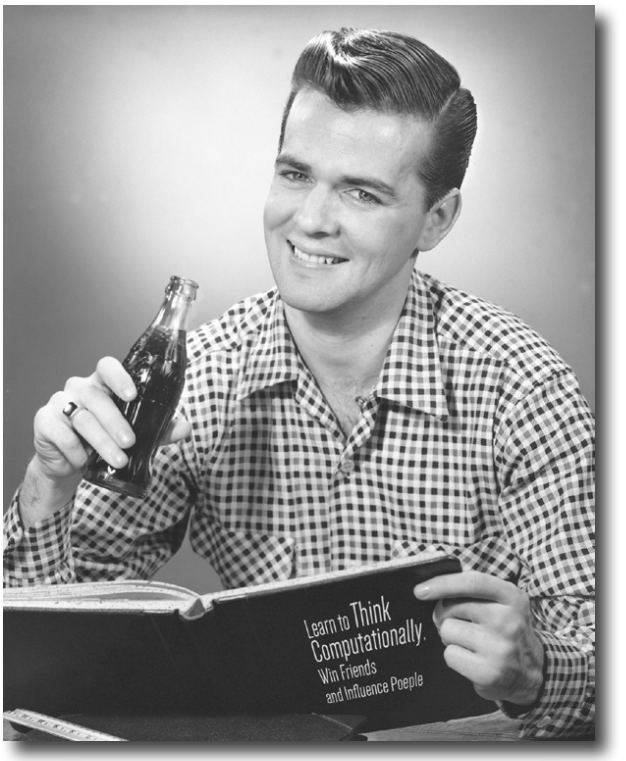
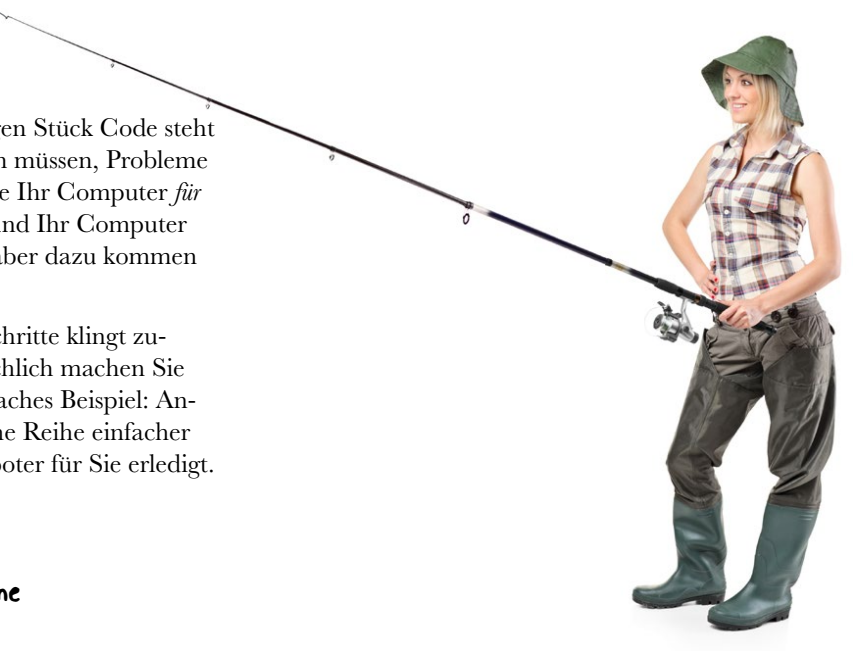
Erste Schritte



Durch die Fähigkeit, rechnerisch zu denken, haben Sie die Kontrolle. Jeder weiß, dass die Welt um einen herum immer vernetzter, konfigurierbarer, programmierbarer und eben **rechnerischer** wird. Entweder Sie bleiben passiv, oder Sie lernen zu programmieren. Sobald Sie es können, sind Sie Regisseur und Schöpfer – Sie sagen dem Computer, was er für Sie tun soll. Sobald Sie es können, bestimmen Sie Ihr Schicksal selbst (oder können zumindest Ihre Rasensprinkleranlage über das Internet steuern). Aber wie lernt man zu programmieren? Zuerst müssen Sie lernen, **rechnerisch** zu denken. Dann schnappen Sie sich eine **Programmiersprache**, um mit Ihrem Computer, Mobilgerät oder eigentlich allem mit einer CPU sprechen zu können. Und was haben Sie davon? Mehr Zeit, mehr Fähigkeiten und mehr kreative Möglichkeiten, die Dinge zu tun, die Sie wirklich tun wollen. Dann mal los …

Immer der Reihe nach



Zwischen Ihnen und Ihrem ersten richtigen Stück Code steht zunächst, dass Sie sich damit beschäftigen müssen, Probleme in kleine erreichbare Ziele aufzuteilen, die Ihr Computer *für Sie ausführen* kann. Natürlich müssen Sie und Ihr Computer dafür eine gemeinsame Sprache finden, aber dazu kommen wir erst etwas später.

Das Aufteilen eines Problems in kleine Schritte klingt zunächst nach einer neuen Fertigkeit, tatsächlich machen Sie das aber jeden Tag. Nehmen wir ein einfaches Beispiel: Angenommen, Sie wollten das Angeln in eine Reihe einfacher Anweisungen aufteilen, die dann ein Roboter für Sie erledigt. Hier unser erster Versuch:

Teilen wir das Angeln in eine Reihe einfacher Schritte auf.

Wir führen die Schritte nacheinander aus.

1. Köder am Haken befestigen.
2. Leine auswerfen.
3. Pose beobachten, bis sie untertaucht.
4. Anhaken und Fisch einholen.
5. Fertig mit dem Angeln? Dann gehe nach Hause, ansonsten zurück zu Schritt 1.

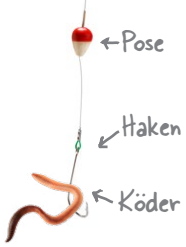
Manche Schritte sind einfache Anweisungen, wie: »Wirf die Leine aus!«

Bei einigen Anweisungen muss unter bestimmten Bedingungen gewartet werden, bevor es weitergeht.

Diese Anweisung wird nur ausgeführt, wenn die Pose in der vorigen Anweisung untergetaucht ist.

Anweisungen können auch Entscheidungen treffen, z. B. ob es Zeit ist, nach Hause zu gehen oder weiterzuangeln.

Beachten Sie, dass Anweisungen wiederholt werden können, wie hier: Wenn wir nicht nach Hause gehen, springen wir zurück zum Anfang und wiederholen die Anweisungen, um einen weiteren Fisch zu fangen.



Stellen Sie sich diese Anweisungen als eine Art **Rezept** zum Angeln vor. Wie jedes Rezept enthält es mehrere Arbeitsschritte. Werden diese der Reihe nach abgearbeitet, wird am Ende ein Ergebnis oder sogar ein Erfolg stehen (hier hoffen wir, ein paar Fische zu fangen).

Wie Sie sehen, bestehen die meisten Schritte aus einfachen Anweisungen, wie »Leine auswerfen« oder »Fisch einholen«. Einige Anweisungen sind anders, weil sie von einer Bedingung abhängen, z. B. »ist die Pose über oder unter Wasser?«. Anweisungen können auch die Reihenfolge der Arbeitsschritte beeinflussen, wie: »Wenn Sie noch nicht genug geangelt haben, gehen Sie zurück zum Anfang und befestigen einen neuen Köder am Haken.« Oder wie wäre es mit einer Bedingung zum Beenden wie: »Wenn du mit Angeln fertig bist, gehe nach Hause.«?

Sie werden merken, dass einfache Anweisungen wie diese die Grundlage für das Programmieren bilden. Jede App oder andere Software, die Sie je benutzt haben, ist nichts anderes als eine (manchmal ziemlich lange) Folge einfacher Anweisungen, die dem Computer sagen, was er tun soll.

ÜBUNG

Echte Rezepte sind keine reinen *Handlungsanweisungen*, denn sie enthalten auch eine Reihe von Objekten, die für die Zubereitung einer bestimmten Speise nötig sind (wie Messbecher, Schneebesen, Küchenmaschinen und natürlich die Zutaten). Welche Objekte werden in unserem Angelrezept verwendet? Kreisen Sie alle Objekte des Angelrezepts auf der vorigen Seite ein und überprüfen Sie Ihre Antwort am Ende des Kapitels, bevor Sie weiterlesen.

Dies ist ein Arbeitsbuch. Sie können etwas hineinschreiben. Und das sollen Sie sogar.

Spitzen Sie Ihren Bleistift

Eines müssen Sie noch wissen: Computer tun immer **genau** das, was Sie Ihnen sagen – nicht mehr und auch nicht weniger. Sehen Sie sich das Rezept zum Angeln auf der vorigen Seite noch mal an. Welche Probleme gäbe es, wenn Sie der Roboter wären und die Anweisungen genau befolgen müssten? Glauben Sie, dass dieses Rezept wirklich zum Erfolg führte?

- ☐ A. Wurden keine Fische gefangen, werden Sie eine sehr lange Zeit angeln (eigentlich für immer).
- ☐ B. Löst sich der Köder vom Haken, werden Sie das nie erfahren oder ihn ersetzen.
- ☐ C. Was passiert, wenn wir keinen Köder mehr haben?
- ☐ D. Haben wir angegeben, was mit dem Fisch zu tun ist, nachdem wir ihn eingeholt haben?
- ☐ E. Was passiert mit der Angelrute?
- ☐ F. _____

Fallen Ihnen weitere Probleme ein?

Falls Sie keine Ahnung vom Angeln haben: Dies ist eine Pose, gelegentlich auch als Schwimmer bezeichnet. Beißt ein Fisch an, geht sie unter Wasser.

Die Antworten zu den »Spitzen Sie Ihren Bleistift«-Übungen gibt es am Kapitelende.

Ich habe dieses Technikbuch gekauft, um programmieren zu lernen, und Sie fangen an, mir was über Rezepte zu erzählen? Das klingt ja nicht gerade vielversprechend, oder gar technisch.

Tatsächlich ist ein Rezept ideal geeignet, um eine Reihe von Anweisungen für einen Computer zu beschreiben. Möglicherweise begegnet Ihnen der Begriff sogar hier und da in Programmierbüchern für Fortgeschrittene. Oh Mann … es gibt sogar Bücher über allgemeine Softwareentwicklung, die als Kochbücher bezeichnet werden. Aber »technisch« können wir natürlich auch. Ein Informatiker oder Softwareentwickler würde ein Rezept üblicherweise als **Algorithmus** bezeichnen. Was ist ein Algorithmus? Um ehrlich zu sein, nicht viel mehr als ein Rezept – eine Folge von Anweisungen, die ein Problem lösen. Anfangs werden Algorithmen oft noch nicht in einer richtigen Programmiersprache, sondern in **Pseudocode** geschrieben.

Zum Pseudocode kommen wir später noch genauer …

Aber egal, ob Sie nun von einem Rezept, Pseudocode oder einem Algorithmus sprechen, es geht darum, eine allgemeine Beschreibung zur Lösung eines Problems zu finden. Das passiert, bevor Sie sich den Details widmen und Code schreiben, den ein Computer verstehen und ausführen kann.

Wie Sie sehen, wird das Programmieren dadurch einfacher und weniger fehleranfällig.

In diesem Buch setzen wir diese Begriffe übrigens synonym ein, wie es gerade sinnvoll ist. In Ihrem nächsten Vorstellungsgespräch sollten Sie vermutlich die Begriffe *Algorithmus* oder sogar *Pseudocode* benutzen, um einen höheren Antrittsbonus rauszuschlagen (wobei das Wort *Rezept* auch vollkommen in Ordnung ist).

So, wie es viele Rezepte für das gleiche Gericht gibt, so gibt es auch mehrere Algorithmen, die das gleiche Problem lösen. Einige schmecken besser als andere.

Code-Magneten

Eine kleine Übung zu ~~Rezepten~~ Algorithmen. Wir haben den Algorithmus des Von Kopf bis Fuß-Restaurants für die Zubereitung eines Omeletts aus drei Eiern an den Kühlschrank geheftet. Blöderweise hat irgendjemand den Code durcheinandergebracht. Können Sie die Magneten wieder in die richtige Reihenfolge bringen, damit unser Algorithmus funktioniert? Bedenken Sie, dass es im Von Kopf bis Fuß-Restaurant Omeletts mit und ohne Käse gibt. Vergessen Sie nicht, Ihre Antwort am Ende des Kapitels zu überprüfen.

Ordnen Sie die Magneten so an, dass der Algorithmus funktioniert.

```
Sofern der Kunde Käse bestellt hat:
Mit Käse bestreuen
Solange die Eier nicht komplett vermischt sind:
Omelett auf den Teller legen
Solange die Eier nicht komplett gar sind:
Pfanne vom Herd nehmen
Eier umrühren
Servieren
Pfanne vorwärmen
Eier schlagen
Drei Eier aufschlagen und in eine Schüssel geben
Eier in die Pfanne geben
```

Wie Programmieren funktioniert

Angenommen, der Computer soll eine Aufgabe für Sie erledigen. Diese Aufgabe müssen Sie in Einzelschritte aufteilen, die der Computer versteht. Aber wie können Sie den Computer *tatsächlich anweisen*, etwas zu tun? Mit einer Programmiersprache. Sie beschreibt Ihre Aufgabe in Begriffen, die sowohl *Sie als auch der Computer* verstehen. Bevor wir uns aber so richtig intensiv mit Programmiersprachen befassen, wollen wir uns die Schritte ansehen, die beim Schreiben von Code tatsächlich ausgeführt werden müssen:

1 Erstellen Sie Ihren Algorithmus

Hier übersetzen wir das Problem oder die Aufgabe in ein allgemeines Rezept, einen Pseudocode oder einen Algorithmus, der die Schritte beschreibt, die der Computer ausführen muss, um das gewünschte Ergebnis zu erreichen.

1. Köder am Haken befestigen.
2. Leine auswerfen.
3. Pose beobachten, bis sie untertaucht.
4. Anhaken und Fisch einholen.
5. Fertig mit dem Angeln? Dann gehe nach Hause, ansonsten zurück zu Schritt 1.

In diesem Schritt skizzieren wir unsere Lösung, bevor wir uns an die Schwerarbeit machen, ihn in eine Programmiersprache zu übersetzen.

2 Schreiben Sie Ihr Programm

Als Nächstes übersetzen Sie Ihr Rezept in die Anweisungen einer Programmiersprache. Dies ist die Stufe des *Programmierens*. Das Ergebnis nennen wir ein *Programm* oder einfach »Ihren Code« (bzw. formeller den *Quellcode*).

```
def hook_fish():
    print('I got a fish!')

def wait():
    print('Waiting...')

print('Get worm')
print('Put worm on hook')
print('Throw in lure')

while True:
    response = input('Is bobber underwater? ')
    if response == 'yes':
        is_moving = True
        print('I got a bite!')
        hook_fish()
    else:
        wait()
```

In diesem Schritt wird >>gecodet<<, Ihr Algorithmus wird in Code (kurz für Quellcode) umgewandelt, den wir im nächsten Schritt ausführen wollen.

3 Führen Sie Ihr Programm aus

Zum Schluss übergeben Sie Ihren Quellcode dem Computer. Der beginnt, Ihre Anweisungen auszuführen. Je nach verwendeter Programmiersprache wird dieser Vorgang als *Interpretieren*, *Laufenlassen*, *Auswerten* oder *Ausführen* Ihres Codes bezeichnet.

Wir verwenden diese Begriffe oft gleichbedeutend.

Wenn Ihr Quellcode fertig ist, können Sie ihn ausführen. Klappt alles und haben Sie Ihren Code gut gestaltet, gibt der Computer das gewünschte Ergebnis zurück.

Sprechen wir überhaupt die gleiche Sprache?

Stellen Sie sich eine **Programmiersprache** als eine Fachsprache für bestimmte Computeraufgaben vor. Mit Programmiersprachen können Sie Ihre Rezepte so klar und präzise beschreiben, dass selbst Computer sie verstehen.

Zum Lernen einer Programmiersprache brauchen Sie zwei Dinge: Was können Sie mit der Sprache ausdrücken, und was bedeutet das? Ein IT-Experte würde das als **Syntax** und **Semantik** der Programmiersprache bezeichnen. Behalten Sie diese Begriffe einfach im Hinterkopf; wir kommen später darauf zurück.

Wie bei gesprochenen Sprachen gibt es auch bei Programmiersprachen eine *große Vielfalt*. Und wie Sie vielleicht schon gemerkt haben, benutzen wir in diesem Buch die Programmiersprache Python. Wir wollen ein besseres Gefühl für Programmiersprachen im Allgemeinen und Python im Besonderen entwickeln …

Entspannen Sie sich

Keine Sorge. Sie müssen erst mal keinen Code schreiben, schließlich haben Sie das ganze Buch ja noch vor sich. Im Moment machen wir uns nur mit dem Aussehen und der Funktionsweise des Codes vertraut. In diesem Kapitel geht es lediglich darum, sich voll und ganz darauf einzulassen.

Die Techniken dieses Buchs können übrigens mit fast allen Programmiersprachen umgesetzt werden, die Ihnen in Zukunft begegnen werden.

DU SAGST TOMATE …

Auf der linken Seite sehen Sie einige Anweisungen in deutscher Sprache, rechts daneben sehen Sie die Anweisungen in einer Programmiersprache. Verbinden Sie die deutsche Anweisung mit der entsprechenden Codeübersetzung. Die erste Verbindung haben wir schon für Sie hergestellt. Überprüfen Sie Ihre Lösung auf jeden Fall am Ende des Kapitels, bevor Sie weiterlesen.

Gib »Hi there« auf dem Bildschirm aus.

Ist die Temperatur höher als 26 Grad, gib die Meldung »Wear shorts« auf dem Bildschirm aus.

Ein Einkaufszettel mit Brot, Milch und Eiern.

Fünf Drinks einschenken.

Frage den Benutzer: What is your name?

```
for num in range(0, 5):
        pour_drink()
```

```
name = input('What is your name? ')
```

```
if temperature > 26:
        print('Wear shorts')
```

```
grocery_list = ['bread', 'milk', 'eggs']
```

```
print('Hi there')
```

Die Welt der Programmiersprachen

Wenn Sie dieses Buch lesen, haben Sie beiläufig vielleicht schon von verschiedenen Programmiersprachen gehört. In der Informatikabteilung Ihres Buchladens finden Sie Titel zu Java, C, C++, LISP, Scheme, Objective-C, Perl, PHP, Swift, Clojure, Haskell, COBOL, Ruby, Fortran, Smalltalk, BASIC, Algol, JavaScript und natürlich Python, um nur ein paar zu nennen. Vielleicht fragen Sie sich, woher all die Namen stammen. Sie haben viel gemeinsam mit den Namen von Rockbands und haben für diejenigen, die sie entwickelt haben, oft eine bestimmte Bedeutung. Zum Beispiel Java: Die Sprache wurde, wenig überraschend, nach einer Kaffeesorte benannt (der ursprünglich gewünschte Name Oak war bereits vergeben). Haskell ist nach einem Mathematiker benannt, und C heißt so, weil sie der Nachfolger der Sprachen A und B bei den Bell Labs war. Aber warum gibt es so viele Sprachen, und was sind ihre Eigenheiten? Sehen wir, was einige Programmierer über die von ihnen verwendeten Sprachen zu sagen haben:

Ich baue den ganzen Tag iPhone-Apps und mag es, wie Objective-C und C sich ähneln, wobei Objective-C viel dynamischer und objektorientiert ist. Außerdem lerne ich Apples neue Programmiersprache **Swift**.

Bei **Java** denke ich anstelle von Low-Level-Code über Objekte nach. Viele Dinge wie Speicherverwaltung und Threading werden automatisch erledigt.

Ich lebe in der Welt von WordPress, das in **PHP** geschrieben ist. Deshalb ist PHP für mich die Sprache der Wahl. Meine Leute bezeichnen PHP als Skriptsprache, aber sie tut alles, was ich brauche.