3 Berechnungen und Variablen

Du hast Python installiert und weißt, wie man die Python-Shell startet. Jetzt kannst Du etwas damit machen. Wir fangen mit ein paar einfachen Berechnungen an und wenden uns dann den Variablen zu. *Variablen* sind eine Möglichkeit, Dinge in einem Computerprogramm zu speichern. Sie helfen uns dabei, nützliche Programme zu schreiben.

3.1 Mit Python rechnen

Wenn jemand Dich jemand nach dem Produkt einer Multiplikation (wie z.B. 8×3,57) fragt, würdest Du normalerweise zum Taschenrechner greifen oder es auf dem Papier schriftlich ausrechnen. Du kannst aber auch die Python-Shell verwenden, um die Berechnung durchzuführen. Wir machen das jetzt einmal.

Starte die Python-Shell durch Doppelklick auf das IDLE-Icon oder, wenn Du Ubuntu nutzt, auf das IDLE-Icon im **Programme**-Menü. Nach dem Prompt gibst Du diese Gleichung ein:

```
>>> 8 * 3.57
28.56
```

Achtung: Wenn Du eine Multiplikation in Python eingibst, musst Du das Sternzeichen (*) statt eines Multiplikationszeichens (x) verwenden. Als Dezimaltrennzeichen musst Du in Python den Punkt (.) nehmen, kein Komma (,).

Wie wäre es, wenn wir jetzt eine Gleichung eingeben, die noch nützlicher ist?

Jason Briggs, Python kinderleicht!, dpunkt.verlag, ISBN 978-3-86490-344-1

Stell Dir vor, Du findest beim Graben im Garten eine Tasche mit 20 Goldmünzen. Am nächsten Tag schleichst Du Dich in den Keller und steckst die Münzen in die dampfgetriebene Kopiermaschine, die Dein Großvater erfunden hat (glücklicherweise passen genau 20 Münzen hinein). Du hörst ein Rollen und Stampfen, und nach ein paar Stunden kommen 10 weitere glänzende Goldmünzen heraus.

Wie viele Goldmünzen hättest Du in Deiner Schatzkiste, wenn Du das ein Jahr lang jeden Tag machen würdest? Auf dem Papier gerechnet, könnte das so aussehen:

$10 \times 365 = 3650$
$20 + 3650 = 3670$

Natürlich könnte man diese Berechnungen mit einem Taschenrechner oder schriftlich ganz einfach durchführen, aber in der Python-Shell geht das genauso gut. Als Erstes multiplizieren wir die 10 Münzen mit 365 Tagen eines Jahres und bekommen 3650. Danach zählen wir unsere 20 Original-Münzen dazu und erhalten 3670.

```
>>> 10 * 365
3650
>>> 20 + 3650
3670
```

Was wäre nun aber, wenn eine Elster Deine glänzenden Goldmünzen in Deinem Schlafzimmer entdecken würde und jede Woche hineingeflogen käme und dabei jeweils drei Münzen stehlen würde? Wie viele Münzen hättest Du dann nach einem Jahr? So sieht die Berechnung in der Shell aus:

```
>>> 3 * 52
156
>>> 3670 - 156
3514
```

Als Erstes multiplizieren wir 3 Münzen mit der Anzahl der Wochen eines Jahres, also 52. Das Ergebnis ist 156. Diese Zahl ziehen wir von unserer Gesamtanzahl von Münzen nach einem Jahr (3670) ab. So haben wir 3514 Münzen nach einem Jahr.

Dies war ein sehr einfaches Programm. In diesem Buch wirst Du lernen, wie man diese Konzepte beim Programmeschreiben immer weiter ausbaut und noch nützlichere Programme schreibt.

Operatoren in Python

In Python kann man Multiplikationen, Additionen, Subtraktionen und Divisionen in der Shell durchführen, aber auch andere mathematische Operationen, die wir jetzt nicht besprechen. Die grundlegenden Symbole, die Python für mathematische Operationen benutzt, heißen *Operatoren*. Sie sind in Tabelle 3–1 aufgeführt

Symbol	Operation
+	Addition
-	Subtraktion
*	Multiplikation
/	Division

Tab. 3–1 Grundlegende Operatoren in Python

Der *Vorwärtsschrägstrich* (/) wird bei Divisionen verwendet, da er an den Bruchstrich erinnert. Wenn Du zum Beispiel 100 Piraten und 20 große Fässer hättest und wissen möchtest, wie viele Piraten Du in ein Fass stecken müsstest, könntest Du 100 Piraten durch 20 Fässer teilen (100÷20) und `100/20` in die Shell eingeben. Der Vorwärtsschrägstrich ist derjenige, der nach rechts fällt. (Du findest ihn auf der Tastatur über der Ziffer 7.)

Die Rangfolge der Operationen

Um die Rangfolge von Operationen in einer Programmiersprache zu bestimmen, benutzen wir Klammern. Als *Operation* bezeichnet man alles, was Operatoren benutzt. Multiplikation und Division haben einen höheren Rang als Addition und Subtraktion, sie werden also als Erstes ausgeführt. Oder anders gesagt: Wenn Du in Python eine Gleichung eingibst, werden die Multiplikationen und Divisionen vor den Additionen und Subtraktionen ausgeführt.

Im folgenden Beispiel werden die Zahlen 30 und 20 zuerst multipliziert, und zu dem Produkt wird dann die Zahl 5 addiert.

```
>>> 5 + 30 * 20
605
```

Diese Gleichung bedeutet: »Multipliziere 30 mit 20 und addiere zum Produkt 5 dazu.« Das Ergebnis ist 605. Die Reihenfolge der Operationen können wir durch Klammern um die ersten beiden Zahlen ändern, und zwar so:

```
>>> (5 + 30) * 20
700
```


Jason Briggs, Python kinderleicht!, dpunkt.verlag, ISBN 978-3-86490-344-1

Das Ergebnis dieser Gleichung ist jetzt 700 (und nicht mehr 605), da die Klammern Python sagen, dass es zuerst die Operation innerhalb der Klammer ausführen soll und erst danach die Operation außerhalb der Klammer. Dieses Beispiel sagt also: »Addiere 5 zu 30, und multipliziere die Summe mit 20.«

Klammern können auch verschachtelt werden. Das heißt, dass Klammern innerhalb von Klammern verwendet werden können, z. B. so:

```
>>> ((5 + 30) * 20) / 10
70.0
```

In diesem Fall berechnet Python erst, was innerhalb der innersten Klammern steht, danach die Anweisung in den äußeren Klammern und zum Schluss die Division: »Addiere 5 zu 30, multipliziere die Summe mit 20, und teile das Produkt durch 10.« So läuft es ab:

- 5 addiert zu 30 ergibt 35.
- 35 mit 20 multipliziert, ergibt 700.
- 700 durch 10 dividiert, ergibt am Ende 70.

Ohne Klammern wäre das Ergebnis ein klein wenig anders:

```
>>> 5 + 30 * 20 / 10
65.0
```

In diesem Fall wird 30 erst mit 20 multipliziert (ergibt 600) und 600 durch 10 geteilt (ergibt 60). Zum Schluss wird 5 addiert, und es kommt 65 dabei heraus.

Achtung!

Achte darauf, dass Multiplikation und Division immer vor Addition und Subtraktion durchgeführt werden (»Punktrechnung geht vor Strichrechnung«) – es sei denn, dass Klammern die Rangfolge der Operationen regeln.

3.2 Variablen sind wie Bezeichnungen

Beim Programmieren steht das Wort *Variable* für einen Platz, an dem Informationen wie Zahlen, Text, Listen von Zahlen und Text usw. gespeichert werden. Eine andere Art, sich eine Variable vorzustellen, ist die, dass sie eine Bezeichnung für etwas ist.

Um zum Beispiel eine Variable mit dem Namen `fred` zu erzeugen, nehmen wir ein Gleichheitszeichen (=) und sagen Python, für welche Information die Variable eine Bezeichnung sein soll. Hier erzeugen wir jetzt die Variable `fred` und sagen, dass sie für die Zahl 100 steht (was nicht heißt, dass eine andere Variable nicht den gleichen Wert haben könnte):

```
>>> fred = 100
```

Um herauszufinden, für welchen Wert eine Variable steht, gibst Du in der Shell den Befehl `print` und danach den Namen der Variable in Klammern ein, und zwar so:

```
>>> print(fred)
100
```

Wir können Python auch sagen, dass die Variable `fred` geändert werden soll, sodass sie für etwas anderes steht. So zum Beispiel ändert man `fred` in die Zahl 200:

```
>>> fred = 200
>>> print(fred)
200
```

In der ersten Zeile sagen wir, dass fred für die Zahl 200 steht. In der zweiten Zeile fragen wir, für was `fred` steht, um uns die Änderung bestätigen zu lassen. Python gibt das Ergebnis in der letzten Zeile aus.

Wir können auch mehr als eine Bezeichnung (mehr als eine Variable) für die gleiche Sache verwenden:

```
>>> fred = 200
>>> john = fred
>>> print(john)
200
```

In diesem Beispiel sagen wir Python, dass wir den Namen (oder die Variable) `john` benutzen wollen, um die gleiche Sache damit zu bezeichnen wie mit `fred`. Dazu setzen wir einfach ein Gleichheitszeichen zwischen `john` und `fred`.

Natürlich ist `fred` wahrscheinlich kein sehr guter Name für eine Variable, da er kaum etwas darüber aussagt, wofür die Variable gebraucht wird. Statt `fred` nennen wir unsere Variable jetzt `Anzahl_der_Münzen`:

```
>>> Anzahl_der_Münzen = 200
>>> print(Anzahl_der_Münzen)
200
```

So ist klar, dass wir von 200 Münzen reden.

Die Namen der Variablen können aus Buchstaben, Zahlen und dem Unterstrich (_) bestehen, dürfen aber nicht mit einer Zahl beginnen. Man kann alles – von einzelnen Buchstaben (wie `a`) bis zu langen Sätzen – als Variablennamen verwenden.

Variablennamen dürfen aber keine Leerzeichen enthalten. Benutze daher einen Unterstrich, um Wörter zu trennen.

Manchmal, wenn man etwas Schnelles macht, sind kurze Variablennamen am besten. Der Name, für den Du Dich entscheidest, sollte so aussagekräftig sein, wie er gerade sein muss.

Jetzt, da Du weißt, wie man Variablen erzeugt, schauen wir uns an, wie man sie benutzt.

Jason Briggs, Python kinderleicht!, dpunkt.verlag, ISBN 978-3-86490-344-1

3.3 Variablen benutzen

Erinnerst Du Dich an die Gleichung, mit der wir herausgefunden haben, wie viele Münzen Du nach einem Jahr hast, wenn die komische Erfindung Deines Großvaters im Keller auf wundersame Weise neue Münzen kopiert? Wir hatten diese Rechnungen (nachdem die diebische Elster auftauchte):

```
>>> 20 + 10 * 365
3670
>>> 3 * 52
156
>>> 3670 - 156
3514
```

Wir können daraus eine einzige Programmzeile machen:

```
>>> 20 + 10 * 365 - 3 * 52
3514
```

Was wäre, wenn wir aus den Zahlen Variablen machen würden? Versuche doch einmal, Folgendes einzugeben:

```
>>> gefundene_Münzen = 20
>>> kopierte_Münzen = 10
>>> gestohlene_Münzen = 3
```

Diese Eingaben erzeugen die Variablen `gefundene_Münzen`, `kopierte_Münzen` und `gestohlene_Münzen`.

Jetzt geben wir die Gleichung noch einmal so ein:

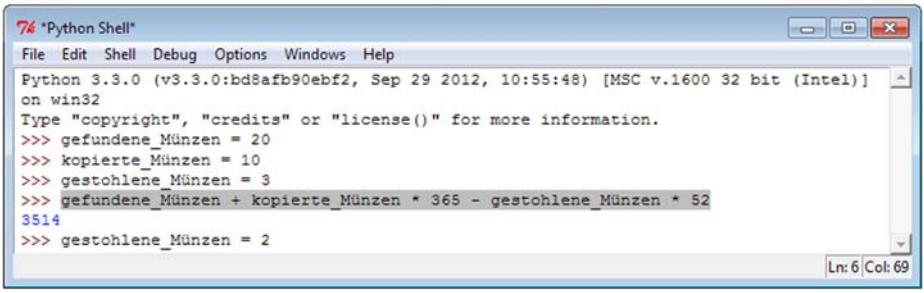
```
>>> gefundene_Münzen + kopierte_Münzen * 365 - gestohlene_Münzen * 52
3514
```

Wie Du siehst, ergibt dies das gleiche Ergebnis. Was soll das Ganze jetzt? Hier kommt die Magie der Variablen ins Spiel. Was wäre, wenn Du eine Vogelscheuche vor Deinem Fenster aufstellst, und die Elster jedes Mal nur noch zwei statt drei Münzen stiehlt? Wenn wir eine Variable einsetzen, können wir die Variable, die für diese Zahl steht, einfach ändern, sodass sie sich überall, wo sie in der Gleichung steht, ändert. Wir können die Variable `gestohlene_Münzen` in 2 ändern, indem wir Folgendes eingeben:

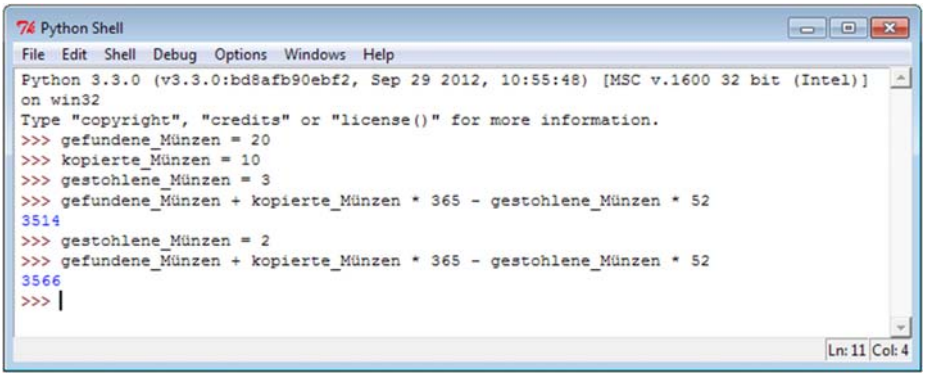
```
>>> gestohlene_Münzen = 2
```

Wir können die Gleichung wie folgt kopieren und einfügen, um das Ergebnis zu berechnen:

1. Wähle den Text, den Du kopieren möchtest, aus, indem Du mit der Maus am Anfang der Zeile klickst und dann (halte die Maustaste weiter gedrückt) bis zum Ende der Zeile ziehst. Danach sieht es aus wie hier:



2. Halte die Ctrl-Taste gedrückt (wenn Du einen Mac benutzt, ist es die ⌘-Taste), und drücke gleichzeitig auf C um den ausgewählten Text zu kopieren. (Ab jetzt sage ich dazu nur noch Ctrl-C.)
3. Klicke auf die letzte Prompt-Zeile (nach `gestohlene_Münzen = 2`).
4. Halte jetzt wieder die Ctrl-Taste gedrückt, und drücke gleichzeitig V, um den ausgewählten und kopierten Text einzufügen. (Ab jetzt sage ich dazu nur noch Ctrl-V.)
5. Drücke die Enter-Taste, um das neue Ergebnis zu sehen:



Ist das nicht viel einfacher, als die ganze Gleichung noch einmal einzugeben? Auf jeden Fall!

Du kannst auch ausprobieren, andere Variablen zu ändern, und dann durch Kopieren (Ctrl-C) und Einfügen (Ctrl-V) der Berechnung schauen, wie sich die Änderungen bemerkbar machen. Es könnte ja sein, dass – wenn man zum richti-

Jason Briggs, Python kinderleicht!, dpunkt.verlag, ISBN 978-3-86490-344-1

gen Zeitpunkt auf die Seitenteile der Erfindung Deines Großvaters haut – jedes Mal 3 zusätzliche Münzen ausgespuckt werden und Du auf diese Weise nach einem Jahr 4661 Münzen hast:

```
>>> kopierte_Münzen = 13
>>> gefundene_Münzen + kopierte_Münzen * 365 - gestohlene_Münzen * 52
4661
```

Es ist natürlich so, dass Variablen bei einer solch einfachen Gleichung immer noch nur *ein klein wenig* nützlich sind. Sie sind noch nicht so *richtig* nützlich geworden. Bis jetzt solltest Du Dir nur einfach merken, dass Variablen eine Möglichkeit sind, Dinge zu bezeichnen, die man später wieder braucht.

3.4 Was Du gelernt hast

In diesem Kapitel hast Du gelernt, wie man einfache Gleichungen mit Python-Operatoren erstellt und wie man mit Klammern die Rangfolge von Operationen bestimmt (die Reihenfolge, nach der Python die Teile der Gleichung berechnet). Anschließend haben wir Variablen erzeugt, um Werte zu bezeichnen, und diese Variablen in unseren Berechnungen eingesetzt.