Didaktische Anforderungen an die erste Programmiersprache

Nik Klever
Hochschule Augsburg
Fakultät für Informatik

Inhalt

- Einführung und Motivation
- Erfolgsgeschichten
- Brainstorming

Motivation

- Programmiererfahrung seit 1972
  - Assembler, Algol60, Algol68, Pascal, Fortran, C, C++, Perl, Python, Java, ...
- Lehrbeauftragter für Datenkommunikation und Netzwerk- und Webprogrammierung seit 1996
  - Zope und Python seit 1998
- Lehre im Studiengang "Interaktive Medien" seit 2001
  - "Interaktive Media" ist ein Studiengang, der zu 50% aus Informatik-Fächern und zu 50% aus Design-Fächern besteht
  - die Studierenden beginnen ihre Programmiererfahrung mit einer zweisemestrigen Java-Ausbildung
  - seit der Umstellung auf den Bachelorstudiengang (2006) mit dem Versuch, die Studierenden bereits in den Grundlagenfächern zu motivieren, Python für die Netzwerk- und Webprogrammierung (seit 2008 mit web2py) zu verwenden

„meine“ Anforderungen

„meine“ Anforderungen an eine
erste Programmiersprache

- benutzerfreundliche Bedienung
- Einfachheit
- schnelle Erfolge
- Open Source, damit die Motivation, wann und wo gelernt wird nicht durch fehlende Lizenzen eingeschränkt wird

Allen B. Downey

Professor of Computer Science am
Franklin W. Olin College of Engineering, Needham, Mass.

Autor der Bücher

- Think Python: How to think like a computer scientist, 2012
- Think Complexity: Complexity Science and Computational Modeling, 2012.
- Think Stats: Probability and Statistics for Programmers, 2011.
- Complexity and Computation, 2011.
- Think Java: How to think like a computer scientist, 2011.
- Python for Software Design, 2009.
- How to think like a computer scientist: C++ Version, 2009.
- Learning Perl the Hard Way, 2009.
- The Little Book of Semaphores, 2009.
- Think Python: An Introduction to Software Design, 2009.
- Physical Modeling in MATLAB, 2008.
- How to think like a computer scientist: Learning with Python, 2002.
- How to Think Like a Computer Scientist, 1999

Geschichte seines Buchs „Think Python: How to Think Like a Computer Scientist“

In January 1999 I was preparing to teach an introductory programming class in Java. I had taught it three times and I was getting frustrated. The failure rate in the class was too high and, even for students who succeeded, the overall level of achievement was too low.

One of the problems I saw was the books. They were too big, with too much unnecessary detail about Java, and not enough high-level guidance about how to program. And they all suffered from the trap door effect: they would start out easy, proceed gradually, and then somewhere around Chapter 5 the bottom would fall out. The students would get too much new material, too fast, and I would spend the rest of the semester picking up the pieces.

Two weeks before the first day of classes, I decided to write my own book. My goals were:

- **Keep it short. It is better for students to read 10 pages than not read 50 pages.**
- **Be careful with vocabulary. I tried to minimize the jargon and define each term at first use.**
- **Build gradually. To avoid trap doors, I took the most difficult topics and split them into a series of small steps.**
- **Focus on programming, not the programming language. I included the minimum useful subset of Java and left out the rest.**

I needed a title, so on a whim I chose „How to Think Like a Computer Scientist“.

Alan B. Downey

My first version was rough, but it worked. Students did the reading, and they understood enough that I could spend class time on the hard topics, the interesting topics and (most important) letting the students practice.

I released the book under the GNU Free Documentation License, which allows users to copy, modify, and distribute the book.

What happened next is the cool part. Jeff Elkner, a high school teacher in Virginia, adopted my book and translated it into Python. **He sent me a copy of his translation, and I had the unusual experience of learning Python by reading my own book.** As Green Tea Press, I published the first Python version in 2001.

In 2003 I started teaching at Olin College and I got to teach Python for the first time. The contrast with Java was striking. Students struggled less, learned more, worked on more interesting projects, and generally had a lot more fun.

http://www.greenteapress.com/thinkpython/

„adaptiert“ von Allen B. Downey

- In der Kürze liegt die Würze, d.h. Studierende sollten besser 10 Zeilen Code lesen als keine 50 Zeilen.
- Vorsichtig mit dem Wortschatz umgehen, d.h. die Syntax an den bekannten Wortschatz anpassen:
  - **for i in liste** anstatt **for (j=1; j<len; j++)**
- Schritt-für-Schritt Entwicklung: d.h. kleine, einfache Funktionen anstatt eine große Klasse (→ funktionale Programmierung)
- Fokussieren auf die Programmierung, nicht die Programmiersprache

Folgerung: ausführbaren Pseudocode benutzen, d.h. Python

edx.org Online Kurse

- Zusammenschluß der Universitäten
  - MIT
  - Harvard
  - Berkeley
  - University of Texas

  für Online Kurse
- Kurs 6.00x von MITx: Introduction to Computer Science and Programming
  - 6.00x will be using the Python programming language, version 2.7.

Learn. Think. Do.

Higher Education for Free

Gegründet von Sebastian Thrun, Professor für künstliche Intelligenz an der Stanford University, Google Fellow und Entwickler und Initiator diverser Projekte wie den autonom fahrenden Kfz oder dem Google Brillen-Projekt Glass

Kurs CS 101

- Kurs CS 101 Introduction to Computer Science
- Verwendet moderne didaktische Konzepte:
- **Fall-orientiert**: Entwicklung einer Suchmaschine
- **Quizzes**: Insgesamt 7 Units mit ca. 35-40 Teileinheiten mit den abschliessenden Quizzes
- **Python** als Programmiersprache

Learning to program – why Python

- Chris Blew erzählt in dem Udacity Blog, warum Udacity Python als Programmiersprache verwendet:
  - http://blog.udacity.com/2012/05/learning-to-program-why-python.html
- „Udacity's courses so far have been using Python because it is overall, the most convenient language for teaching and learning.“
- „For example object-oriented, imperative, and even functional programming paradigms can all be introduced using Python. The dynamic typing system makes it easy to try advanced practices like "aspect-oriented" design and "inversion of control" through "dependency injection" without complicated libraries. Somehow, amazingly, Python handles all of these crazy ideas with the same grace it handles "print 'hello world.'""“
- „It is important to keep in mind that time spent learning Python doubles as time spent learning C#, Ruby, or Java.“

Eric Raymond: Why Python

Eric Raymond hat im Linuxjournal Nr. 73 vom **30.4.2000** den Artikel „**Why Python ?**“ veröffentlicht (http://www.linuxjournal.com/article/3882):

- „This was my first clue that, in Python, I was actually dealing with an exceptionally good design.“
- „It's remarkable enough when implementations of simple techniques work exactly as expected the first time; but my first metaclass hack in a new language, six days from a cold standing start? Even if we stipulate that I am a fairly talented hacker, this is an amazing testament to Python's clarity and elegance of design.“
- „The long-term usefulness of a language comes not in its ability to support clever hacks, but from how well and how unobtrusively it supports the day-to-day work of programming. The day-to-day work of programming consists not of writing new programs, but mostly reading and modifying existing ones. So the real punchline of the story is this: weeks and months after writing fetchmailconf, I could still read the fetchmailconf code and grok what it was doing without serious mental effort.“

Raspberry Pi

- Die Raspberry Pi Foundation, die sich das Ziel gesetzt hat, die Informatik-Ausbildung an britischen Schulen zu verbessern, hat Python zur offiziell unterstützten Lehrsprache gemacht.
  - http://raspberrycenter.de/handbuch/python-programmierung

Brainstorming

- Was sind Eure Erfahrungen im Lernen von Python gewesen ?
- Welche Erfolgsstories gibt es außerdem ?
- Welche didaktischen Argumente sprechen für Python als erste Programmiersprache ?