TECHNISCHE UNIVERSITÄT DRESDEN

DIL

Fakultät Informatik – Institut für Software- und Multimediatechnik, AG Didaktik der Informatik/Lehrerbildung

Bachelorarbeit im Fach Informatik

PYTHON IM SCHULINFORMATIKUNTERRICHT

Joachim Birk
Geb.-Datum: 19. März 1991
Mat.-Nr.: 3670763
Immatrikulationsjahr: WS 2010/11
Fachsemester: 6
Studiengang: Lehramt Bachelor Allgemeinbildende Schulen
Fachrichtung: Englisch/Informatik
Kontakt: Joachim.Birk@mailbox.tu-dresden.de

Betreut durch:
Prof. Dr. Steffen Friedrich
und:
Dr. Holger Rohland

Eingereicht am 06. August 2013

Sommersemester 2013

INHALTSVERZEICHNIS

1 EINLEITUNG

„Informations- und Kommunikationstechniken gelten als Schlüsseltechnologien unserer Epoche, die Informatik bildet hierzu die Grundlage. Mit der Beherrschung von Methoden und Werkzeugen der Informatik entsteht neben Schreiben, Lesen und Rechnen eine vierte Kulturtechnik." Mit diesen Worten beschreiben die *Bildungsstandards Informatik* die Rolle dieses Fachgebiets und heben vor allem ihre Bedeutung in der Schule hervor [18, S.2].

Als eine der Kernfähigkeiten der Informatik ist das logische und strukturierte Denken und damit auch die Fähigkeit zur Algorithmierung alltäglicher Probleme zu nennen. In vereinfachter Form lernen die Schüler den Algorithmusbegriff im sächsischen Informatikunterricht bereits in der achten Klasse kennen (vgl. [26, S.8]). In den Klassen neun und zehn, in denen Informatik in den Profillehrplan integriert ist, fordert der sächsische Lehrplan, dass die Schüler „aufbauend auf dem Algorithmusbegriff Grundlagen der Programmierung beherrschen" [26, S.11].

In den Vorgaben der Klassen elf und zwölf intensiviert sich die Anforderung noch einmal, wobei hier zwischen dem normalen Grundkurs und dem Kurs für die Schüler des sprachlichen Profils unterschieden wird. Während man in ersterem die „Kenntnis einfacher und komplexer Algorithmen- und Datenstrukturen sowie die Umsetzung dieser unter Verwendung von Programmiersprachen" [26, S.19] fordert, baut der letztere auf den Vorgaben der Profilstufe auf und verlangt gleichfalls die „Beherrschung der Grundlagen der Programmierung" [26, S.12].

Allen Vorgaben gemein ist, dass zwar ab der neunten Klasse die Auseinandersetzung mit einer Programmiersprache gefordert wird, letztlich die Wahl eines geeigneten Werkzeugs aber vom Lehrer frei getroffen werden kann. Schaut man sich in der deutschen Schulinformatik um, sieht man eine

Vielzahl an genutzten Sprachen, mit all ihren Vor- und Nachteilen. Man ist weit entfernt von einer universell einsetzbaren Lösung:

Die perfekte Programmiersprache für den Informatikunterricht ist noch nicht gefunden (vgl. [12, S.5]).

In Sachsen dominiert zurzeit noch das 1972 als Lehrsprache erschienene *Pascal* die Schulinformatiklandschaft, auch *Java* und Varianten von *C* finden Verwendung. Eine vergleichbare Situation findet sich auch in den anderen Bundesländern, es existieren viele verschiedene Ansätze, jedoch keine gemeinsame Lösung.

Bevor man anfangen kann, Programmiersprachen hinsichtlich ihrer Tauglichkeit für den Schulunterricht zu bewerten – was diese Arbeit im weiteren Verlauf anstrebt – steht die Frage nach Kriterien, die eine Schulinformatiksprache aufweisen sollte. Das wichtigste Kriterium sollte für jede Programmiersprache ein Aufbau sein, der in seinem Wesen so einfach wie möglich und nur so komplex wie nötig ist, um den Schülern den Einstieg sowie den Lehrern die Hilfestellung und Korrektur der Aufgaben zu erleichtern. Schlüsselwörter und Syntax sollten einfach und eindeutig gehalten sein, ein Programm im besten Fall für andere in kürzester Zeit verständlich sein.

Hierzu ist es vor allem notwendig, mit möglichst wenig Komplexität zu beginnen. Ein Programmtext einer perfekten Lernsprache sollte kein einziges Wort oder Satzzeichen enthalten, das nicht noch in derselben Unterrichtsstunde erklärt werden kann. Meiner Meinung nach disqualifiziert sich hier beispielsweise *C++*, wenn man allein die verwendeten Begriffe im *Hallo-Welt-Programm* zählt, dessen Erklärung sich der Unterrichtende für eine der kommenden Stunden aufsparen muss.

```
1 // Dateiname: hallowelt.cpp
2 #include <iostream>
3 using namespace std;
4 int main(void)
5 {
6     cout << "Hallo Welt \n";
7     cin.get();
8     return 0;
9 }
```

Python hingegen benötigt für die gleiche Ausgabe die das *C++*-Programm erzeugt nur eine einzige Zeile.

```
1 print("Hallo Welt!")
```

Dem Schüler muss hier statt vieler unbekannter Schlüsselwörter nur `print` erklärt werden, darauf aufbauend kann die eigentliche Programmierung schnell komplexer werden, ohne sich lange mit den Eigenheiten einer bestimmten Programmiersprache aufhalten zu müssen.

Hilbert Meyer formuliert in seinen Veröffentlichungen die *Zehn Merkmale guten Unterrichts*. Neben einer klaren Strukturiertheit gibt er an, dass vor allem der „Anteil echter Lernzeit“ entscheidend sei [15, S.23]. Bei den Unterrichtseinheiten der Informatik, die sich mit der Programmierung auseinandersetzen, sei es daher wichtig, dass die Schüler den Großteil der Zeit für strukturiertes Denken beziehungsweise dem aktiven Lösen von Problemen durch Algorithmen aufwenden.

Im Gegensatz hierzu erfordern viele der bisher genutzten Programmiersprachen eine intensive Auseinandersetzung mit der jeweiligen Syntax und dem Gewöhnen an die Konventionen wie Schlüsselwörter oder Klammersetzung. Vollständig maximieren könnte man die echte Lernzeit nur, wenn man in Pseudocode programmieren könnte oder auch für höhere Klassen vollständig auf grafische Programmierumgebungen wie *Scratch* oder *Kara* setzen könnte, was jedoch an der geforderten Komplexität der in höheren Klassen unterrichteten Programme scheitert.

Die Programmiersprache *Python* ist ein Kandidat, der dem Ideal eines Pseudocodes sehr nahe kommt. Obwohl die Sprache aus einer übersichtlichen Syntax und einfachsten Schlüsselwörtern besteht, handelt es sich doch um eine hohe und komplexe Programmiersprache.

Das folgende Beispiel zeigt eine in *Python* geschriebene rekursive Fakultätsberechnung

```
1 # Dateiname: fakultaet.py
2 def fakultaet(x):
3     if x > 1:
4         return x * fakultaet(x - 1)
5     else:
6         return 1
7
8 print(fakultaet(6))
```

Besonders anzumerken ist die formale Struktur, die zur fehlerfreien Kompilierung unbedingt eingehalten werden muss. Zeilenumbrüche und Einrückungen werden von *Python* als Teil des Codes behandelt und ermöglichen es somit, neben der sehr übersichtlichen Gestaltung, nahezu ohne Klammersetzung auszukommen. So wird es den Schülern erleichtert, strukturierte Programme zu schreiben.

Bei *Python* handelt es sich um eine OpenSource-Entwicklung, die kontinuierlich verbessert wird. Neben der Tatsache, dass die Sprache also kostenneutral in Schulen eingesetzt werden kann, existiert eine große Community, die auch deutsche Hilfeseiten, Dokumentationen und Foren anbietet. Anzumerken ist allerdings, dass der Großteil der verfügbaren Literatur sowie natürlich auch die meisten Internetquellen, wie bei vielen informatischen Themen, überwiegend in englischer Sprache verfasst werden.

Im Verlauf dieser Arbeit wird die Eignung der Programmiersprache *Python* für den schulischen Informatikunterricht ausführlicher diskutiert sowie auf deren Eigenschaften genauer eingegangen. Die aufgestellten Kriterien für eine in der Schule genutzten Sprache sollen noch genauer definiert werden, im Anschluss Vorschläge für Aufgaben und die Verwendung der Sprache gegeben werden.

Diese Arbeit verwendet *Python* in Version 3. Kürzere Programmbeispiele sind so notiert, wie sie in der *IDLE* stehen würden; es folgt eine sofortige Rückmeldung. Bei längeren Sequenzen steht ein Dateiname in der ersten Zeile, welcher die jeweilige Programmdatei auf dem beiliegenden Datenträger adressiert. Im Anhang findet sich unter Punkt A.1 auf Seite 45 eine Übersicht sämtlicher Programmierbeispiele, die der Arbeit beiliegen.

5 FAZIT

Diese Arbeit beschäftigte sich mit der Frage, ob sich die Programmiersprache *Python* für den Schulinformatikunterricht eignet und ob sie eine reelle Alternative zu den bisher verwendeten Sprachen wie *Java*, *Pascal* oder *C* ist.

Hierzu wurde die Sprache zunächst vorgestellt, die Historie beleuchtet sowie Vorgänger und Beweggründe für die Entwicklung genannt. Es schlossen sich der Aufbau und ein Überblick über die Eigenschaften an, welche schon andeuteten, wie bedienungsfreundlich und simpel diese Sprache gehalten ist. Anschließend wurden Anwendungsbereiche *Pythons* beleuchtet und es wurde gezeigt, dass sich die Sprache vor allem im angloamerikanischen Raum schon als Lehrsprache etabliert hat.

Um auf die Kernfrage der Unterrichtseignung *Pythons* eingehen zu können, sollte zunächst die Frage geklärt werden, welche Kriterien eine geeignete Unterrichtssprache mitbringen muss. Hierzu dienten vor allem die Überlegungen Werner Hartmanns und Raimon Reichelts als Grundlage. Neben einer einheitlichen, klaren Struktur standen hier auch Aspekte wie die Menge der verfügbaren Literatur, die allgemeine „Mächtigkeit" der Sprache sowie die Verfügbarkeit und Lizenzierung der Software im Vordergrund. Anschließend wurden die zu erfüllenden Anforderungen mit den Eigenschaften von *Python* verglichen. Besonders mit dem Hauptargument „Reduzierung und Ausblendung der Komplexität" konnte *Python* hier punkten. Auch die Visualisierung ist mit der klaren Struktur, dem mächtigen Debugger und den verständlichen Fehlermeldungen sichergestellt. Literatur ist vor allem in englischer Sprache erschienen, aber auch deutsche Werke sind reichlich vorhanden.

Der Ruf der Skriptsprachen als langsam und unflexibel trifft auf *Python* nach der durchgeführten Analyse nicht zu. Zwar sind die Befehle kurz gehalten, vieles ist automatisiert und die benötigte Laufzeit für Programme ist länger als etwa bei *C*; nichtsdestotrotz handelt es sich bei *Python* um eine

hohe, mächtige und vollständige Programmiersprache, die sich vor allem für Programmiereinsteiger eignet. Die OpenSource-Lizenz macht sie für Bildungseinrichtungen attraktiv, und ihre Aktualität und stetige Weiterentwicklung macht sie zu einer sehr guten Wahl, um Schülern das Programmieren zu lehren.

Um mit den Worten der *Python Software Foundation* zu schließen:

> „It is still common to start students with a procedural and statically typed language such as Pascal, C, or a subset of C++ or Java. Students may be better served by learning Python as their first language. Python has a very simple and consistent syntax and a large standard library and, most importantly, using Python in a beginning programming course lets students concentrate on important programming skills such as problem decomposition and data type design. With Python, students can be quickly introduced to basic concepts such as loops and procedures."
>
> (Python Software Foundation, 2013 [19, Unterseite General FAQ])

Es sei bei Schülern immer noch üblich, mit prozeduralen Programmiersprachen mit statischer Typisierung wie *Pascal*, *C* oder einer Abwandlung von *C++* oder *Java* zu beginnen. Schüler seien mit *Python* als erster Sprache besser bedient. *Python* habe eine sehr einfache und konstante Syntax und eine große Standardbibliothek. Vor allem könnten sich Schüler durch die Benutzung *Pythons* in einem Einsteigerkurs auf die wichtigen Programmierfähigkeiten wie die Problemzerlegung und das Datentypdesign konzentrieren. Mit *Python* könnten Schülern schnell Grundkonzepte wie Schleifen oder Prozeduren nähergebracht werden.

Dieser Aussage schließe ich mich an. Es gibt keinen Grund, *Python* nicht im Schulinformatikunterricht einzusetzen. Dies trifft auf den Programmiereinstieg in jüngeren Klassenstufen ebenso zu wie auf komplexere Anforderungen in höheren Klassen.

QUELLEN- UND LITERATURVERZEICHNIS

[1] AUTODESK: *Python in Maya*. Autodesk, `http://download.autodesk.com/global/docs/maya2013/en_us/files/Python_Python_in_Maya.htm`, 2013, zuletzt abgerufen am 09. Juli 2013.

[2] DU BOULAY, BENEDICT, TIM O'SHEA und JOHN MONK: *The black box inside the glass box: presenting computing concepts to novices*. International Journal of Human-Computer Studies, 51(2): 265 – 277, 1999.

[3] ELKNER, JEFFREY: *Using Python in a High School Computer Science Program*. Technischer Bericht, Yorktown High School, Arlington, Virginia, 2000.

[4] GALILEO PRESS: *Python — Das umfassende Handbuch*. Galileo Press, `http://www.heise.de/download/python-das-umfassende-handbuch.html`, 2008, zuletzt abgerufen am 15. Juli 2013.

[5] GUERTS, L. J. M.: *An Overview of the B Programming Language*. Stichting Mathematisch Centrum, Amsterdam, 1982.

[6] HAMILTON, NAOMI: *The A-Z of Programming Languages: Python*. Computerworld, `http://www.computerworld.com.au/article/255835/a-z_programming_languages_python`, 2008, zuletzt abgerufen am 15. Juli 2013.

[7] HARTMANN, WERNER und RAIMOND REICHERT: *System.out.println („Ich hab genug von 'Hello World'!"); – Programmieren für Einsteiger/innen*. infosense, Wettingen, 2011.

[8] KEMNITZ, GÜNTER: *Informatik für Schüler (11)*. TU Clausthal, `http://techwww.in.tu-clausthal.de/site/Lehre/Schuelerinformatik/`, 2012, zuletzt abgerufen am 21. Juli 2013.

[9] KEMNITZ, GÜNTER: *Informatik für Schüler (12)*. TU Clausthal, `http://techwww.in.tu-clausthal.de/site/Lehre/Schuelerinformatik13/`, 2012, zuletzt abgerufen am 21. Juli 2013.

[10] KOKAVECZ, BERND: *Didaktische Vorzüge von Python*. `http://www.b.shuttle.de/b/humboldt-os/python/vorwort/python2.html`, 2001, zuletzt abgerufen am 16. Juli 2013.

[11] LANDESAKADEMIE FÜR FORTBILDUNG UND PERSONALENTWICKLUNG AN SCHULEN BADEM-WÜRTTEMBERG: *Linux als Betriebsystem für die Arbeitsplatzrechner in Schulen?* Landesakademie für Fortbildung und Personalentwicklung an Schulen Badem-Württemberg, `http://lehrerfortbildung-bw.de/werkstatt/opensource/os/schule/linux/`, 2009, zuletzt abgerufen am 15. Juli 2013.

[12] LEDERER, DOMINIQUE: *Python und Zope als Unterrichtswerkzeuge*. Grin, München, 2007.

[13] M. TIM JONES: *Application development for the OLPC laptop*. IBM Developer Works, `http://www.ibm.com/developerworks/linux/tutorials/l-sugarpy/l-sugarpy-pdf.pdf`, 2007, zuletzt abgerufen am 09. Juli 2013.

[14] MAXON COMPUTER PUBLIC RELATIONS: *MAXON announces acquisition of Py4D*. MAXON Computer, `http://www.maxon.net/en/news/press-releases/singleview/article/maxon-announces-acquisition-of-py4d.html`, 2010, zuletzt abgerufen am 09. Juli 2013.

[15] MEYER, HILBERT: *Was ist guter Unterricht?* Cornelsen, Berlin, 8. Aufl. Auflage, 2011.

[16] MICHAEL MARKERT ET AL.: *Das Python-Tutorial*. `http://tutorial.pocoo.org`, 2013, zuletzt abgerufen am 15. Juli 2013.

[17] PEMBERTON, STEVEN: *The ABC Programming Language: a short introduction*. Centrum Wiskunde & Informatica Amsterdam, `http://homepages.cwi.nl/~steven/abc/`, 2012, zuletzt abgerufen am 08. Juli 2013.

[18] PUHLMANN, HERRMANN, TORSTEN BRINDA, MICHAEL FOTHE, STEFFEN FRIEDRICH, BERNHARD KOERBER, GERHARD RÖHNER und CARSTEN SCHULTE: *Grundsätze und Standards für die Informatik in der Schule. Bildungsstandards Informatik für die Sekundarstufe I*. Gesellschaft für Informatik (GI) e.V., 2008.

[19] PYTHON SOFTWARE FOUNDATION: *Python v3.3.2 documentation*. Python Software Foundation, `http://docs.python.org`, 2013, zuletzt abgerufen am 09. Juli 2013.

[20] PYTHON SOFTWARE FOUNDATION: *Associate Membership Program*. Python Software Foundation, `https://psfmember.org`, 2013, zuletzt abgerufen am 15. Juli 2013.

[21] PYTHON SOFTWARE FOUNDATION: *PyPI - the Python Package Index*. Python Software Foundation, `https://pypi.python.org/pypi`, 2013, zuletzt abgerufen am 15. Juli 2013.

[22] PYTHON SOFTWARE FOUNDATION: *The Python Wiki*. Python Software Foundation, `http://wiki.python.org`, 2013, zuletzt abgerufen am 15. Juli 2013.

[23] PYTHON SOFTWARE FOUNDATION: *Python.org – Homepage*. Python Software Foundation, `http://python.org`, 2013, zuletzt abgerufen am 15. Juli 2013.

[24] REICHERT, RAIMOND, JÜRG NIEVERGELT und WERNER HARTMANN: *Programmieren mit Kara - ein spielerischer Zugang zur Informatik*. Springer, Berlin, 2., überarb. und erw. Aufl. Auflage, 2005.

[25] SÄCHSISCHE LANDESBIBLIOTHEK — STAATS- UND UNIVERSITÄTSBIBLIOTHEK DRESDEN: *Suchabfrage Python Programmiersprache*. Sächsische Landesbibliothek — Staats- und Universitätsbibliothek Dresden, `http://katalog.slub-dresden.de/primo_library/libweb/action/search.do?ct=facet&fctN=facet_tlevel&fctV=available&fctN=facet_lang&fctN=facet_rtype&fctV=ger&fctV=books&tab=default_tab&dstmp=&srt=rank&ct=facet&indx=1&vl(freeText0)=python%20programmiersprache&fn=search&vid=SEM`, 2013, zuletzt abgerufen am 15. Juli 2013.

[26] SÄCHSISCHES STAATSMINISTERIUM FÜR KULTUS UND SPORT: *Lehrplan Gymnasium Informatik*. Dresden, 2011.

[27] SÄCHSISCHES STAATSMINISTERIUM FÜR KULTUS UND SPORT: *Lehrplan Gymnasium Mathematik*. Dresden, 2011.

[28] TIOBE SOFTWARE BV: *TIOBE Programming Community Index for July 2013*. TIOBE Software BV, `http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html`, 2013, zuletzt abgerufen am 09. Juli 2013.