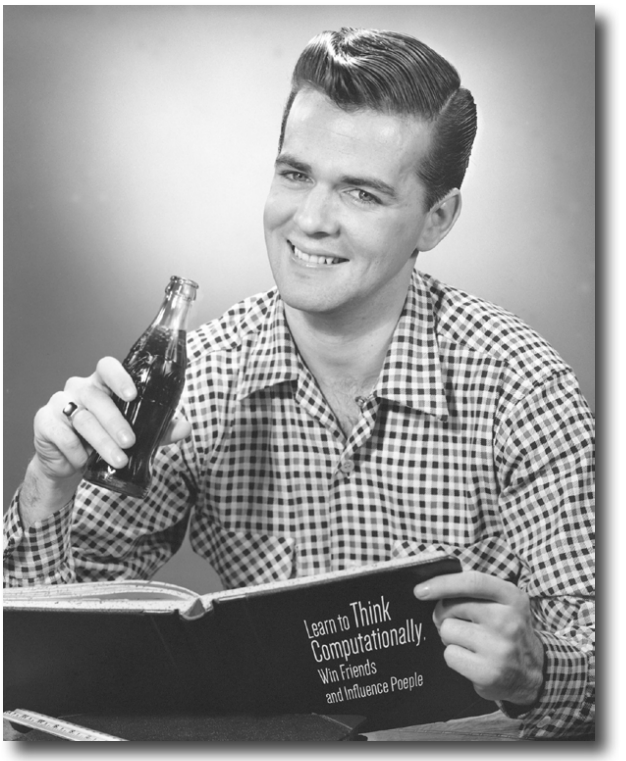
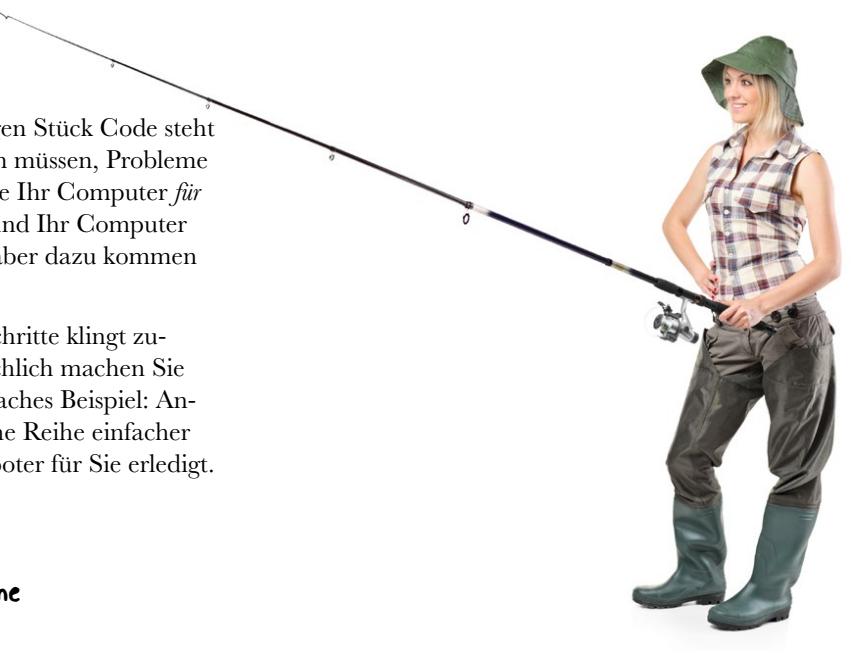
Erste Schritte



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

Immer der Reihe nach



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

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

Teilen wir das Angeln in eine Reihe einfacher Schritte auf.

Wir führen die Schritte nacheinander aus.

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

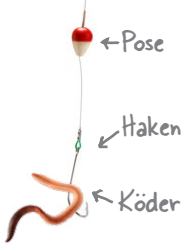
Manche Schritte sind einfache Anweisungen, wie: »Wirf die Leine aus!«

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

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

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

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



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

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

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

ÜBUNG

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

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

Spitzen Sie Ihren Bleistift

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

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

Fallen Ihnen weitere Probleme ein?

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

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

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

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

Zum Pseudocode kommen wir später noch genauer …

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

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

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

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

Code-Magneten

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

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

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

Wie Programmieren funktioniert

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

1 Erstellen Sie Ihren Algorithmus

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

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

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

2 Schreiben Sie Ihr Programm

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

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

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

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

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

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

3 Führen Sie Ihr Programm aus

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

Wir verwenden diese Begriffe oft gleichbedeutend.

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

Sprechen wir überhaupt die gleiche Sprache?

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

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

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

Entspannen Sie sich

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

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

DU SAGST TOMATE …

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

Gib »Hi there« auf dem Bildschirm aus.

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

Ein Einkaufszettel mit Brot, Milch und Eiern.

Fünf Drinks einschenken.

Frage den Benutzer: What is your name?

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

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

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

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

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

Die Welt der Programmiersprachen

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

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

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

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

Ich benutze meistens die Programmiersprache **C**. Ich schreibe Teile von Betriebssystemen, die supereffizient sein müssen. In meinem Job zählt jeder CPU-Zyklus und jede Speicheradresse.

Nennen Sie mich verkopft, aber ich liebe **Scheme – und Sprachen im Stil von LISP**. Für mich geht es hauptsächlich um Funktionen hoher Ordnung und Abstraktion. Ich freue mich, dass Sprachen wie **Clojure** tatsächlich in der Praxis benutzt werden.

Ich bin Webentwickler. Daher ist **JavaScript** meine Hauptsprache. Es ist die De-facto-Sprache aller Webbrowser und wird außerdem benutzt, um Backend-Webservices zu programmieren.

Ich bin Systemadministrator und verwende Perl, um z.B. verschiedene Systemskripte zu schreiben. Perl ist knapp und dabei sehr ausdrucksstark. Mit wenig Code können Sie eine Menge erledigen.

Wer die Wahl hat ...

Wie Sie sehen, gibt es diverse Sprachen und Meinungen. Dabei haben wir moderne Sprachen gerade einmal oberflächlich gestreift. Außerdem sieht man, dass diese Sprachen eine Menge an Fachausdrücken mit sich bringen. Je weiter Sie fortschreiten, desto mehr werden auch diese Begriffe einen Sinn für Sie ergeben. Im Moment ist für Sie nur wichtig, dass es eine Vielzahl von Programmiersprachen gibt, die täglich mehr werden.

Was sollen wir also in diesem Buch benutzen? Wie schon gesagt, wir wollen hauptsächlich, dass Sie lernen, *rechnerisch* zu denken. Egal welche Sprache Ihnen in der Zukunft begegnen wird, wir wollen Ihnen das Lernen erleichtern. Trotzdem müssen wir mit *irgendeiner Sprache* anfangen, und wie Sie schon wissen, ist dies Python. Also warum? Unsere Freunde oben auf dieser Seite haben es schon gesagt: Python gilt als eine der besten Anfängersprachen, weil sie so gut lesbar und konsistent ist. Sie ist außerdem eine sehr mächtige Sprache. Egal was Sie (jetzt oder außerhalb dieses Buchs) damit machen wollen, für fast alle Aufgaben gibt es Unterstützung in Form von Codeerweiterungen (die wir *Module* oder *Bibliotheken* nennen). Außerdem hat Python eine sehr freundliche und hilfsbereite Entwicklergemeinschaft. Einige Entwickler sagen sogar, Python mache einfach mehr *Spaß* als andere Sprachen. Was kann da schon schiefgehen?

Spitzen Sie Ihren Bleistift

Sehen Sie, wie einfach es ist, Python zu schreiben

Sie kennen Python noch nicht, aber ich bin sicher, Sie können erraten, wie dieser Python-Code funktioniert. Sehen Sie sich die Zeilen gut an und versuchen Sie, herauszufinden, was hier passiert. Schreiben Sie Ihre Antworten unten hin. Wenn Sie stecken bleiben, finden Sie die Antworten auf der folgenden Seite. Die erste Zeile haben wir schon für Sie ausgefüllt.

`customers = ['Jimmy', 'Kim', 'John', 'Stacie']`	Eine Liste der Kunden erstellen.
`winner = random.choice(customers)`	
`flavor = 'vanilla'`	
`print('Congratulations ' + winner +` `  ' you have won an ice cream sundae!')`	
`prompt = 'Would you like a cherry on top? '`	
`wants_cherry = input(prompt)`	
`order = flavor + ' sundae '`	
`if (wants_cherry == 'yes'):`	
`  order = order + ' with a cherry on top'`	
`print('One ' + order + ' for ' + winner +` `  ' coming right up...')`	

Python-Ausgaben

```
Congratulations Stacie you have won an ice cream sundae!
Would you like a cherry on top? yes
One vanilla sundae with a cherry on top for Stacie coming
right up...
```

Das sollte helfen. Dies sind die Ausgaben unseres Codes. Glauben Sie, die Ausgaben sind bei jeder Ausführung des Codes gleich?

Spitzen Sie Ihren Bleistift. Lösung

Sehen Sie, wie einfach es ist, Python zu schreiben

Sie kennen Python noch nicht, aber ich bin sicher, Sie können erraten, wie dieser Python-Code funktioniert. Sehen Sie sich die Zeilen gut an und versuchen Sie, herauszufinden, was hier passiert. Schreiben Sie Ihre Antworten unten hin. Die erste Zeile haben wir schon für Sie ausgefüllt.

`customers = ['Jimmy', 'Kim', 'John', 'Stacie']`	Eine Liste der Kunden erstellen.
`winner = random.choice(customers)`	Einen dieser Kunden zufällig auswählen.
`flavor = 'vanilla'`	Den Namen oder die Variable flavor mit dem Text 'vanilla' füllen.
`print('Congratulations ' + winner + ' you have won an ice cream sundae!')`	Eine Glückwunschnachricht auf dem Bildschirm ausgeben, die den Namen der Gewinnerin enthält. Ist Kim die Gewinnerin, lautet die Meldung beispielsweise >>Congratulations Kim you have won an ice cream sundae!<<
`prompt = 'Would you like a cherry on top? '`	Den Namen oder die Variable prompt mit dem Text >>Would you like a cherry on top?<< füllen.
`wants_cherry = input(prompt)`	Den Benutzer auffordern, etwas Text einzugeben und ihn wants_cherry zuzuweisen. Soll der Benutzer etwas eingeben, wird diese Aufforderung zuerst angezeigt (wie in den Python-Ausgaben zu sehen).
`order = flavor + ' sundae '`	Den Namen oder die Variable order mit dem Text 'vanilla', gefolgt von 'sundae' füllen.
`if (wants_cherry == 'yes'): order = order + ' with a cherry on top'`	Hat der Benutzer auf die Frage 'Would you like a cherry on top?' mit ja ('yes') geantwortet, füge den Text >> with a cherry on top<< zu order hinzu.
`print('One ' + order + ' for ' + winner + ' coming right up...')`	Nachricht ausgeben, dass die Bestellung des Gewinners sofort serviert wird (>>is coming right up<<).

Python-Ausgaben

```
Congratulations Stacie you have won an ice cream sundae!
Would you like a cherry on top? yes
One vanilla sundae with a cherry on top for Stacie coming
right up...
```

PS: Wenn Sie den Code unbedingt eintippen müssen, fügen Sie am Anfang der Datei `import random` hinzu, bevor Sie sie ausführen. Was diese Anweisung tut, werden wir später erklären. Im Moment ist das Ausführen des Codes allerdings noch nicht erforderlich und auch nicht besonders nützlich. Einige von Ihnen werden es trotzdem ausprobieren. Wir kennen Sie!

Code mit Python schreiben und ausführen

Nachdem wir über Code gesprochen und sogar schon welchen gesehen haben, wollen wir jetzt überlegen, wie man echten Code schreibt und ausführt. Je nach Sprache und Umgebung gibt es dafür eine Reihe verschiedener Modelle. Versuchen wir, ein Gefühl für das Schreiben und Ausführen Ihres Python-Codes zu bekommen:

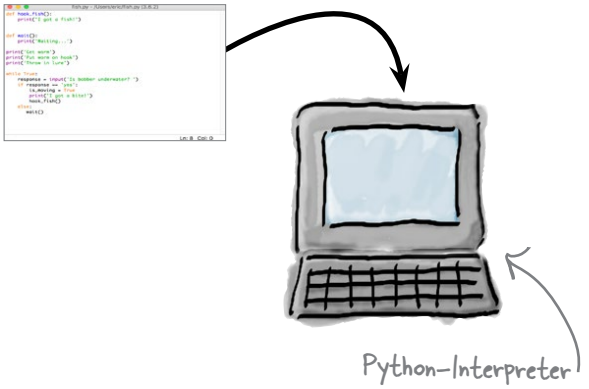
1 Ihren Code schreiben

Zuerst müssen Sie Ihren Code in einen Editor eingeben und speichern. Das kann ein Texteditor wie Notepad (Windows) oder TextEdit (Mac) sein. Allerdings benutzen die meisten Entwickler zum Schreiben ihres Codes spezielle Editorprogramme, die IDE (Integrated Development Environment, integrierte Entwicklungsumgebung) genannt werden. Warum? Weil IDEs wie Textverarbeitungsprogramme viele nette zusätzliche Fähigkeiten wie die automatische Vervollständigung bestimmter Python-Schlüsselwörter, das Hervorheben der Syntax (oder von Fehlern) und eingebaute Testmöglichkeiten besitzen. Praktischerweise liegt Python bereits eine eigene IDE namens IDLE bei, die wir uns gleich ansehen werden.

Ihr Code

Dies ist Pythons IDLE-Editor.

Ihr Code



2 Ihren Code ausführen

Um Ihren Code auszuführen, müssen Sie ihn einfach dem Python-Interpreter übergeben. Das ist ein Programm, das sich um alles kümmert, was zum Ausführen Ihres Codes nötig ist. Zu den Details werden wir gleich kommen. Auf den Interpreter können Sie entweder über IDLE oder direkt über die Kommandozeile Ihres Computers zugreifen.

3 Wie Ihr Code interpretiert wird

Wir benutzen Python als Sprache, die Sie wie auch Ihr Computer verstehen. Und soweit wir gelernt haben, gibt es einen Interpreter, der Ihren Code liest und ausführt. Hierfür übersetzt der Interpreter Ihren Code hinter den Kulissen in Maschinencode, der direkt von der Hardware Ihres Rechners ausgeführt werden kann. Sie müssen nicht wissen, wie – es reicht, wenn Sie verstehen, dass der Interpreter jede Ihrer Python-Anweisungen für Sie ausführt.

Der Interpreter führt Ihren Code aus.

Es gibt keine Dummen Fragen

F: Warum benutzt man nicht einfach eine natürliche Sprache, um Computer zu programmieren? Spezielle Programmiersprachen wären dann nicht mehr nötig.

A: Ja, das wäre wirklich schick. Leider sind natürliche Sprachen voller Mehrdeutigkeiten, was ihre Übersetzung in Maschinencode sehr erschwert. Forscher arbeiten an dem Problem, sind aber noch weit davon entfernt, natürliche Sprachen als Programmiersprachen verwenden zu können. Außerdem bevorzugen Programmierer Sprachen, die sich weniger »natürlich« anfühlen und stattdessen auf die Programmierung spezialisiert sind.

F: Warum gibt es überhaupt mehrere Programmiersprachen?

A: Technisch gesehen, sind alle Programmiersprachen gleich. Sie berechnen die gleichen Dinge, um Ihnen zu dienen. Wie gesprochene Sprachen unterscheiden sich auch Programmiersprachen in ihrer Ausdruckskraft. Einige Aufgaben (z. B. das Erstellen von Websites) lassen sich mit bestimmten Sprachen besser erledigen als mit anderen. Manchmal ist die Wahl der Programmiersprache auch nur von Ihrem Geschmack oder der Verwendung einer bestimmten Methodik abhängig – oder es ist einfach die Sprache, die Ihr Chef sich ausgesucht hat. Auf eines können Sie aber zählen: Die Anzahl der Sprachen wächst immer weiter, weil die Programmiersprachen sich weiterentwickeln.

F: Ist Python nur eine Spielzeugsprache für Anfänger? Ist es überhaupt sinnvoll, Python zu lernen, wenn ich später als Softwareentwickler arbeiten will?

A: Python ist eine ernsthafte Sprache und wird in vielen Produkten verwendet, die Sie (vermutlich) kennen und lieben. Außerdem ist Python eine der wenigen professionellen Sprachen, die sich ausgezeichnet für Anfänger eignen. Warum? Weil Python im Vergleich zu vielen anderen Sprachen die Dinge ohne große Umstände und trotzdem konsistent angeht (diese Eigenschaften werden Sie im Laufe der Zeit besser verstehen, sobald Sie mehr Erfahrung mit Python und anderen Sprachen haben).

F: Worin besteht der Unterschied zwischen dem Lernen einer Programmiersprache und rechnerischem Denken? Ist Zweiteres nur eine Informatiksache?

A: Rechnerisches Denken ist ein bestimmter Denkansatz zum Lösen von Problemen, der tatsächlich aus der Informatik stammt. Beim rechnerischen Denken lernen wir, das Problem zu zerlegen, um daraus Algorithmen zum Lösen der Teilprobleme zu erstellen und die Lösungen zu verallgemeinern. Dadurch lassen sich auch größere Probleme lösen. Oft soll ein Computer Algorithmen für uns auszuführen. Und genau da kommt das Programmieren ins Spiel. Programmieren ist der Weg, einen Algorithmus in den Computer (oder irgendein anderes Rechengerät, z. B. Ihr Smartphone) hineinzubekommen. Beide geht Hand in Hand. Mit rechnerischem Denken können wir Lösungen für Probleme finden, die wir programmieren wollen. Das Programmieren wiederum gibt uns die Möglichkeit, unsere Lösungen an einen Computer weiterzugeben. Übrigens kann das rechnerische Denken auch helfen, wenn Sie nicht programmieren.

KOPF-NUSS

Woher, glauben Sie, stammt der Name Python?
Wählen Sie die wahrscheinlichste Antwort unten aus:

- ☐ **A.** Der Erfinder von Python liebte Schlangen und hat vorher eine weniger erfolgreiche Sprache namens Cobra geschaffen.
- ☐ **B.** Der Erfinder von Python liebte Kuchen (engl. »pie«) und wählte deshalb den Namen Pie-thon.
- ☐ **C.** Die britische Komikertruppe Monty Python inspirierte zu dieser Namensgebung.
- ☐ **D.** Python ist ein Akronym für Programming Your Things, Hosted On the Network.
- ☐ **E.** Der Name Python ist durch die Laufzeitumgebung Anaconda inspiriert, auf der die Sprache basiert.

C. Monty Python ist die britische Komikertruppe, die die Show Monty Python's Flying Circus erfand. Als freiwillige Hausaufgabe dürfen Sie sich gerne ein paar Shows ansehen. Selbst wenn die Truppe nicht so Ihr Ding ist, werden Sie in der Python-Gemeinschaft gelegentlich Anspielungen auf Monty Python finden.

Eine kurze Geschichte von Python

Python 1.0

In den Niederlanden gab es am nationalen Forschungszentrum für Mathematik und Informatik ein großes Problem: Den Wissenschaftlern war das Lernen von Programmiersprachen zu schwer. Selbst für diese hoch spezialisierten Wissenschaftler waren die meisten aktuellen Programmiersprachen verwirrend und inkonsistent. Als Ausweg entwickelte das Institut eine neue Sprache namens »ABC« (Sie haben gedacht, wir sagen jetzt »Python«, oder?), die wesentlich leichter zu erlernen sein sollte. ABC war sogar einigermaßen erfolgreich. Dann kam ein aufstrebender junger Entwickler namens Guido van Rossum. Nachdem er sich ein Wochenende lang ausgiebig alte Monty-Python-Folgen angesehen hatte, war er überzeugt davon, dass er die Dinge, die er von ABC gelernt hatte, noch verbessern könnte. Also schuf er Python. Der Rest ist Geschichte.

Hinweis vom Herausgeber: Den »Rest« finden Sie in den folgenden paar Absätzen.

Python 2.0

Mit der Version 2.0 wuchs Python heran und erhielt eine Vielzahl neuer Funktionen, die die wachsende Entwicklergemeinschaft unterstützen sollten. Da Python inzwischen eine wirklich weltweit verbreitete Sprache war, wurde 2.0 so erweitert, dass Zeichensätze von Sprachen verarbeitet werden konnten, die weit außerhalb der typischen englischen Buchstaben liegen. Außerdem wurden viele technische Sprachaspekte verbessert. Hierzu gehörte etwa die Speicherverwaltung und eine bessere Unterstützung für bestimmte Datentypen wie Listen und Zeichenstrings.

Das Python-Entwicklungsteam arbeitete außerdem hart daran, Python einer ganzen Entwicklergemeinschaft zu öffnen, die bei der Verbesserung und Implementierung der Sprache helfen konnten.

Okay, den Teil über das Monty-Python-Wochenende haben wir uns einfach ausgedacht.

Python 3.0

Niemand ist perfekt. Und so bemerkten die Schöpfer von Python irgendwann, dass es durchaus noch Verbesserungsbedarf gab. Obwohl Python möglichst unkompliziert sein will, zeigte die Erfahrung, dass das Sprachdesign noch verbessert werden konnte. Anderes war in die Jahre gekommen und musste entfernt werden.

Diese Änderungen hatten zur Folge, dass einige Aspekte von Python 2 nicht weiter unterstützt werden konnten. Trotzdem sorgten die Schöpfer von Python dafür, dass Python 2.0-Code auch weiterhin funktioniert. Machen Sie sich also keine Sorgen, wenn Sie mit Python 2-Code arbeiten müssen. Python 2 ist immer noch gesund und munter. Die Zukunft von Python liegt aber trotzdem in der Version 3.

Wir sind absolut davon überzeugt, dass fliegende Autos mit Python programmiert werden.

1994 **2000** **2008** **Die Zukunft!**

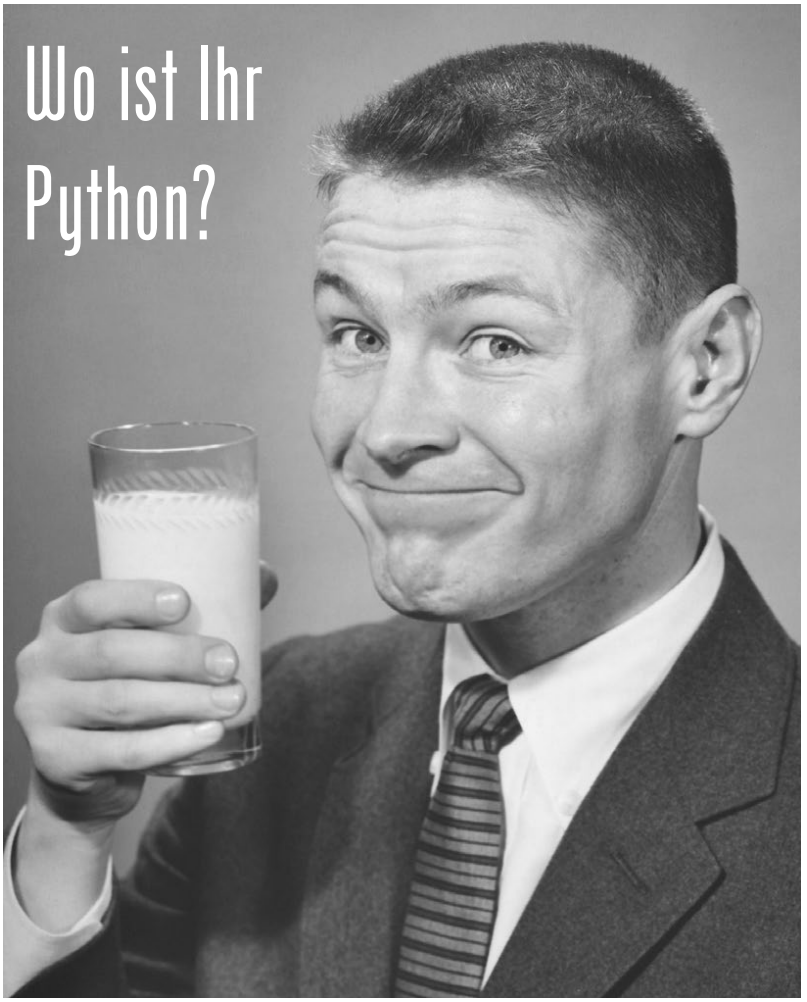
Das klingt, als gäbe es zwei Python-Versionen: 2 und 3. Welche benutzen wir, und welche Unterschiede gibt es?

Gute Frage. Es gibt tatsächlich zwei Python-Versionen. Beim Druck dieses Buchs waren das die Versionen 3.6 und 2.7.

Wir sehen das so: Wenn Sie sich die beiden Sprachen aus ca. drei Kilometern Höhe betrachten, gibt es erstaunlich viele Ähnlichkeiten, und Sie werden vermutlich überhaupt keine Unterschiede feststellen. Trotzdem *gibt* es diese Unterschiede. Das heißt, wenn Sie nicht auf die von Ihnen verwendete Version achten, kommen Sie leicht ins Stolpern. Wir benutzen in diesem Buch die neueste Python-Version, also 3, denn wir zeigen Ihnen lieber gleich den Weg in die Zukunft.

Jetzt gibt es aber schon eine Menge Python 2-Code da draußen. Sollten Sie Softwareentwickler werden, wird Ihnen früher oder später auch Python 2-Code begegnen. Zum Beispiel in einem heruntergeladenen Modul oder als Teil des Codes, den Sie weiterpflegen sollen. Nach diesem Buch sind Sie in einer guten Position, bei Bedarf die kleinen Unterschiede zwischen Python 2 und 3 zu erkennen.

Wenn wir von Python 3 oder 2 sprechen, meinen wir die jeweils aktuelle Version (beim Schreiben des Buchs 3.6 und 2.7).

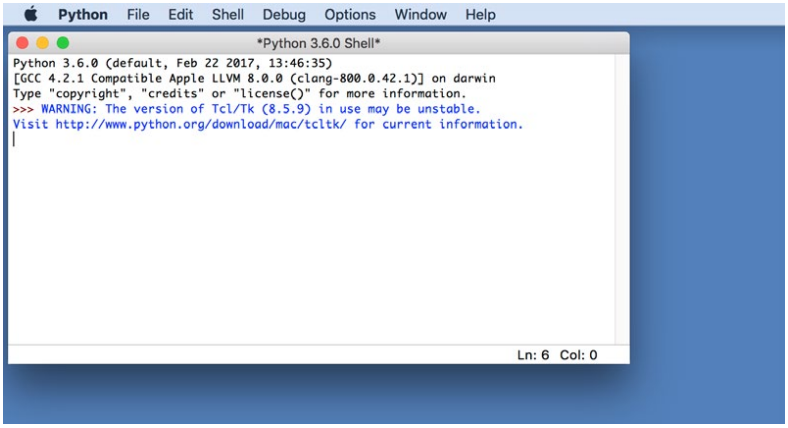


Ohne dass Sie Python installieren, werden wir nicht weiterkommen. Falls Sie es noch nicht getan haben, ist jetzt der richtige Zeitpunkt gekommen. Lesen Sie hierfür den Abschnitt »Sie müssen Python installieren« in der Einführung zu diesem Buch. Falls Sie einen Mac oder Linux benutzen, ist es gut möglich, dass Python bei Ihnen schon installiert ist – allerdings mit einer recht hohen Chance, dass Sie Version 2 und nicht 3 haben. Egal ob Sie mit macOS, Windows oder Linux arbeiten, Sie werden spätestens jetzt Python in der Version 3 installieren müssen.

Tun Sie das als Allernächstes. Sobald Python bei Ihnen läuft, können wir uns mit richtigem Code befassen.

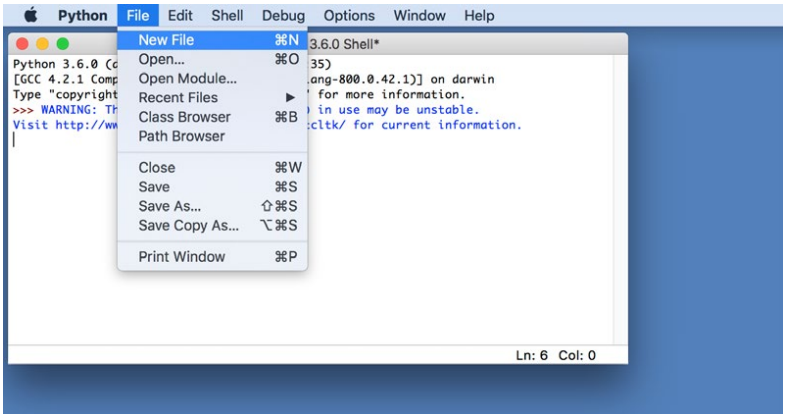
Python auf Herz und Nieren prüfen

Nachdem Python installiert ist, wollen wir es auch benutzen. Wir beginnen mit einem kleinen Testprogramm, um zu sehen, ob alles richtig funktioniert. Hierfür müssen Sie einen Editor einsetzen, um Ihr Programm einzugeben. Erst danach können Sie es ausführen. Und da kommt IDLE, der Python-Editor (bzw. die Python-IDE), ins Spiel. Öffnen Sie IDLE, wie in der Einführung gezeigt. Zur Erinnerung: Auf dem Mac finden Sie IDLE im Ordner **`Programme`** > **`Python 3.x`**. Unter Windows benutzen Sie den Start-Button, um zu Alle Programme zu navigieren. Sie finden IDLE im Python 3.x-Menü.



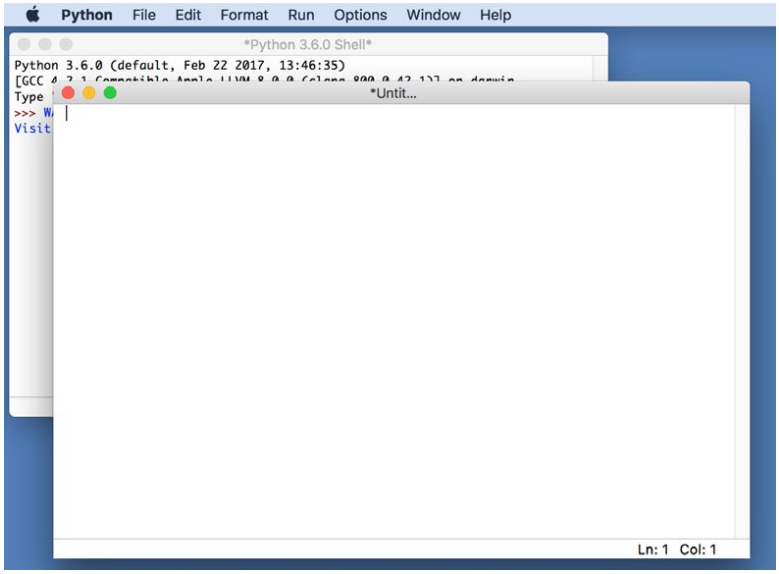
Wenn Sie IDLE zum ersten Mal öffnen, zeigt er einen interaktiven Interpreter an, der Python-Shell genannt wird. Wenn Sie sehr neugierig sind, geben Sie *1+1* (d. h. eins plus eins) ein und drücken dann die Enter-Taste.

Beim ersten Start von Python wird ein interaktives Fenster namens Python-Shell angezeigt. Hier können Sie Python-Anweisungen direkt eingeben. Sie können aber auch einen Editor benutzen. Wählen Sie hierfür den Befehl **`File`** > **`New File`** aus dem IDLE-Menü, und es erscheint ein leeres Editorfenster.



Benutzen Sie den Menübefehl File > New File, um ein neues Fenster für die Eingabe Ihres Python-Codes zu öffnen.

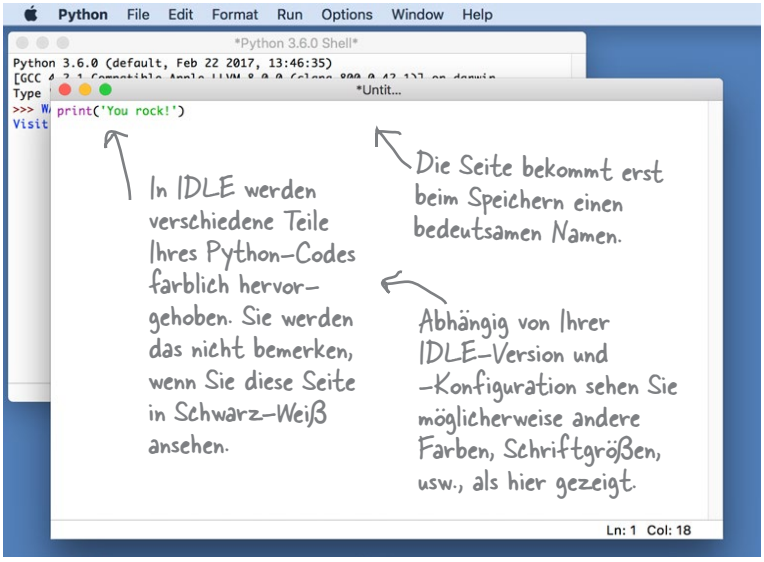
IDLE funktioniert wie ein Textverarbeitungsprogramm, das Python-Code versteht. Es hilft Ihnen durch die Hervorhebung von Python-Schlüsselwörtern, durch Codeformatierung und, wo angemessen, durch die automatische Vervollständigung von Python-Schlüsselwörtern, um die Codeeingabe zu vereinfachen.



Nach der Auswahl von New File sollten Sie ein neues leeres Fenster sehen, das das Fenster der Python-Shell überlagert.

Nach dem Anlegen einer neuen Datei erhalten Sie ein neues leeres Editorfenster. Hier geben Sie eine Zeile Code ein, um die Sache auszuprobieren, zum Beispiel:

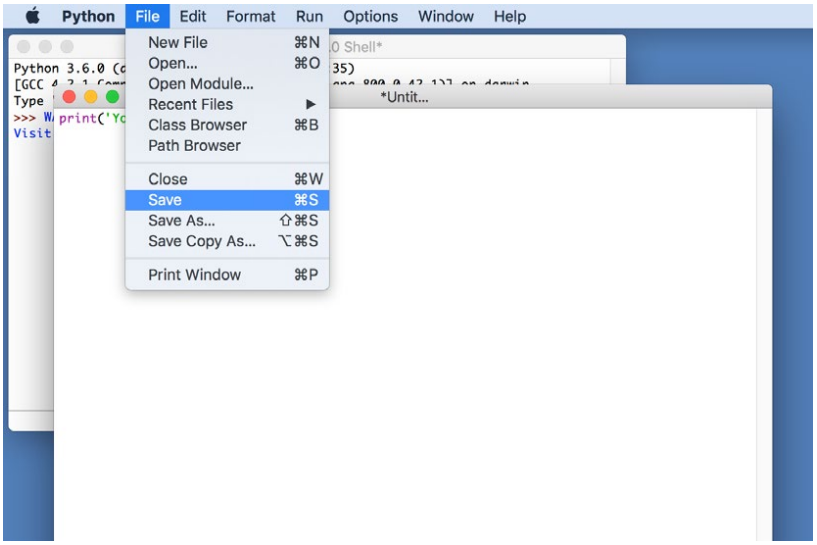
```
print('You rock!')
```



Achten Sie genau auf Schreibweise und Zeichensetzung, da Python und andere Sprachen diese Fehler meist nicht verzeihen.

Ihre Arbeit speichern

Nachdem Sie Ihre erste Codezeile eingegeben haben, wollen wir sie speichern. Hierfür wählen Sie einfach den Befehl **`File`** > **`Save`** aus dem IDLE-Menü:

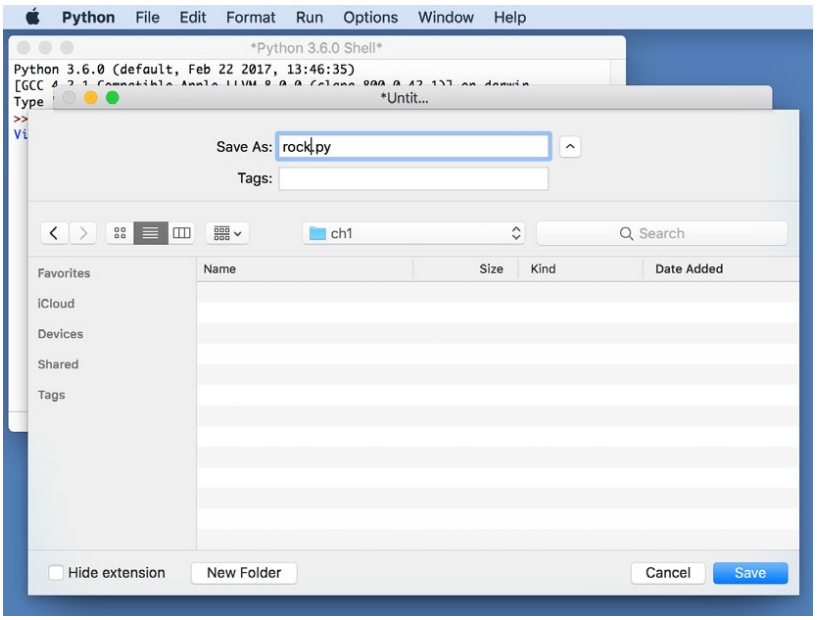


Vor dem Ausführen müssen wir Ihren Code erst speichern. In IDLE finden Sie den Menübefehl Save unter dem File-Menü. Wählen Sie diese Option und geben Sie Ihrem Quellcode einen Namen. Beim Schreiben von Python-Code benutzen wir übrigens die Dateiendung >.py<.

Quellcode, Quelldatei, Code und Programm sind einfach verschiedene Namen für die Dateien, die Ihren Code enthalten.

Wenn Sie Ihrem Code einen Namen geben, vergessen Sie nicht die Dateiendung *.py* . Wir haben uns für den Namen *rock.py*. entschieden. Und auch wenn das hier nicht zu sehen ist, haben wir für Kapitel 1 einen eigenen Ordner mit dem Namen *ch1* angelegt. Wir empfehlen Ihnen, das ebenfalls zu tun.

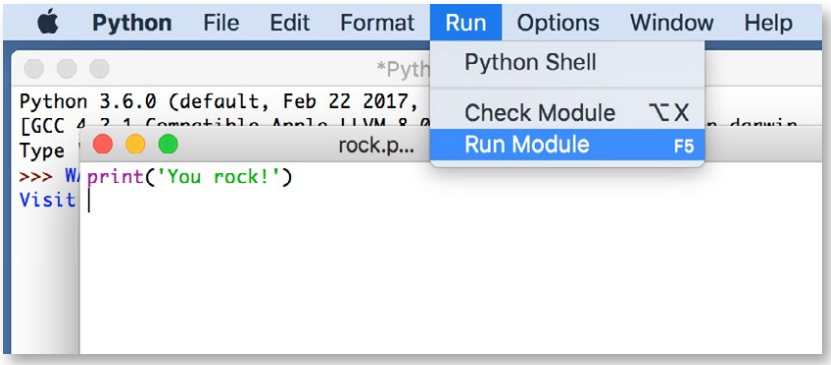
Vermutlich sollten Sie Ihre Dateien nach dem gleichen Schema benennen, wie wir das für dieses Buch getan haben. Werfen Sie einen Blick auf Seite xxxii in der Einführung zu diesem Buch, falls noch nicht geschehen.



Sobald Sie den Dateinamen und einen Ordner für Ihren Code haben, klicken Sie auf Save.

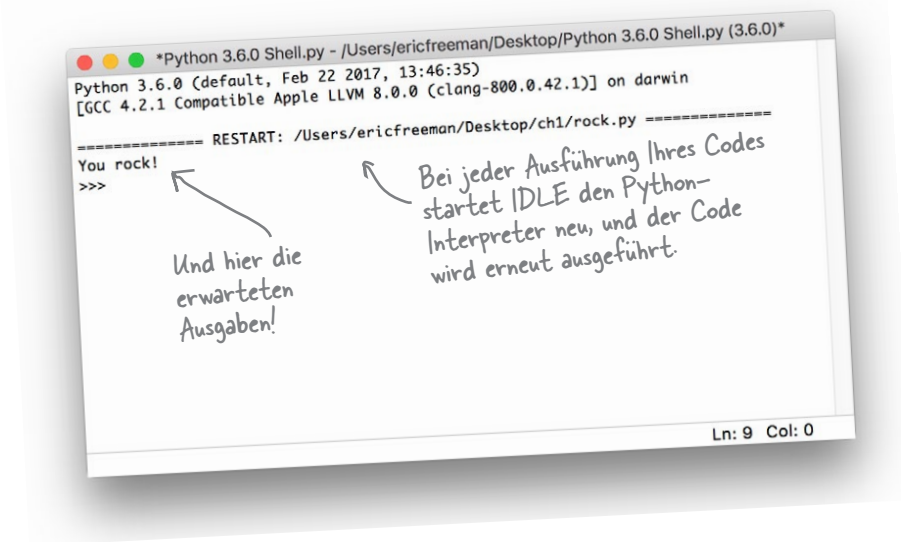
Hier kommt alles zusammen. Nachdem Sie Ihren Code gespeichert haben, wählen Sie den Menübefehl **Run > Run Module**. Die Ausgaben Ihres Programms können Sie in Pythons Shell-Fenster beobachten.

Führen Sie Ihren Code aus, indem Sie den Befehl Run Module aus dem Run-Menü wählen. Wenn Sie Ihren Code noch nicht gespeichert haben, wird IDLE Sie nun dazu auffordern.



Herzlichen Glückwunsch zu Ihrem ersten Python-Programm!

Sie haben Python installiert, Sie haben Ihr erstes Stück Python-Code mit IDLE eingegeben, und Sie haben auch gleich Ihr erstes Python-Programm ausgeführt. Der Code ist zwar noch nicht besonders komplex, aber irgendwo muss man halt anfangen. Die gute Nachricht: Damit können wir uns schon an unser erstes ernsthaftes Businessprogramm wagen.



Hatten Sie eine andere Ausgabe als »You rock!«?

Beim Schreiben und Testen passiert schnell mal ein Fehler. Wenn Sie beim ersten Versuch keinen Erfolg hatten, gewöhnen Sie sich besser gleich daran: Das Beseitigen von Fehlern im Code gehört zum Entwickleralltag. Hier ein paar Möglichkeiten:

- *Meldet Python einen Fehler wie* `invalid syntax` *(ungültige Syntax), überprüfen Sie Ihren Code auf falsche Zeichensetzung, fehlende Klammern und Ähnliches. IDLE hebt den fraglichen Code farblich hervor. So können Sie die Fehler meistens schon schnell finden.*
- *Meldet Python einen Fehler wie* `NameError: name 'prin' is not defined`*, sollten Sie Ihren Code auf Tippfehler, z. B. falsch geschriebene Namen, überprüfen (in unserem Beispiel das Wort* `print`*).*
- *Meldet IDLE einen Fehler wie* `EOL while scanning`*, so heißt das normalerweise, dass Sie bei der Zeichenkette* `'You rock'` *ein Anführungszeichen vergessen haben. Stellen Sie sicher, dass auf beiden Seiten des Strings ein einfaches Anführungszeichen steht, wie hier:* `'You rock!'`*.*
- *Und wenn Sie wirklich nicht weiterkommen, versuchen Sie es mal auf der Communityseite zu diesem Buch unter: wickedlysmart.com.*

Es gibt keine Dummen Fragen

F: Warum brauchen wir zum Ausführen unseres Codes das Run-Modul?

A: Python ruft eine Datei mit Python-Code auf, die *Modul* genannt wird. Die Anweisung lautet einfach: »Führe sämtlichen Python-Code in meiner Datei aus.« Module sind eine weitere Möglichkeit, Code zu speichern, um den sich Python erst später kümmern soll.

F: Was heißt Ein- und Ausgabe genau?

A: Im Moment haben wir es nur mit einfachen Ein- und Ausgaben zu tun. Unsere Ausgabe ist der Text, den Ihr Programm erzeugt und dann im Python-Shell-Fenster angezeigt hat. Umgekehrt ist eine Eingabe der Text, der im Shell-Fenster an Ihr Programm übergeben wird. Grundsätzlich sind fast alle Ein- und Ausgabearten möglich, wie solche mit einer Maus oder einem Touchpad, Grafiken, Töne usw.

F: Ich habe kapiert, dass ich per print Text an den Benutzer übergeben kann. Aber warum dieser Name? Der klingt wie eine Anweisung, etwas auszudrucken (engl. to print).

A: Es ist noch gar nicht so lange her, da fanden Computerausgaben viel öfter über einen Drucker statt als auf dem Bildschirm. Damals war der Name **print** daher sinnvoller als heute. Natürlich reicht das für Python nicht als Ausrede. Dafür ist es noch zu jung. Allerdings ist **print** in vielen Sprachen der Standardbefehl für eine Ausgabe. **print** erzeugt also eine Bildschirmausgabe (mithilfe des Python-Shell-Fensters). Und ein Beispiel für **input**, bei dem Benutzereingaben über die Python-Shell entgegengenommen werden, haben Sie übrigens auch schon gesehen.

F: Ist print die einzige Möglichkeit, Dinge mit Python auszugeben?

A: Nein, es ist nur die einfachste Methode. Computer und Programmiersprachen besitzen eine Vielzahl an Ein- und Ausgabemöglichkeiten, auch Python. Mit Python (und den meisten anderen Sprachen) können Sie Dinge auf Webseiten, im Netzwerk, in Dateien auf Speichermedien, Grafik- und Audiogeräten und vielen anderem ausgeben.

F: Okay. Bei der Benutzung von print schreibe ich etwas wie print('Moin allerseits'). Was passiert da genau?

A: Sie benutzen eine Python-eigene Funktion, um eine Ausgabe vorzunehmen. Das heißt, Sie weisen eine Funktion mit dem Namen **print** an, den Text innerhalb der Anführungszeichen zu nehmen und auf der Python-Shell auszugeben. Wir werden später genauer erklären, was Funktionen sind, was Text ist und so weiter. Jetzt reicht es zu wissen, dass Sie **print** benutzen können, um alles, was Sie wollen, auf der Shell auszugeben.

Python im Gespräch

Interview der Woche: Meinen Sie es ernst?

Von Kopf bis Fuß: Willkommen, Python. Wir freuen uns schon darauf, Sie näher kennenzulernen.

Python: Danke. Ich freue mich, hier zu sein.

Von Kopf bis Fuß: Ihr Name stammt also von einer Comedy-Truppe, und Sie sind als Anfängersprache bekannt. Mal ehrlich: Kann man Sie überhaupt ernst nehmen?

Python: Eigentlich werde ich für alles Mögliche benutzt, von der Chipfertigung für Programme, die bei der Produktion großer Kinofilme geholfen haben (hat etwa gerade jemand »George Lucas« gesagt? Wir sicher nicht …), bis zum Ansteuern von Schnittstellen für die Flugsicherung. Ich könnte noch mehr aufzählen … Klingt das ernsthaft genug?

Von Kopf bis Fuß: Nun gut, wenn Sie so eine ernsthafte Sprache sind, wie können Sie dann derart einfach für Neueinsteiger sein? Die Projekte, die Sie beschreiben, klingen doch ziemlich komplex. Sollen wir wirklich glauben, dass keine komplexe Hardcore-Sprache nötig ist, um das alles zu schaffen?

Python: Einer der Gründe dafür, dass Anfänger *und Profis* mich gleichermaßen schätzen, ist, dass mein Code so schlicht und gut lesbar ist. Haben Sie sich mal eine Sprache wie Java angesehen? Pfui Deibel. Meine Güte! Der Aufwand, den Sie treiben müssen, um einfach nur »Hallo Welt!« zu sagen. In Python geht das mit einer einzigen Codezeile.

Von Kopf bis Fuß: Okay. Sie sind gut lesbar. Wunderbar, aber was heißt das überhaupt?

Python: Nachdem ich Java schon erwähnt habe, hier ein kleines Beispiel. Angenommen, Sie wollten den Benutzer einfach mit »Moin!« begrüßen. In Java macht man das so:

```
class HelloWorldApp {

  public static void main(String[] args) {

    System.out.println("Moin!");

    }

  }
```

Da muss man erst mal durchsteigen. Ich nenne das komplett unlesbar, besonders für Leute, die gerade erst programmieren lernen. Was zum Henker soll das alles bedeuten? Ist das wirklich nötig? Und jetzt sehen Sie sich mal meine Version an, also natürlich in Python geschrieben:

```
print('Moin!')
```

Ich denke mal, wir sind uns einig, dass das deutlich einfacher und lesbarer ist. Ein Blick auf die Zeile reicht, um zu wissen, was hier passiert. Und das war noch ein simples Beispiel. Im Großen und Ganzen nimmt jeder Python – fast schon wie Englisch – als klar und konsistent wahr.

Von Kopf bis Fuß: Was meinen Sie mit »konsistent«?

Python: Konsistenz heißt, dass es in der Sprache nur wenige Überraschungen gibt. Anders gesagt: Wenn Sie einen kleinen Teil der Sprache verstanden haben, werden die anderen Teile so funktionieren, wie Sie es erwarten. Nicht alle Sprachen sind so.

Von Kopf bis Fuß: Ich möchte noch mal auf etwas zurückkommen, das Sie vorhin gesagt haben. Sie haben da ein paar recht esoterische Beispiele genannt, wie Flugsicherung, Chipherstellung, die Verwendung als Hauptsoftware für das Spaceshuttle und so weiter. Das klingt alles ziemlich industriell und sehr spezialisiert. Ich bin mir nicht sicher, ob Python wirklich die beste Sprache für unsere Leser ist.

Python: Das Spaceshuttle? Das haben Sie sich ausgedacht. Ich habe Ihnen Beispiele für Dinge genannt, die Sie *als ernst ansehen könnten*, nachdem Sie vorhin meinten, das sei Python nicht. Python wird unter anderem häufig für die Erstellung von Websites, das Schreiben von Spielen und sogar für die Erstellung von Desktopprogrammen verwendet.

Von Kopf bis Fuß: Themawechsel, okay? Jemand hat mir einen Tipp gegeben. Unsere Quellen sagen, es gibt eigentlich *zwei Versionen* von Python, und die sind auch noch … oh Mann, wie sage ich das jetzt … *miteinander inkompatibel*. Wie um alles in der Welt können Sie das konsistent nennen?

Python: Wie alles, müssen auch Sprachen wachsen und sich entwickeln. Und ja, es gibt zwei Versionen von Python, 2 und 3. Version 3 enthält alles Neue, was noch nicht Teil von Version 2 war. Allerdings gibt es Möglichkeiten, die Dinge rückwärtskompatibel zu machen. Wir zeigen Ihren Lesern gern, wie das geht …

Von Kopf bis Fuß: … leider ist unsere Zeit gerade um. Wir freuen uns schon auf unseren nächsten Hinterhalt – ich meine natürlich, die nächste Gelegenheit, mit Ihnen zu sprechen.

Python: Vielen Dank. Das Vergnügen war ganz meinerseits … denke ich.

Mit meinem neuen Phrasendrescher können Sie reden wie der Boss oder die Leute aus der Marketingabteilung …

Okay. Jetzt wird's ernst. Wir wollen unsere erste echte Businessapplikation mit Python schreiben. Dieser Phrasendrescher-Code wird Sie beeindrucken.

Entspannen Sie sich

Im Ernst. Wir wollen, dass Sie sich auf den Code voll und ganz einlassen. Sehen Sie sich die Codezeilen an, lesen Sie die Beschreibungen und lassen Sie die Informationen in sich hineinfließen. Wir haben ein paar Python-Kommentare eingefügt. Das sind einfach ein paar Hinweise nach den #-Zeichen, damit Sie besser verstehen, was der Code hier tut. Kommentare sind eine Art hilfreicher Pseudocode. Später in diesem Buch werden wir Kommentare noch genauer betrachten. Wenn Sie glauben, den Code einigermaßen verstanden zu haben, gehen Sie weiter zur nächsten Seite, wo wir Ihnen die Sache etwas detaillierter erklären.

1 **# Python mitteilen, dass wir ein paar Zufallsfunktionen**
 # benutzen wollen, indem wir das random-Modul importieren

```
import random
```

2 **# Drei Listen erstellen: eine mit Verben, eine mit**
 # Adjektiven und eine mit Substantiven

```
verbs = ['Leverage', 'Sync', 'Target',
         'Gamify', 'Offline', 'Crowd-sourced',
         '24/7', 'Lean-in', '30,000 foot']

adjectives = ['A/B Tested', 'Freemium',
              'Hyperlocal', 'Siloed', 'B-to-B',
              'Oriented', 'Cloud-based',
              'API-based']

nouns = ['Early Adopter', 'Low-hanging Fruit',
         'Pipeline', 'Splash Page', 'Productivity',
         'Process', 'Tipping Point', 'Paradigm']
```

3 **# Aus jeder Liste ein Verb, Adjektiv und Substantiv auswählen**

```
verb = random.choice(verbs)
adjective = random.choice(adjectives)
noun = random.choice(nouns)
```

4 **# Die Phrase erstellen, indem wir die Wörter »addieren«**

```
phrase = verb + ' ' + adjective + ' ' + noun
```

5 **# Die Phrase ausgeben**

```
print(phrase)
```

Phrasendrescher

Einfach gesagt, übernimmt das Programm drei Listen mit Wörtern, wählt aus jeder Liste zufällig ein Wort aus und kombiniert diese zu einer Phrase (die Sie als Marketing-Slogan für Ihr nächstes Start-up-Unternehmen benutzen können). Danach wird die Phrase ausgegeben. Keine Sorge, wenn Sie jetzt noch nicht jedes Detail Ihres Programms verstehen. Schließlich sind Sie erst auf Seite 25 von über 600. Es geht erst mal darum, sich mit dem Code vertraut zu machen:

1. Die `import`-Anweisung teilt Python mit, dass wir weitere eingebaute Funktionen einsetzen wollen, die sich im Modul `random` befinden. Damit werden die Möglichkeiten Ihres Codes erweitert – in diesem Fall um die Fähigkeit, Dinge zufällig auszuwählen. Wie `import` im Detail funktioniert, werden wir weiter unten in diesem Buch noch sehen.

2. Danach müssen wir drei Listen erstellen. Die Deklaration einer Liste ist einfach. Umgeben Sie jedes Listenelement mit Anführungszeichen und umgeben Sie die Liste mit eckigen Klammern, wie hier:

```
verbs = ['Leverage', 'Sync', 'Target',
         'Gamify', 'Offline', 'Crowd-sourced',
         '24/7', 'Lean in', '30,000 foot']
```

   Hier weisen wir der Liste einen Namen zu, z. B. `verbs`, damit wir später im Code wieder darauf zurückgreifen können.

Slogans für Ihr nächstes Python-basiertes Start-up-Unternehmen

24/7 Freemium Productivity

Lean-in Hyperlocal Splash Page

Gamify Siloed Early Adopter

Offline API-based Process

Crowd-sourced Cloud-based Pipeline

3. Als Nächstes müssen wir aus jeder Liste ein zufälliges Wort auswählen. Hierfür benutzen wir `random.choice`. Diese Funktion übernimmt eine Liste und wählt zufällig ein Element aus. Dieses weisen wir dann dem passenden Namen (`verb`, `adjective` oder `noun`) zu, damit wir später darauf zurückgreifen können.

   random.choice ist eine weitere Python-eigene Funktion, zu der wir später im Buch noch mehr lernen werden.

4. Dann müssen wir die Phrase zusammenbauen, indem wir Verb (verb), Adjektiv (adjective) und Substantiv (noun) zusammenbauen. In Python benutzen wir dafür das Pluszeichen. Vergessen Sie nicht, dass auch die Leerzeichen zwischen den Wörtern gebraucht werden. Sonst bekommen wir Phrasen wie »Lean-in-Cloud-basedPipeline«.

   Informatiker nennen dieses Zusammenbauen von Text auch »Verkettung« oder englisch »Concatenation«. Ein hübsches kleines Wort, das wir hier noch häufiger sehen werden.

5. Zum Schluss geben wir die Phrase mit der print-Anweisung in der Python-Shell aus, *und schon sprechen wir perfektes Marketing-Chinesisch.*

Den Code in die Maschine bekommen

Nachdem Sie ein Programm in IDLE eingegeben haben, wollen wir die einzelnen Schritte noch einmal betrachten. Hierfür legen wir mit dem Menübefehl **`File`** > **`New File`** eine neue Python-Datei an. Danach geben Sie den Code der vorletzten Seite in den Editor ein.

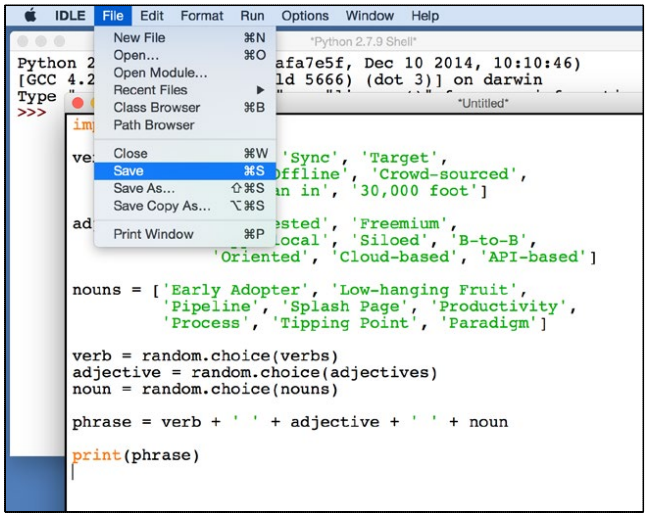
Hier ein Beispiel für die Hervorhebung eines Tippfehlers. Wenn Ihnen das passiert, sollten Sie noch einmal genau nachsehen, ob Ihr Code korrekt ist.

Achten Sie besonders auf die verwendeten Wörter und die Zeichensetzung. Wir werden später noch genau darauf eingehen. Im Moment reicht es, wenn Sie sich schon mal daran gewöhnen.

IDLE markiert häufige Fehler automatisch. Je nach Fehler kann es beim Schreiben oder beim Ausführen Ihres Codes zu Fehlermeldungen kommen. Hier wurde ein fehlendes Anführungszeichen bemerkt und markiert. In so einem Fall sollten Sie Ihren Code noch einmal überprüfen und korrigieren.

Um zu zeigen, was IDLE macht, haben wir hier absichtlich einen Fehler eingebaut. Wenn Sie den Code der vorletzten Seite richtig eingegeben haben, sollte dieser Fehler nicht auftauchen.

Sobald der Code fertig eingegeben ist, wollen wir ihn speichern. Benutzen Sie hierfür einfach den IDLE-Menübefehl **`File`** > **`Save`** und geben Sie der Datei den Namen *phraseomatic.py* (der englische Name des Programms).



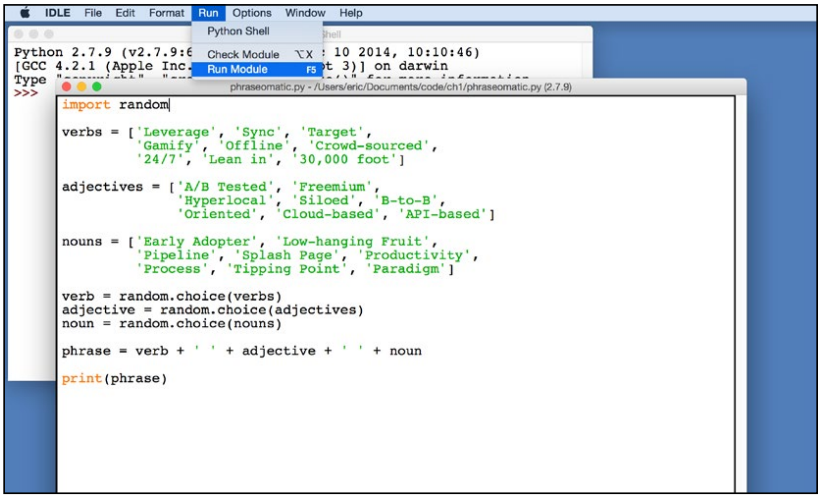
Wie Sie sehen, hebt IDLE Codeteile je nach ihrer Funktion farblich hervor.

Nach dem Drücken der Enter-Taste fügt IDLE außerdem die nötigen Einrückungen ein.

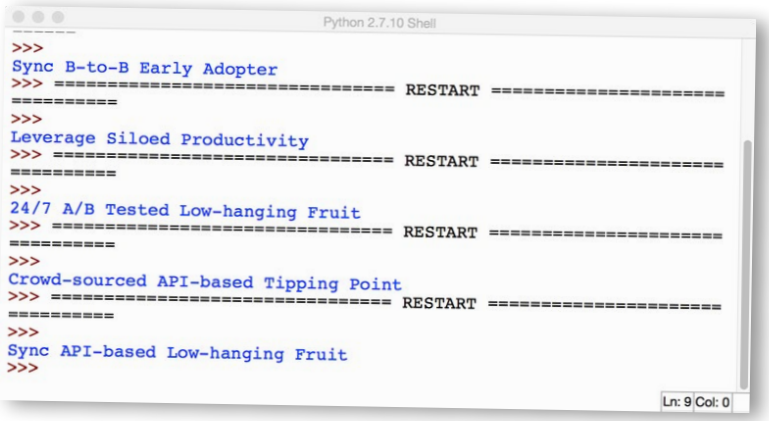
Wir benutzen zusätzliche Leerzeichen und Zeilenumbrüche, um den Code lesbarer zu machen. Python ignoriert Whitespace-Zeichen (auf eine Ausnahme kommen wir noch zu sprechen).

Jetzt wollen den Phrasendrescher mal laufen lassen. Gehen wir die Schritte noch einmal genau durch, damit Sie es auch wirklich verstehen. Nach dem Speichern Ihres Codes wählen Sie den Menübefehl **Run > Run Module**. Danach finden Sie Ihren nächsten Marketing-Slogan direkt im Shell-Fenster.

Um Ihren Code auszuführen, benutzen Sie den Befehl Run Module aus dem Run-Menü. Wenn Sie Ihren Code noch nicht gespeichert haben, wird IDLE Sie jetzt dazu auffordern.



Wir haben den Phrasendrescher ein paarmal laufen lassen. Hier sind die Ergebnisse.



Um den Phrasendrescher erneut laufen zu lassen, klicken Sie zuerst auf das Codefenster und wählen dann erneut den Menübefehl Run > Run Module.

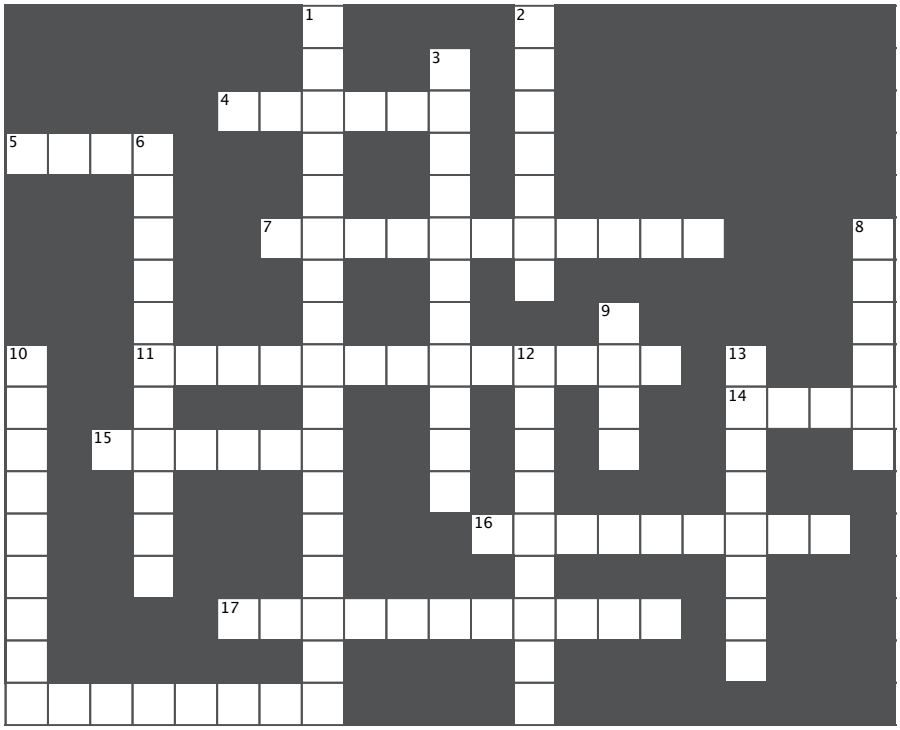
Punkt für Punkt

- Um Code zu schreiben, müssen Sie das Problem in mehrere einfache Aktionen aufteilen, die es lösen.
- Diesen Satz von Aktionen bezeichnen wir als Algorithmus oder, weniger formal, als Rezept zur Problemlösung.
- Aktionen haben die Form von Anweisungen, die sehr einfache Aufgaben ausführen, Entscheidungen treffen oder den Ablauf des Algorithmus beeinflussen, z. B. indem Teile des Codes wiederholt werden.
- Rechnerisches Denken ist eine bestimmte Art, über Problemlösungen nachzudenken, und stammt ursprünglich aus der Informatik.
- Programmieren ist die Übersetzung der Einzelschritte eines Algorithmus in eine Programmiersprache, die von einem Computer ausgeführt werden kann.
- Algorithmen werden manchmal in menschenlesbarem Pseudocode ausgedrückt, bevor sie in eine tatsächliche Programmiersprache übersetzt werden.
- Programmiersprachen sind spezielle Sprachen, mit denen Aufgaben für einen Computer beschrieben werden können.
- Natürliche Sprachen sind wegen ihrer vielen Mehrdeutigkeiten nicht gut als Programmiersprachen geeignet.
- Es gibt viele Programmiersprachen. Alle haben ihre Vor- und Nachteile, benutzen aber die gleiche Rechenleistung.
- Der Name Python hat nichts mit einer Schlange zu tun, sondern damit, dass ihr Erfinder die Comedytruppe Monty Python mag.
- Neue und erfahrene Programmierer wissen Pythons sauberes und konsistentes Design zu schätzen.
- Es gibt zwei Versionen von Python, 2 und 3. In diesem Buch geht es um Python 3 (auch wenn die Unterschiede oftmals nur gering sind).
- Python-Code wird mit einem Interpreter ausgeführt. Dieser übersetzt die menschenlesbaren Anweisungen in Maschinencode, den Ihr Computer direkt ausführt.
- Python stellt einen Editor namens IDLE zur Verfügung, der speziell auf das Schreiben von Python-Code zugeschnitten ist.
- Durch die Verwendung von Leerzeichen können Sie die Lesbarkeit eines Python-Programms erhöhen.
- **input** und **print** sind zwei von Python bereitgestellte Funktionen, mit denen Sie einfache Shell-basierte Ein- und Ausgaben vornehmen können.

Programmierer-Kreuzworträtsel

Jetzt soll Ihre rechte Hirnhälfte auch etwas zu tun bekommen.

Hier sehen Sie ein ganz normales Kreuzworträtsel. Alle Wörter stammen aus Kapitel 1.



Waagerecht

4. Eine der besten Anfängersprachen.
5. Nach Kaffee benannte Sprache.
7. Technischer Name für Rezept.
11. Direkt vom Computer ausführbarer Code.
14. Anderer Ausdruck für Algorithmus.
15. Flying _____.
16. Ein Programm laufen lassen.
17. Denkweise, die dieses Buch vermittelt.
18. Menschenlesbarer Code.

Senkrecht

1. Zu mehrdeutig zum Programmieren.
2. Steht im Von Kopf bis Fuß-Restaurant auf der Speisekarte.
3. Hiermit wird der Python-Code ausgeführt.
6. Wir übersetzen das Problem in ein _____ Rezept.
8. Eine Firma, die Python einsetzt.
9. Name der Python-IDE.
10. Wird an einen Interpreter oder Compiler übergeben.
12. Hier finden Sie Unterstützung.
13. Ein anderes Wort für Quellcode.

Spitzen Sie Ihren Bleistift
Lösung

Echte Rezepte sind keine reinen *Handlungsanweisungen*, denn sie enthalten auch eine Reihe von Objekten, die für die Zubereitung einer bestimmten Speise nötig sind (wie Messbecher, Schneebesen, Küchenmaschinen und natürlich die Zutaten). Welche Objekte werden in unserem Angelrezept verwendet?

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

Spitzen Sie Ihren Bleistift
Lösung

Eines müssen Sie noch wissen: Computer tun immer **genau** das, was Sie Ihnen sagen – nicht mehr und auch nicht weniger. Sehen Sie das Rezept zum Angeln auf Seite 2 noch mal an. Welche Probleme gäbe es, wenn Sie der Roboter wären und die Anweisungen genau befolgen müssten? Glauben Sie, dass dieses Rezept wirklich zum Erfolg führen würde?

- ☑ A. Wurden keine Fische gefangen, werden Sie eine sehr lange Zeit angeln (eigentlich für immer).
- ☑ B. Löst sich der Köder vom Haken, werden Sie das nie erfahren oder ihn ersetzen.
- ☑ C. Was passiert, wenn wir keinen Köder mehr haben?

Sieht aus, als müssten Sie alles anhaken!

- ☑ D. Haben wir angegeben, was mit dem Fisch zu tun ist, nachdem wir ihn eingeholt haben?
- ☑ E. Was passiert mit der Angelrute?
- ☑ F. Gibt es weitere Angaben dazu, was ein guter Wurf ist? Müssen wir die Leine erneut auswerfen, wenn der Köder auf einem Seerosenblatt landet?

Typischerweise müssen wir den Fisch >>anhaken<<, bevor wir die Leine einholen. Es wird aber nicht gesagt, wie man das macht.

Wie finden wir heraus, ob wir mit dem Angeln fertig sind? Nach einer bestimmten Zeit? Wenn wir keine Köder mehr haben? Irgendetwas anderes? An vielen Stellen gehen wir in unserem Rezept von Annahmen aus. Sie haben bestimmt noch viele weitere Anweisungen gefunden, die eigentlich fehlen.

Code-Magneten, Lösung

Wir haben den Algorithmus des Von Kopf bis Fuß-Restaurants für die Zubereitung eines Omeletts aus drei Eiern an den Kühlschrank geheftet. Blöderweise hat irgendjemand den Code durcheinandergebracht. Können Sie die Magneten wieder in die richtige Reihenfolge bringen, damit unser Algorithmus funktioniert? Bedenken Sie, dass es im Von Kopf bis Fuß-Restaurant Omeletts mit und ohne Käse gibt.

Hier sind die richtig geordneten Magneten!

Es gibt mehrere korrekte Variationen, die hier funktionieren. Stellen Sie einfach sicher, dass Sie unsere Lösung verstehen und dass Ihre gegebenenfalls abweichende Lösung einen logischen Sinn ergibt.

```
Pfanne vorwärmen
```

```
Drei Eier aufschlagen und in eine Schüssel geben
```

Zu Beginn richten wir alles ein: das Vorheizen der Pfanne und das Aufschlagen der Eier.

```
Solange die Eier nicht komplett vermischt sind:
```

Dann schlagen wir die Eier, bis sie gut verrührt sind.

```
    Eier schlagen
```

Wie Sie sehen, haben wir das Schlagen eingerückt, um anzuzeigen, dass es so lange ausgeführt wird, bis die Eier komplett verrührt sind. Falls Sie das anders gekennzeichnet haben, ist es auch in Ordnung.

```
Eier in die Pfanne geben
```

Dann geben wir die Eier in die vorgeheizte Pfanne.

```
Solange die Eier nicht komplett gar sind:
```

Und braten sie, bis sie gar sind.

```
    Eier umrühren
```

Beachten Sie, dass wir das Schlagen der Eier nur eingerückt haben, um zu zeigen, dass sie so lange geschlagen werden, bis sie komplett verrührt sind. Haben Sie das auf andere Weise vermerkt – auch okay.

```
Sofern der Kunde Käse bestellt hat:
```

Hat der Kunde Käse bestellt, bestreuen wir das Omelett damit.

```
    Mit Käse bestreuen
```

Wir haben auch den Käse-Teil eingerückt, weil er nur ausgeführt wird, wenn der Kunde Käse bestellt hat.

```
Pfanne vom Herd nehmen
```

Unabhängig davon nehmen wir die Pfanne vom Herd und legen das Omelett auf den Teller.

```
Omelett auf den Teller legen
```

```
Servieren
```

Zum Schluss können wir das Omelett endlich servieren.

Du sagst Tomate ... Lösung

Auf der linken Seite sehen Sie einige Anweisungen in deutscher Sprache, rechts daneben sehen Sie die Anweisungen in einer Programmiersprache. Verbinden Sie die deutsche Anweisung mit der entsprechenden Codeübersetzung. Die erste Verbindung haben wir schon für Sie hergestellt.

Gib »Hi there« auf dem Bildschirm aus.

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

Ein Einkaufszettel mit Brot, Milch und Eiern.

Fünf Drinks einschenken.

Frage den Benutzer: What is your name?

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

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

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

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

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

Programmierer-Kreuzworträtsel, Lösung