3 open source Python GUI frameworks

Originalartikel

Backup

<html> <p>There comes a time in the journey of most any programmer when they are ready to branch out past the basic&#160;examples and start to build a graphical interface to their program.</p><p>In Python, the steps to get started with GUI programming are not terribly complex, but they do require the user to begin making some choices. By its nature as a general purpose programming language with interpreters available across every common operating system, Python has to be fairly agnostic as to the choices it presents for creating graphical user interfaces.</p> <p>Fortunately, there are many options available for programmers looking to create an easy way for users to interact with their programs. Bindings exist for several UI frameworks on a variety of platforms, including those native to Windows, Mac, and Linux, and many that will work across all three.</p> <p>Before we go any further, let me play devil's advocate for just a moment and ask: Does it really make sense for your application to have a traditional graphical user interface at all? For some programs, the answer is obvious. If your application is inherently graphical in nature, and is either optimized for or just makes sense to be run locally on a single machine, then yes, you probably should build a desktop GUI. Many times, this is made obvious by what you're designing.</p> <p>But for more general purpose programs, don't count out either the command line or a web interface just yet. The command line offer many advantages&#8212;speed, remote access, reusability, scriptability, and control&#8212;which may be more important for your application's users than a graphical interface, and there are many libraries like <a href=„http://click.pocoo.org/5/“ target=„_blank“>Click</a>, <a href=„http://builtoncement.com/“ target=„_blank“>Cement</a>, and <a href=„http://docs.openstack.org/developer/cliff/“ target=„_blank“>Cliff</a> that make it easier to design great command line programs.</p> <p>Similarly, a web interface, even for a program meant to be run locally, might be an option worth considering, particular if you think your users may wish to host your application remotely, and projects like <a href=„https://www.djangoproject.com/“ target=„_blank“>Django</a>, <a href=„http://flask.pocoo.org/“ target=„_blank“>Flask</a>, or <a href=„http://www.pylonsproject.org/“ target=„_blank“>Pyramid</a> all make this straightforward. You can even use a library like <a href=„https://github.com/r0x0r/pywebview“ target=„_blank“>pywebview</a> to put a thin wrapper around a web application in a more native GUI window.</p> <p>Or you can use a framework like <a href=„http://pyforms.readthedocs.io/en/latest/“ target=„_blank“>Pyforms</a> to build a consistent experience across the web, command line, and desktop, all with a single code base.</p> <p>Still sure you want to build a GUI? Great, here are three fantastic open source libraries to get you started.</p> <h2>PyQt</h2> <p><a href=„https://riverbankcomputing.com/software/pyqt/intro“ target=„_blank“>PyQt</a> implements the popular <a href=„http://www.qt.io/“ target=„_blank“>Qt</a> library, and so if you are familiar with Qt development in another language, perhaps from developing native applications for KDE or another Qt-using desktop environment, you may already be familiar with Qt. This opens up the possibility of developing applications in Python which have a familiar look and feel across many platforms, while taking advantage of the tools and knowledge of the large Qt community.</p> <p>PyQt is dual licensed under both a commercial and <a href=„http://www.gnu.org/licenses/gpl-3.0.en.html“ target=„_blank“>GPL</a> license, not unlike Qt project itself, and the primary company supporting PyQt offers a <a href=„https://www.riverbankcomputing.com/commercial/license-faq“ target=„_blank“>license FAQ</a> to help understand what this means for your application.</p> <h2>Tkinter</h2> <p>If there were a single package which might be called the „standard“ GUI toolkit for Python, it would be

<a href=„http://tkinter.unpythonic.net/wiki/“ target=„_blank“>Tkinter</a>. Tkinter is a wrapper around <a href=„http://www.tcl.tk/“ target=„_blank“>Tcl/Tk</a>, a popular graphical interface and language pairing first popularized in the early 90s. The advantage of choosing Tkinter is the vast number of resources, including books and code samples, as well as a large community of users who may be able to help you out if you have questions. <a href=„https://docs.python.org/2/library/tkinter.html“ target=„_blank“>Simple examples</a> are easy to get started with and fairly human-readable.</p> <p>Tkinter is available under the <a href=„http://tkinter.unpythonic.net/wiki/Tkinter“ target=„_blank“>Python license</a>, on top of the BSD license of Tcl/Tk.</p> <h2>WxPython</h2> <p><a href=„http://www.wxpython.org/“ target=„_blank“>WxPython</a> brings the <a href=„http://wxwidgets.org/“ target=„_blank“>wxWidgets</a> cross-platform GUI library from its native C++ to Python. WxPython is a slightly more modern approach to, which looks a little more native than Tkinter across different operating systems as it does not attempt to create its own set of widgets (although these can be themed to look much like native components). It's fairly easy to get started with as well, and has a growing developer community. You may need to bundle wxPython with your applications, as it is not automatically installed with Python.</p> <p>WxPython uses the <a href=„http://www.wxwidgets.org/about/licence/“ target=„_blank“>wxWindows Library License</a> of its parent project, which is <a href=„https://opensource.org/licenses/wxwindows.php“ target=„_blank“>OSI approved</a>.</p> <hr/><p>These are not the only choices you have available to you, not even by a long shot. For more options, check out the „<a href=„https://wiki.python.org/moin/GuiProgramming“ target=„_blank“>GUI programming in Python</a>“ page on the official Python Software Foundation wiki, where you will find dozens of other options. While beginners will want to look out for and probably avoid projects which are only partial implementations, or those longer actively maintained, you'll find plenty of good fits for a variety of situations.</p> <p>While many of the options out there are for more traditional applications that take on the windowed look and feel of their parent desktop environment, there are also times when you may wish to do something completely different, for example, within a video game. There are great libraries for these situations too, like <a href=„http://www.pygame.org/wiki/gui“ target=„_blank“>pygame</a> and <a href=„https://bitbucket.org/pyglet/pyglet/wiki/Home“ target=„_blank“>pyglet</a>.</p> <p>Do you have a favorite that we didn't mention here? Let us know in the comments below!</p> </html>