

dpunkt.verlag



Al Sweigart

Eigene Spiele programmieren: Python lernen

Der spielerische Weg zur Programmiersprache

Was sind E-Books von dpunkt?

Unsere E-Books sind Publikationen im PDF- oder ePub-Format, die es Ihnen erlauben, Inhalte am Bildschirm zu lesen, gezielt nach Informationen darin zu suchen und Seiten daraus auszudrucken.

Sie benötigen zum Ansehen den Acrobat Reader oder ein anderes adäquates Programm bzw. einen E-Book-Reader.

E-Books können Bücher (oder Teile daraus) sein, die es auch in gedruckter Form gibt (bzw. gab und die inzwischen vergriffen sind). (Einen entsprechenden Hinweis auf eine gedruckte Ausgabe finden Sie auf der entsprechenden E-Book-Seite.)

Es können aber auch Originalpublikationen sein, die es ausschließlich in E-Book-Form gibt. Diese werden mit der gleichen Sorgfalt und in der gleichen Qualität veröffentlicht, die Sie bereits von gedruckten dpunkt.büchern her kennen.

Was darf ich mit dem E-Book tun?

Die Datei ist nicht kopiergeschützt, kann also für den eigenen Bedarf beliebig kopiert werden. Es ist jedoch nicht gestattet, die Datei weiterzugeben oder für andere zugänglich in Netzwerke zu stellen. Sie erwerben also eine Ein-Personen-Nutzungslizenz.

Wenn Sie mehrere Exemplare des gleichen E-Books kaufen, erwerben Sie damit die Lizenz für die entsprechende Anzahl von Nutzern.

Um Missbrauch zu reduzieren, haben wir die PDF-Datei mit einem Wasserzeichen (Ihrer E-Mail-Adresse und Ihrer Transaktionsnummer) versehen.

Bitte beachten Sie, dass die Inhalte der Datei in jedem Fall dem Copyright des Verlages unterliegen.

Wie erhalte ich das E-Book von dpunkt?

Sobald der Bestell- und Bezahlvorgang abgeschlossen ist, erhalten Sie an die von Ihnen angegebene Adresse eine Bestätigung. Außerdem erhalten Sie von dpunkt eine E-Mail mit den Downloadlinks für die gekauften Dokumente sowie einem Link zu einer PDF-Rechnung für die Bestellung.

Die Links sind zwei Wochen lang gültig. Die Dokumente selbst sind mit Ihrer E-Mail-Adresse und Ihrer Transaktionsnummer als Wasserzeichen versehen.

Wenn es Probleme gibt?

Bitte wenden Sie sich bei Problemen an den [dpunkt.verlag](mailto:dpunkt.verlag@dpunkt.de)
e-mail: ebooks@dpunkt.de
fon: 06221/1483-0.

Papier
plus⁺
PDF.

Zu diesem Buch – sowie zu vielen weiteren dpunkt.büchern – können Sie auch das entsprechende E-Book im PDF-Format herunterladen. Werden Sie dazu einfach Mitglied bei dpunkt.plus⁺:

www.dpunkt.plus

Al Sweigart

Eigene Spiele programmieren – Python lernen

Der spielerische Weg zur Programmiersprache



dpunkt.verlag

Al Sweigart

Lektorat: Gabriel Neumann

Fachgutachterin (amerikanische Ausgabe): Ari Lacenski

Copy-Editing: Claudia Lötschert

Übersetzung & Satz: G&U Language & Publishing Services GmbH, www.gundu.com

Herstellung: Nadine Thiele

Umschlaggestaltung: Helmut Kraus, www.exclam.de

nach der Originalvorlage von No Starch Press

Druck und Bindung: M.P. Media-Print Informationstechnologie GmbH, 33100 Paderborn

Bibliografische Information der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

ISBN:

Print 978-3-86490-492-9

PDF 978-3-96088-322-7

ePub 978-3-96088-323-4

mobi 978-3-96088-324-1

1. Auflage 2017

Translation Copyright für die deutschsprachige Ausgabe © 2017 dpunkt.verlag GmbH

Wieblinger Weg 17

69123 Heidelberg

Copyright © 2017 by Al Sweigart. Title of English-language original: Invent Your Own Computer Games with Python, 4th Edition, ISBN 978-1-59327-795-6, published by No Starch Press.

German-language edition copyright © 2017 by dpunkt.verlag. All rights reserved.

Die vorliegende Publikation ist urheberrechtlich geschützt. Alle Rechte vorbehalten. Die Verwendung der Texte und Abbildungen, auch auszugsweise, ist ohne die schriftliche Zustimmung des Verlags urheberrechtswidrig und daher strafbar. Dies gilt insbesondere für die Vervielfältigung, Übersetzung oder die Verwendung in elektronischen Systemen.

Es wird darauf hingewiesen, dass die im Buch verwendeten Soft- und Hardware-Bezeichnungen sowie Markennamen und Produktbezeichnungen der jeweiligen Firmen im Allgemeinen warenzeichen-, marken- oder patentrechtlichem Schutz unterliegen.

Alle Angaben und Programme in diesem Buch wurden mit größter Sorgfalt kontrolliert. Weder Autor noch Verlag können jedoch für Schäden haftbar gemacht werden, die in Zusammenhang mit der Verwendung dieses Buches stehen.

5 4 3 2 1 0

Inhalt

Der Autor	xx
Die Fachgutachterin der amerikanischen Ausgabe	xx
Danksagung	xxi
Einleitung	1
Zielgruppe	2
Der Aufbau dieses Buches	3
Die Codebeispiele	5
Zeilennummern und Einrückungen	5
Lange Codezeilen	6
Python herunterladen und installieren	6
IDLE starten	8
Online Hilfe finden	9
1 Die interaktive Shell	11
Einfaches Rechnen	12
Integer und Fließkommazahlen	13
Ausdrücke	13
Ausdrücke auswerten	14
Syntaxfehler	15
Werte in Variablen speichern	15
Zusammenfassung	19

2 Programme schreiben	21
Stringwerte	22
Stringverkettung	23
Programme im Dateieditor von IDLE schreiben	23
Das Hello-World-Programm schreiben	24
Das Programm speichern	25
Das Programm ausführen	26
So funktioniert das Hello-World-Programm	27
Kommentare	27
Funktionen: Miniprogramme innerhalb von Programmen	28
Das Ende des Programms	29
Variablennamen	30
Zusammenfassung	31
3 Zahlen raten	33
Ein Beispieldurchlauf des Zahlenratespiels	34
Der Quellcode für das Zahlenratespiel	35
Das Modul random importieren	36
Zufallszahlen mit der Funktion random.randint() erzeugen	37
Den Spieler begrüßen	38
Flusssteuerungsanweisungen	39
Code in Schleifen wiederholen	39
Blöcke	39
for-Schleifen	41
Die Vermutung des Spielers abrufen	42
Werte mit den Funktionen int(), float() und str() umwandeln	42
Boolesche Werte	44
Vergleichsoperatoren	45
Bedingungen	45
Experimente mit booleschen Werten, Vergleichsoperatoren und Bedingungen	46
Der Unterschied zwischen = und ==	47
if-Anweisungen	48
Schleifen mit break vorzeitig abbrechen	48
Wenn der Spieler gewonnen hat	48
Wenn der Spieler verloren hat	49
Zusammenfassung	50

4 Ein Kalauerprogramm	53
Ein Beispieldurchlauf von Jokes	54
Der Quellcode für Jokes	54
Funktionsweise des Codes	55
Maskierungszeichen	55
Einfache und doppelte Anführungszeichen	56
Der Schlüsselwortparameter end der Funktion print()	57
Zusammenfassung	58
5 Im Reich der Drachen	59
Spielverlauf von Dragon Realm	59
Ein Beispieldurchlauf von Dragon Realm	60
Das Flussdiagramm für Dragon Realm	60
Quellcode von Dragon Realm	61
Die Module random und time importieren	63
Funktionen	63
def-Anweisungen	63
Funktionen aufrufen	64
Wohin mit Funktionsdefinitionen?	64
Mehrzeilige Strings	65
while-Schleifen	65
Boolesche Operatoren	66
Der Operator and	67
Der Operator or	68
Der Operator not	68
Ausdrücke mit booleschen Operatoren auswerten	69
Rückgabewerte	70
Globaler und lokaler Gültigkeitsbereich	71
Funktionsparameter	72
Das Ergebnis anzeigen	73
Die Höhle mit dem freundlichen Drachen bestimmen	74
Die Hauptschleife des Spiels	75
Die Funktionen aufrufen	75
Neue Runde	76
Zusammenfassung	77

6 Der Debugger	79
Fehlerarten	80
Der Debugger	81
Den Debugger starten	81
Das Programm im Debugger schrittweise durchlaufen	82
Fehler finden	86
Haltepunkte setzen	89
Haltepunkte verwenden	90
Zusammenfassung	92
7 Galgenmännchen: Entwurf mit einem Flussdiagramm	93
Die Regeln für Galgenmännchen	94
Ein Beispieldurchlauf von Galgenmännchen	94
ASCII-Grafik	95
Programme mit Flussdiagrammen entwerfen	96
Das Flussdiagramm zeichnen	97
Verzweigungen in Flussdiagrammen	98
Das Spiel beenden oder neu starten	100
Erneut raten	101
Rückmeldung an den Spieler	102
Zusammenfassung	103
8 Galgenmännchen: Der Code	105
Der Quellcode von Galgenmännchen	106
Das Modul random importieren	109
Konstanten	109
Der Datentyp für Listen	110
Zugriff auf Listenelemente über den Index	110
Listenverkettung	112
Der Operator in	112
Methoden	113
Die Listenmethoden reverse() und append()	113
Die Methode split()	114
Ein Wort aus der Liste auswählen	114
Die Grafik anzeigen	115
Die Funktionen list() und range()	116
Listen- und Stringslices	117
Das zu erratende Wort mit Leerstellen anzeigen	118

Die Rateversuche des Spielers abrufen	120
Die Stringmethoden lower() und upper()	121
Die while-Schleife verlassen	122
elif-Anweisungen	122
Die Eingabe eines gültigen Werts sicherstellen	123
Dem Spieler eine weitere Runde anbieten	124
Die Funktionen im Galgenmännchen-Programm	125
Die Hauptschleife des Spiels	126
Die Funktion displayBoard() aufrufen	126
Den Spieler raten lassen	126
Gehört der Buchstabe zu dem Wort?	127
Hat der Spieler gewonnen?	127
Fehlversuche handhaben	128
Hat der Spieler verloren?	128
Das Spiel beenden oder zurücksetzen	129
Zusammenfassung	130
9 Galgenmännchen: Erweiterungen	131
Mehr Rateversuche hinzufügen	132
Dictionarys	132
Die Größe eines Dictionarys mit len() bestimmen	133
Der Unterschied zwischen Dictionarys und Listen	134
Die Dictionary-Methoden keys() und values()	135
Dictionarys in Galgenmännchen	135
Zufällige Auswahl aus einer Liste	136
Elemente aus Listen entfernen	137
Mehrfachzuweisung	139
Die Wortkategorie ausgeben	140
Zusammenfassung	140
10 Tic-Tac-Toe	143
Ein Beispieldurchlauf von Tic-Tac-Toe	144
Der Quellcode von Tic-Tac-Toe	145
Das Programm entwerfen	149
Das Spielbrett in Form von Daten darstellen	150
Die Strategie der Spiel-KI	150
Das Modul random importieren	152
Das Spielbrett anzeigen	152

Den Spieler zwischen X und O wählen lassen	153
Den ersten Zug auswürfeln	154
Eine Markierung auf dem Spielbrett vornehmen	155
Listenverweise	155
Listenverweise in makeMove()	158
Hat der Spieler gewonnen?	159
Die Spielbrettdaten duplizieren	161
Ist ein Feld belegt?	161
Einen Zug vom Spieler abrufen	162
Kurzschlussauswertung	163
Einen Zug von einer Zugliste auswählen	165
Der Wert None	166
Die KI gestalten	167
Prüfung auf einen Gewinnzug des Computers	167
Prüfung auf einen Gewinnzug des Spielers	168
Ecken, Mitte und Seitenfelder prüfen	169
Ist das Brett voll?	169
Die Hauptschleife des Spiels	170
Den Spielerbuchstaben auswählen und den ersten Zug auslösen	170
Der Zug des Spielers	171
Der Zug des Computers	172
Eine weitere Spielrunde anbieten	173
Zusammenfassung	173
11 Bagels	175
Ein Beispieldurchlauf von Bagels	176
Der Quellcode von Bagels	177
Das Flussdiagramm für Bagels	179
Import von random() und Definition von getSecretNum()	179
Ziffern durcheinanderwürfeln	180
Reihenfolge von Listenelementen mit random.shuffle() ändern	180
Die Geheimzahl aus durcheinandergewürfelten Ziffern zusammenstellen	181
Erweiterte Zuweisungsoperatoren	181
Berechnungen für die Hinweise	182
Die Listenmethode sort()	183

Die Stringmethode <code>join()</code>	184
Besteht der String nur aus Ziffern?	185
Das Spiel starten	185
Stringinterpolation	186
Die Hauptschleife des Spiels	187
Die Vermutung des Spielers abrufen	188
Die Hinweise gewinnen	188
Auf Sieg oder Niederlage prüfen	188
Nach einer weiteren Runde fragen	189
Zusammenfassung	189
12 Kartesische Koordinaten	191
Raster und kartesische Koordinaten	192
Negative Zahlen	194
Das Koordinatensystem eines Computerbildschirms	196
Rechentricks	196
Trick 1: Ein Minuszeichen verschlingt ein Pluszeichen zu seiner Linken	197
Trick 2: Zweimal Minus gibt Plus	197
Trick 3: Zwei Zahlen in einer Addition können den Platz tauschen	197
Beträge und die Funktion <code>abs()</code>	198
Zusammenfassung	199
13 Sonar-Schatzsuche	201
Ein Beispieldurchlauf der Sonar-Schatzsuche	203
Der Quellcode für die Sonar-Schatzsuche	205
Entwurf des Programms	210
Die Module <code>random</code> , <code>sys</code> und <code>math</code> importieren	211
Ein neues Spielbrett erstellen	211
Das Spielbrett zeichnen	213
Die x-Koordinaten am oberen Rand des Spielbretts zeichnen	213
Das Meer zeichnen	214
Eine Meeresreihe ausgeben	215
Die x-Koordinaten am unteren Rand ausgeben	216
Die zufällig verteilen Schatztruhen erstellen	216
Die Gültigkeit eines Zugs bestimmen	217

Einen Zug auf dem Spielbrett machen	217
Die nächstgelegene Schatztruhe finden	218
Werte mit der Listenmethode <code>remove()</code> entfernen	220
Den Zug des Spielers abrufen	222
Die Spielanleitung ausgeben	224
Die Hauptschleife des Spiels	224
Den Spielzustand anzeigen	225
Den Zug des Spielers verarbeiten	226
Eine Schatztruhe finden	226
Hat der Spieler gewonnen?	227
Hat der Spieler verloren?	227
Das Programm mit der Funktion <code>sys.exit()</code> beenden	228
Zusammenfassung	228
14 Die Caesar-Chiffre	231
Kryptografie und Verschlüsselung	232
Funktionsweise der Caesar-Chiffre	233
Ein Beispieldurchlauf der Caesar-Chiffre	234
Der Quellcode für die Caesar-Chiffre	235
Die maximale Schlüssellänge festlegen	236
Auswahl zwischen Ver- und Entschlüsselung	237
Die Nachricht vom Benutzer abrufen	237
Den Schlüssel vom Benutzer abrufen	237
Die Nachricht verschlüsseln oder entschlüsseln	238
Strings mit der Methode <code>find()</code> finden	239
Die einzelnen Buchstaben ver- bzw. entschlüsseln	240
Das Programm starten	241
Brute-Force-Entschlüsselung	241
Den Brute-Force-Modus hinzufügen	242
Zusammenfassung	243
15 Reversegam	245
Die Spielregeln von Reversegam	246
Ein Beispieldurchlauf von Reversegam	249
Der Quellcode von Reversegam	251
Module importieren und Konstanten einrichten	257
Die Datenstruktur für das Spielbrett	257

Die Datenstruktur für das Spielbrett auf dem Bildschirm ausgeben . . .	258
Eine neue Datenstruktur für das Brett erstellen	259
Züge auf Gültigkeit prüfen	259
Die acht Richtungen prüfen	260
Prüfen, ob Steine gedreht werden müssen	262
Prüfen auf gültige Koordinaten	263
Eine Liste mit allen gültigen Zügen bekommen	263
Die Funktion bool() aufrufen	264
Die Punktzahl des Spielbretts ermitteln	265
Die Steinwahl des Spielers abfragen	266
Entscheiden, wer beginnt	266
Einen Spielstein auf dem Brett platzieren	267
Die Datenstruktur des Spielbretts kopieren	268
Bestimmen, was eine Ecke ist	268
Den Spielzug abfragen	268
Der Zug des Computers	270
Strategische Züge in Ecken	271
Eine Liste der besten Züge erstellen	271
Die Punkte auf dem Bildschirm ausgeben	273
Das Spiel starten	273
Ein Patt erkennen	274
Der Spieler ist am Zug	274
Der Spielzug des Computers	276
Die Spielschleife	277
Fragen, ob der Spieler erneut spielen möchte	278
Zusammenfassung	278
16 Reversegam-KI-Simulation	279
Der Computer spielt gegen sich selbst	280
Ein Beispieldurchlauf von Simulation 1	281
Der Quellcode für Simulation 1	281
Die Spielereingaben entfernen und einen Computerspieler hinzufügen	283
Den Computer mehrfach gegen sich selbst spielen lassen	284
Ein Beispieldurchlauf von Simulation 2	284
Quellcode für Simulation 2	284
Den Überblick über mehrere Spiele behalten	285

Die Funktionsaufrufe print() auskommentieren	286
Die KI mit Prozentsätzen bewerten	287
Die verschiedenen KI-Algorithmen vergleichen	289
Quellcode für Simulation 3	289
Wie die KI in Simulation 3 funktioniert	291
Die KI miteinander vergleichen	294
Zusammenfassung	296
17 Grafik einsetzen	299
pygame installieren	300
Hello World in pygame	301
Beispieldurchlauf des pygame Hello World	301
Quellcode für pygame Hello World	302
Das Modul pygame importieren	303
pygame initialisieren	304
Das pygame-Fenster einrichten	304
Tupel	305
Surface-Objekte	305
Farb-Variablen einrichten	306
Im pygame-Fenster Text ausgeben	307
Text mit Schriftarten formatieren	307
Ein Font-Objekt ausgeben	308
Die Textposition mit Rect-Attributen festlegen	309
Ein Surface-Objekt mit Farbe füllen	311
Die Zeichenfunktionen von pygame	311
Ein Polygon zeichnen	312
Eine Linie zeichnen	313
Einen Kreis zeichnen	313
Eine Ellipse zeichnen	314
Ein Rechteck zeichnen	314
Pixel färben	315
Die blit()-Methode für Surface-Objekte	316
Das Surface-Objekt auf dem Bildschirm ausgeben	316
Ereignisse und die Game-Schleife	316
Ereignis-Objekte erhalten	317
Das Programm beenden	318
Zusammenfassung	318

18 Grafiken animieren	319
Beispieldurchlauf des Animationsprogramms	320
Der Quellcode für das Animationsprogramm	320
Kästen bewegen und abprallen lassen	322
Die Konstanten einrichten	323
Konstanten für die Richtung	324
Konstanten für die Farbe	325
Datenstrukturen für die Kästen einrichten	325
Die Game-Schleife	326
Wenn der Spieler aufhören möchte	326
Die Kästen bewegen	327
Einen Kasten abprallen lassen	328
Kästen an ihren neuen Positionen im Fenster zeichnen	329
Das Fenster auf den Bildschirm zeichnen	330
Zusammenfassung	330
19 Kollisionserkennung	331
Beispieldurchlauf der Kollisionserkennung	332
Der Quellcode für die Kollisionserkennung	333
Die Module importieren	335
Das Programm mit einem Zeitgeber steuern	335
Das Fenster und die Datenstrukturen einrichten	336
Variablen zur Bewegungsverfolgung einrichten	337
Ereignisse verarbeiten	338
Das KEYDOWN-Ereignis	339
Das Ereignis KEYUP	342
Den Spieler teleportieren	342
Neue Nahrungsquadrate hinzufügen	343
Den Spieler im Fenster bewegen	344
Den Spieler im Fenster zeichnen	344
Auf Kollisionen prüfen	345
Die Nahrungsquadrate im Fenster zeichnen	346
Zusammenfassung	346

20 Toneffekte und Bilder	347
Bilder mit Sprites hinzufügen	348
Ton- und Bilddateien	349
Beispieldurchlauf des Sprites- und Sound-Programms	349
Der Code für das Sprites- und Sounds-Programm	350
Das Fenster und die Datenstruktur einrichten	353
Ein Sprite hinzufügen	353
Die Größe eines Sprites verändern	354
Die Musik und die Toneffekte einrichten	354
Sound-Dateien hinzufügen	354
Den Ton ein- und ausschalten	355
Den Spieler im Fenster zeichnen	355
Auf Kollisionen prüfen	356
Die Kirschen im Fenster zeichnen	356
Zusammenfassung	357
21 Dodger mit Ton und Grafik	359
Wiederholung: Die grundlegenden Datentypen in pygame	360
Ein Beispieldurchlauf von Dodger	361
Der Quellcode von Dodger	361
Die Module importieren	366
Die Konstanten einrichten	366
Funktionen definieren	367
Das Spiel beenden und unterbrechen	367
Kollisionen mit Gegnern feststellen	368
Text in das Fenster zeichnen	369
pygame initialisieren und das Fenster einrichten	370
Schrift-, Klang- und Bildobjekte einrichten	371
Den Startbildschirm anzeigen	372
Das Spiel starten	373
Die Spielschleife	374
Tastaturereignisse	375
Mausbewegungen	376
Neue Gegner hinzufügen	377
Die Figur des Spielers und die Gegner bewegen	378
Die Cheats	379

Gegner entfernen	380
Das Fenster zeichnen	380
Den Punktestand ausgeben	381
Die Spielerfigur und die Gegner zeichnen	381
Auf Kollisionen prüfen	382
Game Over	383
Das Spiel abwandeln	383
Zusammenfassung	384
Stichwortverzeichnis	385

Für Caro

Der Autor

Al Sweigart ist Softwareentwickler, Fachbuchautor und ein echter Froud, der weiß, wo sein Handtuch ist. Er hat mehrere Programmierbücher für Einsteiger geschrieben, darunter *Routineaufgaben mit Python automatisieren* und *Cooler Spiele mit Scratch*, die ebenfalls beim dpunkt.verlag erschienen sind. Auf seiner Website <https://inventwithpython.com/> stehen seine Bücher unter einer Creative-Commons-Lizenz auch kostenlos zur Verfügung.

Die Fachgutachterin der amerikanischen Ausgabe

Ari Lacenski ist Entwicklerin für Android-Anwendungen und Python-Software. Sie lebt in Bay Area, wo sie auf <http://gradlewhy.ghost.io/> über Android-Programmierung schreibt und als Mentorin für Women Who Code dient.

Danksagung



Ohne die herausragende Arbeit des Teams von No Starch Press wäre dieses Buch nicht möglich gewesen. Ich danke meinem Herausgeber Bill Pollock und meinen Lektoren Laurel Chun, Jan Cash und Tyler Ortman für ihre unglaubliche Hilfe während des gesamten Vorgangs, meiner Fachgutachterin Ari Lacenski für ihre gründliche Durchsicht und Josh Ellingson für ein weiteres hervorragendes Titelbild.

Einleitung



Die ersten Videospiele, die ich im Kindesalter spielte, machten mich süchtig. Ich wollte sie nicht nur spielen, ich wollte selbst so etwas gestalten. Damals fand ich ein Buch, das mir zeigte, wie ich eigene erste Programme und Spiele schreiben konnte. Das machte Spaß und war ganz einfach. Die ersten Spiele, die ich entwickelte, waren ähnlich wie diejenigen in diesem Buch. Sie waren nicht so ausgefeilt wie die Nintendo-Spiele, die meine Eltern für mich kauften, aber sie waren mein eigenes Werk.

Jetzt, als Erwachsener, habe ich immer noch Spaß am Programmieren und werde sogar dafür bezahlt. Doch auch wenn Sie nicht vorhaben, Programmierer zu werden, ist Programmierung eine nützliche und unterhaltsame Fähigkeit. Sie schulen sich dabei, logisch zu denken, planvoll vorzugehen und ihre Ideen kritisch zu betrachten, wenn Sie Fehler in Ihrem Code gefunden haben.

Viele Programmierbücher für Einsteiger fallen in eine von zwei verschiedenen Kategorien. Die erste Art von Büchern konzentriert sich so sehr auf »Spielentwicklungssoftware« oder stark vereinfachende Sprachen, dass man das, was da gelehrt

wird, gar nicht mehr als »Programmieren« bezeichnen kann. Die zweite Kategorie umfasst Bücher, die Programmierung in der Art eines Mathebuchs abhandeln: lauter Lehrsätze und Begriffe, aber nur wenige praktische Anwendungen für die Leser. Das vorliegende Buch verfolgt einen anderen Ansatz und bringt Ihnen das Programmieren am Beispiel von Videospielen bei. Ich stelle Ihnen den Quellcode der Spiele jeweils zu Anfang vor und erkläre dann die Programmierprinzipien anhand dieser Beispiele. Diese Vorgehensweise war für mich selbst von entscheidender Bedeutung, als ich Programmieren lernte. Je mehr ich darüber erfuhr, wie die Programme anderer Leute funktionierten, umso mehr Ideen bekam ich für meine eigenen.

Sie brauchen nicht mehr als einen Computer, den Python-Interpreter (eine kostenlose Software) und dieses Buch. Wenn Sie gelernt haben, wie Sie die Spiele in diesem programmieren, können Sie auch Ihre eigenen entwickeln.

Computer sind unglaubliche Maschinen. Zu lernen, wie man sie programmiert, ist gar nicht so schwer, wie die meisten Leute glauben. Ein *Computerprogramm* ist eine Folge von Anweisungen, die ein Computer verstehen kann, ähnlich wie ein Buch eine Folge von Sätzen ist, die der Leser verstehen kann. Um einen Computer anzuweisen, etwas zu tun, müssen Sie ein Programm in einer Sprache schreiben, die er versteht. In diesem Buch lernen Sie die Programmiersprache Python kennen. Es gibt noch viele andere Programmiersprachen, die Sie lernen können, z. B. BASIC, Java, JavaScript, PHP oder C++.

Als Kind habe ich BASIC gelernt, aber neuere Programmiersprachen wie Python lassen sich noch einfacher lernen. Python wird auch von professionellen Programmierern sowohl bei der Arbeit als auch für Hobbyprojekte verwendet. Installation und Verwendung sind außerdem völlig kostenlos. Sie brauchen lediglich eine Internetverbindung, um die Sprache herunterzuladen.

Da Videospiele nichts anderes sind als Computerprogramme, bestehen sie ebenfalls aus Anweisungen. Im Vergleich zu den Spielen für die Xbox, die PlayStation oder den Nintendo wirken diejenigen, die Sie in diesem Buch erstellen werden, ziemlich simpel. Sie weisen keine anspruchsvolle Grafik auf, sondern sind absichtlich einfach gehalten, sodass Sie sich drauf konzentrieren können, Programmieren zu lernen. Außerdem müssen Spiele nicht kompliziert sein, um Spaß zu machen.

Zielgruppe

Programmieren an sich ist nicht schwer. Dagegen kann es durchaus schwer sein, Anleitungen zu finden, die einem sagen, wie man interessante Dinge programmiert. Andere Computerbücher behandeln viele Themen, die Neulinge gar nicht brau-

chen. Dieses Buch zeigt Ihnen, wie Sie Ihre eigenen Spiele programmieren. Damit erwerben Sie eine nützliche Fähigkeit und haben außerdem unterhaltsame Spiele, die Sie stolz vorstellen können. Dieses Buch richtet sich an folgende Leser:

- Anfänger ohne jegliche Vorkenntnisse, die sich selbst das Programmieren beibringen möchten
- Kinder und Jugendliche, die durch das Schreiben von Spielen programmieren lernen möchten
- Erwachsene und Lehrer, die anderen Programmieren beibringen möchten
- Jegliche Personen, ob jung oder alt, die anhand einer professionellen Programmiersprache programmieren lernen wollen

Der Aufbau dieses Buches

In den meisten Kapiteln dieses Buches wird jeweils ein neues Spielprojekt vorgestellt und erläutert. Einige wenige Kapitel sind zusätzlichen interessanten Themen gewidmet, z. B. dem Debugging. Neue Programmierprinzipien werden jeweils dann vorgestellt, wenn sie in den Spielen vorkommen. Die Kapitel sollten in der vorliegenden Reihenfolge gelesen werden. Die folgende kurze Aufstellung zeigt, was Sie in den einzelnen Kapiteln erwartet:

Kapitel 1: Die interaktive Shell erklärt, wie Sie die interaktive Shell von Python verwenden können, um Code zeilenweise auszuprobieren.

Kapitel 2: Programme schreiben erklärt, wie Sie im Dateieditor von Python vollständige Programme schreiben.

In **Kapitel 3: Zahlen raten** programmieren Sie das erste Spiel in diesem Buch. Dabei muss der Spieler eine Geheimzahl raten und erhält jeweils den Hinweis, ob seine Vermutung zu hoch oder zu niedrig war.

In **Kapitel 4: Ein Kalauerprogramm** schreiben Sie ein einfaches Programm, das dem Benutzer mehrere Flachwitze erzählt.

In **Kapitel 5: Im Reich der Drachen** programmieren Sie ein Ratespiel, in dem der Spieler zwischen zwei Höhlen wählen muss: In einer der beiden wohnt ein freundlicher Drache, in der anderen ein hungriger.

Kapitel 6: Der Debugger erklärt, wie Sie Probleme in Ihrem Code mithilfe des Debuggers lösen.

Kapitel 7: Galgenmännchen: Entwurf mit einem Flussdiagramm beschreibt, wie Sie Flussdiagramme zur Planung anspruchsvollerer Programme einsetzen, hier für das in den folgenden Kapiteln beschriebene Galgenmännchen-Spiel.

In **Kapitel 8: Galgenmännchen: Der Code** schreiben Sie das Galgenmännchen-Spiel anhand des Flussdiagramms aus Kapitel 7.

In **Kapitel 9: Galgenmännchen: Erweiterungen** erweitern Sie das Galgenmännchen-Spiel, indem Sie den Python-Datentyp der Dictionaries verwenden.

In **Kapitel 10: Tic-Tac-Toe** lernen Sie, wie Sie ein Tic-Tac-Toe-Spiel schreiben, bei dem eine künstliche Intelligenz gegen den menschlichen Benutzer antritt.

In **Kapitel 11: Bagels** schreiben Sie das Denksportspiel Bagels, in dem der Spieler anhand von Hinweisen eine Geheimzahl erraten muss.

Kapitel 12: Kartesische Koordinaten erklärt das kartesische Koordinatensystem, das Sie in nachfolgenden Spielen verwenden werden.

In **Kapitel 13: Sonar-Schatzsuche** schreiben Sie ein Spiel, in dem Sie das Meer nach verlorenen Schatztruhen absuchen.

In **Kapitel 14: Die Caesar-Chiffre** schreiben Sie ein einfaches Verschlüsselungsprogramm, mit dem Sie Geheimnachrichten verschlüsseln und entschlüsseln können.

In **Kapitel 15: Reversegam** programmieren Sie ein anspruchsvolles Reversi-ähnliches Spiel mit einer fast unschlagbaren künstlichen Intelligenz als Gegner.

Kapitel 16: Reversegam: AI-Simulation erweitert das Reversegam-Spiel aus Kapitel 15, sodass nun verschiedene KIs gegeneinander antreten.

Kapitel 17: Grafik einsetzen führt das Python-Modul pygame ein und zeigt, wie Sie damit 2-D-Grafiken zeichnen können.

Kapitel 18: Grafiken animieren zeigt, wie Sie mit pygame Grafiken animieren.

In **Kapitel 19: Kollisionserkennung** lernen Sie, wie Sie in 2-D-Spielen Zusammenstöße zwischen Objekten erkennen können.

In **Kapitel 20: Sounds und Bilder verwenden** erweitern Sie Ihre einfachen pygame-Spiele um Klänge und Bilder.

Kapitel 21: Dodger mit Ton und Grafik wendet den Stoff aus den Kapiteln 17 bis 20 für ein animiertes Spiel namens Dodger an.

Die Codebeispiele

Die meisten Kapitel dieses Buches beginnen mit einem Beispieldurchlauf des Programms, das in dem Kapitel vorgestellt wird. Dieser Durchlauf zeigt, wie das Programm aussieht, wenn Sie es ausführen. Die Eingaben des Benutzers sind durch Fettdruck gekennzeichnet.

Ich rate Ihnen, den Code der einzelnen Programme jeweils selbst in den IDLE-Dateieditor einzugeben, anstatt ihn einfach nur herunterzuladen und zu kopieren. Wenn Sie sich die Zeit nehmen, den Code abzutippen, werden Sie mehr davon im Kopf behalten.

Zeilennummern und Einrückungen

Beim Abtippen des Quellcodes aus dem Buch dürfen Sie die Nummern zu Beginn der einzelnen Zeilen *nicht* mit eingeben! Nehmen wir an, Sie sehen die folgende Codezeile:

```
9. number = random.randint(1, 20)
```

Hier dürfen Sie weder 9. noch das darauffolgende Leerzeichen eingeben, sondern nur Folgendes:

```
number = random.randint(1, 20)
```

Die Zeilennummern stehen hier nur, um in den Erklärungen auf die einzelnen Zeilen verweisen zu können. Sie gehören nicht zum Quellcode des Programms.

Abgesehen von diesen Zeilennummern jedoch müssen Sie den Code genau so eingeben, wie er in dem Buch erscheint. Einige Zeilen sind mit vier, acht oder mehr Leerzeichen eingerückt. Diese Leerzeichen zu Anfang einer Zeile wirken sich darauf aus, wie Python die Anweisungen deutet, weshalb es sehr wichtig ist, sie einzuschließen.

Betrachten Sie dazu das folgende Beispiel. Die Leerzeichen sind durch dicke Punkte (•) dargestellt, sodass Sie sie sehen können:

```
while guesses < 10:  
    ••••if number == 42:  
        ••••••••print('Hello')
```

Die erste Zeile ist nicht eingerückt, die zweite um vier und die dritte um acht Leerzeichen. In den tatsächlichen Beispielen in dem Buch sind die Leerzeichen zwar nicht durch solche Punkte gekennzeichnet, aber da alle Zeichen in IDLE die gleiche Breite aufweisen, können Sie die Anzahl der Leerzeichen einfach dadurch

Wenn Sie mit Windows arbeiten, laden Sie den MSI-Installer für Windows x86-64 von <https://www.python.org/downloads/release/python-344/> herunter und doppelklicken darauf. Möglicherweise müssen Sie das Administratorkennwort für Ihren Computer eingeben.

Folgen Sie den Anweisungen, die der Installer anzeigt, um Python zu installieren. Die Vorgehensweise ist wie folgt:

1. Wählen Sie *Install for All Users* und klicken Sie auf *Weiter*.
2. Bestätigen Sie die Installation im Ordner `C:\Python34`, indem Sie auf *Weiter* klicken.
3. Klicken Sie auf *Weiter*, um den Abschnitt *Customize Python* zu überspringen.

Wenn Sie mit OS X arbeiten, laden Sie den MSI-Installer für Mac OS X 64 Bit/32 Bit von <https://www.python.org/downloads/release/python-344/> herunter und doppelklicken darauf. Möglicherweise müssen Sie das Administratorkennwort für Ihren Computer eingeben.

Folgen Sie den Anweisungen, die der Installer anzeigt, um Python zu installieren. Die Vorgehensweise ist wie folgt:

1. Wenn Sie eine Warnung der Art »Python.mpkg kann nicht geöffnet werden, da es von einem nicht verifizierten Entwickler stammt« erhalten, klicken Sie bei gedrückter `ctrl`-Taste auf die Datei *Python.mpkg* und wählen *Öffnen* aus dem dann angezeigten Menü. Möglicherweise müssen Sie das Administratorpasswort für Ihren Computer eingeben.
2. Klicken Sie im Willkommensabschnitt auf *Continue* und dann auf *Agree*, um die Lizenzverarbeitung zu akzeptieren.
3. Wählen Sie *Macintosh HD* (bzw. den Namen Ihrer Festplatte) aus und klicken Sie auf *Install*.

Wenn Sie Ubuntu verwenden, könne Sie Python wie folgt aus dem Ubuntu Software Center heraus installieren:

1. Öffnen Sie das Ubuntu Software Center.
2. Geben Sie in das Suchfeld oben rechts in dem Fenster **Python** ein.
3. Wählen Sie *IDLE (Python 3.4 GUI 64 bit)* aus.
4. Klicken Sie auf *Install*. Möglicherweise müssen Sie das Administratorpasswort Computer eingeben, um die Installation abzuschließen.

Sollten die angegebenen Vorgehensweisen nicht funktionieren, finden Sie alternative Installationsanweisungen für Python 3.4 auf <https://www.nostarch.com/inventwithpython/>.

IDLE starten

IDLE steht für Interactive Development Environment, also »interaktive Entwicklungsumgebung«. Es handelt sich dabei um eine Art Textverarbeitungssystem zum Schreiben von Python-Programmen. Je nach Betriebssystem müssen Sie zum Starten dieser Software jeweils unterschiedlich vorgehen:

- Unter Windows klicken Sie auf das Startmenü unten links, geben **IDLE** ein und wählen *IDLE (Python GUI)*.
- Unter OS X öffnen Sie den Finder und klicken auf *Programme*. Doppelklicken Sie anschließend auf *Python 3.x* und dann auf das IDLE-Symbol.
- Unter Ubuntu und anderen Linux-Distributionen öffnen Sie ein Terminal-Fenster und geben `idle3` ein. Es kann auch sein, dass Sie auf *Applications* am oberen Bildschirmrand und dann auf *Programming* und *IDLE3* klicken können.

Das Fenster, das Sie beim Start von IDLE sehen, ist die *interaktive Shell* (siehe Abbildung). Wenn Sie hinter der Eingabeaufforderung `>>>` Python-Anweisungen in die Shell eingeben, führt Python sie aus. Anschließend erscheint wieder die Eingabeaufforderung `>>>`, um auf Ihre nächste Anweisung zu warten.



Abb. E-1 Die interaktive Shell von IDLE

Online Hilfe finden

Die Quellcodedateien und andere Materialien zu diesem Buch finden Sie auf <https://www.nostarch.com/inventwithpython/>. Wenn Sie Fragen zur Programmierung im Zusammenhang mit diesem Buch haben, können Sie auf <https://reddit.com/r/inventwithpython/> nachschauen oder sich über al@inventwithpython.com an mich wenden (in englischer Sprache).

Bevor Sie Ihre Fragen stellen, sollten Sie aber Folgendes unternehmen:

- Wenn Sie bei einem Programm aus diesem Buch, das Sie abgetippt haben, Fehlermeldungen erhalten, prüfen Sie, ob sich Tippfehler eingeschlichen haben. Das können Sie mit dem Vergleichswerkzeug (Diff Tool) machen, das auf <https://www.nostarch.com/inventwithpython#diff> zur Verfügung steht. Kopieren Sie Ihren Code in das Diff-Tool. Sollte es Unterschiede zwischen Ihrer Version des Codes und dem im Buch geben, werden sie Ihnen angezeigt.
- Schauen Sie im Web nach, ob jemand bereits die gleiche Frage gestellt (und eine Antwort darauf erhalten) hat.

Je besser Sie Ihre Programmierfragen formulieren, umso besser können andere Ihnen helfen. Gehen Sie daher in solchen Fällen immer wie folgt vor:

- Erklären Sie, was Sie zu tun versuchen, wenn der Fehler auftritt. Dadurch können potenzielle Helfer erkennen, ob Sie vielleicht völlig vom Wege abgekommen sind.
- Kopieren Sie die gesamte Fehlermeldung und Ihren Code.
- Geben Sie an, welches Betriebssystem und welche Version Sie benutzen.
- Beschreiben Sie, was Sie bereits getan haben, um das Problem zu lösen. Dadurch versichern Sie anderen, dass Sie schon selbst Anstrengungen unternommen haben, um der Sache auf den Grund zu gehen.
- Bleiben Sie höflich. Fordern Sie keine Hilfe und drängen Sie Ihre Helfer nicht, schnell zu antworten.

Nun sind Sie bereit zu lernen, wie Sie Ihre eigenen Computerspiele programmieren!

1

Die interaktive Shell



Bevor Sie Spiele entwickeln können, müssen Sie erst einige elementare Programmierprinzipien lernen. In diesem Kapitel erfahren Sie daher, wie Sie die interaktive Shell von Python nutzen und grundlegende Rechenoperationen durchführen.

Themen in diesem Kapitel

- Operatoren
- Integer und Fließkommazahlen
- Werte
- Ausdrücke
- Syntaxfehler
- Werte in Variablen speichern

Einfaches Rechnen

Starten Sie IDLE nach der Anleitung im Abschnitt »IDLE starten« in der Einleitung. Als Erstes wollen wir Python nutzen, um einige einfache Rechenaufgaben zu lösen. Die interaktive Shell lässt sich wie ein Taschenrechner benutzen. Geben Sie $2 + 2$ an der Eingabeaufforderung `>>>` der Shell ein und drücken Sie die Eingabetaste (die je nach Tastatur `Enter` oder `Return` heißen kann). Abbildung 1–1 zeigt, wie die Rechenaufgabe in der interaktiven Shell angezeigt wird. Wie Sie sehen, antwortet die Shell mit der Zahl 4.

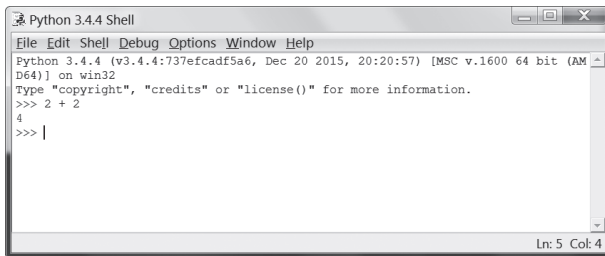


Abb. 1–1 Eingabe von $2 + 2$ in die aktive Shell

Diese Rechenaufgabe ist ein einfacher Programmbefehl. Das Pluszeichen weist den Computer an, die beiden Zahlen 2 und 2 zu addieren. Der Computer erledigt diese Aufgabe und gibt in der folgenden Zeile die Zahl 4 aus. Tabelle 1–1 nennt weitere mathematische Symbole, die in Python zur Verfügung stehen.

Operator	Operation
+	Addition
-	Subtraktion
*	Multiplikation
/	Division

Tab. 1–1 Mathematische Operatoren

Das Minuszeichen subtrahiert Zahlen, das Sternchen multipliziert sie und der Schrägstrich teilt sie. Bei der Verwendung auf diese Weise werden $+$, $-$, $*$ und $/$ als *Operatoren* bezeichnet. Sie weisen Python an, was es mit den Zahlen auf beiden Seiten tun soll.

Integer und Fließkommazahlen

Integer sind ganze Zahlen wie 4, 99 oder 0. Bei *Fließkommazahlen* dagegen handelt es sich um Dezimalbrüche wie 3.5, 42.1 oder 5.0 (wobei in Programmiersprachen statt des im Deutschen üblichen Kommas ein Punkt verwendet wird). In Python ist 5 ein Integer und 5.0 eine Fließkommazahl.

Diese Zahlen werden als *Werte* bezeichnet. (Wir werden später noch andere Arten von Werten kennenlernen.) In der Rechenaufgabe, die Sie in die Shell eingegeben haben, sind 2 und 2 Integerwerte.

Ausdrücke

Die Rechenaufgabe $2 + 2$ ist ein Beispiel für einen *Ausdruck*. Wie Abbildung 1–2 zeigt, bestehen Ausdrücke aus Werten (den Zahlen), die durch Operatoren (die Rechenzeichen) verbunden sind, um daraus einen neuen Wert zu bilden, der im Code verwendet werden kann. Computer können Millionen von Ausdrücken in Sekundenschnelle berechnen.

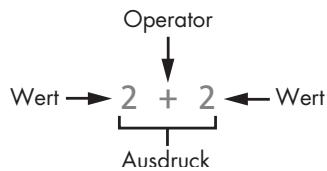


Abb. 1–2 Ausdrücke bestehen aus Werten und Operatoren

Versuchen Sie, einige Ausdrücke in die interaktive Shell einzugeben. Drücken Sie nach jedem dieser Ausdrücke :

```
>>> 2+2+2+2
10
>>> 8*6
48
>>> 10-5+6
11
>>> 2 +      2
4
```

Diese Ausdrücke sehen alle wie ganz normale Rechenaufgaben aus. Beachten Sie aber die Leerzeichen in dem Beispiel $2 + \quad 2$. In Python können Sie beliebig viele Leerzeichen zwischen Werte und Operatoren schreiben. Eingaben in die aktive Shell müssen jedoch immer am Anfang der Zeile beginnen (ohne Leerzeichen davor).

Ausdrücke auswerten

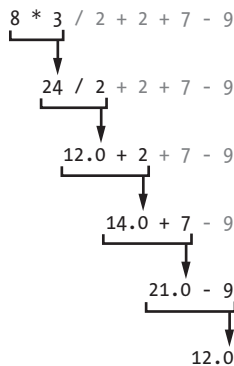
Wenn ein Computer den Ausdruck $10 + 5$ verarbeitet und den Wert 15 zurückgibt, so spricht man davon, dass er den Ausdruck *auswertet*. Das bedeutet, dass der Ausdruck auf einen einzigen Wert reduziert wird. Wenn Sie eine Rechenaufgabe lösen, reduzieren Sie sie damit auch auf einen einzigen Wert, nämlich das Ergebnis. Die Ausdrücke $10 + 5$ und $10 + 3 + 2$ werden beide zu 15 ausgewertet.

Wenn Python einen Ausdruck auswertet, führt es die Operationen in der gleichen Reihenfolge aus, wie Sie es beim Rechnen tun. Dabei gelten die folgenden Regeln:

- Teile des Ausdrucks, die in Klammern stehen, werden zuerst ausgewertet.
- Multiplikation und Division erfolgen vor Addition und Subtraktion.
- Die Auswertung erfolgt von links nach rechts.

Der Ausdruck $1 + 2 * 3 + 4$ ergibt daher 11 und nicht 13, da zuerst $2 * 3$ berechnet wird. Lautete die Aufgabe stattdessen $(1 + 2) * (3 + 4)$, würde sich 21 ergeben, da $(1 + 2)$ und $(3 + 4)$ in Klammern stehen und daher vor der Multiplikation ausgewertet werden.

Ausdrücke können eine beliebige Größe aufweisen, werden aber stets zu einem einzigen Wert ausgewertet. Auch einzelne Werte können Ausdrücke sein. So wird beispielsweise der Ausdruck 15 zu dem Wert 15 ausgewertet. Der Ausdruck $8 * 3 / 2 + 2 + 7 - 9$ ergibt 21. Dabei erfolgt die Berechnung in den folgenden Schritten:



Der Computer führt zwar alle diese Schritte durch, doch in der interaktiven Shell sehen Sie nichts davon, sondern nur das Ergebnis:

```
>>> 8 * 3 / 2 + 2 + 7 - 9
12.0
```

Ausdrücke mit dem Divisionsoperator `/` werden stets zu einer Fließkommazahl ausgewertet, hier beispielsweise $24 / 2$ zu 12.0. Mathematische Operationen, an

denen auch nur ein Fließkommawert beteiligt ist, ergeben wiederum Fließkommawerte. In unserem Beispiel wird deshalb $12.0 + 2$ zu 14.0 ausgewertet.

Syntaxfehler

Wenn Sie `5 +` in die interaktive Shell eingeben, erhalten Sie die folgende Fehlermeldung:

```
>>> 5 +
SyntaxError: invalid syntax
```

Das liegt daran, dass `5 +` kein Ausdruck ist. Ausdrücke bestehen aus Werten, die durch Operatoren verbunden sind, und der Operator `+` erwartet einen Wert davor *und* dahinter. Wenn ein erwarteter Wert fehlt, wird eine Fehlermeldung angezeigt.

`SyntaxError` bedeutet, dass Python den Befehl nicht versteht, da Sie ihn falsch eingegeben haben. Bei der Programmierung geht es nicht einfach darum, dem Computer Befehle zu erteilen; man muss auch schon genau wissen, wie man diese Befehle auf die richtige Weise gibt!

Machen Sie sich jedoch keine Sorgen über solche Fehler. Durch diese Art falscher Angaben können Sie den Computer nicht beschädigen. Geben Sie den Befehl einfach an der nächsten Eingabeaufforderung (`>>>`) korrekt in die Shell ein.

Werte in Variablen speichern

Wenn ein Ausdruck zu einem Wert ausgewertet wird, können Sie diesen Wert später nutzen, wenn Sie ihn in einer *Variablen* speichern. Eine solche Variable können Sie sich als eine Schachtel vorstellen, in der Sie den Wert ablegen.

Eine *Zuweisungsanweisung* speichert einen Wert in einer Variablen. Geben Sie dazu den Namen der Variablen ein, dann ein Gleichheitszeichen – den sogenannten *Zuweisungsoperator* – und schließlich den Wert, den Sie speichern wollen:

```
>>> spam = 15
>>>
```

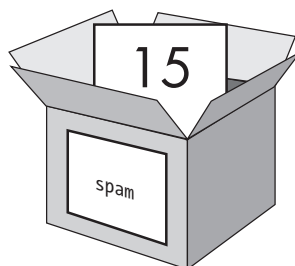


Abb. 1-3 Variablen fungieren wie Schachteln, in denen Sie Werte aufbewahren können.

Wenn Sie `Enter` drücken, scheint nichts zu passieren. In Python wissen Sie, dass ein Befehl erfolgreich ausgeführt wurde, wenn keine Fehlermeldung erscheint. Es wird einfach wieder die Eingabeaufforderung `>>>` angezeigt, sodass Sie den nächsten Befehl eingeben können.

Im Gegensatz zu Ausdrücken werden *Anweisungen* nicht zu einem Wert ausgewertet. Das ist der Grund, aus dem in der nächsten Zeile der interaktiven Shell nach `spam = 15` nichts mehr angezeigt wird. Wenn Ihnen nicht ganz klar ist, ob es sich bei einer Ihrer Eingaben um einen Ausdruck oder eine Anweisung handelt, so denken Sie daran, dass Ausdrücke immer zu einem einzelnen Wert ausgewertet werden. Alles andere sind Anweisungen.

In Variablen können Sie Werte festhalten, aber keine Ausdrücke. Betrachten Sie zum Beispiel die Ausdrücke in den Anweisungen `spam = 10 + 5` und `spam = 10 + 7 - 2`. Beide Ausdrücke werden zu 15 ausgewertet, und beide Anweisungen speichern den Wert 15 in der Variablen `spam`.

Die Beispielvariablen in diesem Kapitel tragen nichtssagende Namen wie `spam`, `eggs` und `bacon`, doch für die Variablen in Ihren Programmen sollten Sie Namen verwenden, die die enthaltenen Daten beschreiben. Stellen Sie sich vor, Sie würden bei einem Umzug alle Kartons einfach mit *Sachen* beschriften. Sie würden nichts mehr wiederfinden!

Python erstellt die Variable, wenn sie zum ersten Mal in einer Zuweisungsanweisung verwendet wird. Um zu prüfen, ob sich ein Wert in der Variablen befindet, geben Sie den Variablennamen in die interaktive Shell ein:

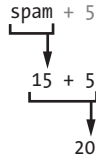
```
>>> spam = 15
>>> spam
15
```

Der Ausdruck `spam` wird zu dem Wert ausgewertet, der sich in der Variablen `spam` befindet, also 15.

Variablen können Sie auch in Ausdrücken verwenden. Geben Sie Folgendes in die interaktive Shell ein:

```
>>> spam = 15
>>> spam + 5
20
```

Hier haben Sie zunächst den Wert der Variablen `spam` auf 15 gesetzt, sodass die Eingabe `spam + 5` dem Ausdruck `15 + 5` entspricht. Die Auswertung von `spam + 5` läuft wie folgt ab:



Es ist nicht möglich, eine Variable schon vor der Zuweisungsanweisung zu verwenden, mit der sie erstellt wird. Sollten Sie das versuchen, so gibt Python die Fehlermeldung `NameError` aus, da eine Variable des angegebenen Namens noch nicht existiert. Der gleiche Fehler ergibt sich auch, wenn Sie den Variablennamen falsch schreiben:

```
>>> spam = 15
>>> spma
Traceback (most recent call last):
  File "<pysHELL#8>", line 1, in <module>
    spma
NameError: name 'spma' is not defined
```

Der Fehler tritt auf, da es nur eine Variable namens `spam` gibt, aber nicht `spma`.

Den Wert, der in einer Variablen gespeichert ist, können Sie mit einer weiteren Zuweisungsanweisung ändern. Probieren Sie das folgende Beispiel in der interaktiven Shell aus:

```
>>> spam = 15
>>> spam + 5
20
>>> spam = 3
>>> spam + 5
8
```

Wenn Sie zum ersten Mal `spam + 5` eingeben, wird der Ausdruck zu 20 ausgewertet, da in `spam` der Wert 15 gespeichert ist. Dann allerdings geben Sie `spam = 3` ein, wodurch der Wert 15 in der »Schachtel« der Variablen gegen den Wert 3 ausgetauscht oder mit ihm *überschrieben* wird, da eine Variable immer nur einen Wert auf einmal enthalten kann. Der Wert von `spam` lautet jetzt 3, weshalb der Ausdruck `spam + 5` bei der erneuten Eingabe zu 8 ausgewertet wird. Beim Überschreiben wird praktisch der vorherige Wert aus der Variablenschachtel herausgenommen und der neue hineingelegt, wie Abbildung 1–4 illustriert.

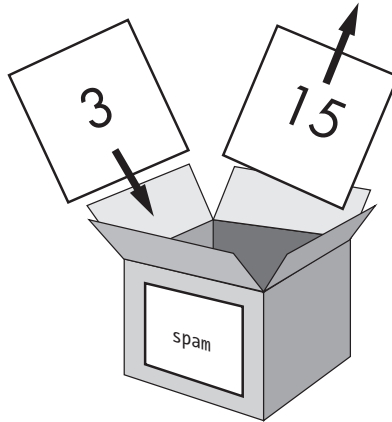


Abb. 1–4 Der Wert 15 in spam wird durch den Wert 3 überschrieben.

Sie können den Wert in spam sogar dazu verwenden, spam selbst einen neuen Wert zuzuweisen:

```
>>> spam = 15
>>> spam = spam + 5
>>> spam
20
```

Die Zuweisungsanweisung `spam = spam + 5` bedeutet: »Der neue Wert der Variablen spam ist der alte Wert von spam plus 5.« Um den Wert in spam wiederholt um 5 zu erhöhen, geben Sie Folgendes in die interaktive Shell ein:

```
>>> spam = 15
>>> spam = spam + 5
>>> spam = spam + 5
>>> spam = spam + 5
>>> spam
30
```

In der ersten Anweisung dieses Beispiels speichern Sie den Wert 15 in spam. In der nächsten addieren Sie 5 zu dem Wert von spam und weisen den neuen Wert `spam + 5`, der 20 ergibt, der Variablen spam zu. Wenn Sie das dreimal tun, enthält spam schließlich den Wert 30.

Bis jetzt haben wir uns nur mit einer einzigen Variablen beschäftigt, doch Sie können in Ihren Programmen so viele Variablen verwenden, wie Sie benötigen. Als Beispiel wollen wir den beiden Variablen `eggs` und `bacon` unterschiedliche Werte zuweisen:

```
>>> bacon = 10
>>> eggs = 15
```

Die Variable `bacon` enthält jetzt den Wert 10, während sich in `eggs` der Wert 15 befindet. Jede Variable stellt eine eigene Schachtel mit einem eigenen Wert dar, wie Sie in Abbildung 1–5 sehen.

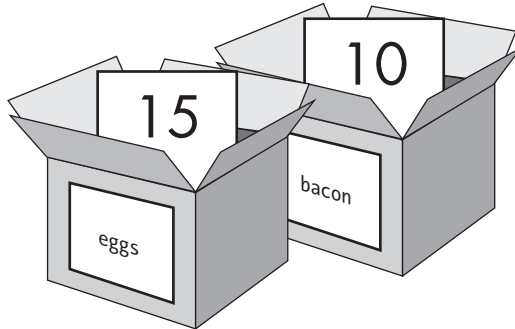


Abb. 1–5 Unterschiedliche Werte in den Variablen `bacon` und `eggs`

Geben Sie jetzt `spam = bacon + eggs` in die interaktive Shell ein und prüfen Sie dann den Wert von `spam`:

```
>>> bacon = 10
>>> eggs = 15
>>> spam = bacon + eggs
>>> spam
25
```

Der Wert in `spam` beträgt jetzt 25. Bei der Addition von `bacon` und `eggs` werden deren Werte addiert, also 10 und 15. Da Variablen Werte enthalten und keine Ausdrücke, wurde `spam` der Wert 25 und nicht der Ausdruck `bacon + eggs` zugewiesen. Nachdem Sie mit der Anweisung `spam = bacon + eggs` den Wert 25 zu `spam` zugewiesen haben, haben anschließende Änderungen von `bacon` oder `eggs` keinen Einfluss auf `spam`.

Zusammenfassung

In diesem Kapitel haben Sie die Grundlagen zum Schreiben von Python-Programmbefehlen gelernt. Da Computer nicht denken können und nur bestimmte Befehle verstehen, müssen Sie Python genau sagen, was es tun soll.

Ausdrücke sind Werte (wie 2 oder 5), die durch Operatoren (wie + und -) verbunden sind. Python kann Ausdrücke auswerten, wobei sie auf einen einzigen Wert reduziert werden. Werte können Sie auch in Variablen speichern, sodass das Programm sich diese Werte merkt und später verwenden kann.