Es muss nicht immer Java sein: Python für mobile Anwendungen

Nokia hat mit Python für S60 eine für mobile Geräte neue Programmiersprache veröffentlicht, welche eine interessante Alternative zur herkömmlichen Softwareentwicklung in Java oder C/C++ darstellt. Wir stellen Python für S60 kurz vor und testen den Einsatz von Python sowohl für rechenintensive als auch kommunikationsorientierten Beispiele. Dabei kommen wir zum Schluss, dass sich Python vor allem für Anwendungen ohne grosse Ansprüche an die Rechenleistung und ganz allgemein für ein Rapid Prototyping eignet.

Oliver Ruf | oliver.ruf@fhnw.ch

Die Vor- und Nachteile von plattformspezifischen Maschinenprogrammen, welche zum Beispiel durch Kompilation von C/C++-Programmen entstehen, sind hinlänglich bekannt: sehr hohe Performanz bei mangelnder Plattformunabhängigkeit. Das Problem der fehlenden Unterstützung von verschiedensten Prozessortypen und Plattformen ist bei der Software-Entwicklung in Java weitestgehend durch den Einsatz von plattformunabhängigem Bytecode gelöst. Dieser Zwischencode wird von der Java Virtual Machine (VM) interpretiert oder just in time (JIT) kompiliert. Somit reduziert sich die Plattformabhängigkeit auf die VM. Der grosse Nachteil dabei ist die verminderte Performanz in rechenintensiven Anwendungen.

Python ist im Gegensatz zu C/C++ und Java eine reine Interpretersprache. Andere bekannte Interpretersprachen sind beispielsweise Perl, Ruby, PHP und auch BASIC. Die Programme werden also nicht kompiliert, sondern direkt zur Laufzeit interpretiert. Sie benötigen daher, ähnlich zu Java, eine plattformspezifische Laufzeitumgebung, um die Programme, auch Skripte genannt, auszuführen. Alleine die Tatsache, dass Python-Programme interpretiert werden, weist darauf hin, dass mit einer verminderten Performanz zu rechnen ist. Wie gross diese Einbusse ist und welche Vorzüge durch die Verwendung eines Interpreters anstatt eines Compilers resultieren, wollen wir hier genauer untersuchen.

Erfolg durch Einfachheit

Bevor sich ein Programmierer an die Lösung eines Problems macht, wäre es wünschenswert, dass er die verschiedenen Vor- und Nachteile der unterschiedlichen Programmierparadigmen und Programmiersprachen im gegebenen Problemfeld untersucht. Denn nicht immer ist der erste Griff in die Sprachenkiste der Richtige; doch fast immer gilt, dass ein falscher Griff später sehr teuer bezahlt werden muss. Falls sich also verschiedene Programmiersprachen anbieten, weil sie technisch gut unterstützt werden und weil sich die Programmierer damit wohl fühlen, so sollten sich die Entwickler unter anderem die Frage stellen: Wie schnell und wie effizient kann ich das gegebene Problem mit der Programmiersprache P lösen? Die redliche Beantwortung dieser Frage müsste zu einem Mix an Programmiersprachen führen, abhängig davon, welches Problem es zu lösen gilt. Denn gewisse Programmiersprachen, wie C/C++, bieten ein grosses Mass an Effizienz, sind aber im Entwicklungsprozess umständlich zu handhaben. Andere Programmiersprachen, wie beispielsweise Python, punkten hier, denn der Entwicklungsprozesses hängt dabei von verschiedenen Faktoren ab, unter anderem von der Einfachheit der Sprache, den unterstützten Konzepten, den beiliegenden Bibliotheken und dem benötigten Aufwand um ein lauffähiges Programm zur Ausführung zu bringen. Ein erstes Gefühl der Einfachheit Pythons erhalten wir dadurch, dass wir die Seitenzahl der Sprachdefinitionen vergleichen. Während für Python knapp hundert Seiten ausreichen [PySpc], benötigt es für Java bereits über sechshundert Seiten um die Sprache zu definieren [JaSpc]. Kurz zusammengefasst heisst das, dass sich keine der heute verbreiteten Programmiersprachen in einem absoluten Sinn als „die Beste“ behaupten kann.

Um einen ersten Eindruck von Python vermitteln zu können, haben wir in Listing 1 den Klassiker Fibonacci programmiert. Anders als in vielen anderen strukturierten Programmiersprachen werden in Python die strukturellen Blöcke lediglich durch das Einrücken von Zeilen definiert und es sind keine expliziten Klammern oder Schlüsselwörter als Begrenzung nötig. Die Art der Einrückung ist somit vorgegeben und kann nicht dem persönlichen Geschmack des Programmierers überlassen werden. Dies wird vor allem bei der for-Schleife im Beispiel deutlich sichtbar: alle

```
import time

def fibonacci(n):
    if   n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fibonacci(n - 1) + fibonacci(n - 2)
measurings = 100
amount     = 30
totaltime  = 0.0
for i in range( measurings ):
    begin = time.clock()
    fibonacci(amount)
    end = time.clock()
    totaltime += end - begin
average = 1000 * totaltime / measurings # in ms
```

Listing 1: Fibonacci mit Zeitmessung

nachfolgenden Anweisungen, welche einfach eingerückt sind, gehören zum Schleifenkörper. Für Java- oder Pascal-Programmierer ist der Quellcode etwas gewöhnungsbedürftig aber die Lesbarkeit ist insgesamt verbessert. Es besteht dadurch aber auch die Gefahr eines fälschlichen Einrückens und somit einer fehlerhaften Interpretation.

Variablen werden generell dynamisch alloziert und der Speicher automatisch verwaltet, so dass keine explizite Instanzierung notwendig ist. Es wird auch nicht vorgeschrieben, auf welche Art die Programme geschrieben werden müssen, und Python lässt dem Entwickler dadurch die Wahl funktional, objektorientiert oder aspektorientiert zu programmieren. Es ist sogar möglich, diese Paradigmen zu mischen.

Diese einfachen Grundsätze sind nicht unwesentlich, weil sich dadurch in kürzester Zeit ein lauffähiges Programm erstellen lässt.

Wenn man also eine spontane Idee hat und nur kurz mal ausprobieren möchte, ob sich diese auch als Programm umsetzten lässt, so wünscht man sich einen möglichst einfachen Entwicklungsprozess ohne schwerfällige Entwicklungsumgebung und ohne langwierigen Kompilations- und Bindeprozess. Ist ein solcher nicht vorhanden, verwirft man die Idee zu leicht oder verschiebt sie auf später.

Die Einfachheit des Python-Entwicklungsprozesses lässt sich gut im Vergleich zu Java am Standardprogramm „Hello, World!“ zeigen. Im Java-Programm (Listing 2) benötigen wir eine in einer Klasse eingebetteten main-Prozedur mit dem entsprechenden Print-Aufruf.

Dieses Java-Programm muss in der Datei «Hello.java» (entsprechend dem Klassennamen) gespeichert und danach kompiliert werden. Die Kompilation geschieht mit dem Befehl javac Hello.java, welcher die Bytecode-Datei Hello.class erstellt. Um das Ganze dann auch auszuführen, muss man die VM mit dem Aufruf java Hello dazu anweisen. Das Resultat ist, dass „Hello, World!“ auf der Konsole ausgegeben wird. Bei Java-Applikationen für Mobiltelefone muss die Funktion zusätzlich noch

```
public class Hello
{
  public static void main(
      String[] args )
  {
    System.out.println(
      „Hello, World!“ );
  }
}
```

Listing 2: „Hello World“ mit Java

Abb. 1: „Hello, World!“ in der Python-Konsole auf einem Nokia E61

Abb. 2: Das Nokia E61

in einem MIDlet eingebaut werden, was weiteren Aufwand bedeutet.

Bei Python erfüllt lediglich ein einzelner Befehl nach Aufstarten des Interpreters seinen Dienst (siehe Abb. 1). Dieser reduzierte initiale Aufwand spielt speziell bei kleinen Applikationen ein wichtiges Kriterium, fällt aber bei grösseren Projekten nicht mehr stark ins Gewicht, da das Verhältnis zum restlichen Quellcode mit dessen Zunahme abnimmt. Daher eignet sich Python besonders gut für Rapid Prototyping.

Wie eingangs schon angesprochen, ist ein weiteres wichtiges Merkmal von erfolgreichen Programmiersprachen die Güte der zugehörigen Bibliotheken. Ähnlich wie die Standardbibliothek von Java [JaLib] besitzt auch Python eine umfangreiche Standardbibliothek [PyLib] mit vielen nützlichen Funktionen. Im Python-Jargon heissen die Bibliothekseinheiten Module und reichen von einfachen Datenstrukturen über Internet- und Netzwerk-Funktionen bis hin zu XML-Parsern und Kompressionsalgorithmen.

Python für Mobiltelefone

Im Januar 2006 veröffentlichte Nokia für die S60 Serie ihrer Mobiltelefone eine Implementierung der Programmiersprache Python in der Version 2.2 [SFS60]. Dass sich Nokia gerade für Python als Skripting-Sprache entschieden hatte, hängt damit zusammen, dass Python generell eine Schnittstelle zu C/C++-Programmen besitzt. Somit besteht die Möglichkeit, vorhandenen C++-Code in eine von Python aus aufrufbare DLL zu kompilieren und dort zu verwenden. Die Python-Implementierung von Nokia enthält einen Interpreter, einen Grossteil der Standardbibliothek und ein API für die Geräte. Nokia stellt diese Implementierung als open source unter einer Apache License V2.0 frei zur Verfügung [ApLic]. Eine aktive Community wie auch Nokia selbst arbeiten kontinuierlich an der Weiterentwicklung der Implementierung.

Als zentrale Stelle dient der Community die offizielle Webseite der open-source-Projekte von Nokia [NoWik], welche als Wiki realisiert ist und somit allen Benutzern die Möglichkeit bietet, Seiten ändern und erweitern zu können. Als Support- und Diskussionsplattform stehen ein offizielles Forum [NoFor] und ein IRC-Chat zur Verfügung.

Eines der Kernstücke von Python für S60 ist die Nokia-API. Mit viel Aufwand hat Nokia ihre C/C++-APIs nach Python exportiert. Diese C/C++-APIs bieten Zugriff auf die meisten Funktionen des Telefons. Entsprechend gibt es auch in Python für S60 diverse Module für alle möglichen Funktionen des entsprechenden mobilen Geräts: Module für GUI- und Grafik-Funktionalität, der Zugriff auf die interne Kamera, die Möglichkeit SMS-Nachrichten zu versenden, oder auch Bluetooth-Verbindungen aufzubauen sind einige Beispiele dafür.

Im Gegensatz dazu sind bei Python für S60 noch nicht alle Module der Standardbibliothek umgesetzt worden. Die Betonung liegt hier speziell auf dem „noch nicht“, denn die Entwickler von Nokia arbeiten an der Erweiterung der bisherigen Umsetzung und veröffentlichen mittlerweile fast monatlich neue Versionen.

Als konkretes Beispiel sind die XML-Funktionen der Standard-Bibliothek noch nicht auf dem Mobiltelefon verfügbar. Ein übliches import xml endet abrupt mit einem ImportError. Gerade hier zeigt sich die Community von der flexibelsten Seite. Einer der interessierten Benutzer hat zu diesem Zweck ein anderes freies XML-Modul, cElementTree, für die mobile Plattform übersetzt und bietet dieses zum herunterladen auf seiner Webseite an [PyXml]. Wie dieses Beispiel zeigt, füllen die Benutzer die noch vorhandenen Lücken durch eigene Module und Programme. Dennoch bedeutet dies für den Programmierer immer zusätzlichen Aufwand; vor der Verwendung eines Moduls lohnt es sich zu überprüfen, ob es auf der mobilen Platt-

Abb. 3: Gefundene Suchresultate zur Titelsuche „Python in a Nutshell“

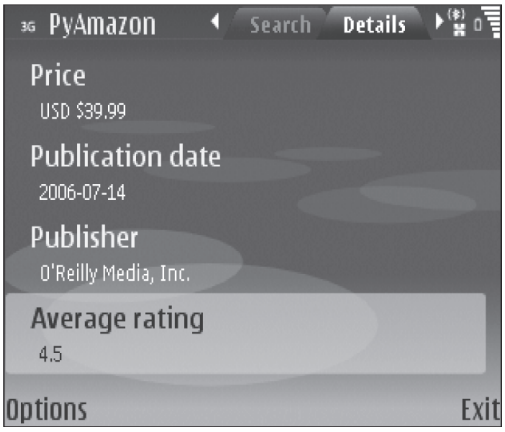

Abb. 4: Daten zu einem Buch

form verfügbar ist. Falls nicht, ist eine Suche im Internet oder eine eigene Alternative notwendig.

PyAmazon

Viele Leute bestellen heute ihre Bücher direkt von zuhause über das Internet. Dies tun sie vor allem dann, wenn sie genau wissen, welches Buch sie kaufen wollen. Das gezielte und ungezielte Schmökern in Büchern ist aber nur in einer Bibliothek oder Buchhandlung richtig gut möglich. Wenn man nun vor dem endlosen Regal steht und vor sich einen Stapel Bücher zum gleichen Thema hat, so fällt es einem oft nicht einfach, sich für eins der Bücher zu entscheiden. Dieser Entscheid könnte durch vorhandene, abrufbare Rezensionen beeinflusst werden. Bei Amazon zum Beispiel können die Kunden direkt zu jedem Artikel eine Bewertung abgeben. Gleichzeitig können andere Kunden diese dann beurteilen, ob sie sinnvoll und zutreffend ist, wodurch man eine ehrliche und gute Rezension auch erkennen kann.

Um solche Rezensionen und andere Buchinformationen direkt im Buchladen abrufen zu können, haben wir unter Einsatz von Python für S60 die kleine, mobile Applikation PyAmazon entwickelt [PyAma]. PyAmazon bietet die Möglichkeit, Buchtitel bei Amazon abzufragen (siehe Abb. 3) und den Preis, Bewertungen und andere Daten in einer benutzerfreundlichen Darstellung zu präsentieren (siehe Abb. 4). Durch die Eigenentwicklung unter Verwendung von Python haben wir die Möglichkeiten und Grenzen von Python für S60 kennengelernt und das ganze open source Projekt von Nokia genauer unter die Lupe nehmen können.

Die Bücherinformationen fragen wir über den von Amazon angebotenen E-Commerce Dienst (ECS) ab. Dabei kann die ECS-Schnittstelle auf zwei verschiedene Arten angesprochen werden: entweder mit dem Simple Object Access Protocol (SOAP) oder über den Representational State Transfer (REST). Während man bei SOAP entsprechende Parameter in XML verpackt und an die Schnittstelle schickt, kodiert man bei REST-Abfragen die benötigten Parameter direkt in der URL.

In PyAmazon haben wir uns für die REST-Methode entschieden, da die ganze Kodierung der Parameter in XML entfällt und dadurch auch weniger Daten übermittelt werden müssen. Dieser Umstand ist speziell bei mobilen Applikationen nach wie vor wichtig, da die Kosten für den Datentransfer recht hoch sind.

Den Python-Code des Kerns der REST-Abfrage ist in Listing 3 dargestellt. Mit dem Standard-Modul urllib kann über die Funktion urlopen(...) ein HTTP-Request an die URL geschickt werden. Dabei werden die Parameter speziell kodiert und mittels der Variablen post_data mitgeschickt. Als Rückgabewert erhält man dann einen Datenstrom f. Mit dem anschliessenden Befehl f.read() wird die Antwort empfangen und der Variablen response zugewiesen. f.close() schliesst danach die offene Verbindung wieder. Bei den Antworten der ECS-Schnittstelle handelt es sich immer um in XML gespeicherte Daten. Der Umfang der Antworten des Amazon-Webservices ist direkt von den

```
# url:       URL als String
# post_data: codierte Ab-
frageparameter

f = urllib.urlopen(url, post_data)
response = f.read()
f.close()
```

Listing 3: URL-Abfrage mit dem Standard-Modul urllib

```
import math
import time

pi          = 3.14159265359
measurings = 100
degrees     = 360
degsteps    = 10
totaltime   = 0.0
for m in range( measurings ):
    begin = time.clock()
    for i in range( degrees*degsteps ):
        angle = (i * pi) / (180.0 * degsteps )
        math.sin( angle )
        math.cos( angle )
        math.tan( angle )
    end = time.clock()
    totaltime += end - begin
totaltime *= 1000 # <- convert seconds to ms
average = int( totaltime / measurings )
```

Listing 4: Berechnung von Winkelfunktionen

verwendeten Parametern abhängig. Die vielfältigen Möglichkeiten und Kombinationen werden gut in der Amazon ECS-Dokumentation [AmECS] beschrieben.

Für eine kommunikationsorientierte Applikation wie PyAmazon ist Python für S60 eine gute Wahl. Da die Applikation hauptsächlich dem Beschaffen und Darstellen von Information dient und keine rechenintensiven Verarbeitungen benötigt, ist die Performanz völlig ausreichend. Das Abfragen und Empfangen der Informationen über das Internet benötigt dabei die meiste Zeit. Zudem sind alle wichtigen Funktionen in den verschiedenen Modulen vorhanden. Auch die Benutzerschnittstelle kann mit Hilfe der Nokia-API und den vorhandenen Komponenten effizient erstellt werden.

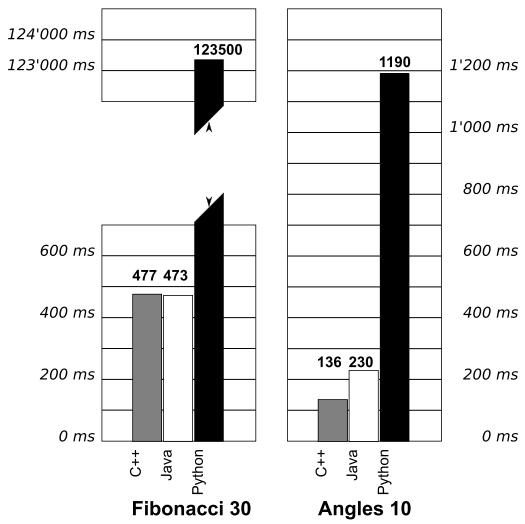

Abb. 5: C++ versus Java versus Python

Performanz

Die Flexibilität einer interpretierten Sprache muss üblicherweise mit einem Verlust an Performanz bezahlt werden. Um die Grösse dieses Verlustes abschätzen zu können, haben wir zwei unterschiedliche Benchmark-Programme je einmal in Python, Java und C/C++ implementiert.

Beim ersten Programm handelt sich um die mehrfache Berechnung einer Fibonacci-Zahl mittels einer zweifach rekursiven Funktion (siehe Listing 1). Mit der Hilfsfunktion fibonacci(n) wird die n-te Fibonacci-Zahl berechnet. Im Hauptprogramm wird diese Funktion hundertmal wiederholt aufgerufen und für jeden Aufruf die Zeit gemessen. Diese Zeiten werden aufsummiert, so dass schliesslich ein gemittelter Wert ausgegeben werden kann. Bei diesem Programm handelt es sich weniger um ein rechenintensives Programm, aber durch die Rekursion wird die Effizienz der Funktionsaufrufe auf die Probe gestellt.

Beim zweiten Programm werden die Winkelfunktionen Sinus, Kosinus und Tangens für verschiedene Winkel berechnet und dabei die Rechenkapazität ausgelotet (siehe Listing 4). Wiederum wird die Zeit gemessen und über hundert Iterationen gemittelt.

Beide Programme bringen wir in Python, Java und C++ auf einem Mobiltelefon Nokia E61 zur Ausführung. Das E61 ist ein neueres S60-Gerät der dritten Generation und verfügt über einen ARM9-kompatiblen Prozessor mit etwa 200 MHz und 25 MB Arbeitsspeicher für Programme (Abb. 2). Zudem verfügt noch kaum ein Gerätemodell über eine Hardwareunterstützung für Fliesskommaarithmetik und deswegen muss diese entweder durch die vorhandene Fixkommaarithmetik ersetzt oder aufwändig emuliert werden.

Die Ergebnisse dieses Versuchs sind in Abb. 5 klar ersichtlich. Der Performanzverlust von Python für S60 im Vergleich zu Java und C++ ist sehr deutlich. Während die Rechenperformanz beim

Programm mit den Winkelfunktionen im Vergleich mit Java um einen Faktor fünf, und im Vergleich mit C++ um fast einen Faktor neun gelitten hat, so haben die vielen rekursiven Funktionsaufrufe bei Fibonacci gleich ein 260faches an Zeit benötigt.

Wenn man aber den Arbeitsaufwand betrachtet, sieht die Sache etwas anders aus. Während man die beiden Python-Programme direkt wie in Listing 1 resp. Listing 4 auf das Mobiltelefon kopieren und ausführen kann, müssen die Java-Versionen zuerst in ein MIDlet verpackt und dann als Paket auf dem Telefon installiert werden.

Fazit und Ausblick

Mit dem Python für S60 Projekt hat Nokia eine ganz neue Art von Softwareentwicklung für Mobiltelefone möglich gemacht. Mit ihrer gut dokumentierten API ermöglichen sie den einfachen Zugriff auf die Funktionen des Telefons. Die Umsetzung der Python-Standardbibliothek ist im Allgemeinen sehr gut, lediglich einige wenige Module fehlen noch. Durch die laufende Weiterentwicklung des Projekts ist längerfristig damit zu rechnen, dass auch die fehlenden Module von Nokia noch umgesetzt werden. Zwischenzeitlich lassen sich durch die Hilfe der aktiven Community die meisten dieser Schwierigkeiten umgehen.

Die klare Struktur und die dynamischen Eigenschaften dieser Programmiersprache führen schnell und einfach zu Resultaten und man sieht während der Entwicklung schön, wie sich die einzelnen Teile zu einem Ganzen zusammenfügen. Es ist eine elegante Art zu programmieren.

Wenn man die Performanz von kompilierten Programmen mit der von interpretierten Skripten vergleicht, sind die Skript-Programme deutlich langsamer. Die geringere Performanz ist bei vielen, vor allem kommunikationsorientierten Applikationen für den Benutzer nicht spürbar und ausreichend. Bei rechenintensiven Applikationen muss sich der Entwickler überlegen, wie gut sich der rechenaufwändige Teil separieren und auslagern lässt, so dass er als externes Modul in C/C++ realisiert werden kann. Während auf handelsüblichen Computern die Rechenkapazität und Speichermengen auch bei Python-Programmen für fast jeden Zweck ausreichend sind, fallen die Ressourcen auf Mobiltelefonen recht knapp aus.

Für grössere Projekte lohnt sich der generelle Einsatz von C/C++ oder Java, insbesondere wenn die Applikation performanzkritisch oder rechenintensiv ist.

Aus unserer Sicht eignet sich Python für S60 allerdings sehr gut, um schnell einen ersten Prototypen einer Applikation zu erstellen. Und wer sich einmal intensiver mit Python auseinandergesetzt hat, wird sich schnell damit anfreunden und die Vorzüge dieser dynamischen Programmiersprache zu schätzen lernen.

Referenzen

[AmECS] Dokumentation Amazon ECS Dienst, http://developer.amazonwebservices.com/connect/kbcategory.jspa?categoryID=19

[ApLic] Apache Lizenzen, http://www.apache.org/licenses

[JaLib] Dokumentation der Java-Standardbibliothek, http://java.sun.com/j2se/1.4.2/docs/api/

[JaSpc] Java Language Specification, http://java.sun.com/docs/books/jls/

[PyAma] PyAmazon, http://pys60.ifi.ch/PyAmazon/

[PyLib] Dokumentation der Python-Standardbibliothek, http://docs.python.org/lib/

[PySpc] Python Language Specification, http://docs.python.org/ref/ref.html

[PyWeb] Offizelle Python Webseite, http://www.python.org

[PyXml] cElementTree für PyS60, http://ssalmine.googlepages.com/somepys60extensions

[NoFor] Offizielles Nokia Forum, http://discussion.forum.nokia.com/forum/forumdisplay.php?forumid=102

[NoWik] Offizielles Nokia Wiki, http://wiki.opensource.nokia.com/projects/Python_for_S60

[SFS60] PyS60 auf SourceForge, http://sourceforge.net/projects/pys60/