Vorwort

Zur zweiten Auflage

Die zweite Auflage bleibt der schon im Vorwort zur ersten Auflage beschriebenen Konzeption des Buchs treu: Nämlich eine Einführung in die Informatik praktisch zu betreiben und mittels typischer Werkzeuge der Informatik, einem Betriebssystem und einer Programmiersprache, erfahrbar zu machen.

Das Buch wurde in der zweiten Auflage auf Python 3 umgestellt, um viele aktuelle Informationen ergänzt und von Tippfehlern und inhaltlichen Fehlern bereinigt.

Rückmeldungen zu Inhalt und Form können jederzeit gerne an den Autor direkt per Mail an `tobias.haeberlein@gmx.de` gegeben werden.

Viel Spaß beim Lesen und Ausprobieren.

Tobias Häberlein — Blaustein im Oktober 2016

Zur ersten Auflage

Es gibt eine Vielzahl exzellenter Bücher, die eine Einführung in die Informatik vermitteln, und dabei teilweise ein weites Feld von Themen abdecken. Jedoch vermisse ich in all diesen Informatik-Einführungen ein didaktisches Konzept des Ausprobierens: Eine einheitliche Art, alle präsentierten Konzepte selbst zu erfahren, mit Ihnen zu arbeiten und zu experimentieren und so die eigentlichen Probleme der Informatik besser zu verstehen. Insbesondere die praktische Informatik lebt vom Ausprobieren verschiedener Lösungswege, Experimentieren mit Programmkonstrukten und Algorithmen, und allgemein vom "Selber-Machen". Dieser Tatsache wird durch die Vielzahl der in diesem Buch gestellten Aufgaben Rechnung getragen. Diese befinden sich immer an genau der Stelle im Buch, die den zu übenden Stoff behandelt. Ein großer Teil der zugehörigen Lösungen ist zu finden unter `www.TobiasHaeberlein.net`.

Die wichtigsten Werkzeuge, um diese Konzepte der praktischen Informatik am Rechner auszuprobieren, sind das *Betriebssystem* und eine *Programmiersprache*. Aus diesem Grund geben die ersten beiden Kapitel dieses Buches, Kapitel 2 und 3, eine Einführung zum einen in ein für das Erlernen von Informatik-Konzepten geeignetes Betriebssystem (nämlich Linux) und zum anderen in eine für Anfänger besonders geeignete höhere Programmiersprache (nämlich Python). Nochmals im Detail:

Kapitel 2 (Unix/Linux und Shell-Programmierung) vermittelt eine Betriebssystem-nahe Sicht auf einen Rechner und lehrt die die Verwendung der einfachen ungetypten Skriptsprache der Bash. Hier soll der Umgang mit einem Betriebssystem *vollständig* ohne den Gebrauch graphischer Hilfsmittel (wie graphischer

Dateinavigator, Desktop, Maus, etc.) eingeübt werden. Meiner Erfahrung nach ist die (anfänglich) ausschließliche Verwendung der „nackten" Shell als Schnittstelle zum Betriebssystem ein effektiver Einstieg in den Umgang mit einem Betriebssystem.

Kapitel 3 (Python-Programmierung) geht – ausgehend von der ungetypten Skriptsprache der Bash – über zu einer höheren und typisierten Programmiersprache. Mit dieser werden zunächst ähnliche Problemstellungen (Dateimanipulation, Stringmanipulation, Ein- und Ausgabe) angegangen. Es wäre durchaus auch denkbar gewesen, eine andere Skriptsprache, wie etwa PHP, Perl oder awk und sed zu verwenden. Verglichen mit diesen ist jedoch Python die „höchste" Programmiersprache, bietet die größte Auswahl an Programmierparadigmen und besitzt zudem eine besonders für Anfänger eingängige Syntax. Zudem bietet sie, was vielleicht für das didaktische Konzept diese Buches am wichtigsten ist, eine interaktive Shell-artige Umgebung, in der man direkt mit den erstellten Funktionen „spielen" kann.

Ich möchte ausdrücklich darauf hinweisen, dass dieses Buch weder gedacht ist als umfassende Bash-Einführung, noch als umfassende Python-Einführung. Sowohl Bash als auch Python sind nichts weiter als didaktische Werkzeuge, um den Leser die wichtigsten Konzepte der Informatik „erleben" zu lassen.

Einige der vorgestellten Programmiertechniken (insbesondere die Betonung der Listenkomprehensionen) haben eine bewusst gewählte Färbung des Funktionalen Programmierparadigmas. Meine Erfahrung zeigt, dass dem späteren Programmierstil von Informatik-Anfängern eine stärkere Betonung des Funktionalen Programmierparadigmas in der Anfangsausbildung gut tut. Zudem bietet die Lehre von funktionalen Programmierkonstrukten einen effektiven Zugang zu der Art von Abstraktionsvermögen, die sich Anfänger so früh wie möglich aneignen müssen. Informatikanfänger machen häufig den Fehler, algorithmische Probleme zu „kleinschrittig" und Spaghetti-Code-artig zu lösen.

An folgenden beiden Lösungen der Aufgabe, eine Funktion zu schreiben, die alle Quadratzahlen von 1 bis n als Liste zurückliefert, kann man gut sehen, was ich hiermit meine. Listing 1 gibt ein von mir häufig gesehenes Beispiel einer zwar korrekten, aber Spaghetti-Code-artig implementierten „Anfänger"lösung. Listing 2 zeigt eine besser verständliche (und daher auch professionellere, „informatischere") Lösung der Aufgabenstellung. Der Leser soll auf dieser Reise durch das „Informatik-Denken" auch lernen, wie man algorithmische Problemstellungen professionell und elegant lösen kann.

```
def quadListe1(n):
  ergebnisLst=[]
  i = 1
  while i<=n:
    quadrat = i*i
    i = i+1
    ergebnisLst.append(quadrat)
  return ergebnisLst
```

Listing 1. „Anfänger“lösung

```
def quadListe2(n):
  return [i*i for i in range(1,n+1)]
```

Listing 2. „Informatiker“lösung

Eine knapp 250-seitige Einführung in die Informatik, ausgelegt auf eine 4-6 SWS-Vorlesung, muss sich entscheiden, welche Teilgebiete der Informatik für die Einführung ausgewählt werden. Neben der Funktionsweise eines Betriebssystems und der Konzepte einer höheren Programmiersprache, die beide als Basis für alle weiteren Kapitel dienen, sind dies hier:

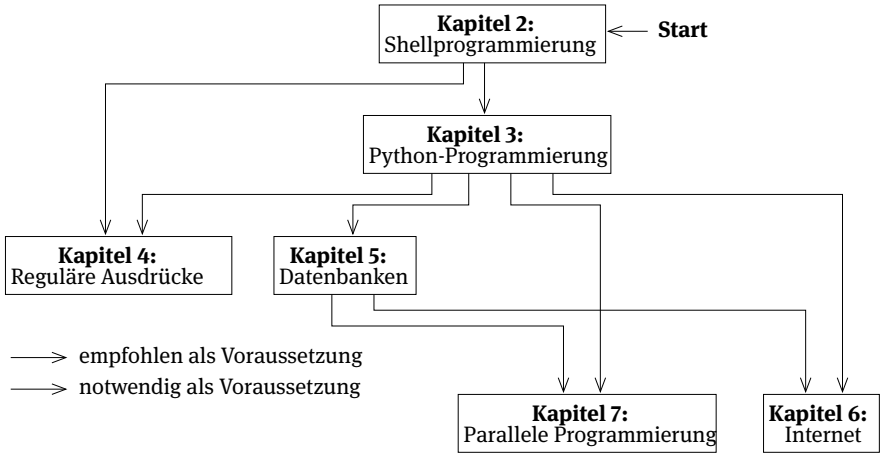
Kapitel 4 (Programmierung mit regulären Ausdrücken) Einführung in die Verwendung regulärer Ausdrücke. Auch hier gilt wieder, dass dies keinesfalls als umfassende Darstellung der Verwendung regulärer Ausdrücke zu verstehen ist, sondern vielmehr als Übung im Umgang mit für die Informatik typischen formalen Konstrukten. Zusätzlich stellt dies für den späteren Programmieralltag eine nützliche Übung dar.

Kapitel 5 (Datenbanken und Datenbankprogrammierung) Einführung in Konzepte der Datenpersistenz. Dies stellt eine notwendige Grundlage vieler typischer Anwendungen in der Informatik dar. Zu diesem Themengebiet zählen unter anderem Datenbanken, Datenbankmanagementsysteme, SQL und No-SQL-Datenbanken und -Datenbanksprachen.

Kapitel 6 (Internet und Internetprogrammierung) Einführung in die Funktionsweise des Internet und in die Internetprogrammierung. Hierzu zählen Socketprogrammierung, HTML-Syntax und HTML-Generierung Kommunikation mit und Funktionsweise eines Web-Servers und Entwurf einfacher interaktiver Web-Seiten.

Kapitel 7 (Nebenläufige und Parallele Programmierung) Einführung in die Programmierung nebenläufiger und paralleler Algorithmen. Dies ist ein anspruchsvolles Thema, das gerade aufgrund der Komplexität des Zusammenspiels und der Synchronisation nebenläufiger und paralleler Programmteile seinen Reiz hat. Zusäztlich erlangt die parallele Programmierung immer größere Relevanz in Zeiten der Mehrkern-Rechner des Cloud-Computing und der leistungsfähigen feingranular parallelisierbaren graphischen Prozessoren.

Die folgende Abbildung stellt die Abhängigkeiten zwischen den einzelnen Kapiteln dar:



Vorschläge für eine auf diesem Buch basierenden Einführungsvorlesung:

2-SWS-Vorlesung: (a) Kapitel 2 → Kapitel 3 → Kapitel 6
(Die Basiskapitel und für die Studierenden interessante Internet-Inhalte)
(b) Kapitel 2 → Kapitel 3 → Kapitel 4
(Die Basiskapitel und eine zur theoretischen Informatik affinen Anwendung)
(c) Kapitel 2 → Kapitel 3 → Kapitel 5 (Die Basiskapitel und Datenbankgrundlagen)

4-SWS-Vorlesung: (a) Alle Kapitel – je nachdem wie flott präsentiert wird, kann der Stoff jedoch über Umfang einer 4-SWS-Vorlesung hinausgehen.
(b) Man kann durchaus eines der Kapitel 4, 5, 6 oder 7 einfach weglassen.

6-SWS-Vorlesung: Meiner Erfahrung nach ist der in diesem Buch vermittelte Stoff sehr gut geeignet für eine Vorlesung mit Umfang 6 SWS.

Wie oben schon erwähnt, kann ein 200-seitiges Einführungsbuch nicht alle Themengebiete der Informatik abdecken. Diese Einführung blendet insbesondere aus: alle technischeren Aspekte der Informatik (Implementierung von Betriebssystemen, Funktionsweise eines Rechners und Computernetzwerke), Boolesche Logik und die theoretischeren Aspekte der Informatik (Algorithmik, Logik, Komplexitätstheorie, usw.) sowie einige Aspekte der praktischen Informatik (UML, Objekt-orientierter Soft-

wareentwurf, viele Aspekte des Software-Engineering, Vorgehensmodelle, Software-Qualitätssicherung, Projektmanagement, usw.).

Ich wünsche Ihnen allen so viel Spaß beim Lesen, Lernen oder Lehren wie ich beim Schreiben dieses Buchs hatte!

Tobias Häberlein — Blaustein im Mai 2011