Rapid GUI Application Development with Python

Volker Kuhlmann

Kiwi PyCon 2009 — Christchurch

7 November 2009

Abstract

Options and tools for rapid desktop GUI application development using Python are examined, with a list of options for each stage of the tool chain.

Keywords

kiwipycon, kiwipycon09, Python, GUI, RAD, GUI toolkit, GUI builder, IDE, application development, multi-platform

1 Introduction

GUI options are examined for Python with respect to performance, ease of use, runtime, and platform portability. Emphasis is put on platform-independence and free open source, however, commercial options are included.

This paper is a summary of the conference presentation. The related slides are available online[2]. A full paper is planned.

The following components of the tool chain are discussed:

- Python interpreter
- GUI toolkit
- IDE (Integrated Development Environment)
- GUI builder
- Debugger
- Database
- Testing
- Application packaging

2 Python Interpreter

Getting the Python interpreter up and running is the easiest part of the tool chain, and is achieved by simply installing the latest stable version for your platform—currently 2.6 or 3.1. The 3.x version is not totally backwards compatible. Pre-compiled packages are available for each platform.

For a well-written easy to understand introduction to Python programming, see the book by Hetland[3].

3 GUI toolkit

The GUI toolkit provides the graphical user interface (GUI) elements, or widgets, like buttons, scrollbars, text boxes, label fields, tree view lists, file-open dialogues and so on. These widgets are an integral and essential part of the user interface.

Most of these widget collections are available as a system library and as such are used not only by Python. The best known ones are Qt and GTK, others are Motif/lesstif, Tk, athena. Java has its own. The main differences are in features, ease of use, native look-and-feel, and license.

To use these widget libraries with Python, a wrapper is needed.

The main GUI toolkits for Python are Tkinter, wxPython, PyGTK and PyQt/PySide. Polo published a very good discussion of their merits[4]. A detailed reference for Python Qt programming is in the book by Summerfield[5].

4 IDE

The Integrated Development Environment is like a workbench. It always consists of an editor, and provides quick access to other tools. Often it also gives quick access to relevant documentation.

Desirable editor features include:

- Code completion, e.g. as soon as an identifier is typed enough to be unambiguous.
- Calltips—a popup with e.g. available methods and attributes when typing a '.' after an object name.

The IDE should give easy access to tools like:

- Debugger. Hovering over a variable to inspect (and modify) its value is a convenient feature.
- Code profiler
- Report generators, e.g. for code or database structures.
- Version control system.

Common choices of IDEs are:

- Eclipse / PyDev
  FOSS, well supported. Top-ranking.
- BoaConstructor
  Includes GUI builder. For wxWidgets.
- Wing IDE
  Commercial (free for established FOSS projects).
- Eric
  Includes GUI builder (Qt Designer).
- Komodo
  Commercial.
- Spyder
  Aimed at science users.
- Emacs/vim
  Not really modern as far as IDE goes, but still used by many.

5 GUI builder

A GUI builder allows arrangement of widgets graphically. It generates code or a description (typically in XML format) which can then be used by the GUI toolkit. These are specific to the GUI toolkit. Popular choices (by toolkit) are:

- wxPython
  - BoaConstructor

  - XRCed
  - WxDesigner (commercial)
- PyGTK
  - wxGlade / Autoglade
  - Gazpacho
- PyQt / PySIDE
  - Qt Designer
- Tkinter
  - SpecTCL with a suitable conversion tool

6 Debugger

The debugger can be integrated into the IDE, or stand-alone. IDEs with integrated debuggers are Eclipse (using PyDev), Wing IDE, or Komodo. Stand-alone debuggers are the winpdb GUI to common command-line debuggers like pydb, or ddd (Unix/Linux only).

7 Database

All commonly used databases can also be used from Python programs. Bindings are used to interface to these databases. Common database choices are SQLite, which keeps all data in a single file and needs very little if any setting up, or the more heavy-duty MySQL and PostgreSQL.

All can be accessed through a common uniform Python SQL database API, which allows to write programs to be more or less independent of which DB is used, as long as only those features are used which are offered by all databases.

A higher-level database-interface is desirable which does not require the programmer to write SQL directly, to reduce development time (write other Python code rather than debug SQL statements).

buzhug implements a database in pure Python which is accessed without SQL, but it is not as performant.

8 Testing

A large number of stand-alone solutions exist for different purposes and methodologies like unit testing or black box or white box testing. They are listed in the Python Testing Tools Taxonomy[6].

9 Application Packaging

Several methods exist for publishing Python applications. Ideally one would use the target-platform's preferred method, e.g. a package file to be installed by the system's package manager, or executable to run. Creating the installables is somewhat tedious, however it is necessary when distributing only pre-compiled binaries. A platform-independent method for deploying the application easily would be a user-friendly solution—but there does not seem to be one. See Richard Jones' excellent summary[7].

10 Summary

Writing good-looking GUI applications with Python quickly is possible. The most important decisions are which GUI toolkit and which IDE to use.

Installing the full tool chain is time-consuming because investigating the available options and deciding what is best-suited takes too long. The full version of this paper will attempt to improve this situation by listing combinations for each of the tool chain components and their respective merits.

References

[1] http://creativecommons.org/licenses/by-nc-sa/3.0/.

[2] http://volker.top.geek.nz/linux/presentation/RapidGUIwithPython.pdf, http://volker.top.geek.nz/linux/presentation/RapidGUIwithPython-notes.pdf.

[3] M. L. Hetland, *Beginning Programming: From Novice to Professional*, 2nd ed. Apress, Sep. 2008.

[4] G. Polo, "PyGTK, PyQT, Tkinter and wxPython comparison," *The Python Papers*, vol. 3, no. 1, pp. 26–37, Mar. 2008. [Online]. Available: http://pythonpapers.org/archive/TPP3_1.pdf

[5] M. Summerfield, *Rapid GUI Programming with Python and Qt*, 1st ed. Prentice Hall PTR, Oct. 2007.

[6] http://pycheesecake.org/wiki/PythonTestingToolsTaxonomy.

[7] http://www.mechanicalcat.net/richard/log/Python/Sane_Python_application_packaging.