# Variablen und Operatoren

Inhalt



Unser erstes Programm hat eine kurze Begrüßung in Textform auf dem Bildschirm ausgegeben. Diese Aufgabe hättest du natürlich auch deutlich einfacher mit einem Textverarbeitungsprogramm wie Word erledigen können. Wie du sicher schon vermutet hast, sind Computerprogramme nicht auf die Ausgabe von einfachen Texten begrenzt. Sie können viele weitere, weitaus komplexere Aufgaben übernehmen. Dazu nehmen sie verschiedene Werte auf und bearbeiten diese. Hierbei kommen Variablen und Operatoren zum Einsatz.

## Variablen

Variablen sind Platzhalter im Programmcode, die verschiedene Werte annehmen können.

Um die Verwendung von Variablen zu veranschaulichen, werfen wir nochmals einen Blick auf das Programm zur Begrüßung zum Python-Kurs. Dieses gibt bisher lediglich eine kurze Begrüßungs-Floskel aus. Im nächsten Schritt möchten wir diese personalisieren, indem wir den Namen des Anwenders hinzufügen. Außerdem soll ein weiterer Text hinzukommen, der den Anwender dazu auffordert, mit dem Kurs zu beginnen. Diese soll ebenfalls den Namen enthalten. Mit den bisherigen Kenntnissen ist es ganz einfach möglich, dieses Programm zu schreiben:

```
print("Michael, willkommen zum Python-Kurs!")
print("Vielen Dank Michael, dass du an diesem Kurs teilnimmst.")
print("Beginne damit, deine eigenen Programme zu schreiben.")
```

Dieses Programm funktioniert zwar einwandfrei, doch ist davon auszugehen, dass wir nicht nur einen einzelnen Anwender zum Kurs begrüßen möchten, sondern viele verschiedene. Nun wäre es selbstverständlich möglich, den Namen für jeden Anwender von Hand zu ändern, doch wäre das sehr aufwendig – insbesondere wenn man davon ausgeht, dass der Text anschließend noch weitergeht und viele weitere persönliche Anreden enthält.

An dieser Stelle kommen Variablen ins Spiel. Diese stellen einen Platzhalter dar und können einen beliebigen Wert annehmen.

## Namen geben

Um eine Variable zu verwenden, musst du dir zunächst einen Namen für sie überlegen. Dafür kannst du verschiedene Buchstaben, Zahlen sowie den Unterstrich (_) verwenden. Du musst lediglich darauf achten, dass der Name nicht mit einer Zahl beginnen darf und außerdem darfst du den gleichen Namen nicht für unterschiedliche Variablen verwenden.

Besonders praktisch ist es, sprechende Namen zu wählen, damit du den Überblick über den Inhalt behältst. Für den Namen des Anwenders wäre beispielsweise folgender Variablenname sinnvoll: `nameAnwender`

Python unterscheidet zwischen Groß- und Kleinschreibung. Die Variable `nameanwender` ist daher nicht die gleiche wie `nameAnwender`

## Wert zuweisen

Nachdem du dir einen Namen für die Variable überlegt hast, musst du ihr einen Wert zuweisen. Wenn du einer Variablen das erste Mal einen Wert gibst, wird das in der Informatik als Initialisierung bezeichnet. Bevor du in Python auf eine Variable zugreifen kannst, musst du sie immer initialisieren. Das geschieht durch das Gleichheitszeichen. Wenn es sich beim Inhalt um Text handeln soll, musst du diesen in Anführungszeichen setzen. Bei Zahlen ist das nicht notwendig. Der Befehl sieht daher wie folgt aus:

```
nameAnwender = "Michael"
```

Wenn du den Inhalt jetzt in den `print`-Befehl einfügen willst, musst du hier einfach den Variablennamen nennen – ohne Anführungszeichen:

```
print(nameAnwender)
```

Auf diese Weise lässt sich der Wert der Variable (hier: Name des Anwenders, also `Michael`) bereits ausgeben. Das Ziel besteht jedoch darin, diesen Namen in den Text einzufügen. Dazu musst du den Namen im Text durch den Variablennamen ersetzen. `Michael` wird also durch `nameAnwender` ersetzt. Danach musst du ein Pluszeichen (+) und anschließend den übrigen Text in Anführungszeichen (`" "`) hinzufügen:

```
print(nameAnwender + ", willkommen zum Python-Kurs!")
```

Anstelle des Pluszeichens wäre es auch möglich, ein Komma zu verwenden. Dieses führt jedoch dazu, dass automatisch ein Leerzeichen nach der Variablen eingefügt wird. Das ist in diesem Fall jedoch nicht erwünscht, da nach der Variablen direkt ein Komma und kein Leerzeichen stehen soll. Wenn du jedoch mehrere Variablen hintereinander ausgeben willst, ist das Komma sehr praktisch, da du auf diese Weise das Leerzeichen nicht manuell einfügen musst.

Nach dem gleichen Muster ist es nun möglich, die zweite Zeile, die ebenfalls den Namen verwendet, mit einer Variablen auszustatten. Das komplette Programm sieht dann wie folgt aus:

```
nameAnwender = "Michael"
print(nameAnwender + ", willkommen zum Python-Kurs!")
print("Vielen Dank " + nameAnwender + ", dass du an diesem Kurs
teilnimmst.")
print("Beginne damit, deine eigenen Programme zu schreiben.")
```

Wenn du jetzt einen weiteren Anwender begrüßen willst, musst du lediglich einmal den Wert der Variablen ändern (bspw. von `Michael` auf `Lena`). Der Text passt dann alle Stellen mit einer persönlichen Anrede automatisch an.

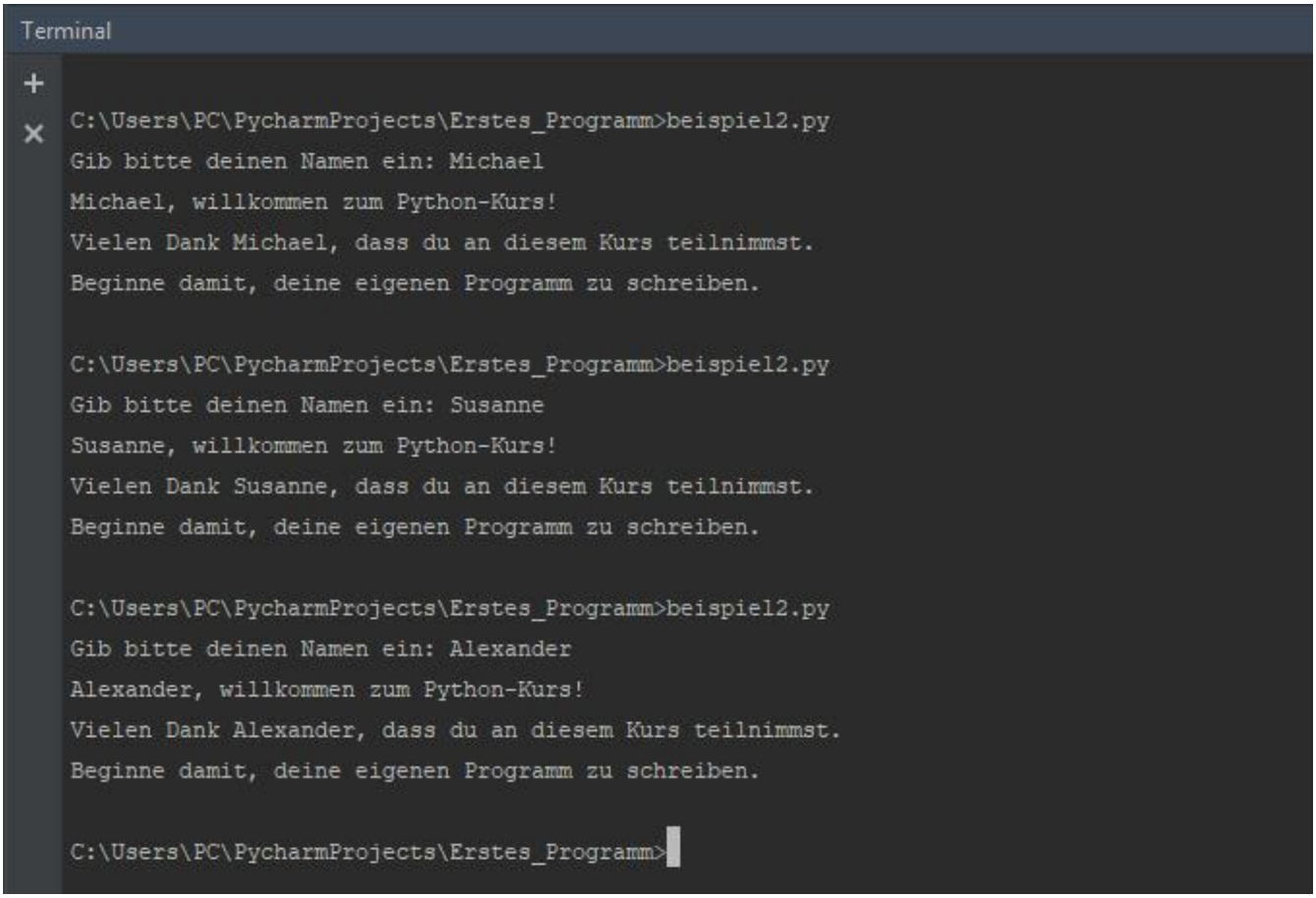
## Nutzer-Interaktion

Variablen ermöglichen es auch, mit dem Anwender zu kommunizieren. Du kannst zum Beispiel zu Beginn des Programms den Nutzer nach seinem Namen fragen und den Wert in einer Variablen speichern. Danach gibt das Programm den Text mit dem Namen aus, den der Anwender selbst eingegeben hat.

Für diese Aufgabe benötigst du den input-Befehl. Dafür musst du einfach den Schlüsselbegriff `input` und danach eine runde Klammer (`( )`) einfügen. Darin kannst du einen Text erstellen, der dem Anwender erklärt, welchen Wert er eingeben soll. Um den Inhalt einer Variablen zuzuweisen, musst du den `input`-Befehl einfach nach deren Namen und einem Gleichheitszeichen einfügen:

```
nameAnwender = input("Gib bitte deinen Namen ein: ")
```

Auf diese Weise speicherst du die Eingabe des Anwenders in der Variablen. Der Rest des Programms bleibt unverändert. Probiere nun das Programm mit verschiedenen Namen aus und überprüfe, wie es darauf reagiert.




Die Ausgabe des Programms mit benutzerdefinierten Namen

# Datentypen

Variablen können viele verschiedene Informationen aufnehmen.

## String

In den bisherigen Beispielen haben wir stets Text in den Variablen gespeichert. Dieser Variablentyp wird in Python als String bezeichnet – abgekürzt als `str`.

## Integer und float

Es gibt jedoch noch einige weitere Möglichkeiten. Du kannst in einer Variablen auch eine Zahl abspeichern – entweder eine ganze Zahl (`int` für integer) oder eine Fließkommazahl (`float` für floating point number). Dazu musst du einfach die Zahl nach dem Gleichheitszeichen einfügen – Zahlen allerdings ohne Anführungszeichen:

```
alter = 25
preis = 2.49
```

Die Variable `alter` ist vom Typ Integer (kurz: `int`). Die Variable `preis` ist vom Typ

Fließkommazahl (kurz: `float`).

Python erkennt daran automatisch, dass es sich um einen anderen Datentyp handelt. Wenn Python den Inhalt einer Variablen als Zahl abspeichert, kannst du damit anschließend mathematische Operationen durchführen.

Du kannst Zahlen auch in Anführungszeichen schreiben. Das führt jedoch dazu, dass Python sie als Text und damit als String behandelt. Das bedeutet, dass du damit keine mathematischen Operationen durchführen kannst.

Das Komma (`,`) wird in Python, wie im Englischen, als Punkt (`.`) geschrieben.

Einer der großen Vorteile von Python besteht darin, dass du hierbei den Datentyp nicht angeben musst. Das übernimmt Python (genau genommen der Python Interpreter) bei der Ausführung automatisch. Die meisten anderen Programmiersprachen sind in dieser Hinsicht deutlich strikter. In Python musst du dich beispielsweise nicht entscheiden, ob eine Zahl eine ganze Zahl sein soll oder ob sie auch Nachkommastellen enthalten darf. Dies macht das Programmieren deutlich einfacher und effizienter. Außerdem kannst du im Verlauf eines Programms einer Variablen verschiedene Daten zuweisen. Darüber hinaus gibt es noch einen weiteren Typ:

### Boolesche Variablen

Boolesche Variablen, auch Booleans genannt, dienen dazu, Wahrheitswerte aufzunehmen und kommen beim Programmieren sehr häufig zum Einsatz. Diese Variablen können nur zwei verschiedene Werte annehmen: `True` und `False` (Großschreibung beachten!). Du kannst diese Variablen verwenden, um einen bestimmten Zustand zu registrieren – beispielsweise ob sich der Nutzer zum Kurs angemeldet hat. Dafür kannst du die Variable angemeldet einführen und ihr den Wert `True` (angemeldet also True) zuweisen, wenn der Nutzer sich angemeldet hat. Ist dies nicht der Fall, kannst du ihr den Wert `False` geben. So ist es später möglich, den Ablauf des Programms an den Wert der Variable anzupassen – beispielsweise indem du bestimmte Teile nur für angemeldete Nutzer zugänglich machst.

## Die Bedeutung der Datentypen

Datentypen haben in Python eine deutlich geringere Bedeutung als in anderen Programmiersprachen, da der Interpreter die Typisierung automatisch durchführt, den Datentyp also selbstständig erkennt. Dennoch ist es in einigen Fällen wichtig, auf den Typ zu achten. Als Beispiel hierfür sollst du ein Programm erstellen, das den Nutzer nach einer Zahl fragt und danach den doppelten Wert ausgibt. Nach den bisherigen Kenntnissen könnte das Programm wie folgt aussehen:

```
zahl = input("Gib bitte eine Zahl ein: ")
print("Doppelter Wert: " + zahl * 2)
```

Für die Multiplikation kommt in Python das Stern-Symbol (*) zum Einsatz.

Wenn du jetzt das Programm ausführst, dann berechnet es jedoch nicht den doppelten Wert, sondern gibt einfach die Ziffern der Zahl zweimal hintereinander aus. Das liegt daran, dass der `input`-Befehl die Werte stets als Zeichenkette speichert. Bei der Multiplikation einer Zeichenkette fügt das Programm die Zeichen in der entsprechenden Anzahl aneinander. Das kannst du ausprobieren, indem du einmal anstatt einer Zahl ein Wort eingibst. Daran siehst du, dass dieses Programm den Wert nicht mit dem gewünschten Datentyp abspeichert.

Um das zu ändern, musst du den Datentyp umformen. Dafür gibt es mehrere Möglichkeiten.

Es ist möglich, diese Umformung direkt vorzunehmen. Das ist im Prinzip nicht schwierig – so lange du genau weißt, welchen Datentyp die Eingabe haben wird.

Wenn du dir sicher bist, dass der Anwender einen Integer-Wert eingibt, musst du dafür die Bezeichnung `int` vor den `input`-Befehl setzen:

```
zahl = int(input("Gib bitte eine Zahl ein: "))
print("Doppelter Wert:", zahl * 2)
```

Um den Ausgabetext und die Variable zu verbinden, muss nun ein Komma stehen. Das Pluszeichen ist nur zur Verbindung von zwei Zeichenketten (eine Zeichenkette ist eine Abfolge von Buchstaben, Ziffern und anderen Zeichen – daran ersichtlich, dass sie in Anführungszeichen steht) zulässig. Das ist ein weiteres Beispiel dafür, dass auch in Python der Datentyp manchmal eine wichtige Rolle spielt.

Das Problem bei dieser Art der Umformung besteht darin, dass du genau wissen musst, welchen Typ die Eingabe haben wird. Gibt der Anwender einen anderen Wert ein, führt das zu einem Abbruch des Programms. Weitere Details zu diesem Umformungsverfahren erfährst du in der Einheit Konvertierungen zwischen Datentypen.

Universeller lässt sich der eval-Befehl einsetzen. Dieser kann unterschiedliche Datentypen aufnehmen und umformen.

Dabei gilt es allerdings zu berücksichtigen, dass dieser hinsichtlich der Sicherheit Risiken mit sich bringen kann. Wenn der Anwender bei der Eingabe Python-Code eingibt, führt das Programm diesen ebenfalls aus. Diese Eigenschaften können sich Hacker zunutze machen, um dessen Funktionsweise zu manipulieren oder andere Schäden anzurichten. Da unsere Programme nur für den Eigengebrauch bestimmt sind und dieser Aspekt daher zu vernachlässigen ist, soll die Verwendung dennoch vorgestellt werden.

Der `eval`-Befehl wandelt die Zeichenkette in eine Zahl um – unabhängig davon, ob es sich um eine ganze Zahl oder um eine Fließkommazahl handelt. Folgendes Programm gibt nun den doppelten Wert aus:

```
zahl = eval(input("Gib bitte eine Zahl ein: "))
print("Doppelter Wert:", zahl * 2)
```

# Operatoren

Eine Variable bleibt im Verlauf eines Programms nicht immer gleich. Stattdessen ist es häufig notwendig, ihren Wert zu verändern. Hierzu kommen sogenannte Operatoren zum Einsatz. Einige Operatoren haben wir in den bisherigen Beispielen bereits verwendet.

## Zuweisungsoperator

Der wichtigste von ihnen ist der Zuweisungsoperator – der aus dem Gleichheitszeichen (=) besteht. Dieser ermöglicht es, einer Variablen einen Wert zuzuweisen. Das ist nicht nur zu Beginn des Programms möglich. Du kannst einer Variablen auch im weiteren Verlauf immer wieder einen neuen Wert zuweisen:

```
name = "Michael"
print("Hallo, " + name)
name = "Susanne"
print("Hallo, " + name)
```

Dabei wird der alte Wert überschrieben. Bei der zweiten Ausgabe gibt der `print`-Befehl daher die Begrüßung mit den neuen Namen (hier: `Susanne`) aus.

## Mathematische Operatoren

Hinzu kommen mathematische Operatoren. Dabei handelt es sich um die vier Grundrechenarten. Die Verwendung von Addition mit Plus (+) und Subtraktion mit Minus (-) funktioniert wie gewohnt. Für Multiplikationen kommt das Sternsymbol (*) und für Divisionen der Schrägstrich (/) zum Einsatz. Außerdem gibt es den Modulo-Operator. Dieser besteht aus dem Prozentzeichen (%) und gibt den Rest der ganzzahligen Division an. Beispiele:

```
x = 4
y = x + 2
x = y / 3
x = 5 % 2
x = x * 5
```

Auf diese Weise sind viele verschiedene Rechenoperationen zwischen Zahlen und Variablen möglich. Dabei kann eine Variable auch Bezug auf sich selbst nehmen. Das zeigt das letzte Beispiel. Dieser Befehl weist der Variablen x das Fünffache ihres bisherigen Werts zu. Da solche Ausdrücke in Python recht häufig vorkommen, gibt es dafür auch eine Kurzform: `x *= 5` Analog dazu bestehen auch für die anderen Grundrechenarten entsprechende Ausdrücke:

```
x += y entspricht x = x + y
x -= y entspricht x = x – y
x *= y entspricht x = x * y
x /= y entspricht x = x / y
```

Wichtig: Mathematische Operatoren sind teilweise auch auf Zeichenketten anwendbar. Allerdings haben sie dabei einen anderen Effekt:

- Das Pluszeichen (+) setzt zwei Zeichenketten zusammen.
- Das Sternsymbol (*) wiederholt die Zeichenkette mit der entsprechenden Anzahl.

Es ist wichtig, dieses Prinzip verstanden zu haben!

Wenn du bis hier alles verstanden und die Einheit abgeschlossen hast, kannst du jetzt dein nächstes Projekt programmieren: Den Taschenrechner Teil 1.

Danach kannst du einfach an diese Stelle zurückkehren und mit der normalen Struktur des Kurses weitermachen.

---

Prüfe deine Skills: □□□ Aufgabe 2