Python and Instant GUIs

Python has quite a few GUI tool-kits for creating large and powerful applications. But what if you want to quickly hack together a small script, without having to learn the API for a full-fledged GUI tool-kit? Read on...

There is a plethora of GUI tool-kits like GTK+, Qt, wxWidgets, FLTK, etc, which are available for different programming and scripting languages. Developing GUIs with scripting languages saves you the tedious compile-link-run cycles of languages like C, C++, etc, besides letting you run the same code on multiple platforms. Python, one of the most popular scripting languages, is great for GUI programming because of its elegance and since it has bindings for almost all popular GUI tool-kits. But most of these require a high level of initial learning, an unnecessary hurdle for when your needs are simple.

Most tool-kits are based on an event-driven model; applications built on these don't follow any predefined sequence of actions, but depend on the user's interaction with widgets in the application. In contrast, there's a model where you determine the presentation sequence for all widgets, or a grouping of widgets, for some general or frequently-required operation. The user interacts with these one at a time, and the application flow is strictly serial, from start to end. This is very useful for scripting short task-flows that proceed through the same steps each time, but with varying user inputs.

A third consideration is that the heavy dependencies of some tool-kits may make the distribution of simple applications problematic or 'bloated'. Sometimes you only need a minimalist approach—to put a few widgets together for a quick and easy GUI, without any hassles. Here are a couple of possible solutions.

EasyGUI is a wrapper over Python's official GUI tool-kit, Tkinter; PyZenity wraps the GTK+ dialogue box's utility, Zenity. These provide the most common widgets used in serial-flow GUI programming. These tools are simple to use, yet powerful. Let's get to know them better.

Note: I used the Ubuntu 11.10 64-bit edition to test the examples presented in this article.

Quick and easy GUIs with EasyGUI

Tkinter is an event-driven tool-kit, and takes time to learn and program—but the EasyGUI wrapper hides all the rocket science. In this set of many general GUI dialogue boxes, each one has its own event loop, and is independent of the others. It's a more feature-rich, true graphics-mode counterpart of shell utilities like *dialog*, *whiptail*, etc. EasyGUI is a standalone single-file Python module, plus a demo application based on Tkinter. It has no dependency

Figure 1: EasyGUI application in action

except Tkinter, which comes with Python on Windows, so EasyGUI works out of the box—but you have to install Tkinter on GNU/Linux since most distributions don't have it installed by default. In Ubuntu, a simple *sudo apt-get install python-tk* is all you need.

To get a feel of EasyGUI, download the tarball (the link is given in *References*), uncompress it, *cd* to the extracted directory, and run *python easygui.py*. The application (Figure 1) presents a listing for different dialogue boxes; click to try them out, one by one. The dialogue boxes are modal; only one kind is active at a time. As you can see, EasyGUI can create the GUI dialogue boxes listed in Table 1.

Table 1: EasyGUI dialogue boxes

boolbox	Presents the choice of 'Yes' or 'No'
buttonbox	Dialogue box to show programmer-defined buttons
buttonbox with an image	Adds the specified image to the box
ccbox	The *Continue* or *Cancel* dialogue box
choicebox	Select one of the choices shown
codebox	Display formatted text with space in a mono-space font
diropenbox	Directory path selection dialogue box
enterbox	Text entry dialogue box
enterbox with an image	Adds an image display
exceptionbox	Shows details of the most recent exception
fileopenbox, filesavebox	File *Open/Save* dialogue boxes
indexbox	*Buttonbox* dialogue box that returns the index of the button clicked
integerbox	Accepts an integer between specified upper and lower bounds
msgbox	Shows the specified message
multchoicebox	Selects one of multiple options
multenterbox	Multiple text entry dialogue box

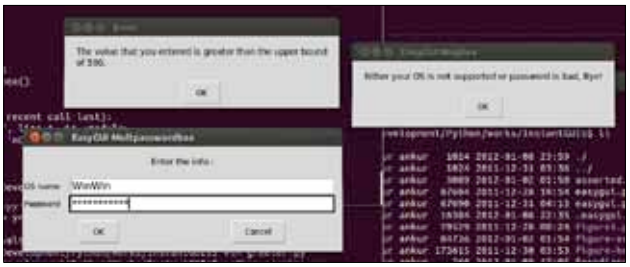

Figure 2: Some EasyGUI dialogue boxes

multpasswordbox	Same as the previous row, but the last field is a password entry
passwordbox	Accepts 'hidden' text (password entry)
textbox	Shows text with proportional font and line wrapping at word breaks
ynbox	Dialogue box to select from 'Yes' or 'No'

Most of these dialogue boxes don't show a title by default, but you can pass a title string as the second parameter or via the *title* = <*dialog title*> named parameter to the respective function. You can pass the text to show in dialogue boxes as the first parameter, or as *msg* = <*dialog message*>. The EasyGUI dialogue boxes return either None or an empty string if the user clicks *Cancel* in the dialogue box (wherever it's present). You can take appropriate decisions in your script based on these return values.

Now, it's time for some working examples. Run *greeter.py*, shown below, with *python greeter.py* in a text console. Enter any password except *fossrulz!!!* or any OS with win in its name, and see how it behaves. You will also notice that the *integerbox* will not accept values that are not between its lower and upper bounds. Figure 2 shows some of the dialogue boxes created by this example:

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

# import all the classes and functions from EasyGUI.
from easygui import *
def main():
    msg1 = 'The greeter needs OS name and password, do you know
both?'
    if 1 == ynbox(msg1, 'EasyGUI Ynbox'):
        msg2 = 'Enter the info:'
        flds = ('OS name', 'Password')
        vals = multpasswordbox(title = 'EasyGUI Multpasswordbox',
msg = msg2, fields = flds)
        if -1 != vals[0].lower().find('win') or 'fossrulz!!!' !=
vals[1]:
            err1 = 'Either your OS is not supported or password
is bad, Bye!'
            msgbox(err1, 'EasyGUI Msgbox')
```

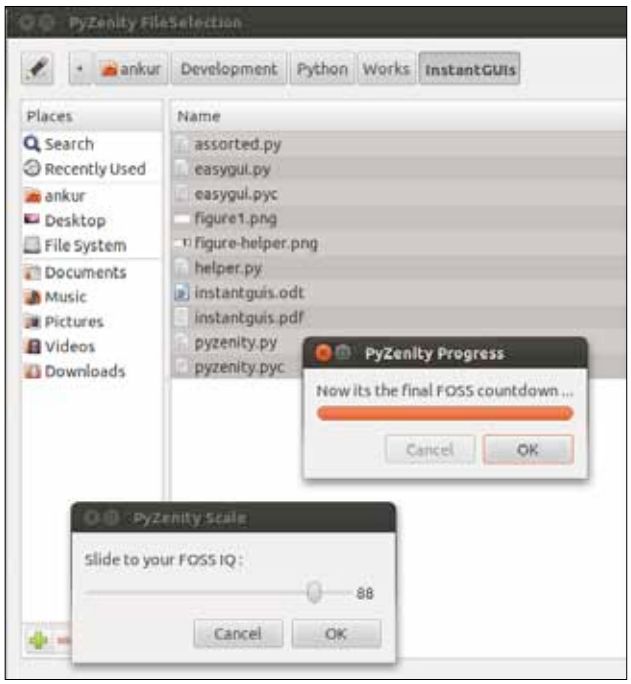

Figure 3: Assorted dialogue boxes provided by PyZenity

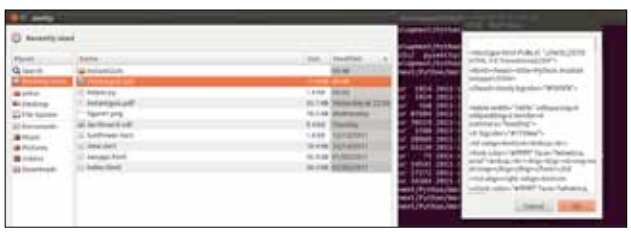

Figure 4: PyZenity helper function dialogue boxes

```
            return
        else:
            msg3 = 'How long do you wanna live with FOSS?'
            yrs = integerbox(msg3, 'EasyGUI Integerbox',
                             lowerbound = 1, upperbound = 500)
        if None != yrs:
            msg = 'You run %s and will live with FOSS for %s
years, long live FOSS and Geekz !!!' %(vals[0], str(yrs))
            textbox('Greetings from FOSS community', 'EasyGUI
Greetings', msg)

if '__main__' == __name__:
    main()
```

The EasyGUI class *EgStore* lets you save and retrieve persistent data. To explore the documentation further, click *Help* in the EasyGUI demo. In fact, the demo is the best example of its usage; read its source for working code examples.

Python meets Zenity with PyZenity

Zenity is a seasoned and well-known command-line utility that creates various GTK+ dialogue boxes to add GUIs to shell scripts. It comes pre-installed with Ubuntu, and you can create very useful applications with its dialogue boxes knitted together. One example is the Ubuntu Customisation Kit (a tool to customise various flavours of Ubuntu live CDs). PyZenity is an object-oriented wrapper over Zenity to expose its simplicity and power to Python. PyZenity is a single Python file; you need the subversion tool to get a copy from its Google Code repository. To install Subversion, run *sudo apt-get install subversion* and then follow the link in *References* to get a read-only copy of *PyZenity*. Just copy *pyzenity.py* from the *pyzenity* checkout directory to your application source folder. You can also install *pyzenity* with *sudo python setup.py install* in the pyzenity checkout directory, for system-wide availability. Please note that the Google Code version is not very recent, but it's easy to add features from the latest releases, as *pyzenity.py* is only around 454 lines. Recent versions of Zenity have additional colour selection, forms and password dialogue boxes that are missing from *pyzenity.py* but these could be added by following the existing dialogue box class code. PyZenity provides the classes in Table 2, to wrap almost all Zenity functionality.

Table 2: PyZenity classes

ZCalender	Creates a calendar dialogue box
ZEntry	An entry dialogue—visible, or password-style hidden text
ZError	An error notification dialogue box
ZInfo	An information notification dialogue box
ZFileSelection	Single or multiple file selection dialogue box
ZList	Check-list or radio-list style dialogue box
ZNotification	Task-bar notification text
ZProgress	Dialogue with progress bar
ZQuestion	A dialogue box requiring an answer to a query
ZText	Lets one read and modify text of a file
ZWarning	Warning dialogue box
ZScale	A dialogue box to accept a value via a slider widget

To see it in action, download *assorted.py* from *http://www.linuxforu.com/article_source_code/may12/instantgui_source_code.zip* and run it (*python assorted.py)* in a text console. Figure 3 shows some Zenity dialogue boxes created by *assorted.py*.

Here is some theory to help you use PyZenity more efficiently. In its object-oriented design, all the classes shown in Table 2 are derived from *ZenityBase*, because

Zenity provides some general-purpose arguments to modify the look and behaviour of dialogue boxes, besides other required and optional parameters for various dialogue boxes. Through this inheritance relationship, you can modify attributes of dialogue boxes by passing *parameter = value* when creating objects of different classes. You can see this in *assorted.py* as well. You can use these parameters to tweak the look and behaviour of the dialogue boxes:

title	Set the dialogue title
window_icon	Add a custom window icon for a dialogue box
width, height	Set width and height of a dialogue box
timeout	Set a time-out (in seconds) for a dialogue box to disappear, if no action is taken within this time.

PyZenity also provides some straightforward helper functions (Table 3) to create various dialogue boxes without any class-based programming—but if you use these, you can't modify dialogue box attributes like the title, height, width, time-out, etc. These come in useful if you want to minimise code and effort, for very small applications.

Table 3: PyZenity helper functions

get_date	Shows date selection dialogue box
get_short_text	Single-line text entry dialogue box
get_hidden_text	Password-style single-line entry dialogue box
show_error	Error notification dialogue box
show_info	Information dialogue box
get_file_path	A single file selection dialogue box
get_save_path	File save dialogue box
ask_question	A dialogue box for 'Yes' or 'No' responses to questions
show_file_text	Displays a dialogue box with specified file content
show_warning	A warning dialogue box
get_scale_value	Gets a slider value set in the dialogue box

Download *helper.py from http://www.linuxforu.com/article_source_code/may12/instantgui_source_code.zip* and run it to see these helper functions in action; some of the resultant dialogue boxes can be seen in Figure 4. Notice how easy it is to use these helper routines. Remember that they return an empty string if either the Cancel button or OK is clicked without entering anything in the entry dialogue boxes.

It's not always necessary to use feature-rich but complicated GUI tool-kits, especially when your requirements are simple. EasyGUI provides many general-purpose programmer-configurable dialogue boxes. On the other hand, PyZenity wraps the good old GTK+-based Zenity dialogue boxes for quick and simple GUI applications. Using both in an application could complement each other, one supplying the dialogue boxes missing in the other. So for your next simple GUI application, consider these simple solutions instead of complicated, resource-hungry GUI tool-kits. END

References

- EasyGUI home page: *http://easygui.sourceforge.net/*
- PyZenity SVN checkout *link: http://code.google.com/p/pyzenity/source/checkout*

By: Ankur Kumar Sharma

The author is a software developer and researcher. He is highly indebted to FOSS, rock music, Zen, the ancient sciences and all other mysterious things for keeping him always on his toes. You could find his other FOSS stuff on *http://www.richnsgeeks.com/* and do provide your valuable feedback.