	275
Solution—The Non-Thread-Safe Method.....	276
Recipe 17-3. How to Use the Clipboard	279
Problem	279
Solution	280
How It Works	282

TABLE OF CONTENTS

- Chapter 18: Grid Recipes 283**
 - Recipe 18-1. Syncing Scrolling Between Two Grids 283
 - Problem 283
 - Solution 283
 - How It Works 285
 - Recipe 18-2. How to Get Selected Cells in a Grid 286
 - Problem 286
 - Solution 287
 - How It Works 290

- Chapter 19: Working with Your Application..... 291**
 - Recipe 19-1. How to Edit Your GUI Interactively Using reload() 291
 - Problem 291
 - Solution 292
 - How It Works 294
 - Recipe 19-2. Updating Your Application with Esky 295
 - Problem 295
 - Solution 295
 - How It Works 301
 - Recipe 19-3. How to Create a Log-in Dialog 310
 - Problem 310
 - Solution 311
 - Using an Instance Variable..... 314

- Chapter 20: Bonus Recipes..... 317**
 - Recipe 20-1. Catching Exceptions from Anywhere..... 317
 - Problem 317
 - Solution 317
 - How It Works 319
 - Creating an Exception-Catching Decorator 319

Recipe 20-2. wxPython's Context Managers	325
Problem	325
Solution	326
How It Works	327
Recipe 20-3. Converting wx.DateTime to Python datetime	331
Problem	331
Solution	331
How It Works	332
Recipe 20-4. Creating a URL Shortener	333
Problem	333
Solution	333
Shortening URLs with Other Shorteners	336
Index.....	343

About the Author



Mike Driscoll started coding in Python in 2006, where his first assignments included porting Windows log-in scripts and VBA to Python, which introduced him to wxPython. He's done back-end programming and front-end user interfaces, writes documentation for wxPython, and currently maintains an automated testing framework in Python. He also owns the popular site “Mouse vs Python” at pythonlibrary.org and has written for the Python Software Foundation and DZone and published *Python 101* and *Python 201*.

About the Technical Reviewers



Kevin Ollivier is a software developer who has been working with Python for nearly 20 years. He has been an avid supporter of open source and has contributed to numerous projects, including wxPython. When he's not coding, he's usually either reading, catching up on the latest anime and superhero shows, or gaming. In addition to coding work that he performs for various clients, he is currently working on an educational role-playing game (RPG) called BrightSparc. You can learn more about him and his projects at his company web site: <http://kosoftworks.com>.



Andrea Gavana has been programming Python for almost 15 years, and dabbling with other languages since the late 1990s.

He graduated from university with a Master's Degree in Chemical Engineering, and he is now a Senior Reservoir Engineer working for Maersk Oil in Copenhagen, Denmark.

Andrea enjoys programming at work and for fun, and he has been involved in multiple open source projects, all Python-based.

One of his favorite hobbies is Python coding, but he is also fond of cycling, swimming, and cozy dinners with family and friends.

This is his first book as technical reviewer.

Acknowledgments

I just wanted to take a moment and say thank you to some of the people who have helped me in writing this book.

My technical reviewers, Andrea Gavana and Kevin Ollivier, were very helpful both in the polishing of this book and in my growth as a Python programmer from practically the beginning of my learning of the language.

The wxPython community itself inspired me to write about Python in general and wxPython in particular. They were always encouraging me when I was just starting out learning Python and wxPython and they still are.

I would also like to thank all my blog readers who have reached out to me over the years and asked me to start writing books.

Robin Dunn, the creator of wxPython, has been very helpful to me personally in figuring out wxPython and in the writing of this work. I have asked him repeatedly for help in regard to some of my code examples that worked in one version of wxPython and not in another, or code that worked in one operating system, but didn't behave the same way somewhere else. He has always been patient with me and pointed me in the right direction.

Finally, I would like to thank my family for their support.

And special thanks to you, dear reader, for picking this book up and giving it a chance.

CHAPTER 1

Introduction

Welcome to my wxPython recipes book! As with most cookbooks, this one is made up of a series of recipes. Some recipes will be more involved than others, but most of the time, the recipe will be a nice bite-sized chunk of information that only covers three to five pages or so. There are more than 50 recipes in this book. I have compiled them over the last eight years from people who have asked questions on the wxPython mailing list, StackOverflow, or e-mailed me directly.

Normally I would spend a lot of time in the introduction going over each section of the book, but since this book is a series of recipes, it won't actually be split into sections. Instead, the recipes will be grouped where possible. For example, I have a number of XRC-related recipes, so they will be kept together as a single chapter.

The recipes will include screenshots of the interfaces that you will be creating. There will be additional screenshots included if and when we change the code inside a recipe. A good example of this is in the Frame Styles recipe where we try out various flags that affect how **wx.Frame** is displayed.

Who Should Read This Book

This book is targeted at people who are already familiar with the Python programming language and also have a basic understanding of wxPython. At the very least, it would be helpful if the reader understands event loops and the basics of creating user interfaces (UIs) with another Python UI toolkit, such as **Tkinter** or **PyQt**.

About the Author

You may be wondering who I am and why I might be knowledgeable enough about Python to write about it, so I thought I'd give you a little information about myself. I started programming in Python in Spring 2006 for a job. My first assignment was to port Windows log-in scripts from Kixtart to Python. My second project was to port VBA code (basically a graphical user interface, or GUI, on top of Microsoft Office products) to Python, which is how I first got started in wxPython. I've been using Python ever since, doing a variation of back-end programming and desktop front-end UIs. Currently I am writing and maintaining an automated test framework in Python.

I realized that one way for me to remember how to do certain things in Python was to write about them and that's how my Python blog came about: www.blog.pythonlibrary.org/. As I wrote, I would receive feedback from my readers and I ended up expanding the blog to include tips, tutorials, Python news, and Python book reviews. I work regularly with Packt Publishing as a technical reviewer, which means that I get to try to check for errors in the books before they're published. I also have written for the Developer Zone (DZone) and i-programmer web sites as well as the Python Software Foundation. In November 2013, DZone published **The Essential Core Python Cheat Sheet**, which I coauthored. Finally, I have also self-published the following two books:

- **Python 101**, which came out in June 2014.
- **Python 201: Intermediate Python**, which came out in September 2016

Conventions

As with most technical books, this one includes a few conventions that you need to be aware of. New topics and terminology will be in **bold**. You will also see some examples that look like the following:

```
>>> myString = "Welcome to Python!"  
...  

```

The `>>>` is a Python prompt symbol. You will see this in the Python **interpreter** and in **IDLE**. Other code examples will be shown in a similar manner, but without the `>>>`.

Requirements

You will need a working **Python 2** or **Python 3** installation. Most Linux and Mac machines come with Python already installed; however, they might not have Python in their path. This is rare, but if it happens there are lots of tutorials on the Internet that explain how to add Python to your path for your particular operating system. If you happen to find yourself without Python, you can download a copy from <http://python.org/download/>. There are up-to-date installation instructions on the web site, so I won't include any installation instructions in this book for Python itself.

The wxPython toolkit is **not** included with Python. We will look at how to install it here. You will want to use the latest version of wxPython, which at the time of writing, is version 4. It also based on the Phoenix branch of wxPython instead of Classic. You don't really need to know the differences between these other than Phoenix supports Python 2 and 3 while Classic does not.

To install wxPython 4, you can just use pip:

```
pip install wxPython
```

This works great on Windows and Mac. I have noticed that on some versions of Linux, you may see an error or two about missing dependencies, such as webkit. You will need to install the listed dependency and then try installing wxPython again.

Once you're done installing wxPython, we can check to make sure it works with the following script:

```
import platform
import wx

class MyFrame(wx.Frame):
    """
    """

    def __init__(self):
        """Constructor"""
        wx.Frame.__init__(self, None, size=(500, 200),
                           title='Version Info')
        panel = wx.Panel(self)

        py_version = 'Python version: ' + platform.python_version()
        wx_version = 'wxPython version: ' + wx.version()
        os_version = 'Operating System: ' + platform.platform()
```

```
main_sizer = wx.BoxSizer(wx.VERTICAL)
size = (20, -1)
main_sizer.Add(
    wx.StaticText(panel, label=py_version), 0, wx.ALL, 5)
main_sizer.Add(
    wx.StaticText(panel, label=wx_version), 0, wx.ALL, 5)
main_sizer.Add(
    wx.StaticText(panel, label=os_version), 0, wx.ALL, 5)
panel.SetSizer(main_sizer)

self.Show()

if __name__ == '__main__':
    app = wx.App(False)
    frame = MyFrame()
    app.MainLoop()
```

This code should run without error and you will see a simple UI appear on screen. Any additional requirements will be explained later on in the book.

Book Source Code

The book's source code can be found on Github:

https://github.com/driscollis/wxPython_recipes_book_code

Reader Feedback

I welcome feedback about my writings. If you'd like to let me know what you thought of the book, you can send comments to the following address:

comments@pythonlibrary.org

Errata

I try my best not to publish errors in my writings, but it happens from time to time. If you happen to see an error in this book, feel free to let me know by e-mailing me at the following:

`errata@pythonlibrary.org`

Now let's get started!

CHAPTER 2

Working with Images

Recipe 2-1. How to Take a Screenshot of Your wxPython App

Problem

Have you ever thought that it would be cool to have your wxPython code take a screenshot of itself? Well, Andrea Gavana (one of wxPython's core developers) figured out a cool way to do just that and between what he told us on the wxPython mailing list and what I learned from other sources, you will soon learn how to not only take the screenshot but send it to your printer! Once it's all done, you'll have an application that looks like Figure 2-1.

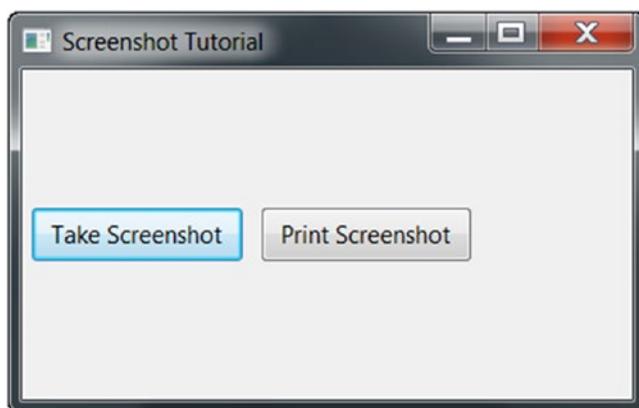


Figure 2-1. Taking a screenshot

Solution

You can tackle this project in several different ways. You could create the code that actually takes the screenshot or you could write an application that calls that code. We will start by creating an application that takes screenshots. Let's take a look.

Listing 2-1. The Code for Taking a Screenshot

```
import sys
import wx
import snapshotPrinter

class MyForm(wx.Frame):
    def __init__(self):
        wx.Frame.__init__(self, None, title="Screenshot Tutorial")

        panel = wx.Panel(self)
        screenshotBtn = wx.Button(panel, label="Take Screenshot")
        screenshotBtn.Bind(wx.EVT_BUTTON, self.onTakeScreenShot)
        printBtn = wx.Button(panel, label="Print Screenshot")
        printBtn.Bind(wx.EVT_BUTTON, self.onPrint)

        sizer = wx.BoxSizer(wx.HORIZONTAL)
        sizer.Add(screenshotBtn, 0, wx.ALL|wx.CENTER, 5)
        sizer.Add(printBtn, 0, wx.ALL|wx.CENTER, 5)
        panel.SetSizer(sizer)

    def onTakeScreenShot(self, event):
        """
        Takes a screenshot of the screen at given pos & size (rect).

        Method based on a script by Andrea Gavana
        """
        print('Taking screenshot...')
        rect = self.GetRect()

        # adjust widths for Linux (figured out by John Torres
        # http://article.gmane.org/gmane.comp.python.wxpython/67327)
```

```

if sys.platform == 'linux2':
    client_x, client_y = self.ClientToScreen((0, 0))
    border_width = client_x - rect.x
    title_bar_height = client_y - rect.y
    rect.width += (border_width * 2)
    rect.height += title_bar_height + border_width

# Create a DC for the whole screen area
dcScreen = wx.ScreenDC()

# On Windows and Mac, we can just call GetAsBitmap on the
wx.ScreenDC
# and it will give us what we want.
bmp = dcScreen.GetAsBitmap().GetSubBitmap(rect)

if not bmp.IsOk():
    # Create a Bitmap that will hold the screenshot image later on
    # Note that the Bitmap must have a size big enough to hold the
    screenshot
    # -1 means using the current default colour depth
    bmp = wx.EmptyBitmap(rect.width, rect.height)

    #Create a memory DC that will be used for actually taking the
    screenshot
    memDC = wx.MemoryDC()

    # Tell the memory DC to use our Bitmap
    # all drawing action on the memory DC will go to the Bitmap now
    memDC.SelectObject(bmp)

    # Blit (in this case copy) the actual screen on the memory DC
    # and thus the Bitmap
    memDC.Blit( 0, # Copy to this X coordinate
               0, # Copy to this Y coordinate
               rect.width, # Copy this width
               rect.height, # Copy this height
               dcScreen, # Where to copy from

```

```

        rect.x, # What's the X offset in the original DC?
        rect.y # What's the Y offset in the original DC?
    )

    # Select the Bitmap out of the memory DC by selecting a new
    # uninitialized Bitmap
    memDC.SelectObject(wx.NullBitmap)

    img = bmp.ConvertToImage()
    fileName = "myImage.png"
    img.SaveFile(fileName, wx.BITMAP_TYPE_PNG)
    print('...saving as png!')

def onPrint(self, event):
    """
    Send screenshot to the printer
    """
    printer = snapshotPrinter.SnapshotPrinter()
    printer.sendToPrinter()

# Run the program
if __name__ == "__main__":
    app = wx.App(False)
    frame = MyForm()
    frame.Show()
    app.MainLoop()

```

How It Works

This piece of code creates a frame with two buttons in it. It's a bit boring, but this is just a simple example after all. The part we care about most is the **onTakeScreenshot** method. As I mentioned earlier, it is based on a script by Andrea Gavana. However, I added a conditional from John Torres that makes this script behave better on Linux since it was originally written for Windows. The comments tell the story of the code, so take your time reading them and when you're done, we can move on to how we can send our result to the printer.

The Snapshot Printer Script

Creating a simple application that can take a screenshot and print it isn't that much more work than just taking a screenshot. You will be able to combine this script with the previous one to make a complete screenshot and printing utility.

The printing utility will end up looking something as shown in Figure 2-2.

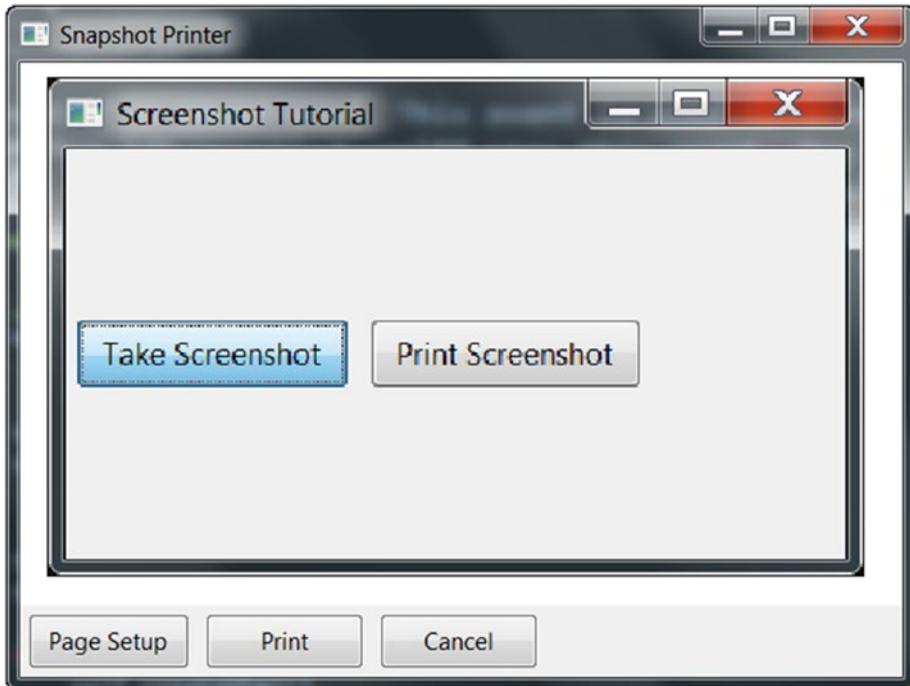


Figure 2-2. *Printing a screenshot*

This initial script actually has the image hard-coded into it, so if you'd like to save the image with a different name, you'll need to add that feature yourself. Let's take a moment to read through the code though, as shown in Listing 2-2:

Listing 2-2. The Application Code That Calls the Screenshot Code

```

# snapshotPrinter.py

import os
import wx
from wx.html import HtmlEasyPrinting, HtmlWindow

class SnapshotPrinter(wx.Frame):

    def __init__(self, title='Snapshot Printer'):
        wx.Frame.__init__(self, None, title=title,
                           size=(650,400))

        self.panel = wx.Panel(self)
        self.printer = HtmlEasyPrinting(
            name='Printing', parentWindow=None)

        self.html = HtmlWindow(self.panel)
        self.html.SetRelatedFrame(self, self.GetTitle())

        if not os.path.exists('screenshot.htm'):
            self.createHtml()
        self.html.LoadPage('screenshot.htm')

        pageSetupBtn = wx.Button(self.panel, label='Page Setup')
        printBtn = wx.Button(self.panel, label='Print')
        cancelBtn = wx.Button(self.panel, label='Cancel')

        self.Bind(wx.EVT_BUTTON, self.onSetup, pageSetupBtn)
        self.Bind(wx.EVT_BUTTON, self.onPrint, printBtn)
        self.Bind(wx.EVT_BUTTON, self.onCancel, cancelBtn)

        sizer = wx.BoxSizer(wx.VERTICAL)
        btnSizer = wx.BoxSizer(wx.HORIZONTAL)

        sizer.Add(self.html, 1, wx.GROW)
        btnSizer.Add(pageSetupBtn, 0, wx.ALL, 5)
        btnSizer.Add(printBtn, 0, wx.ALL, 5)

```

```

    btnSizer.Add(cancelBtn, 0, wx.ALL, 5)
    sizer.Add(btnSizer)

    self.panel.SetSizer(sizer)
    self.panel.SetAutoLayout(True)

def createHtml(self):
    '''
    Creates an html file in the home directory of the application
    that contains the information to display the snapshot
    '''
    print('creating html...')

    html = '''<html>\n<body>\n<center>
<img src=myImage.png width=516 height=314>
</center>\n</body>\n</html>'''
    with open('screenshot.htm', 'w') as fobj:
        fobj.write(html)

def onSetup(self, event):
    self.printer.PageSetup()

def onPrint(self, event):
    self.sendToPrinter()

def sendToPrinter(self):
    self.printer.GetPrintData().SetPaperId(wx.PAPER_LETTER)
    self.printer.PrintFile(self.html.GetOpenedPage())

def onCancel(self, event):
    self.Close()

if __name__ == '__main__':
    app = wx.App(False)
    frame = SnapshotPrinter()
    frame.Show()
    app.MainLoop()

```