Review of Business and Technology Research
Vol. 5, No. 1, 2011, pp. 163-168
ISSN 1941-9414, All rights reserved

COMPARISON OF PYTHON FOR ANDROID AND C-PYTHON

Albert Chan, Fayetteville State University, U.S.A. (achan@uncfsu.edu)

ABSTRACT

Student engagement has been a hot topic in recent years. Many computer science programs around the world have changed their introductory and/or primary programming language to Python. Studies show that students understand the basic programming concepts easier with Python and are hence better engaged. This change to Python is usually combined with other approaches such as media programming, robotics, and recently, mobile computing. At first, mobile computing was not easy with Python, as most platforms do not provide direct access to Python (Objective-C for iPhone/iPad, Java for Blackberry, Java/XML for Android, and C++/C# for Windows Phone). This changed when Google released the Scripting Layer for Android (SL4A), which allows the user to access several popular scripting languages including Python (also called Python for Android). In this article, we compare Python for Android with the standard desktop version of Python. This allows the instructors to determine whether mobile computing is a suitable approach to use with the Python language to attract student interest.

I. Introduction

Student engagement is one of the many important issues faced by educational institutions recently, affecting all subjects including Computer Science. To get students more engaged, many (Computer Science) programs have changed their curriculum to use Python as the primary/introductory programming language. Python is an easy to learn, multi-paradigm programming language (Tucker 2007). It is mainly an interpreted language (with automatic compilation for more advanced usage), so students need not to go through the whole edit/compile/execute cycle to see the results of their programs. Also, the availability of the Python shell allows students to easily experiment with ideas without writing full-blown programs. All these features help students to become more focused on the concepts to be learned, and hence become more productive in relatively shorter time (Goadrich 2011).

To get the students more engaged, many institutions augment the programming course(s) with specialized themes. Some of the most common themes are robotics, media programming, and mobile computing. In principle, any of these themes can be used with any programming language. However, in practice, the easiness of applying a theme with a language depends on many factors, and the choices of themes available to instructors may be restricted.

Applying the robotic theme with Python is relatively easy – as long as the appropriate hardware is available. The scribbler robots, coupled with the fluke boards and a Bluetooth connection, allow the students to control them with Python programs (Kumar 2011). Several colleges have used the scribbler robots to introduce basic computing concepts in their CS0/CS1 courses.

Media programming courses with Python are not difficult either. Guzdial et al. have implemented a media programming system based on Java (Guzdial 2006). They have also ported the system to Python using Jython (a version of Python implemented on top of Java) (Guzdial 2009), and are working on implementing the system on C-Python. Using the systems, students can easily write (Java or Python) programs to manipulate images, sound tracks, and/or video clips.

Mobile computing with Python, on the other hand, used to be quite difficult. Not only does each mobile platform (iPhone, Android, BlackBerry, or Windows Phone) have its own native program language; but each platform also has its own architecture that the programmers must follow. These architectures are usually based on the event

programming paradigm. That means programmers are required to write sufficient event handlers to handle the many events that can happen during the execution life-span of the application.

With the introduction of Scripting Layer for Android (SL4A), along with Python for Android, it is now possible to write mobile applications with Python. The focus of this article is to discuss the use of this facility in the academic environment to help engage students.

Section 2 gives an overview on the Android platform; Section 3 introduces SL4A; while Sections 4 and 5 review Python and Python for Android, respectively. We compare Python for Android and (Desktop) Python in Section 6; and Section 7 concludes our discussion.

II. The Android Platform

Google, along with other handset manufacturers, formed the Open Handset Alliance in 2007 (Darcey 2010). The main goal of the alliance is the development of the next generation wireless platform. Unlike other platforms available at the time, the aimed platform was an open platform that was freely available to everyone. In the following year, the Open Handset Alliance announced the Android platform and launched a beta program for developers. The first Android handset began shipping in late 2008. Today, the latest versions of Android are version 2.x for handsets and version 3.x for tablets (Google 2011a).

Before Android, almost all mobile platforms available were propriety and controlled by a single company (for example, iOS by Apple, Blackberry OS by Research In Motion, and Windows Phone by Microsoft). The availability of Android broke this monopoly. Not only do phone manufacturers now have an additional choice in phone OS, this new choice is also open source – meaning that anyone interested can look into the code to see how it works and maybe even contribute to the development of the OS.

Android is based on a specialized Java virtual machine (the Dalvik Virtual Machine, or DVM), which is implemented on top of a Linux platform. Therefore, almost all Android applications are written in Java. However, the main difference between Android Java applications and desktop Java applications (and mobile applications for other wireless platforms) is that Android also uses XML to describe the layouts of the applications, hence decoupling the views from the applications' functionalities. One obvious advantage is that it is now easier to modify the layouts of the applications without affecting the code that describes the functionalities.

III. Scripting Layer for Android

Mobile Computing used to be confined in the specific language chosen by platforms, for example, Objective C in iOS (for iPhone and/or iPad); Java for Blackberry OS; Java/XML for Android, and C++/C# for Windows Phone. Developers wishing to develop applications for a specific platform must do so using the chosen language(s). Unfortunately, this has several disadvantages. First of all, the learning curves for a new programming language may be steep, especially for amateur programmers. It may also be overkill for some simple, maybe one-time use, applications. Furthermore, for most wireless platforms, the native applications must follow a platform specific, pre-defined framework. Without understanding the framework, it is very difficult, if not impossible, to create applications for the wireless platform. Very often, the frameworks incorporate concepts from event-driven programming. Although event-driven programming may be necessary for any professional programmers who would like to develop modern, professional software, it may not be easily understood by amateur or beginning programmer.

The Scripting Layer for Android (SL4A) is an experimentation environment developed by Google. The purpose of SL4A is to allow programmers to write programs using their scripting languages of choice. Release 3 of SL4A requires the written programs (the scripts) to be run within the environment. Release 4 allows the developers to package the scripts in a way that looks like other native Android applications (Ferrill 2011).

SL4A is a generic scripting platform for Android. It supports several scripting languages, including BeanShell (an interpreted version of the Java language), JRuby, Linux shell, Lua, Perl, PHP, Python, and Rhino. Like the Android platform, SL4A, along with the interpreters it supports, is available as a free download from Google (Google 2011b).

IV. Python

Python is a high level, multi-paradigm, open-source, and general purpose programming language developed in early 1990. Like many other modern languages, Python code is first compiled into byte-code, and the byte-code is then interpreted in a virtual machine. It has built-in high level data structures such as strings, lists, and tuples. It has the usual control mechanism like `if`, `if-else`, `if-elif-else`, `while`, and `for`. It supports multiple level of organizational structure including functions, classes, modules, and packages, as well as easy extension through C/C++ code.

Python uses indentation to mark off blocks of code. This not only reduces the amount of punctuation in program code (for example, `{}` in Java/C/C++ programs, `begin/end` in Pascal programs), but also helps enforce good programming styles. Variables in Python do not require declaration; however, no one is allowed to use un-initialized variables. It supports unlimited precision integers and provides a seamless transition between normal fixed-size integers.

Unlike Java (in which everything must be encapsulated into classes) and C (in which everything must be encapsulated into functions), Python code can be entered directly in the Python shell. This saves a lot of confusing setup work students must perform before they can see their program running. One can even use Python as a super powerful calculator. The following shows the typical "Hello, world!" program written as a one-liner Python code:

```
print "Hello, world!"
```

Therefore, it is very suitable in teaching beginning programming classes. Many universities (including ours) have recently converted their Computer Science curriculum to use Python in CS0, CS1 and CS2.

To quote Lutz and Ascher, Python "is more powerful than Tcl; has a cleaner syntax and simpler design than Perl; is simpler and easier to use than Java and C++; is more powerful and more cross-platform than Visual Basic; and has the dynamic flavor of languages like SmallTalk and Lisp." (Lutz 2004)

Because of its open-source nature, Python is supported by a large community. There are extension packages for almost every task programmers can dream of. Very often, it is just a matter of installing an extension package to have the required functionalities. Examples of such extension packages include NumPy/SciPy, OpenCV, PyOpenGL, and SciPy.

V. Python for Android

Python is one of the many supported languages in SL4A. However, it is not installed with SL4A by default. To use Python for Android, it must be downloaded and installed separately, in addition to the required SL4A installation.

The easiest way to download and install Python for Android is from within SL4A. Once SL4A is installed, start SL4A. At this time, SL4A will start with an empty screen since there are no scripts installed yet. Invoke the menu and select view to display the view dialog. In the view dialog, select Interpreters, to show installed interpreters. The only interpreter installed by default is the Linux shell. Invoke the menu again and select Add to display the add dialog. In the add dialog, select Python. This will download the Python for Android installer to your handset. Once the download is complete, execute the downloaded file to complete the install process. The install process will pull three more files from the download site: one for the core distribution, one for extra modules, and one for example scripts. For the current version, the total size is about 6MB compressed and 15MB expanded (Google 2011c).

After installation, starting SL4A will bring you to the script screens, which should display a handful of example Python scripts (if you have installed other interpreters, there may be example scripts for those languages too). You can click on a script to edit/execute it, or you can use the menu to create a new script.

Figure 1 shows the SL4A screen populated with sample scripts. Figure 2 shows the execution of a textbook Python example program.

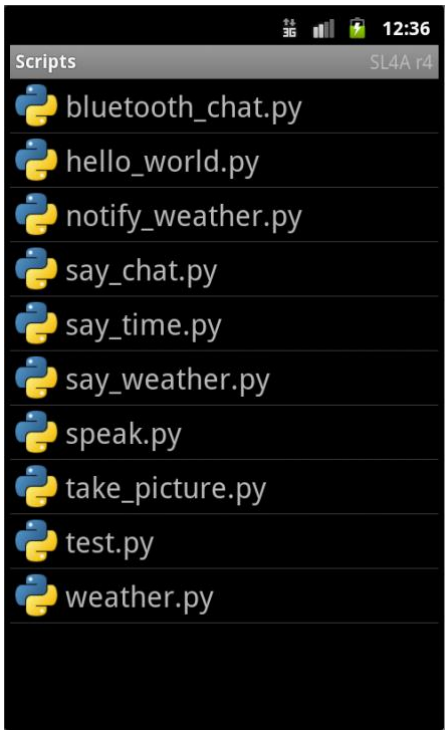

Figure 1: SL4A screen populated with sample Python scripts

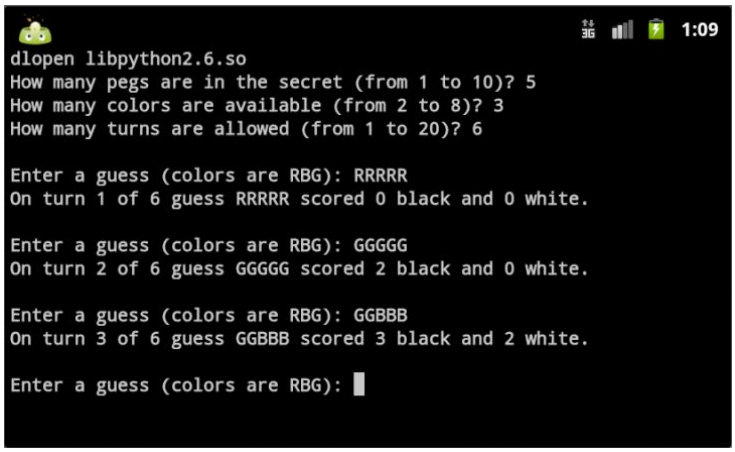

Figure 2: Execution of a textbook Python example program

VI. Comparison

As of this writing, the most current version of Python for Android is 2.6.2. We will use this version to compare with the corresponding Windows version. In this Section, we call the Windows version of Python "Python for Windows," to differentiate it with "Python for Android".

The default installation of Python for Windows comes with 371 modules, while Python for Android only comes with 297 modules. Among them, 283 modules are common to both installations. That means there are 88 modules unique to the Windows installation and 14 modules unique to Python for Android. The lists of modules are obtained by issuing the command `help ("modules")` in both installations.

For our purpose, the differences are asymmetric; that is, we are more interested in the modules included in Python for Windows but not in Python for Android. This is because these modules may mean something that we can do in Python for Windows, but difficult (or even impossible) in Python for Android. It is less important (for our purpose) that something can be done with Python for Android, but not in Python for Windows.

A further investigation in the 88 modules unique to Python for Windows shows that they can be broken down as follows:

- 56 modules are for IDLE, the integrated IDE that comes with Python for Windows.
- 15 modules are for `TkInter`, the wrapper to allow Python code to access facility provided by Tcl/Tk.
- 1 module for 2to3, the tools to convert Python code between 2.x to 3.x.
- 4 modules for using Microsoft Windows specific resources, such as `winsound` (Windows sound service), `msvcrt`, `msilib`, and `nt`.
- 7 private modules used by other modules.
- 2 modules for testing: `test` and `testcode`.

The following table lists the modules in the two implementations:

Modules that are unique in Python for Windows:

AutoComplete	MultiCall	Tkdnd	idle
AutoCompleteWindow	MultiStatusBar	Tkinter	idlever
AutoExpand	ObjectBrowser	ToolTip	keybindingDialog
Bindings	OutputWindow	TreeWidget	lib2to3
CallTipWindow	ParenMatch	UndoDelegator	macosxSupport
CallTips	PathBrowser	WidgetRedirector	msilib
Canvas	Percolator	WindowList	msvcrt
ClassBrowser	PyParse	ZoomHeight	nt
CodeContext	PyShell	_bsddb	rpc
ColorDelegator	RemoteDebugger	_hashlib	run
Debugger	RemoteObjectBrowser	_locale	tabbedpages
Delegator	ReplaceDialog	_msi	test
Dialog	ScriptBinding	_subprocess	testcode
EditorWindow	ScrolledList	_tkinter	textView
FileDialog	ScrolledText	_winreg	tkColorChooser
FileList	SearchDialog	aboutDialog	tkCommonDialog
FixTk	SearchDialogBase	configDialog	tkFileDialog
FormatParagraph	SearchEngine	configHandler	tkFont
GrepDialog	SimpleDialog	configHelpSourceEdit	tkMessageBox
HyperParser	StackViewer	configSectionNameDialog	tkSimpleDialog
IOBinding	Tix	curses	turtle
IdleHistory	Tkconstants	dynOptionMenuWidget	winsound

Modules that are unique in Python for Android:

BeautifulSoup	atom	fcntl	posix	resource	syslog	twitter
android	crypt	gdata	pwd	simplejson	termios	xmpp

VII. Conclusion and Future Work

It is obvious that Python for Android is a pretty comprehensive implementation of the Python language on the Android platform under the SL4A environment; and is very suitable for use in introductory programming courses. The only major missed functionality is the use of windowing toolkit (`TkInter`), which is usually omitted in such introductory courses. Also, IDLE is not available in Python for Android. In fact, a more detailed study shows that IDLE uses many of the facilities provided by `TkInter`, which explains why `TkInter` is supplied with the core distribution of Python for Windows.

Android's SL4A also supports many other scripting languages such as Lua, Perl, and Ruby. We would like to extend the comparison into these languages. This will allow us to establish a much more complete picture on how suitable mobile computing (more precisely, SL4A under the Android platform) is in Computer Science curriculum. We hope this will help to motivate the development of similar facilities on other platforms (iOS, Blackberry OS, Windows Phone, to name a few). Finally, we also hope that this study will motivate the porting of `TkInter` (or even IDLE) to SL4A.

As can be seen in Figure 2, most textbook examples can be run using Python for Android. This is generally true unless the code uses some specialized modules, in which case the specialized modules must be installed separately. Two examples are the `graphics` (Zeller 2004) and `cs1graphics` (Goldwasser 2008) modules used in textbooks. While installing these modules in Python for Windows is easy, it is very challenging to install them in Python for Android. We would like to see this study motivate research in this direction.

VIII. Acknowledgement

The publication of this article is partially supported by the Fayetteville State University's Integrated STEM Academic Success (ISAS) program, which is funded by NSF grant #HRD-1036257.

IX. Bibliography

(Darcey 2010) Darcey and Conder, "*Sams Teach Yourself Android Application Development in 24 Hours*," SAMS Publishing, 2010.

(Ferrill 2011) Ferrill, "*Pro Android Python with SL4A*," APress, 2011.

(Goadrich 2011) Goadrich and Rogers, "*Smart Smartphone Development: iOS versus Android*," in Proceedings of the 42nd ACM Technical Symposium on Computer Science Education, (SIGCSE 2011), 2011.

(Goldwasser 2008) Goldwasser and Letscher, "*Object-Oriented Programming in Python*," Prentice Hall, 2008.

(Google 2011a) Google, "*Android Developers Web Site*," available at URL http://developer.android.com/index.html.

(Google 2011b) Google, "*Android Scripting Web Site*," available at URL http://code.google.com/p/android-scripting.

(Google 2011c) Google, "*Python for Andorid Web Site*," available at URL http://code.google.com/p/python-for-android.

(Guzdial 2006) Guzdial and Ericson, "*Introduction to Computing and Programming with Java: A Multimedia Approach*," Prentice Hall, 2006.

(Guzdial 2009) Guzdial and Ericson, "*Introduction to Computing and Programming with Python: A Multimedia Approach*," 2nd Edition, Prentice Hall, 2009.

(Kumar 2011) Kumar, "*Learning Computing With Robots: Python + Scribbler or Scribbler2*," lulu.com, 2011.

(Lutz 2004) Lutz and Ascher, "*Learning Python*," 2nd Edition, O'Reilly, 2004.

(Tucker 2007) Tucker and Noonan, "*Programming Languages: Principles and Paradigms*," 2nd edition, McGraw Hill Higher Education, 2007.

(Zeller 2004) Zelle, "*Python Programming: An Introduction to Computer Science*," Franklin, Beedle and Associates, 2004.