Python 3

Programmieren für Einsteiger

Michael Bonacina

2. Auflage: Oktober 2018

© dieser Ausgabe 2019 by BMU Media GmbH

ISBN: 978-3-96645-007-2

Herausgegeben durch:
BMU Media GmbH
Hornissenweg 4
84034 Landshut

Python 3

Inhaltsverzeichnis

1. **Einleitung** 9

1.1 Python: eine einfach zu erlernende Programmiersprache 9
1.2 Die Ziele bei der Entwicklung von Python 11
1.3 Die Entwicklungsgeschichte 12
1.4 Eine interpretierte Programmiersprache 13

2. **Die Vorbereitungsmaßnahmen** 16

2.1 Den Python Interpreter installieren 17
2.2 Einen Texteditor für die Erstellung des Codes 20

3. **Interaktive Interpretation: ideal für den ersten Kontakt mit Python** 23

3.1 Den Python-Prompt aufrufen 24
3.2 Erste Befehle ausprobieren 25

4. **Ein Python-Programm in eine eigene Datei schreiben** 29

4.1 Ein Programm für eine einfache Textausgabe 29
4.2 Die Ausführung im Python-Interpreter 31
4.3 Kommentare: hilfreich für das Verständnis des Programms 33
4.4 Übungsaufgabe: eigene Inhalte zum Programm hinzufügen 35

5. **Variablen: unverzichtbar für die Programmierung mit Python** 40

5.1 Die Aufgabe von Variablen in einem Computerprogramm 40
5.2 Variablen in Python verwenden 42

5.3 Den Wert einer Variablen durch eine Eingabe des Nutzers festlegen ... 45
5.4 Dynamische Typisierung: viele Freiheiten bei der Nutzung von Variablen ... 49
5.5 Datentypen sind auch in Python von Bedeutung ... 51
5.6 Übungsaufgabe: Mit Variablen arbeiten ... 55

6. Datenstrukturen in Python 58

6.1 Listen: Mehrere Informationen zusammenfassen ... 59
6.2 Dictionaries: Zugriff über einen Schlüsselbegriff ... 64
6.3 Tupel: unveränderliche Daten ... 67
6.4 Übungsaufgabe: Mit Datenstrukturen arbeiten ... 71

7. Entscheidungen im Programm treffen 76

7.1 Der Schlüsselbegriff if ... 76
7.2 Vergleiche: wichtig für das Aufstellen der Bedingung ... 78
7.3 Die Verknüpfung mehrerer Bedingungen ... 82
7.4 Mit else und elif weitere Alternativen hinzufügen ... 85
7.5 Übungsaufgabe: eigene Abfragen erstellen ... 88

8. Schleifen für die Wiederholung bestimmter Programmteile 95

8.1 Die while-Schleife: der grundlegende Schleifentyp ... 96
8.2 Die for-Schleife: ein mächtiges Instrument in Python ... 99
8.3 break und continue: weitere Werkzeuge für die Steuerung von Schleifen ... 103
8.4 Übungsaufgabe: mit verschiedenen Schleifen arbeiten ... 106

9. Funktionen in Python 111

9.1 Die Vorteile einer Funktion ... 111
9.2 Eine Funktion selbst erstellen ... 112
9.3 Argumente für Funktionen verwenden ... 114
9.4 Einen Rückgabewert verwenden ... 119
9.5 Funktionen in einer eigenen Datei abspeichern ... 123
9.6 Übungsaufgabe: Funktionen selbst gestalten ... 125

10. Mit Modulen aus der Standardbibliothek arbeiten 130

10.1 Was ist die Standardbibliothek und welche Module enthält sie? ... 130
10.2 Die Referenz für die Standardbibliothek ... 132
10.3 Beispiel für ein häufig verwendetes Modul: math ... 134
10.4 Übungsaufgabe: mit der Standardbibliothek arbeiten ... 136

11. Objektorientierung in Python 141

11.1 Objektorientierung: Was ist das? ... 141
11.2 Klassen: die Grundlage der objektorientierten Programmierung ... 144
11.3 Objekte: Instanzen der Klassen ... 147
11.4 Die Kapselung der Daten ... 150
11.5 Methoden: Funktionen für Objekte ... 152
11.6 Klassen- und Objektvariablen ... 157
11.7 Vererbung: ein grundlegendes Prinzip der objektorientierten Programmierung ... 161
11.8 Übungsaufgabe: mit Objekten arbeiten ... 164

12. Die Behandlung von Fehlern und Ausnahmen in Python 170

12.1 Warum ist es wichtig, Fehler und Ausnahmen zu behandeln? .. 170

12.2 try und except: So werden Ausnahmen behandelt 172
12.3 finally: die Ausnahmebehandlung abschließen 178
12.4 Selbst definierte Ausnahmen festlegen 180
12.5 Übungsaufgabe: Programme mit Ausnahmebehandlung schreiben 183

13. Dateien für die Datenspeicherung verwenden 189

13.1 Daten dauerhaft abspeichern: verschiedene Möglichkeiten 189
13.2 Daten in die Datei schreiben 191
13.3 Daten aus der Datei auslesen 193
13.4 Übungsaufgabe: mit Dateien für die Datenspeicherung arbeiten 200

14. Grafische Benutzeroberflächen mit Tkinter erzeugen 205

14.1 Ein erstes einfaches Fenster erstellen 206
14.2 Buttons mit Funktionen hinzufügen 212
14.3 Das Layout der Fenster 216
14.4 Weitere Elemente für die Gestaltung der Fenster 221
14.5 Übungsaufgabe: Programme mit Fenstern selbst gestalten 226

15. Anwendungsbeispiel: Verwaltungsprogramm für Gebrauchtwagenhändler 231

15.1 Die Struktur des Programms festlegen 232
15.2 Das Sortiment anzeigen 235
15.3 Ein neues Auto zum Sortiment hinzufügen 241
15.4 Ein Auto verkaufen 245
15.5 Den Preis eines Fahrzeugs anpassen 249
15.6 Letzte Anpassungen am Hauptprogramm 254
15.7 Ausblick 257

Alle Programmcodes aus diesem Buch sind auch als PDF zum Download verfügbar. Dadurch müssen Sie sie nicht abtippen:

http://bmu-verlag.de/python/

Kapitel 1
Einleitung

Eine Programmiersprache zu erlernen, eröffnet viele Möglichkeiten. Zum einen stellt dies eine interessante intellektuelle Herausforderung dar. Um eine bestimmte Problemstellung mit einem Computerprogramm zu lösen, ist es notwendig, dessen Logik zu erfassen und in einfache Abläufe zu unterteilen. Das schärft die Sinne und ermöglicht eine neue Sichtweise auf viele alltägliche Herausforderungen. Außerdem lassen sich auf diese Weise nützliche Programme für den eigenen Gebrauch schreiben. Darüber hinaus sind gute Programmierkenntnisse sehr förderlich für die berufliche Karriere. Im Bereich der Informatik besteht eine große Nachfrage an Fachkräften mit entsprechenden Kenntnissen. Das Erlernen einer Programmiersprache stellt den ersten Schritt dar, um diese Fähigkeiten zu erwerben. Darüber hinaus erfasst die Digitalisierung immer weitere Bereiche der Arbeitswelt. Das bedeutet, dass auch in vielen weiteren Berufen gute Computerkenntnisse zu einer wichtigen Voraussetzung geworden sind. Wer solide Programmierkenntnisse vorweisen kann, verfügt über eine wichtige Zusatzqualifikation, die ebenfalls für die Karriere sehr förderlich ist.

Dieses Buch stellt einen Einstieg in das Programmieren mit Python dar. Dabei handelt es sich um eine einfach zu erlernende Programmiersprache, die viele Anwendungsmöglichkeiten bietet. Darüber hinaus entspricht sie modernen Programmierprinzipien, die auch auf weitere Sprachen übertragbar sind. Daher stellt es eine gute Wahl dar, das Programmieren mit Python zu erlernen. Das erste Kapitel stellt diese Sprache vor und gibt einen Überblick darüber, welche Vorteile sie bietet.

1.1 Python: eine einfach zu erlernende Programmiersprache

Python gilt als eine sehr leicht zu erlernende Programmiersprache. Der wesentliche Grund dafür liegt in der einfachen Syntax. Der Be-

griff Syntax bezieht sich auf formale Regeln, die die einzelnen Befehle und Ausdrücke eines Computerprogramms befolgen müssen. Selbst erfahrenen Programmierern unterlaufen bei ihrer Arbeit häufig Syntaxfehler. Das hat zeitaufwendige Verbesserungen zur Folge. Bei Anfängern sind diese Fehler besonders häufig. Als besonderes Problem kommt hierbei hinzu, dass diese die Syntaxfehler häufig nur schwer erkennen. Oftmals sind die Befehle eigentlich korrekt verfasst – es fehlt lediglich eine Klammer oder ein Semikolon. Da Anfänger im Bereich der Programmierung diese Fehler jedoch häufig nicht lokalisieren können, ist es nicht möglich, das Programm zum Laufen zu bringen.

Python verwendet daher eine Syntax, die so einfach wie möglich ist. Ein Beispiel hierfür besteht darin, dass diese Programmiersprache fast vollständig auf Klammern verzichtet. Zusammengehörige Blöcke werden hier lediglich durch eine Einrückung gekennzeichnet – die auch in anderen Sprachen üblich ist. Klammern sind hierfür jedoch nicht notwendig. Das merzt eine der häufigsten Fehlerquellen beim Programmieren aus. Auch das Semikolon, das bei vielen anderen Programmiersprachen nach jedem Befehl stehen muss und das häufig vergessen wird, ist bei Python nicht notwendig.

Auch hinsichtlich der Variablentypen ist Python sehr frei. Variablen sind ein wichtiger Bestandteil der meisten Computerprogramme. In vielen Programmiersprachen ist es notwendig, die Variablen zunächst zu deklarieren und ihnen einen festen Datentyp zuzuweisen. Dieser kann später nicht mehr verändert werden. Vergessene Variablendeklarationen und inkompatible Typen stellen dabei eine häufige Fehlerquelle dar. Da Python auf die Deklaration der Variablen verzichtet und den Typ dynamisch vergibt – weshalb es innerhalb des Programms problemlos möglich ist, ihn zu verändern – fällt diese bei dieser Programmiersprache jedoch weg.

Aufgrund der einfachen Syntax haben Anfänger vergleichsweise wenige Probleme damit, das Programmieren mit Python zu erlernen.

1.2 Die Ziele bei der Entwicklung von Python

Python entstand ursprünglich als Programmier-Lehrsprache. Ihr wesentliches Ziel bestand von Anfang an darin, einen möglichst übersichtlichen und einfachen Code zu erzeugen. Das sollte es den Schülern und Studenten erleichtern, das Programmieren zu erlernen. Außerdem sollten dadurch die wesentlichen Bestandteile eines Programms besser zur Geltung kommen. Ein Mittel, um dies zu erreichen, bestand in der Verwendung einer einfachen Syntax. Dieser Punkt wurde bereits im vorherigen Abschnitt angesprochen. Python verfügt jedoch noch über weitere Eigenschaften, die dazu beitragen, dass diese Sprache einen sehr übersichtlichen Code erzeugt, der leicht zu verstehen ist. Darüber hinaus bestand der Ansatz darin, dass er intuitiv und so einfach zu verstehen sein sollte, wie reines Englisch.

Python verwendet im Vergleich zu anderen Programmiersprachen nur sehr wenige Schlüsselwörter. Das hat zur Folge, dass sich Personen, die das Programmieren mit Python erlernen, nur vergleichsweise wenige Begriffe merken müssen. Die Standardbibliothek, die alle grundlegenden Befehle enthält, ist sehr knapp und übersichtlich. Das macht es deutlich einfacher, sich in dieses Werk einzuarbeiten. Trotz dieser Einschränkung ist Python nicht in seinem Funktionsumfang eingeschränkt. Die Standardbibliothek lässt sich leicht erweitern, um alle benötigten Funktionen hinzuzufügen.

Zur Übersichtlichkeit trägt außerdem bei, dass Python einen sehr knappen Programmiercode verwendet. Wenn man ein Programm in Python schreibt, benötigt dieses fast immer deutlich weniger Zeilen, als wenn man ein Programm mit identischen Funktionen in einer anderen Sprache verfasst. Das macht den Quellcode erheblich übersichtlicher. Davon profitieren insbesondere Anfänger, die in ihren Lehrbüchern häufig den Code anderer Programmierer lesen und verstehen müssen.

Die einfachen Strukturen führen nicht nur dazu, dass Python einfach zu verstehen ist. Darüber hinaus ist der zeitliche Aufwand für das Er-

stellen des Codes vergleichsweise gering. Die kurzen Entwicklungszeiten führen dazu, dass sich Python sehr gut für Projekte mit hohem Zeitdruck eignet.

1.3 Die Entwicklungsgeschichte

Python entstand zu Beginn der 90er Jahre. Die Entwicklung geht auf den Niederländer Guido van Rossum zurück. Dieser war zu dieser Zeit am Centrum Wiskunde & Informatica in Amsterdam beschäftigt. Dieses nutzte als Lehrsprache ABC – eine Programmiersprache, die ebenfalls viel Wert auf Einfachheit und Übersichtlichkeit legt. Van Rossum beschloss während der Weihnachtswochen 1989 – solange sein Büro geschlossen blieb – an seinem privaten Computer eine neue Scriptsprache als Nachfolger für ABC zu entwickeln. Diese sollte auch für Programmierer interessant sein, die mit UNIX und C arbeiten und die mit ABC nur wenig anfangen konnten.

Als Arbeitstitel für dieses Projekt wählte van Rossum den Namen Python. Dieser geht jedoch nicht auf die bekannte Schlangenart zurück, sondern auf die Fernsehsendung Monty Python's Flying Circus. Obwohl als Symbol für die Programmiersprache mittlerweile eine Schlange gewählt wurde, ist dieser Ursprung noch in vielen Bereichen zu erkennen. Zahlreiche Lehrbücher und Dokumentationen zu Python sind mit Zitaten der bekannten Komiker-Truppe ausgeschmückt.

Die erste Vollversion dieser Programmiersprache erschien 1994. Im Laufe der folgenden Jahre kam es jedoch zu vielfachen weiteren Versionen, die den Funktionsumfang von Python deutlich erweiterten. Ein wichtiger Schritt war beispielsweise die Einführung der Garbage Collection, die mit der Version 2.0 im Jahre 2000 umgesetzt wurde. Diese Funktion gibt den Speicherplatz, den ein Programm besetzt, automatisch wieder frei. So muss sich der Programmierer nicht um diese Aufgabe kümmern. Das macht das Programmieren wesentlich einfacher.

Bei der Entwicklungsgeschichte ist es wichtig, auch auf den Unterschied zwischen Python 2 und Python 3 einzugehen. Python 3 wurde 2008 eingeführt. Python 2 wurde aber parallel dazu weiterentwickelt. Die letzte Version – Python 2.7 – erschien 2010. Das führte dazu, dass es über zwei Jahre hinweg zwei verschiedene aktuelle Versionen gab. Wer Python lernt, muss sich für eine dieser Versionen entscheiden. Das liegt darin begründet, dass die Entwickler beschlossen, auf eine Abwärtskompatibilität zu verzichten. Das führt dazu, dass Code in Python 2 nicht mit einem Interpreter, der Python 3 verwendet, ausgelesen werden kann. Mittlerweile kommt Python 2 kaum mehr bei neuen Programmen zum Einsatz. Doch sind noch zahlreiche ältere Lehrbücher verbreitet, die auf dieser Version beruhen. Auch ältere Programme sind häufig in Python 2 geschrieben. Dieses Buch verwendet hingegen Python 3 und entspricht damit dem aktuellen Stand. Der Leser muss sich keine weiteren Gedanken um die Unterschiede zwischen den beiden Versionen machen. Es ist lediglich wichtig, zu beachten, dass Code aus anderen Quellen eventuell nicht kompatibel zur aktuellen Version ist. Wer fremden Code verwendet, sollte daher stets darauf achten, um welche Python-Version es sich dabei handelt.

Bei Python handelt es sich um ein Open-Source-Projekt. Das bedeutet, dass die Entwicklung der Programmiersprache auf vielen freiwilligen Helfern beruht. Die Einfachheit und der große Funktionsumfang von Python führten nicht nur dazu, dass sich schnell eine beachtliche Gruppe von Anwendern bildete. Darüber hinaus entstand eine große Community, die Python immer weiter verbessert. Van Rossum wirkte hier bis 2018 als "wohlwollender Diktator auf Lebenszeit". Das bedeutet, dass er bei wichtigen Entscheidungen in der Entwicklung dieser Programmiersprache stets das letzte Wort hatte. Am 12. Juli 2018 gab er jedoch bekannt, sich von dieser Position zurückzuziehen. Dennoch arbeitet er auch weiterhin an diesem Projekt mit.

1.4 Eine interpretierte Programmiersprache

Eine wesentliche Eigenschaft von Python besteht darin, dass es sich hierbei um eine interpretierte Programmiersprache handelt. Leser, die

bislang nur über geringe Kenntnisse im Bereich der Informatik verfügen, werden an dieser Stelle wahrscheinlich nicht wissen, was dies bedeutet. Da dieser Aspekt jedoch sehr wichtig für das Verständnis von Python ist, soll er an dieser Stelle etwas ausführlicher behandelt werden.

Um ein Computerprogramm zu verfassen, kommt gewöhnlicher Text zum Einsatz. Im Unterschied zu einer normalen Sprache gibt es bei einer Programmiersprache jedoch fest vorgegebene Schlüsselbegriffe, die nach einer ebenfalls fest vorgegebenen Struktur zusammengefügt werden müssen. Die Schlüsselbegriffe orientieren sich an der menschlichen Sprache und an mathematischen Symbolen. Wer über grundlegende Englisch- und Mathematikkenntnisse verfügt und ein Computerprogramm liest, kann sich dabei häufig bereits vorstellen, welche Aufgaben dieses ausführen soll – selbst wenn keine Programmierkenntnisse vorhanden sind.

Wenn man jedoch diese abstrakte Ebene verlässt und die Abläufe im Bereich der Hardware betrachtet, ergibt sich ein ganz anderes Bild. Hier bestehen alle Informationen aus Binärcode. Das bedeutet, dass jede Information nur zwei verschiedene Zustände annehmen kann. Auf Hardwareebene handelt es sich dabei in der Regel um elektrische Impulse. In der Informatik kommen als Stellvertreter hierfür die Zahlen 0 und 1 zum Einsatz. Jede einzelne Information stellt ein Bit dar. Eine Kombination aus mehreren Bits wird als Wort bezeichnet. Moderne Computer arbeiten mit 64-Bit-Wörtern. Ältere Modelle nutzen hingegen 32-, 16- oder sogar 8-Bit-Wörter. Jedes einzelne Wort, das den Prozessor erreicht, hat eine ganz spezifische Ausgabe zur Folge. Ein Programm besteht aus einer Abfolge verschiedener Wörter, die zur Folge haben, dass der Prozessor das gewünschte Ergebnis ausgibt. Der Binärcode, der hierfür zum Einsatz kommt, wird als Maschinensprache bezeichnet.

Wenn man nun ein Computerprogramm mit gewöhnlichen Buchstaben und bestimmten Schlüsselbegriffen verfasst, ist es notwendig, dieses in die Maschinensprache zu übersetzen. Sonst ist es nicht möglich, es auszuführen. Hierfür gibt es verschiedene Möglichkeiten. Eine Alternative besteht darin, das Programm nach dem Verfassen in Maschinen-

sprache zu übersetzen. Dieser Prozess wird als Kompilation bezeichnet und erzeugt ein ausführbares Programm. Eine andere Alternative besteht darin, einen Interpreter zu verwenden. Dabei handelt es sich um ein Programm, das den Code während der Ausführung einliest und direkt in Maschinensprache übersetzt.

Beide Alternativen haben gewisse Vor- und Nachteile. Ein einmal kompiliertes Programm muss bei weiteren Ausführungen nicht erneut übersetzt werden. Da dieser Prozess einige Zeit in Anspruch nimmt, steigert das die Effizienz. Allerdings ist es nicht möglich, das entsprechende Programm auf verschiedenen Betriebssystemen auszuführen, da die Kompilation stets an dessen spezifischen Anforderungen ausgerichtet ist. Ein interpretiertes Programm lässt sich hingegen auf jedem Rechner ausführen, der über einen entsprechenden Interpreter verfügt – ganz unabhängig vom Betriebssystem. Außerdem ist der Entwicklungsprozess etwas schneller. Um ein Programm, das in einer Programmiersprache verfasst ist, die einen Compiler verwendet, auszuprobieren, ist es notwendig, es zunächst zu kompilieren. Dieser Prozess führt insbesondere bei umfangreichen Werken zu einer erheblichen Wartezeit. Während der Entwicklungsphase ist es jedoch häufig notwendig, das Programm auszuprobieren, einen kleinen Fehler zu verbessern und es anschließend erneut auszuprobieren. So kommen oftmals mehrere Hundert Kompilier-Vorgänge zusammen. Das verlangsamt den Entwicklungsprozess erheblich. Bei interpretierten Programmen ist es hingegen nicht notwendig, sie zu kompilieren. Es ist möglich, sie direkt mit dem Interpreter auszuführen. Das gestaltet die Entwicklung wesentlich effizienter.

Zusammen mit der einfachen Syntax, dem einfachen Wortschatz und der schnellen und intuitiven Codeerstellung trägt auch dieser Aspekt dazu bei, dass sich Python sehr gut für Anfänger eignet. Daher stellt es eine gute Wahl für die erste Programmiersprache dar.

Kapitel 2
Die Vorbereitungsmaßnahmen

Um mit dem Programmieren zu beginnen, sind einige Vorbereitungsmaßnahmen notwendig. Wie bereits in den vorherigen Abschnitten beschrieben, besteht ein Computerprogramm zunächst lediglich aus Text. Um diesen Text zu schreiben, ist eine geeignete Software notwendig. Die meisten Leser verfügen sicherlich über ein Textverarbeitungsprogramm wie Word, in dem sich Textdokumente verfassen lassen. Diese Programme eignen sich jedoch nicht zum Programmieren. Der Grund dafür liegt darin, dass diese neben dem eigentlichen Text noch viele weitere Informationen enthalten – beispielsweise zur Schriftgröße, zur Schriftart und zu vielen weiteren Formatierungs-Vorgaben. Das führt dazu, dass sie die Daten auf eine Weise speichern, die zum Programmieren ungeeignet ist. Daher ist es notwendig, einen Texteditor zu verwenden.

Anwender, die das Betriebssystem Windows nutzen, verfügen bereits über einen integrierten Texteditor. Dieser trägt den Namen Microsoft Editor – auch bekannt unter dem englischen Begriff NotePad. Dieser eignet sich zwar grundsätzlich dazu, Computerprogramme zu schreiben, doch sind seine Funktionen stark eingeschränkt. Daher empfiehlt es sich in diesem Fall, einen Texteditor mit etwas umfangreicheren Möglichkeiten zu installieren. Linux-Anwender verfügen in der Regel bereits über ein passendes Programm. Bei Ubuntu ist dies gEdit und bei KDE Kate. Auch andere Distributionen sind meistens mit einem passenden Texteditor ausgestattet. Daher können Leser, die einen Linux-Rechner verwenden, das Kapitel 2.1 daher überspringen.

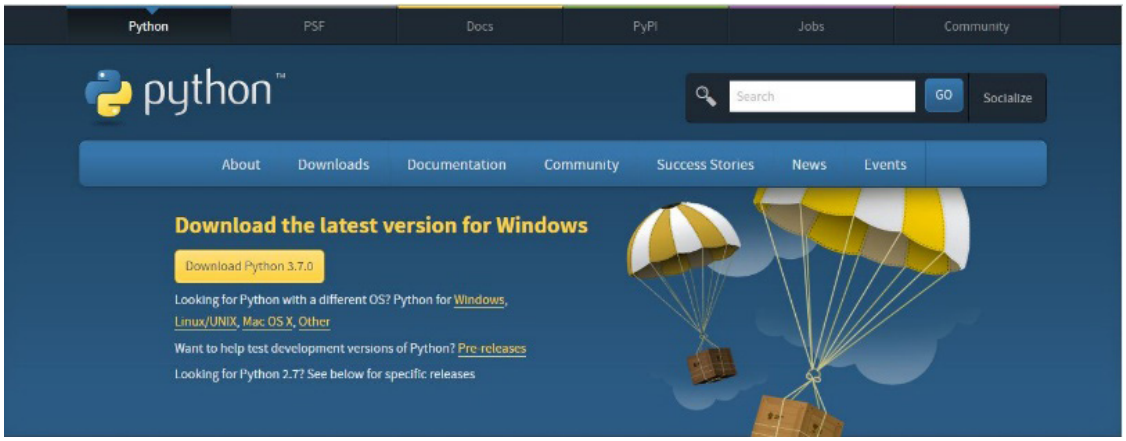
Das Kapitel 2.2 befasst sich mit der Installation des Python-Interpreters. Wie bereits in der Einleitung erwähnt, ist es notwendig, den geschriebe-

nen Code in die Maschinensprache zu übersetzen, um das Programm auszuführen. Hierfür ist es ebenfalls notwendig, eine passende Software auf dem Computer zu installieren. Genau wie der Texteditor ist auch diese kostenfrei erhältlich, sodass keine weiteren Ausgaben entstehen.

2.1 Den Python Interpreter installieren

Die Entwicklergemeinschaft von Python stellt den notwendigen Interpreter unter folgender Adresse zum Download bereit:

https://www.python.org/downloads/



Screenshot 1 Die Internetseite der Python-Entwickler mit dem Download des Interpreters

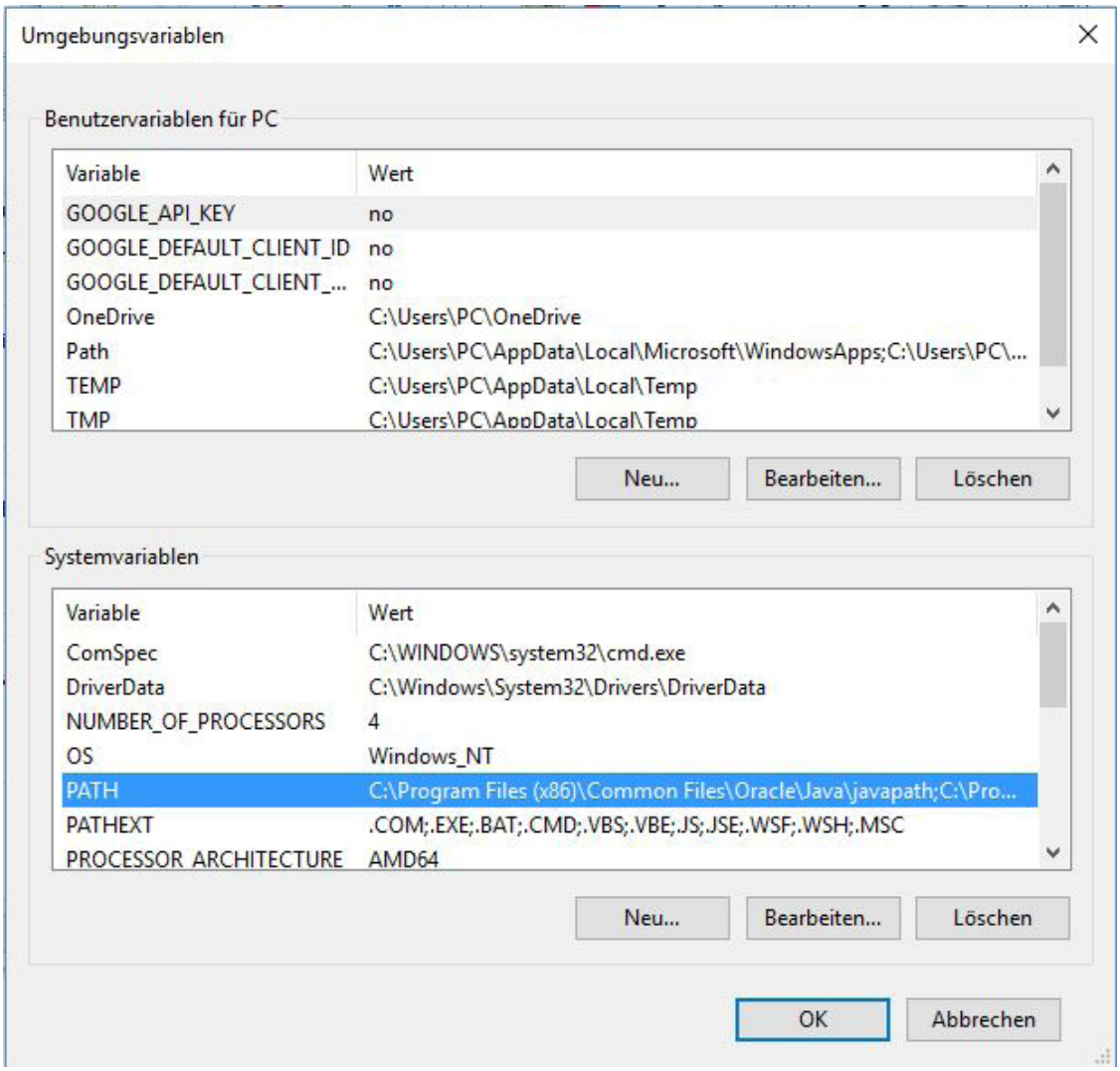
Ein Klick auf die gelbe Schaltfläche löst den Download des Installations-Assistenten aus. In der Regel stellt die Internetseite bereits die passende Version für das Betriebssystem des Besuchers bereit. Dennoch ist es sinnvoll, dies kurz zu überprüfen und gegebenenfalls die passende Ausführung auszuwählen.

Wenn der Download abgeschlossen ist, ist es notwendig, die entsprechende Datei zu öffnen. Daraufhin erscheint folgendes Fenster:

Screenshot 2 Der Installations-Assistent für den Python-Interpreter

In diesem Fenster ist es wichtig, die Checkbox ganz unten am Bildrand anzuklicken (Add Python 3.7 to PATH). Das bewirkt, dass der Installations-Assistent automatisch den Pfad des Python-Interpreters zu den Umgebungsvariablen hinzufügt. Wenn er dies nicht tut, lässt er sich nur öffnen, wenn man sich im Verzeichnis befindet, in dem er installiert wurde. Da es aber nicht sinnvoll ist, die eigenen Programme im Installations-Verzeichnis des Python-Interpreters abzulegen, würde das eine große Einschränkung darstellen. Wenn jedoch der Pfad zu den Umgebungsvariablen hinzugefügt wird, lässt sich der Interpreter auch aus anderen Verzeichnissen öffnen. Nach der Auswahl der Checkbox ist es nur noch notwendig, auf "Install Now" zu klicken und bereits einige Sekunden später ist der Python-Interpreter einsatzbereit.

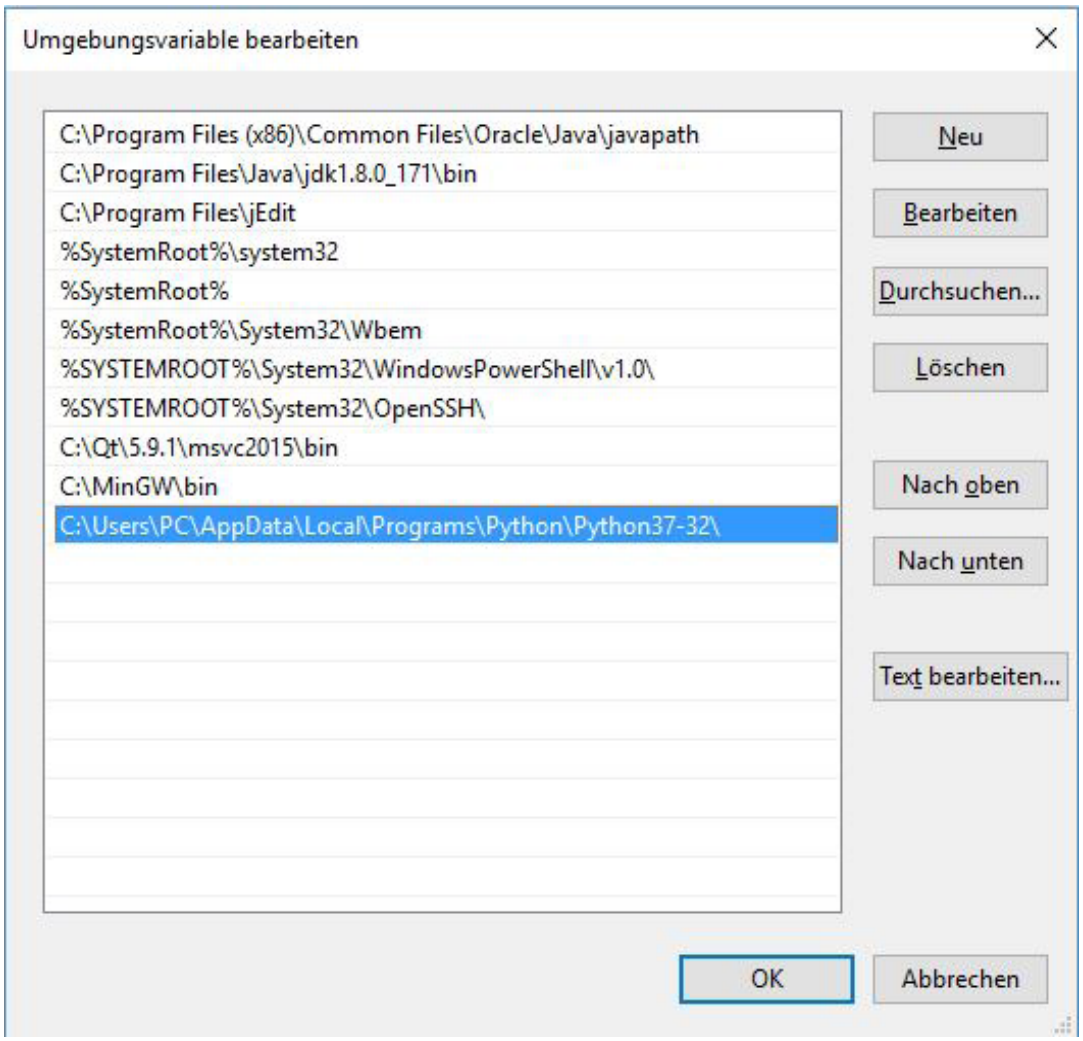
Wenn die Auswahl der Checkbox beim Installieren vergessen wurde, lässt sich die Umgebungsvariable auch nachträglich anpassen. Dazu ist es notwendig, die erweiterten Systemeinstellungen aufzurufen und anschließend die Schaltfläche "Umgebungsvariablen" anzuklicken.



Screenshot 3 Manuelle Einstellung der Umgebungsvariablen

Danach muss die Systemvariable PATH ausgewählt und danach die Schaltfläche “Bearbeiten” angeklickt werden. Nun ist es erforderlich, das Installationsverzeichnis einzutragen. In diesem Fall wäre dies

C:\Users\PC\AppData\Local\Programs\Python\Python37-32\. Das kann sich jedoch je nach Version und Einstellungen des Anwenders ändern.



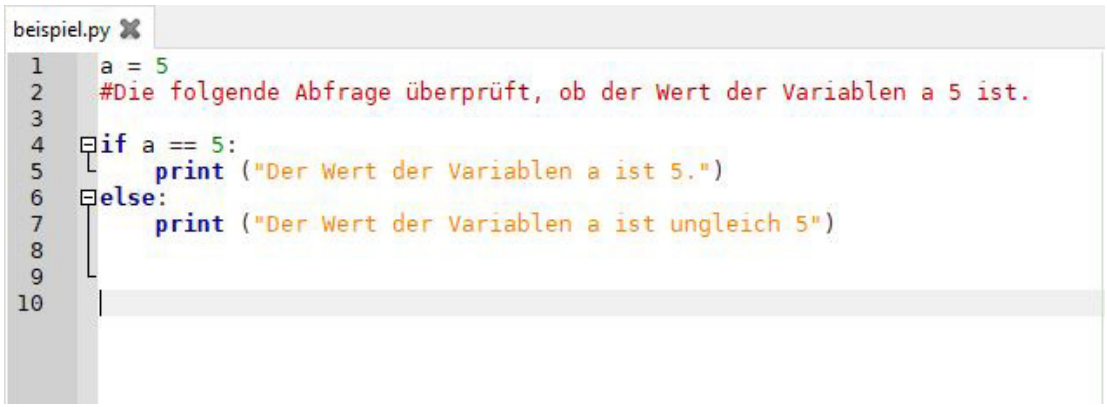
Screenshot 4 Die Eingabe des richtigen Installationsverzeichnisses

2.2 Einen Texteditor für die Erstellung des Codes

Um den Programmcode zu erstellen, ist ein Texteditor notwendig. Dieser bietet normalerweise folgende Vorteile:

- Syntaxhervorhebung: Verschiedene Bestandteile des Codes werden automatisch farbig markiert, um sie leichter zu erkennen.
- Nummerierung der Zeilen: Die Nummerierung der Zeilen gestaltet das Programm übersichtlicher und macht es einfacher, Fehlermeldungen nachzuvollziehen, da diese normalerweise die Zeilennummer angeben.

- Code-Faltung: Einzelne Blöcke lassen sich durch einen Klick auf das seitliche Minuszeichen einklappen. Das macht den Text deutlich übersichtlicher.
- Suchen und Ersetzen: Es ist möglich, bestimmte Elemente im Code zu suchen und automatisch durch einen anderen Befehl zu ersetzen. Das beschleunigt Korrekturen deutlich.
- Automatische Vervollständigung: Bereits nach den ersten Buchstaben eines Befehls öffnet sich ein Fenster, in dem der Programmierer aus verschiedenen passenden Möglichkeiten auswählen kann.
- Automatische Einrückungen: Der Texteditor rückt zusammengehörige Blöcke automatisch ein. Das ist insbesondere bei Python wichtig, da hier die Einrückungen funktionale Bestandteile des Programmcodes sind.
- Kompilieren oder Ausführen: Die Programme lassen sich direkt aus dem Texteditor heraus kompilieren und ausführen.



Screenshot 5 Ein Beispiel für die Darstellung des Programmcodes in einem Texteditor

Je nach gewähltem Programm können noch viele weitere praktische Funktionen hinzukommen. Die Auswahl an verschiedenen Texteditoren ist sehr groß. Einen Überblick über die Möglichkeiten gibt folgende Seite:

https://de.wikipedia.org/wiki/Liste_von_Texteditoren

Im Prinzip ist es möglich, aus dieser Liste einen beliebigen Editor auszuwählen und zu installieren. Wichtig ist es lediglich, darauf zu achten, dass dieser für das verwendete Betriebssystem verfügbar ist.

Für die Beispiele in diesem Buch kommt der Texteditor Geany zum Einsatz. Dieser kann auf folgender Seite heruntergeladen werden:

https://www.geany.org/Download/Releases

Dieser bietet alle genannten Vorteile und darüber hinaus noch einige weitere Funktionen, die das Programmieren erleichtern. Darüber hinaus ist er kostenfrei erhältlich und für alle gängigen Betriebssysteme verfügbar.

Diese Wahl ist jedoch nicht bindend. Die Aufgaben und Beispiele in diesem Buch lassen sich auch mit vielen weiteren Texteditoren bearbeiten. Der Leser kann gerne verschiedene Möglichkeiten ausprobieren und daraufhin selbst entscheiden, welche davon ihm persönlich am besten gefällt.

Kapitel 3

Interaktive Interpretation: ideal für den ersten Kontakt mit Python

Beim Programmieren ist es notwendig, den Programmcode zunächst in eine Textdatei zu schreiben. Danach wird der Code kompiliert oder mit einem Interpreter ausgeführt. Python bietet jedoch noch eine weitere Möglichkeit: die interaktive Interpretation. Dabei wird der Programmcode direkt in den Interpreter geschrieben. Ein Klick auf die Eingabetaste reicht aus, um ihn auszuführen.

Python ist eine der wenigen Programmiersprachen, die diese Möglichkeit anbieten. Diese bringt in vielen Situationen große Vorteile mit sich. Professionelle Entwickler verwenden sie beispielsweise, wenn sie das Verhalten eines ganz bestimmten Programmteils ausprobieren möchten. Anstatt das ganze Programm auszuführen, ist es hierbei möglich, nur einen kleinen Teil in den Interpreter einzufügen. Das macht es einfacher, dessen Verhalten zu analysieren.

Auch wenn der Programmierer zu Beginn noch keine genaue Vorstellung davon hat, wie er das Programm angehen will, ist die interaktive Interpretation sehr praktisch. Auf diese Weise kann er verschiedene Möglichkeiten schnell und einfach ausprobieren und ihre Funktionsweise kennenlernen. Das kann den Entwicklungsprozess gerade bei komplexen Strukturen, bei denen die Lösung nicht unmittelbar auf der Hand liegt, deutlich beschleunigen.

Auch für Anfänger ist die interaktive Interpretation sehr praktisch. Auf diese Weise können sie einfach verschiedene Befehle in den Interpreter schreiben und auf diese Weise das Verhalten der Programmiersprache spielerisch kennenlernen. Daher ist diese Form des Programmierens ideal für den ersten Kontakt mit Python geeignet. Daher soll sie hier zunächst vorgestellt werden. Dennoch ist es wichtig, dabei zu beachten, dass hierbei kein dauerhaftes Programm ent-

steht. Wenn die Befehle ausgeführt sind, gehen sie verloren. Um sie auch zu einem späteren Zeitpunkt wiederholen zu können, ist es notwendig, sie auf die herkömmliche Weise in einen Texteditor zu schreiben.

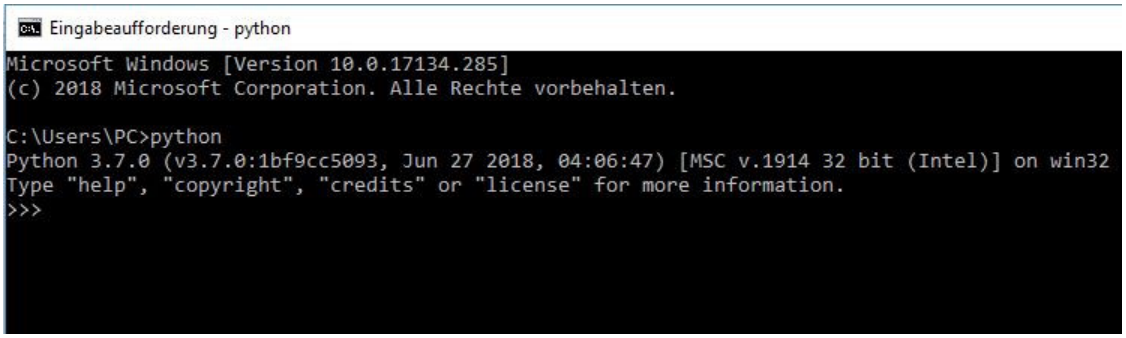
3.1 Den Python-Prompt aufrufen

Um mit der interaktiven Programmierung zu beginnen, ist es notwendig, den Python-Prompt zu öffnen. Dazu ist es zunächst notwendig, einen Kommandozeileninterpreter aufzurufen. Windows-Nutzer finden diesen im Startmenü unter den installierten Apps. Er befindet sich im Ordner Windows-System und trägt den Namen "Eingabeaufforderung". Noch etwas schneller geht es, die Buchstaben cmd in die Windows-Suche einzugeben. Daraufhin erscheint sofort das entsprechende Programm.

Neuere Windows-Versionen bieten noch eine weitere Alternative: die Windows PowerShell (Im Startmenü im Ordner Windows PowerShell). Diese erfüllt genau den gleichen Zweck wie die Eingabeaufforderungen, sie bietet jedoch noch einige Zusatzfunktionen. Es bleibt dem Leser selbst überlassen, eines dieser Programme auszuwählen.

Wer das Betriebssystem Linux verwendet, verfügt ebenfalls bereits über ein passendes Programm. Dieses trägt hier den Namen "Terminal".

Nachdem der Kommandozeileninterpreter aufgerufen wurde, erscheinen zunächst die Versionsangabe und die Copyright-Informationen. Danach zeigt er das Verzeichnis an, in dem sich das Programm befindet. Nun ist es lediglich notwendig, den Begriff python einzugeben und mit der Eingabetaste zu bestätigen. Wenn bei der Installation und bei der Vorgabe für die Umgebungsvariablen im vorherigen Kapitel keine Fehler aufgetreten sind, sollte nun folgende Ausgabe im Fenster erscheinen:

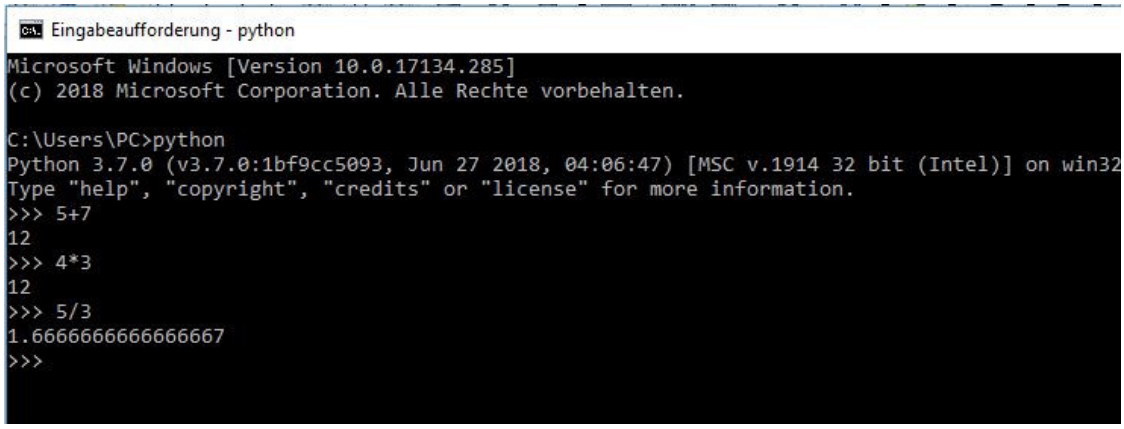


Screenshot 6 Der Python-Prompt für die interaktive Interpretation

Die drei Größerzeichen zu Beginn der Zeile zeigen an, dass der Python-Prompt einsatzbereit ist und auf die Befehle des Anwenders wartet.

3.2 Erste Befehle ausprobieren

Nun ist es möglich, verschiedene Python-Befehle auszuprobieren. Allerdings wird bei den meisten Lesern das Problem bestehen, dass diese bislang noch keinen einzigen Python-Befehl kennen. Daher sollen zunächst einige einfache Rechenaufgaben eingegeben werden – beispielsweise 5+7, 4*3 oder 5/3.



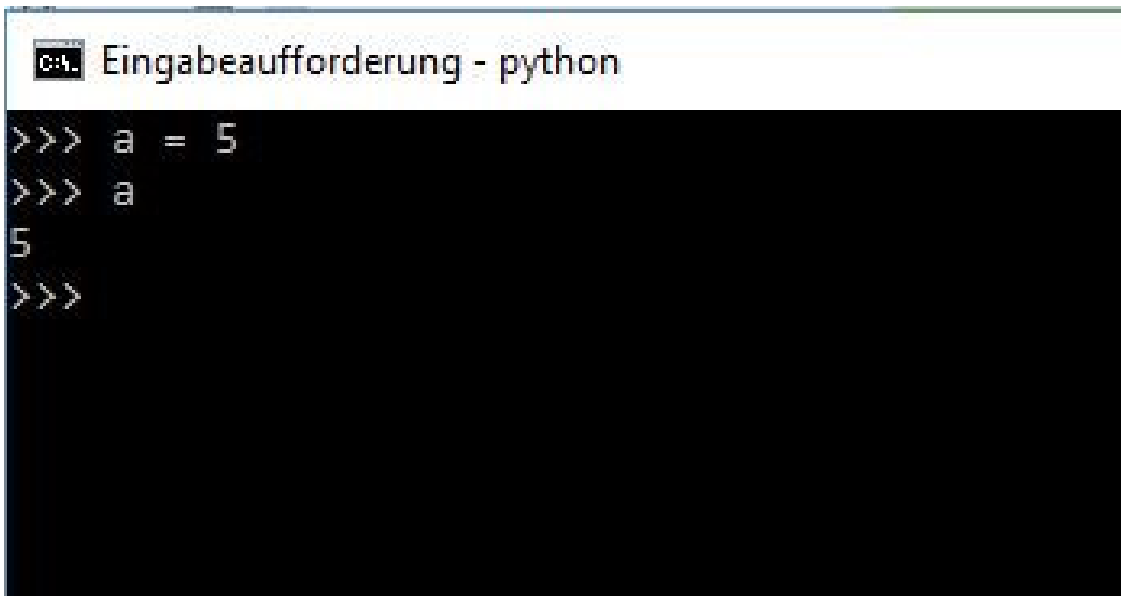
Screenshot 7 Einfache Rechenaufgaben mit dem Python-Prompt

Sobald die Eingabetaste gedrückt wird, erscheint das richtige Ergebnis. Für diese Aufgaben wäre zwar ausreichend, einen Taschenrechner zu

verwenden, anstatt ein eigenes Programm zu schreiben, doch zeigen sich dabei bereits einige Grundzüge von Python. Beispielsweise wird deutlich, dass es hierbei möglich ist, alle Grundrechenarten durchzuführen. Außerdem wird klar, dass für die Multiplikation das Sternsymbol `(*)` und für die Division der Schrägstrich `(/)` zum Einsatz kommt – wie übrigens bei fast allen anderen Programmiersprachen auch.

Dennoch soll nun bereits einen Schritt weiter gegangen werden – durch den Einsatz von Variablen. Dieses Thema wird in einem späteren Kapitel noch ausführlicher behandelt, doch bietet die interaktive Interpretation eine gute Möglichkeit zu einem spielerischen Einsteig.

Zunächst soll mit folgendem Befehl der Variablen `a` der Wert `5` zugewiesen werden: `a = 5`. Nach dem Drücken der Eingabetaste soll daraufhin lediglich der Variablenname `a` eingegeben und erneut durch die Eingabetaste bestätigt werden.



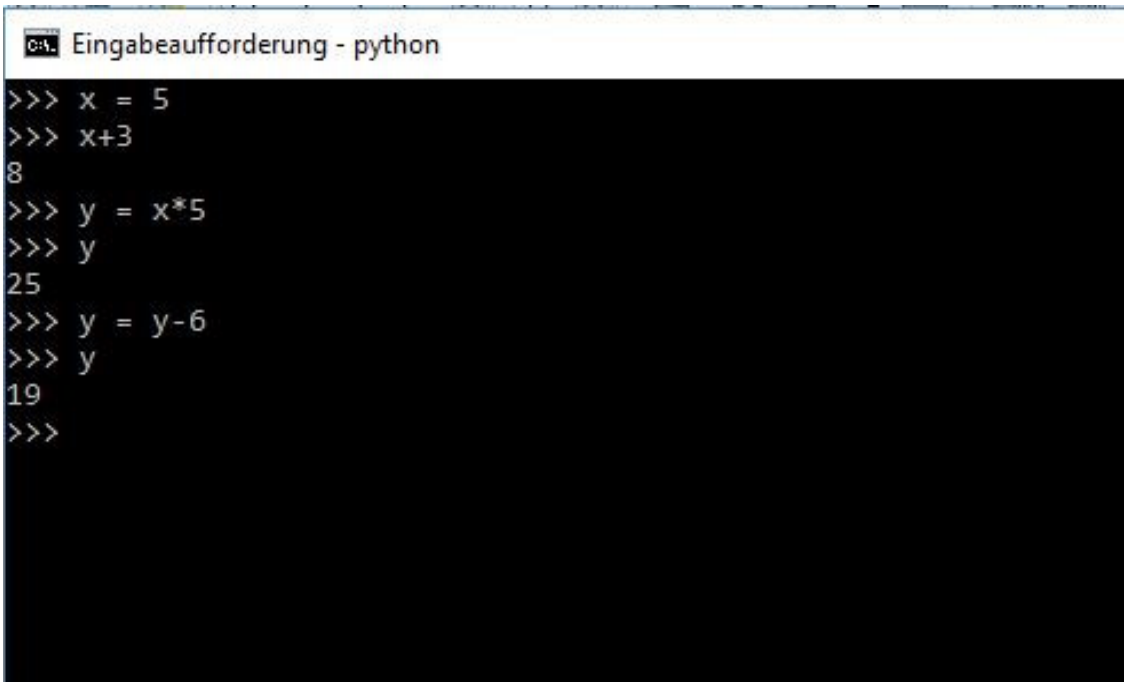
Screenshot 8 Die Verwendung einer Variablen

Nach dem ersten Befehl ist kein Ergebnis sichtbar. Der Python-Prompt zeigt lediglich erneut die drei Größerzeichen an. Dass er dennoch Auswirkungen hatte, wird nach dem nächsten Befehl deutlich. Die Eingabe

des Variablennamens führt dazu, dass der Interpreter deren Wert ausgibt. Daran wird deutlich, dass dieser nun 5 entspricht – genau wie im vorherigen Befehl zugewiesen.

Mit den Variablen lassen sich noch viele weitere Rechenaufgaben durchführen. Für das nächste Beispiel sollen folgende Befehle nacheinander eingegeben werden:


```
x = 5
x+3
y = x*5
y
y = y-6
y
```



Screenshot 9 Verschiedene Operationen mit Variablen

Auf diese Weise lässt sich ganz einfach ausprobieren, wie sich die Variablen in einer Programmiersprache verhalten. Wenn die Funktionsweise an dieser Stelle noch nicht ganz klar sein sollte, stellt dies kein Problem dar. An dieser Stelle soll lediglich ein spielerischer Einstieg erfolgen. Eine genaue Erklärung folgt im weiteren Verlauf dieses Buchs.

Um den Python-Prompt zu beenden, gibt es mehrere Möglichkeiten – beispielsweise `quit()` oder Strg+Z. Alternativ dazu lässt sich selbstverständlich auch das gesamte Fenster schließen, wodurch auch der Kommandozeileninterpreter beendet wird.

Kapitel 4

Ein Python-Programm in eine eigene Datei schreiben

Die interaktive Interpretation des Python-Codes stellt eine gute Möglichkeit dar, um die ersten Erfahrungen mit dieser Programmiersprache zu sammeln und um verschiedene Bestandteile des Programms auszuprobieren. Eine der wesentlichen Eigenschaften eines Computerprogramms besteht jedoch darin, dass es möglich ist, es beliebig oft und zu einem beliebigen Zeitpunkt erneut auszuführen – ohne den Code dafür neu zu schreiben. Das ist bei der Verwendung des Python-Prompts jedoch nicht der Fall. Wenn dieser einmal beendet ist, gehen alle Informationen verloren.

Aus diesem Grund wird in diesem Kapitel die klassische Form des Programmierens vorgestellt: das Verfassen des Quellcodes in einer eigenen Datei. Diese Methode soll im weiteren Verlauf des Buchs für das Erstellen der Beispiele und Übungsaufgaben zum Einsatz kommen. Dennoch ist es wichtig, die interaktive Programmierung nicht aus den Augen zu verlieren. Wenn die Programme später etwas komplexer werden und die Funktionsweise der einzelnen Bestandteile nicht sofort klar wird, ist es sinnvoll, diese interaktiv auszuprobieren.

4.1 Ein Programm für eine einfache Textausgabe

Um das erste Programm in Python zu schreiben, ist es notwendig, den Texteditor zu öffnen. Hier soll nur eine einzige Zeile eingetragen werden: `print ("Willkommen zum Python-Kurs!")`. Diese ist bereits ausreichend, um ein vollständiges Python-Programm zu erzeugen. Das zeigt nochmals deutlich, wie einfach strukturiert diese Programmiersprache ist. Bei den meisten anderen Alternativen sind

selbst für ein derart einfaches Programm mehrere Code-Zeile notwendig, deren Bedeutung dem Anfänger zunächst vollkommen unklar ist und die ihn verwirren. Python verzichtet jedoch auf diesen unnötigen Ballast und erlaubt es, den Befehl ohne weitere Zusätze einzufügen.

Dieses Programm hat die Aufgabe, den Leser zum Python-Kurs zu begrüßen. Dazu soll es den Text "Willkommen zum Python-Kurs!" auf dem Bildschirm ausgeben. Hierfür ist ein print-Befehl notwendig. Dieser bewirkt, dass das Programm den entsprechenden Inhalt im Kommandozeileninterpreter anzeigt.

In der Einleitung wurde zwar erwähnt, dass Python weitestgehend auf Klammern verzichtet, um die Syntax einfacher und übersichtlicher zu gestalten. Bei einigen Befehlen und Funktionen sind diese aber dennoch notwendig. Dies ist beispielsweise beim `print`-Befehl der Fall. Der Grund dafür besteht darin, dass dieser aus mehreren Bestandteilen bestehen kann. Im späteren Verlauf des Buchs sollen der Ausgabe beispielsweise Variablen und weitere Textblöcke hinzugefügt werden. Daher ist es notwendig, dass das Programm genau weiß, wo der Anfangs- und wo der Endpunkt für die Ausgabe ist. Dazu dient die runde Klammer.

Der Text für die Ausgabe muss stets in Anführungszeichen stehen. In diesem Beispiel wurden doppelte Anführungszeichen gewählt. Es wäre jedoch genauso gut möglich, einfache Anführungszeichen zu verwenden. Deren Funktionsweise ist vollkommen identisch. Die Möglichkeit, zwei verschiedene Anführungszeichen einzufügen, ist jedoch hilfreich, wenn der Ausgabetext selbst Anführungszeichen enthalten soll. Wenn hier die gleichen Anführungszeichen verwendet werden, interpretiert das Programm diese als Ende des Texts. Das führt zu einer Fehlermeldung oder zu einer unkorrekten Darstellung. Dieses Problem lässt sich einfach umgehen, indem man die jeweils andere Form der Anführungszeichen verwendet. Wenn man beispielsweise das Wort Python-Kurs in doppelte Anführungszeichen setzen will, würde folgender Ausdruck einen Fehler hervorrufen:

```
print ("Willkommen zum "Python-Kurs!"")
```

Wenn man jedoch für die Markierung des Texts einfache Anführungszeichen verwendet, lässt sich die Nachricht korrekt anzeigen:

```
print ('Willkommen zum "Python-Kurs!"')
```


Abschließend ist es nur noch notwendig, das Programm abzuspeichern. Der Name ist dabei frei wählbar. Das Beispielprogramm in diesem Buch trägt den Namen willkommen. Allerdings kann der Leser hierfür auch eine beliebige andere Bezeichnung wählen. Es ist lediglich wichtig, auf die richtige Dateiendung zu achten. Python-Programme haben stets die Endung .py. Daher lautet der vollständige Dateiname willkommen.py.

4.2 Die Ausführung im Python-Interpreter

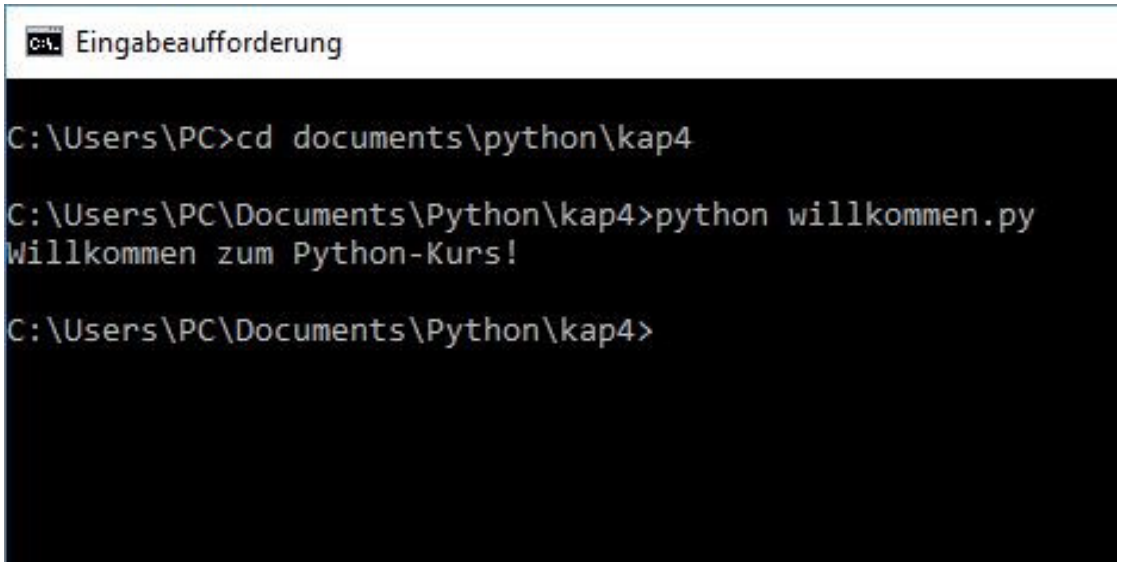
Das erste Programm ist nun bereits geschrieben, allerdings wurde es noch nicht ausgeführt. Um diese Aufgabe zu erledigen, ist es notwendig, den Kommandozeileninterpreter aufzurufen. Dieser zeigt nach dem Öffnen zunächst das Stammverzeichnis an. Um das Programm auszuführen, ist es jedoch notwendig, in das Verzeichnis zu wechseln, in dem das Python-Programm abgespeichert wurde. Dazu ist der Befehl cd gefolgt vom Pfadnamen notwendig. Die einzelnen Verzeichnisse werden dabei durch einen Backslash (\) voneinander getrennt. Unter Windows ist es wichtig, darauf zu achten, dass die vorgefertigten Verzeichnisse stets mit ihrer englischen Bezeichnung angegeben werden. Während beispielsweise der Windows-Explorer das Verzeichnis “Dokumente” anzeigt, wird es im Kommandozeileninterpreter als “Documents” bezeichnet. Wer sich nicht mehr genau an den Pfad erinnert, kann innerhalb eines Verzeichnisses den Befehl dir eintippen. Dieser sorgt dafür, dass der gesamte Inhalt des Ordners angezeigt wird. In diesem Beispiel befindet sich das Programm im Verzeichnis Dokumente. Darin ist das Unterverzeichnis python enthalten, in dem sich

wiederum ein Unterverzeichnis mit der Bezeichnung kap4 befindet. Dem Leser sei an dieser Stelle empfohlen, die gleiche Verzeichnisstruktur zu verwenden. Um dorthin zu wechseln, ist daher folgender Befehl notwendig:

```
cd documents\python\kap4
```

Nun ist es nur noch erforderlich, das entsprechende Programm aufzurufen. Dazu kommt der Begriff python gefolgt vom Dateinamen zum Einsatz:

```
python willkommen.py
```



Screenshot 10 Die Ausführung des ersten Programms

Nachdem der Befehl für die Ausführung eingegeben wurde, gibt das Programm den entsprechenden Text im gleichen Fenster aus. Danach wird es beendet und der Kommandozeileninterpreter wartet auf neue Befehle.

Der Begriff python, der dem Dateinamen vorangestellt wird, ist dabei zwar üblich, aber nicht unbedingt notwendig. Der Kommandozeileninterpreter erkennt auch anhand der Dateiendung, dass es sich hierbei um ein Python-Programm handelt und führt es entsprechend aus.

Tipp: Beim Programmieren ist es sehr oft notwendig, den Programmcode auszuprobieren. Um dafür nicht jedes Mal den kompletten Befehl eintippen zu müssen, ist es sinnvoll, im Kommandozeileninterpreter die Pfeiltaste nach oben zu verwenden. Diese führt dazu, dass automatisch der zuletzt verwendete Befehl angezeigt wird. Das bringt eine erhebliche Arbeitserleichterung mit sich.

4.3 Kommentare: hilfreich für das Verständnis des Programms

Wenn ein professioneller Programmierer ein Programm schreibt, kommt es häufig vor, dass er es nach einiger Zeit überarbeiten muss – beispielsweise wenn das Unternehmen, für das er es entwickelt hat, seine Produktionsabläufe ändert. In diesen Fällen ist es meistens nicht notwendig, das komplette Programm neu zu gestalten. Häufig reicht es aus, einige Teilbereiche zu verändern. Das spart viel Arbeit.

Dabei tritt jedoch oftmals das Problem auf, dass sich der Programmierer nach einigen Jahren nicht mehr an die genauen Details des Programms erinnert. Die verwendeten Funktionen sind häufig sehr abstrakt und ihre Aufgabe wird nicht auf den ersten Blick klar. Besonders gravierend ist dieses Problem, wenn ein anderer Programmierer diese Aufgabe übernimmt, der einen eigenen Programmierstil hat und nicht an die Arbeitsweise des ursprünglichen Verfassers gewöhnt ist.

Um diese Probleme zu reduzieren, ist es sinnvoll, Kommentare zu verwenden. Dabei handelt es sich um gewöhnlichen Text, der in den Quellcode eingefügt wird. Er hat jedoch keinerlei Einfluss auf die Ausführung. Kommentare dienen lediglich dazu, die einzelnen Bestandteile des Programms zu erläutern und um zu erklären, auf welche Weise sie mit anderen Bereichen interagieren. Das macht es erheblich einfacher, die Funktionsweise zu verstehen und Änderungen vorzunehmen.

Darüber hinaus sind Kommentare bei der Erstellung eines Programms hilfreich. Es kommt immer wieder vor, dass dabei ein bestimmter Teil zunächst offengelassen wird, um einen anderen Bereich zu bearbeiten – obwohl dieser erst später folgt. In diesem Fall ist es sinnvoll, einen Kommentar einzufügen, um die Stelle zu markieren und um aufzuzeigen, welche Aufgaben hier noch zu erledigen sind.

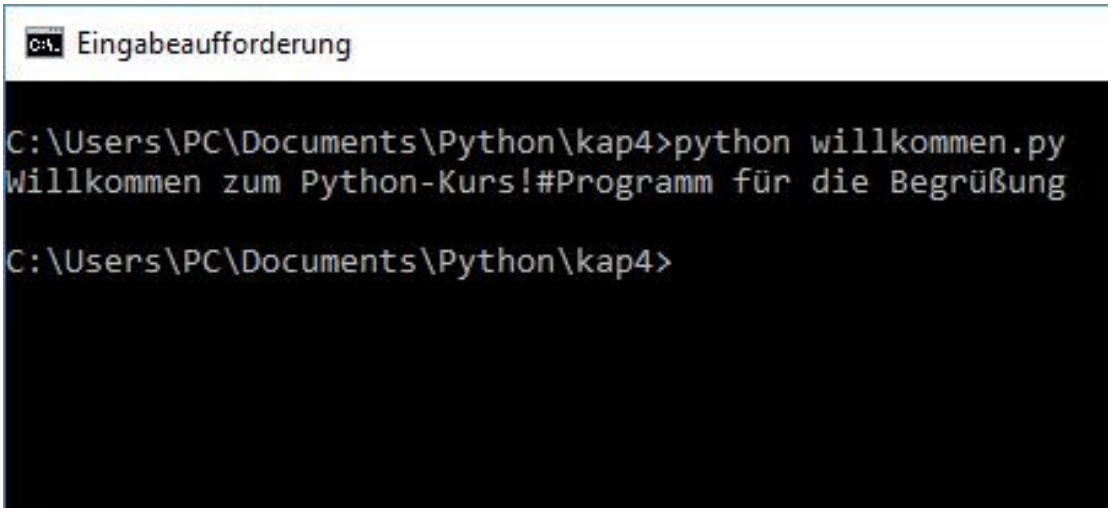
Zwar ist es noch ein weiter Weg bis zur Erstellung professioneller Programme, doch ist es auch für Anfänger sinnvoll, mit Kommentaren zu arbeiten. Auf diese Weise gewöhnen sie sich diese Vorgehensweise an und erzeugen von Beginn an einen leicht verständlichen Programmcode. Darüber hinaus ist es wichtig, die Kommentare im Code einer fremden Quelle zu erkennen. In diesem Buch werden gelegentlich Kommentare in den Quellcode eingefügt, um die Aufgaben der einzelnen Programmteile direkt an der richtigen Stelle zu erläutern. Der Leser muss diese nicht zwingend abtippen, um den Arbeitsaufwand so gering wie möglich zu halten. Es ist jedoch wichtig, dass er die Kommentare erkennt und dass es ihm bewusst wird, dass diese keine Auswirkung auf das Programm haben. Wer nach dem Ende dieses Kurses seine Programmierkenntnisse eigenständig ausweiten will, findet im Internet unzählige Beispiele für Python-Programme, die als Lernmaterial dienen können. Auch hier sind meistens Kommentare enthalten, sodass es auch unter diesem Aspekt wichtig ist, dieses Mittel kennenzulernen.

Die Kennzeichnung eines Kommentars erfolgt in Python durch das Raute-Zeichen `(#)`. Wenn dieses im Quellcode auftaucht, wird der Bereich, der danach folgt, nicht bei der Ausführung berücksichtigt. Die Kennzeichnung gilt nur für die Zeile, in der das Raute-Zeichen steht. Allerdings ist es nicht möglich, Kommentare innerhalb von Anführungszeichen zu verwenden. In diesem Fall interpretiert Python das Zeichen als Text und gibt es auf dem Bildschirm aus. Wenn man dem ersten Programm einen Kommentar hinzufügen will, könnte das folgendermaßen aussehen:

```
#Programm für die Begrüßung
print ("Willkommen zum Python-Kurs!")
```

Wenn man es nun erneut ausführt, tritt keine Veränderung auf: Der Kommentar wird bei der Ausführung ignoriert. Wenn man allerdings den Kommentar in die Anführungszeichen setzt, wird er ebenfalls ausgegeben:

```
print ("Willkommen zum Python-Kurs!#Programm für die Begrüßung")
```

Screenshot 11 Die falsche Verwendung des Kommentars innerhalb der Anführungszeichen

4.4 Übungsaufgabe: eigene Inhalte zum Programm hinzufügen

Am Ende der meisten Kapitel befindet sich ein kurzer Abschnitt mit einigen Übungsaufgaben. Diese dienen dazu, den Inhalt zu vertiefen und selbst anzuwenden. Der Schwerpunkt liegt dabei stets auf den Themen, die im jeweiligen Kapitel besprochen wurden. Doch werden auch die Kenntnisse, die in den vorherigen Bereichen des Buchs vermittelt wurden, als gegeben vorausgesetzt. Wenn für die Bearbeitung einer Aufgabe einmal neue Funktionen und Befehle notwendig sein sollten, die noch nicht behandelt wurden, werden diese in einer Anmerkung kurz erklärt.

Am Ende der Aufgaben befindet sich stets eine Musterlösung. Allerdings ist es empfehlenswert, die Aufgaben selbstständig zu lösen. Nur wenn Sie einmal nicht weiterkommen, sollten Sie diese heranziehen. Außerdem dient sie zum Abgleich, nachdem Sie die Aufgabe erledigt haben. Dabei ist es jedoch wichtig, zu beachten, dass es häufig verschiedene Lösungswege gibt. Die Musterlösung stellt dabei nur eine Alternative dar. Wenn Ihr Programm funktionsfähig ist und die Anforderungen der Aufgabenstellung korrekt umsetzt, stellt es ebenfalls eine korrekte Lösung dar – selbst wenn es von der Musterlösung abweichen sollte.

1. Stellen Sie sich vor, Sie arbeiten als Programmierer und ein Gebrauchtwagenhändler beauftragt Sie damit, ein Programm für die Verwaltung seines Bestands zu schreiben. Erstellen Sie eine Überschrift, eine Begrüßungs-Floskel und einen kurzen Überblick über die Aufgaben, die das Programm übernimmt. Diese sollen jeweils einen eigenen print-Befehl verwenden.

Anmerkung: Wenn der Inhalt zu lang für eine einzelne Zeile ist, können Sie diesen in zwei Abschnitte unterteilen. Jeder Abschnitt muss in Anführungszeichen stehen. Verbinden Sie sie mit einem Plus-Zeichen.

2. Fügen Sie einen Kommentar in dieses Programm ein, der beinhaltet, welche Aufgaben hierfür noch zu erledigen sind.
3. Erstellen Sie ein neues Programm mit zwei `print`-Befehlen. Deren Inhalt soll jeweils 5 + 2 lauten – allerdings einmal in Anführungszeichen und einmal ohne Anführungszeichen. Führen Sie das Programm aus und überlegen Sie sich, worauf die unterschiedlichen Ausgaben zurückzuführen sind.

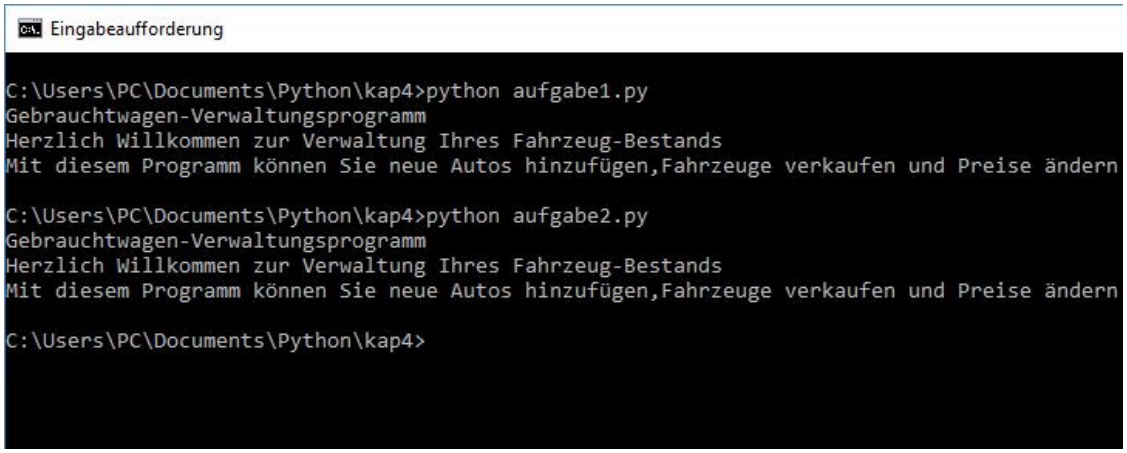
Lösungen:

1.

```
print ("Gebrauchtwagen-Verwaltungsprogramm")
print ("Herzlich Willkommen zur Verwaltung Ihres Fahrzeug-Bestands")
print ("Mit diesem Programm können Sie neue Autos hinzufügen,"+
"Fahrzeuge verkaufen und Preise ändern")
```

2.

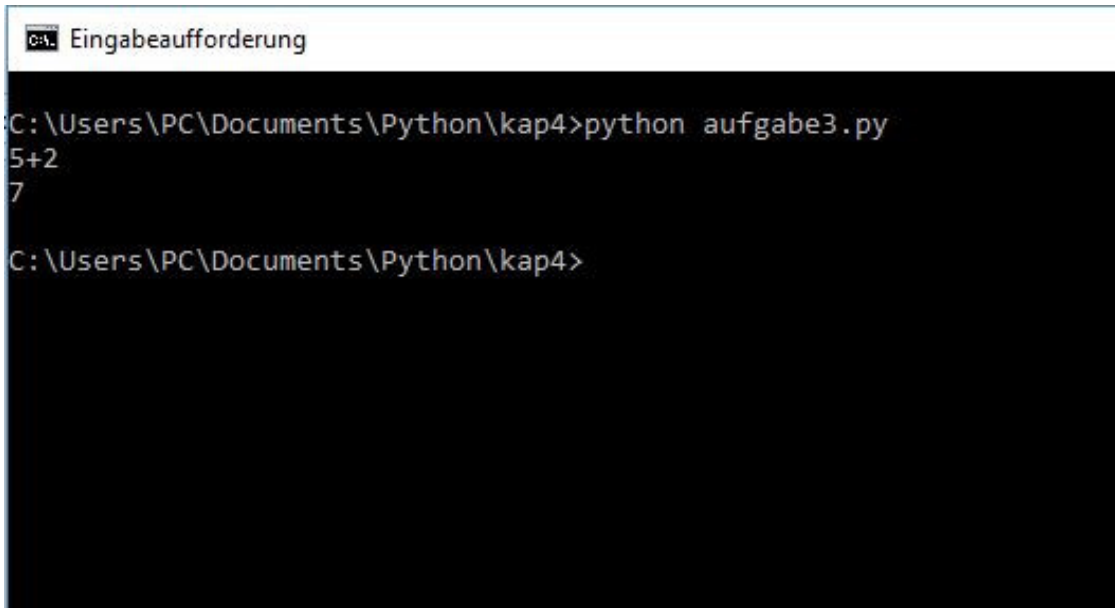

```
print ("Gebrauchtwagen-Verwaltungsprogramm")
print ("Herzlich Willkommen zur Verwaltung Ihres Fahrzeug-Bestands")
print ("Mit diesem Programm können Sie neue Autos hinzufügen,"+
"Fahrzeuge verkaufen und Preise ändern")

#To-Do: Funktionen für das Verkaufen, Hinzufügen von Autos und für
Preisänderungen
```



Screenshot 12 Die Ausgabe ist bei Aufgabe 1 und Aufgabe 2 identisch.

3.

```
print ("5+2")
print (5+2)
```



Screenshot 13 Die Verwendung von Anführungszeichen verändert die Ausgabe

Wenn ein Text in Anführungszeichen steht, gibt Python diesen unverändert wieder. Fehlen diese, geht der Interpreter davon aus, dass es sich hierbei um eine Rechenaufgabe handelt. In diesem Fall berechnet er das Ergebnis aus 5+2 und gibt es aus. Wenn man hier eine Variable einfügt, gibt er deren Inhalt aus. Steht hier ein Funktionsname, berechnet er deren Ergebnis. Sollte es sich beim Inhalt weder um einen mathematischen Ausdruck, noch um eine Variable oder um eine Funktion handeln, kommt es zu einer Fehlermeldung.

Hat Ihnen diese Leseprobe gefallen?

Bestellen Sie das Buch als Taschenbuch oder eBook komfortabel auf Amazon!

eBook: https://amzn.to/2SrbfRt

Taschenbuch: https://amzn.to/2T1teD5

Möchten Sie über neue Bücher und Sonderangebote vom BMU Verlag informiert werden?

Tragen Sie sich jetzt in unseren E-Mail Newsletter ein: https://bmu-verlag.de/