Inhaltsverzeichnis

Variablen und Datentypen

Wozu dienen Variablen?

Programme bestehen aus einer Abfolge von Anweisungen, die der Computer von oben nach unten ausführt. Jede Anweisung kann bei ihrer Ausführung ein Ergebnis erzeugen, beispielsweise wenn wir eine einfache Berechnung durchführen.

```
width * height
```

Das obige Beispiel zeigt eine einfache Berechnung einer Rechteckfläche anhand von Breite und Höhe. Um das Ergebnis dieser Berechnung in späteren Programmschritten nutzen zu können, müssen wir es speichern. Hierfür verwenden wir in der Programmierung eine Variable.

```
area = width * height
```

Im Beispiel oben weisen wir der Variable `area` das Ergebnis der Flächenberechnung zu. Eine Variable funktioniert dabei wie ein Notizzettel, auf dem wir uns etwas Wichtiges aufschreiben. Über den Variablennamen können wir dann jederzeit innerhalb unseres Programms auf den gespeicherten Wert zugreifen. Zum Beispiel können wir ihn auf der Konsole ausgeben:

```
print(area)
```

Oder wir nutzen die Fläche, um das Volumen einer Kiste zu berechnen, nachdem der Benutzer dem Programm die Tiefe mitgeteilt hat:

```
volume = area * depth
```

Konstanten

Eine Konstante ist ein Sonderfall einer Variable. Der wesentliche Unterschied besteht darin, dass einer Konstante nur einmal ein Wert zugewiesen wird, der sich im weiteren Programmverlauf nicht mehr ändert. Dies ähnelt dem Konzept mathematischer Konstanten wie π, die einen unveränderlichen Wert besitzen.

In Python definieren wir Konstanten, indem wir alle Buchstaben des Namens groß schreiben:

```
PI = 3.14159
INPUT_FILE = "data.csv"
```

In Python gibt es technisch gesehen keine echten Konstanten – sie existieren nur als Konvention. Programmierer nutzen die Großschreibung des Namens, um zu signalisieren, dass der Wert nicht mehr verändert werden soll. Wird der Wert dennoch geändert, führt dies zu keinem technischen Fehler. Dies unterscheidet Python von anderen Programmiersprachen wie Java, wo Konstanten tatsächlich unveränderlich sind.

Benennung von Variablen

Den Namen einer Variable vergeben wir übrigens selbst, wenn wir ein Programm schreiben. Anstatt wie im Beispiel oben die Variable `volume` zu nennen, hätten wie sie ebensogut `volumen` nennen können. Grundsätzlich sind wir frei bei der Benennung von Variablen, wenn wir uns an ein paar kleine Regeln halten, die wir gleich besprechen. Es gibt neben diesen Regeln, die wir einhalten *müssen*, weil es sonst zu Fehlern kommt, auch eine Konvention unter Programmierern, sich an bestimmte Abmachungen zu halten. Dazu zählt zum Beispiel, dass wir Variablennamen in Englischer Sprache halten, damit Programme weltweit geteilt und verstanden werden können.

Regeln bei der Benennung von Variablen

Die zwingenden Regeln sind oft spezifisch für die Programmiersprache, die wir verwenden. Allerdings gibt es zwischen den Sprachen bezüglich dieser Regeln eine große Einigkeit. Für Python gelten folgende Regeln, die bei Missachtung zu einem Fehler führen:

- Ein Variablenname muss mit einem Buchstaben beginnen.
- In einem Variablennamen dürfen nur Buchstaben aus dem römischen Alphabet, Zahlen und Unterstriche (_) verwendet werden. Leerzeichen, Umlaute oder andere Sonderzeichen sind nicht erlaubt.

- Ein Variablenname darf kein reserviertes Schlüsselwort der Programmiersprache sein. Beispielsweise können wir keine Variable `print` oder `if` nennen, da diese Wörter in Python bereits eine besondere Bedeutung haben.

Konventionen bei der Benennung von Variablen

Neben den festen Regeln solltest du folgende Konventionen unbedingt berücksichtigen:

- Ein Variablenname sollte mit einem Kleinbuchstaben beginnen. Also `area` und nicht `Area`.
- Variablennamen sollten in englischer Sprache gehalten werden. Also `area` und nicht `flaeche`.
- Variablennamen sollten aussagekräftig sein und ihre Bedeutung klar vermitteln. Ein Name wie `a` anstelle von `area` wäre ungeeignet, auch wenn er kürzer ist. Dank der Autovervollständigung in unserem Code-Editor sind längere Variablennamen kein Problem – sie sind sogar vorzuziehen, da sie die Lesbarkeit des Programms verbessern.
- Wenn ein Variablenname aus zwei oder mehreren Wörtern zusammengesetzt ist, dann solltest du einen Unterstrich zur Trennung der Wörter verwenden. Also `area_rectangle` statt `arearectangle`. Auch das erhöht die Lesbarkeit des Programms.

Dazu sei noch erwähnt, dass Python Groß- und Kleinschreibung unterscheidet. Das heisst konkret, dass die Variablen `area` und `Area` in einem Programm zwei unterschiedliche Variablen sind.

Datentypen

Zeichenketten (*Strings*)

Variablen können verschiedene Arten von Daten speichern. In den obigen Beispielen haben wir Zahlen gespeichert (Fläche, Volumen). Neben Zahlen können wir auch Zeichenketten in Variablen speichern, zum Beispiel eine E-Mail-Adresse:

```
email = "m.mustermann@mail.de"
```

Eine Zeichenkette erkennt man an den Anführungszeichen, die den Wert umschließen. Die Anführungszeichen selbst gehören nicht zum Wert, sondern dienen nur der Abgrenzung. Dabei können sowohl doppelte als auch einfache Anführungszeichen verwendet werden:

```
email = 'm.mustermann@mail.de'
```

Das Besondere an diesem Datentyp ist, dass innerhalb der Anführungszeichen eine beliebige Abfolge von Zeichen stehen darf. Es können also ganze Texte als String abgebildet werden. Aber auch eher kryptische Werte wie Passwörter:

```
password = "%?Zha§-ÄiK,!/"
```

Eine Besonderheit tritt auf, wenn innerhalb einer Zeichenkette ein Anführungszeichen vorkommen soll. Das das Anführungszeichen schon als Begrenzungszeichen fungiert, müssen wir es mit einem Backslash kennzeichnen. Der Backslash fungiert dann als so genanntes Escape-Zeichen und ist selbst nicht Teil der Zeichenkette:

```
password_with_quotation_mark = "abc\"123"
```

Der Wert der Variable oben wäre somit `abc"123`, der Backslash ist selbst nicht sichtbar. Was aber, wenn wir nun einen Backslash in einer Zeichenkette verwenden möchten? Ganz einfach, dann brauchen wir zwei Backslashes. Der erste „escaped“ den zweiten:

```
string_with_backslash = "abc\\123"
```

Der Wert der Variable wäre somit `abc\123`.

Numerische Werte (*Integer* und *Float*)

Zahlen sind in Algorithmen und Programmen von zentraler Bedeutung, da wir regelmäßig Berechnungen durchführen müssen. Wie im Beispiel zu Beginn dieses Tutorials gezeigt, haben wir die Fläche eines Rechtecks berechnet und das Ergebnis in einer Variable gespeichert. Solch ein Ergebnis kann entweder eine ganze Zahl oder eine Zahl mit Dezimalstellen sein. Python und andere Programmiersprachen unterscheiden zwischen ganzen Zahlen und Dezimalzahlen, da beide unterschiedliche Anforderungen an den Speicher stellen. Ganze Zahlen werden in Python als *Integer* bezeichnet, während Dezimalzahlen (oder Gleitkommazahlen) als *Float* bekannt sind.

Ob es sich um den einen oder den anderen numerischen Datentyp handelt, wird anhand des Ausdrucke entschieden, der einer Variable zugewiesen wird. Betrachten wir zunächst den einfachsten Fall, die direkte Zuweisung eines Wertes:

```
width = 5
height = 2.5
```

Im obigen Beispiel wird der Variable `width` eine ganze Zahl zugewiesen, weshalb Python den Datentyp *Integer* wählt. Die Variable `height` erhält eine Dezimalzahl, wodurch Python automatisch den Datentyp *Float* verwendet.

```
area = width * height
```

Welchen Datentyp hat nun die Variable `area`? Da das Produkt einer ganzen Zahl und einer Dezimalzahl eine Dezimalzahl ergeben kann, weist Python automatisch den Datentyp *Float* zu – selbst wenn das Ergebnis theoretisch eine ganze Zahl sein könnte. Python geht auf Nummer sicher. Wäre `height` allerdings ebenfalls eine ganze Zahl, dann wäre auch das Ergebnis `area` vom Typ *Integer*.

Wahrheitswerte (*Boolean*)

Für manche Sachverhalte wollen wir lediglich wissen, ob eine Aussage wahr oder falsch ist. Ist der Kunde bereits 18 Jahre alt? Besitzt der Fahrer die Erlaubnis, einen 3,5-Tonner zu fahren? Wurde die Prüfung bestanden? In solchen Fällen gibt es nur zwei mögliche Antworten: „Ja" oder „Nein" – oder in der Sprache der Logik: „Wahr" oder „Falsch". Für diesen Zweck existiert der Boolesche Datentyp, der nur zwei mögliche Werte annehmen kann: `True` und `False`.

```
is_underage = False
exam_passed = points_reached >= 50
```

Im Codebeispiel in Zeile 1 weisen wir der Variable `is_underage` den Wert `False` zu. Beachtet, dass dieser Datentyp, anders als eine Zeichenkette, keine Anführunsgzeichen benötigt. Die Begriffe `True` und `False` sind so genannte Schlüsselwörter in Python und werden automatisch als Boolsche Werte erkannt.

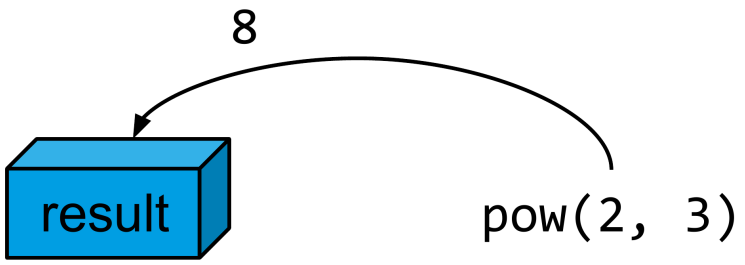
Die zweite Zeile weist keinen direkten Wahrheitswert zu, sondern das Ergebnis eines logischen Ausdrucks. Dieser Ausdruck wird entweder zu wahr oder falsch ausgewertet, abhängig vom aktuellen Wert der Variable `points_reached`. Diese Punktzahl wurde vielleicht in vorherigen Programmzeilen berechnet.

Komplexe Datentypen

Variablen können neben den primitiven Datentypen wie Zeichenketten, Zahlen oder Boolesche Werte auch komplexere Dinge speichern. Im Prinzip lässt sich alles in einer Variable speichern, was in Python existiert. Dazu gehören beispielsweise Funktionen, die wir in einem späteren Tutorial kennenlernen. Auch sogenannte Collections (zu Deutsch: Sammlungen) zählen dazu. Collections sind Strukturen, die mehrere primitive Datentypen enthalten können. Ein Beispiel dafür ist eine Liste von Zahlen. Collections werden wir in einem späteren Tutorial ausführlich behandeln.

Variablen im Computer

Eine Variable lässt sich bildlich als Behälter mit einem Namensetikett vorstellen. In diesen Behälter können wir beliebige Werte ablegen. Mithilfe des Namens finden wir den Behälter jederzeit wieder, können seinen Inhalt einsehen und für weitere Zwecke verwenden. Wir können auch einen neuen Wert in den Behälter legen – dabei wird der bisherige Inhalt entweder gelöscht oder muss zuvor in einen anderen Behälter umgelagert werden.



In Wirklichkeit sind Variablen bestimmte Bereiche im Arbeitsspeicher (RAM) des Computers. Jeder dieser Speicherorte besitzt eine eindeutige Adresse. Bei der Zuweisung eines Wertes zu einer Variable speichert der Computer diesen Wert an einer bestimmten Adresse im RAM. Der Variablenname wird dann mit dieser Adresse verknüpft, damit wir den Wert später über den Namen abrufen können. Eine Variable ist also im Grunde ein Zeiger auf eine bestimmte Stelle im Computerspeicher.