Variablen in Python (Musterlösung)

Jacques Mock Schindler

14.09.2025

Vorbemerkungen: Python als Rechner

Python verfügt über eingebaute mathematische Fähigkeiten. Es kann die Grundrechenarten und kennt die Hierarchie der Operationen. Sie können das überprüfen, in dem Sie in der folgenden Zelle die Rechnung

$$2 + 3 \cdot 4$$

ausführen.

```
2 + 3 * 4
```

```
14
```

Die folgende Tabelle gibt einen Überblick über die direkt in Python verfügbaren mathematischen Funktionen:

Beschreibung	Befehl	Beispiel	Resultat
Addition	`+`	2 + 3	5
Subtraktion	`-`	3 - 2	1
Multiplikation	`*`	3 * 2	6
Division	`/`	3 / 2	1.5
Potenzen	`**`	3 ** 2	9
Wurzeln	`**(1/n)`	16 ** (1/2)	4.0
Ganzzahlige Division	`//`	7 // 2	3
Modulo	`%`	7 % 2	1

Variablen

In Python sind Variablen symbolische Namen für gespeicherte Daten. Variablen verweisen dabei auf den Speicherbereich im Computer, in welchem die entsprechenden Daten physikalisch abgelegt sind. Aus diesem Grund werden Variablen gelegentlich auch als Zeiger

bezeichnet. Was genau für Daten in diesem Speicherbereich abgelegt werden, spielt keine Rolle und kann während der Ausführung eines Programmes auch ändern.

In Python werden Variablen Werte mit dem Gleichheitszeichen zugewiesen. Um der Variable x den Wert 2 zuzuweisen, ist die Eingabe `x = 2` erfoderlich. Die Variable muss links vom Gleichheitszeichen, der zuzuweisende Wert rechts davon stehen.

Überprüfen Sie dies, indem Sie in der folgenden Zelle der Variabel y den Wert 3 und der Variabel z den Wert 4 zuweisen. Anschliessend multiplizieren Sie die beiden Variablen miteinander.

```
y = 3
z = 4
y * z
```

```
12
```

Wenn Variablen neue Werte zugewiesen werden, wird die Referenz auf den Speicherbereich mit dem alten Wert gelöscht. Die Daten, welche ohne Verweis durch eine Variable im Speicher liegen, werden vom in Python eingebauten *Garbage Collector* im Hintergrund gelöscht und der so freigewordene Speicherplatz kann wieder verwendet werden.

Sie können überprüfen, dass Variablen neue Werte zugewiesen werden können, indem Sie in der untenstehenden Zelle die Variabeln y und z addieren. Sie erhalten dann das Resultat 7. Das heisst, den Variablen y und z sind immer noch die Werte 3 und 4 zugewiesen.

```
y + z
```

```
7
```

Wenn Sie in der folgenden Zelle der Variabel y den Wert 5 zuweisen und anschliessend y und z addieren erhalten Sie als neues Resultat 9.

```
y = 5
y + z
```

```
9
```

Variablen können auch Resultate von Berechnungen zugewiesen werden. Ausserdem können Variablen ganze Wörter als Namen haben. Dies ist gegenüber einzelnen Buchstaben vorzuziehen, weil dann aussagekräftige Namen gewählt werden können. Grundsätzich sind die Namen von Variabeln frei wählbar. Es gibt allerdings eine Reihe von reservierten Begriffen, welche in der Programmiersprache Python eine eigene Bedeutung haben. Unzulässig sind ausserdem Namen, die mit Ziffern beginnen.

Für die Darstellung von Namen für Variablen hat sich in Python eingebürgert, Variablen klein zu schreiben und Wörter durch Underlines zu trennen ($this\_is\_a\_valid\_variable$). Diese Darstellung nennt sich *Snake Case*. Zudem werden Variablen meist mit englischen Begriffen bezeichnet.

Weisen Sie in der nächsten Zelle der Variable `result` das Resultat der Rechnung $y + z$ zu und geben Sie das Resultat mit `print(result)` aus. `print()` ist eine Funktion, die Python zur Verfügung stellt. Was Funktionen sind, wird im nächsten Abschnitt erklärt.

```
result = y + z
print(result)
```

```
9
```

Funktionen in Python

Python verfügt über viele bereits vordefinierte Funktionen. Die oben verwendete Funktion `print()` ist ein Beispiel dafür. Um zu demonstrieren, wie Funktionen in Python definiert werden, zeige ich Ihnen als Beispiel eine Funktion, mit der zwei Zahlen addiert werden.

```
# Definition der Funktion
def get_sum(x, y):
    return x + y

# Aufruf der Funktion
result = get_sum(3,4)
# Ausgabe des Resultats des Funktionsaufrufs
print(result)
```

```
7
```

`def` ist das Schlüsselwort für die Definition einer Funktion. `get_sum` ist der von mir gewählte Name dieser Funktion. Für die Wahl des Namens einer Funktion gelten die gleichen Regeln, wie für Variablen. In den Klammern stehen die sogenannten Parameter, welche der Funktion übergeben werden, damit sie etwas damit macht. Mit dem Doppelpunkt wird die *Signatur* der Funktion abgeschlossen. Die *Signatur* zeigt idealerweise, *was* eine Funktion *womit* macht. Sie gibt aber keine Auskunft darüber, wie sie das macht.

Python gruppiert Befehle, die zusammengehören, durch die gleiche Tiefe der Einrückung. Eine Einrückung hat üblicherweise die Tiefe von vier Leerzeichen. Im Beispiel oben gibt es nur eine eingerückte Zeile, weil die Funktion nur aus einem Befehl besteht. Mit `return` gibt die Funktion das Resultat zurück.

Im Beispiel wird das Resultat der Berechnung, welche die Funktion ausführt der Variable `result` zugewiesen. Der Wert der Variable `result` wird mit `print(result)` ausgegeben.

Definieren Sie in der folgenden Zelle eine Funktion, mit der zwei Zahlen multipliziert werden.

```
def get_product(a, b):
    return a * b

product = get_product(3, 4)
print(product)
```

```
12
```

Datentypen

Als nächstes geht es um die Frage, auf welche Inhalte eine Variable zeigen kann.

Im Grundsatz kann eine Variable auf beliebige Inhalte verweisen.

Am einfachsten ist die Verwendung der grundlegenden Datenypen (basic data types), welche Python zur Verfügung stellt. Dies sind (mit ihren englischen Bezeichnungen):

- Integer (Ganzzahl)
- Floating-Point Number (Gleitkommazahl)
- Complex Number (komplexe Zahl)
- String (Zeichenkette)
- Boolean Type (Wahrheitswert)

Darüber hinaus ist es möglich, eigene Datentypen zu programmieren. Hier aber zuerst eine Beschreibung der grundlegenden Datentypen von Python.

Integer

Die Bezeichnung für Integer in Python ist ein kurzes `int`.

Anders als in anderen Progammiersprachen gibt es in Python theoretisch keine Beschränkung, wie gross ein Integer sein kann. Die einzige Grenze ist der Speicherplatz des auf dem der Integer gespeichert werden soll.

Wenn einer Variable ein grosser Integer zugewiesen wird, kann dieser zur besseren Lesbarkeit auch mit einem Underline als Tausendertrennzeichen geschrieben werden (`100_000`).

Um das Auszuprobieren, weisen Sie in der folgenden Zelle der Variable a den Wert von einer Million und der Variable b den Wert von einer Milliarde zu. Anschliessend addieren Sie a und b und weisen das Resultat der Variable big_sum zu. Zum Schluss geben Sie den Wert von big_sum mit der Funktion `print()` aus.

```
a = 1_000_000
b = 1_000_000_000
big_sum = a + b
print(big_sum)
```

```
1001000000
```

Eingegebene Zahlen werden automatisch als Dezimalzahlen interpretiert.

Integers können jedoch auch als Binär-, Oktal- oder Hexadezimalzahlen eingegeben werden. Die Eingabe erfordert dann allerdings ein Präfix, welches das Zahlensysten identifiziert. Die folgende Tabelle stellt die möglichen Präfixe zusammen.

Präfix	Bedeutung	Basis
`0b` (Null + Kleinbuchstabe `b`)	Binärzahl	2
`0B` (Null + Grossbuchstabe `B`)		2
`0o` (Null + Kleinbuchstabe `o`)	Oktalzahl	8
`0O` (Null + Grossbuchstabe `O`)		8
`0x` (Null + Kleinbuchstabe `x`)	Hexadezimalzahl	16
`0X` (Null + Grossbuchstabe `X`)		16

In der folgende Zelle finden Sie ein entsprechendes Beispiel.

Python kann Integer in verschiedenen Zahlensystemen darstellen. Um das Zahlensystem bei der Zuweisung zu spezifizieren, verwenden Sie die entsprechenden Präfixe. Die folgenden Beispiele zeigen die Zuweisung der Zahlen 10, 8, 255 und 16 in den Zahlensystemen Dezimal, Oktal, Hexadezimal und Binär:

```
dezimal      = 10
oktal        = 0o10
hexadezimal  = 0x10
binaer       = 0b10
```

Die Anzeige der Werte der entsprechenden Variablen erfolgt grundsätzlich im Dezimalsystem.

```
b = 0b101010
o = 0o52
x = 0x2a

print(b, o, x)
```

```
42 42 42
```

Gleitkommazahl

Die Bezeichnung für Gleitkommazahlen in Python ist `float`. Python interpretiert Zahlen mit einem Dezimalpunkt als Gleitkommazahlen. Optional können Zahlen mit `e` oder `E` in "wissenschaftlicher" Schreibweise eigegeben werden (1000 = 1e3 bzw. 1e-3 = 0.001).

Weisen Sie in der folgenden Zelle den Variablen `million` und `billionth` die passenden Werte in wissenschaftlicher Schreibweise zu.

```
million = 1e6
billonth = 1e-9
print(million, billonth)
```

```
1000000.0 1e-09
```

Komplexe Zahlen

Python kann auch mit komplexen Zahlen umgehen. Der Abschnitt zu diesem Thema kann wieder aufgegriffen werden, wenn Sie in Mathe die komplexen Zahlen besprochen haben.

String

Zeichenketten (String) werden von Python als `str` bezeichnet.

Zeichenketten sind beliebige Zeichenfolgen. Damit Python Zeichenketten als solche erkennt, müssen sie durch die Verwendung von einfachen oder doppelten Anführungs- und Schlusszeichen als solche gekennzeichnet werden.

`"Ich bin eine Zeichenkette."` oder `'Ich bin auch eine Zeichenkette.'`

Wenn man innerhalb einer Zeichenkette Anführungszeichen braucht, müssen die eingrenzenden Anführungszeichen von der "anderen Sorte" sein (`"It's cool learning Python!"` oder `'Der Lehrer sagt: "Es ist cool Python zu lernen."'`). Eine andere Möglichkeit reservierte Zeichen zu verwenden ist der Gebrauch eines "escape"-Zeichens. In Python ist das der "backslash" (\). Die beiden Beispielsätze von vorher hätten entsprechend auch folgendermassen geschrieben werden können:

`'It\'s cool learning Python!'` bzw. `"Der Lehrer sagt: \"Es ist cool Python zu lernen.\""`

Die Länge von Zeichenketten wird lediglich durch die Speicherkapazität des verwendeten Systems begrenzt. Zeichenketten können nicht nur sehr lang, sondern auch leer sein (`''`).

Zeichenketten können, wie alle Datentypen, Variablen zugewiesen werden. Dies zeigt das folgende Beispiel.

```
# Zuweisung eines Strings zu einer Variabel
standard_greeting = "Hello World"

# Ausgabe der Variabel
print(standard_greeting)
```

```
Hello World
```

Boolean Type

Wahrheitswerte werden in Python als `bool` bezeichnet. Wahrheitswerte können entweder “wahr” oder “falsch” sein.

```
wahr      = True
falsch    = False
```

Die Ausgabe findet sich in der folgenden Zelle.

```
wahr      = True
falsch    = False
print(wahr, falsch)
```

```
True False
```

Welche Warheitswerte sich bei Vergleichen ergeben, kann mit den folgenden Operatoren überprüft werden:

```
==   # Gleichheit
!=   # Ungleichheit
>    # Größer als
<    # Kleiner als
>=   # Größer oder gleich
<=   # Kleiner oder gleich
```

```
2 < 1
```

```
False
```

Wahrheitswerte werden zur Steuerung von Programmflüssen verwendet. Mit einem Wahrheitswert kann zum Beispiel gesteuert werden, wie oft ein Programmteil wiederholt werden soll.

Funktionen mit Type-Hints

Zum Abschluss komme ich noch einmal auf die Definition von Funktionen zurück. In Python können Variablen - anders als zum Beispiel in Java - beliebige Datentypen zugewiesen werden. Wenn Variablen im Verlauf eines Programms mehrfach verwendet werden, können ihnen auch unterschiedliche Datentypen zugewiesen werden. Dies ist allerdings schlechter Programmierstil.

Aus diesem Grund ist es sinnvoll, bei der Definition einer Funktion zu deklarieren, welche Datentypen die Parameter haben und welcher Datentyp der Rückgabewert hat. Dies soll mit dem Beispiel der Funktion `get_quotient` verdeutlicht werden.

```
1 def get_quotient(x : int, y : int) -> float:
2     return x / y
```

Hier wird angegeben, dass die Parameter x und y vom Datentyp `int` sein sollen. Der Datentyp des Rückgabewertes wird hinter `->` geschrieben. Im Beispiel ist der Rückgabewert vom Typ `float`. Das ist so, weil die Funktion zum Beispiel $3/4 = 3.5$ rechnet.

Aber Achtung: die Funktion arbeitet auch dann korrekt, wenn ein anderer als der deklarierte Datentyp übergeben wird. Voraussetzung ist lediglich, dass der Datentyp mit den verwendeten Operationen kompatibel ist. Die “Type-Hints” dienen lediglich der besseren Nachvollziehbarkeit, was die Funktion macht.