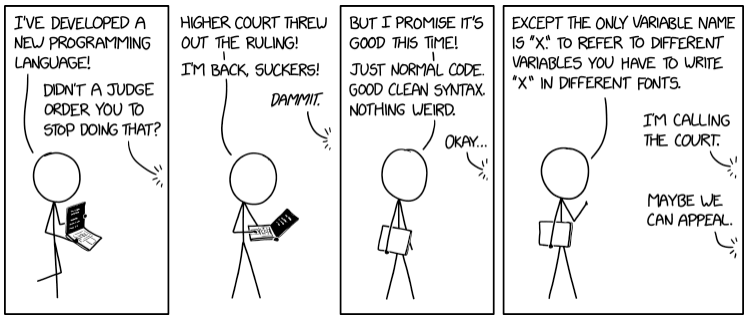
X https://xkcd.com/2309/



The worst is when you run out of monospaced fonts and have to use variable-width variables.

7. Variablen

Für einfachere und vielseitigere Programme müssen wir in der Lage sein, eine Zahl oder einen Text zu speichern. In den Programmiersprachen stehen dafür **benannte Speicherplätze** zur Verfügung. Das Speichern von z.B. Zahlen oder Text dürfen wir nicht mit dem Speichern z.B. eines Textdokuments verwechseln. Ein Speicherplatz ist nur während das Programm läuft verfügbar. Danach sind alle Speicherplätze wieder gelöscht. Die Lernziele für dieses Kapitel sind:

- ☐ Sie definieren die Begriffe Variable, Wert, Literal und Zuweisung.
- ☐ Sie verwenden Variablen, um Python-Programme einfacher und vielseitiger zu gestalten.
- ☐ Sie wissen, welche Variablennamen in Python erlaubt sind.
- ☐ Sie berücksichtigen die Clean-Code-Regeln im Umgang mit Variablen.

7.1 Quadrate mit unterschiedlicher Länge

Schauen wir uns das Programm aus Listing 7.1 an, welches ein Quadrat mit der Seitenlänge 100 zeichnet. Wenn wir nun ein Quadrat mit der Seitenlänge 275 zeichnen möchten, dann müssen wir das Programm an **vier** Stellen anpassen. Listing 7.2 zeigt das angepasste Programm. Optimal wäre das Programm, wenn wir es nur an **einer** Stelle anpassen müssten. Dies können wir erreichen, in dem wir eine **Variable** verwenden.

```
1  import turtle
2
3  turtle.fd(100)
4  turtle.lt(90)
5  turtle.fd(100)
6  turtle.lt(90)
7  turtle.fd(100)
8  turtle.lt(90)
9  turtle.fd(100)
10 turtle.lt(90)
11 turtle.done()
```

Listing 7.1: Quadrat (Länge 100).

```
1  import turtle
2
3  turtle.fd(275)
4  turtle.lt(90)
5  turtle.fd(275)
6  turtle.lt(90)
7  turtle.fd(275)
8  turtle.lt(90)
9  turtle.fd(275)
10 turtle.lt(90)
11 turtle.done()
```

Listing 7.2: Quadrat (Länge 275).

Wir speichern die Zahl für die Seitenlänge des Quadrats in einer Variablen ab und verwenden die Variable anschliessend als Argument bei den vier Funktionsaufrufen. Listing 7.3 zeigt das

verbesserte Programm. Wenn das Programm ausgeführt wird, dann wird in Zeile **drei** die Zahl 100 in der Variablen mit dem Namen a gespeichert. In den Zeilen vier, sechs, acht und zehn wird die Variable a dann durch die gespeicherte Zahl ersetzt.

```
1  import turtle
2
3  a = 100
4  turtle.fd(a)
5  turtle.lt(90)
6  turtle.fd(a)
7  turtle.lt(90)
8  turtle.fd(a)
9  turtle.lt(90)
10 turtle.fd(a)
11 turtle.lt(90)
12 turtle.done()
```

Listing 7.3: Die Variable a speichert die Zahl 100 (`quadrat_var.py`).

```
1  import turtle
2
3  a = 275
4  turtle.fd(a)
5  turtle.lt(90)
6  turtle.fd(a)
7  turtle.lt(90)
8  turtle.fd(a)
9  turtle.lt(90)
10 turtle.fd(a)
11 turtle.lt(90)
12 turtle.done()
```

Listing 7.4: Zeile drei wurde angepasst - nun wird die Zahl 275 gespeichert.

Wir können das Programm nun bequem verändern. Wenn wir ein Quadrat mit der Seitenlänge 275 möchten, dann müssen wir nur noch Zeile drei abändern. Listing 7.4 zeigt das angepasste Programm.

7.2 Was ist eine Variable?

Eine Variable können wir uns als einen Behälter mit einer Beschriftung zur Aufbewahrung von Werten vorstellen. In Abbildung 7.1 sind die beiden Variablen name und age als Behälter dargestellt.

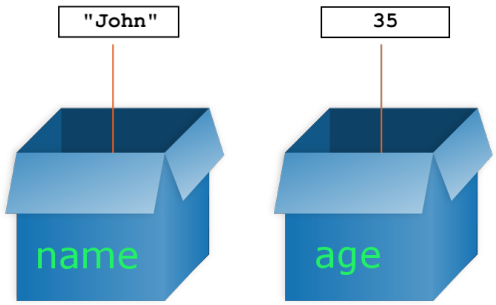


Abbildung 7.1: Zwei Variablen, dargestellt als Behälter mit Namen.[1].

Jeder Behälter in Abbildung 7.1 besitzt eine Beschriftung und einen gespeicherten Wert. Der Behälter mit der Beschriftung name entspricht einer Variablen mit dem Namen name. Darin wird der Wert "John" (ein Text) gespeichert. Die andere Variable hat den Namen age und speichert den Wert 35 (eine Zahl).

Definition 6 — Variable. Eine Variable ist ein Speicherplatz mit einem Namen. In der Variablen kann ein beliebiger Wert gespeichert werden. Dies geschieht in Python mit einer **Zuweisung**. Den gespeicherten Wert können wir auch auslesen. Dazu notieren wir den Namen der Variablen. Den Namen einer Variablen können wir selbst wählen.

[1] Angepasst von http://www.pyworld.in/Python/pythonVariables.html?i=1.

7.3 Was ist ein Wert?

Computerprogramme verarbeiten Daten. Ein **Wert** ist eines der grundlegenden Dinge, mit denen ein Programm arbeitet. Die Zahl `42` oder der Text `"2001: Odyssee im Weltraum"` sind zwei Beispiele für einen Wert. Werte können wir in verschiedene Kategorien aufteilen. Abbildung 7.2 zeigt eine Übersicht der bisher kennengelernten Werte.

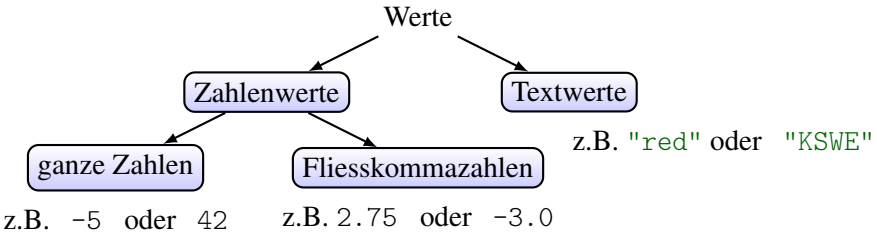


Abbildung 7.2: Diese Darstellung ist nicht abschliessend. Es gibt noch mehr Kategorien.

Einen Wert können wir **direkt darstellen**, in dem wir den Wert in das Programm eintippen.

Definition 7 — Literal. Eine Zeichenfolge, welche zur direkten Darstellung von Werten benutzt wird, nennen wir Literal.

■ **Beispiel 7.1** `-5`, `42`, `2.75`, `-3.0`, `"red"` und `"KSWE"` sind Beispiele für Literale. ■

Neben Literalen ist es auch möglich, durch einen **Funktionsaufruf** einen Wert zu **erzeugen**.

■ **Beispiel 7.2** Listing 7.5 und Listing 7.6 zeigen, wie wir mit einem Funktionsaufruf einen Wert erzeugen. Die Programme beinhalten neben den erzeugten Werten auch Literale (`1`, `101` und `42`).

```
1 import random
2
3 zufallszahl = random.randrange(1, 101)
4 print(zufallszahl)
```

Listing 7.5: Erzeugt eine Zufallszahl zwischen 1 und 100 und gibt diese dann in der Konsole aus.

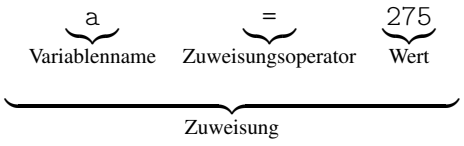
```
1 import math
2
3 quadratwurzel = math.sqrt(42)
4 print(quadratwurzel)
```

Listing 7.6: Berechnet die Quadratwurzel aus 42 und gibt diese dann in der Konsole aus.

■

7.4 Was ist eine Zuweisung?

Wir haben mit `a = 275` einen neuen Befehl kennengelernt. Der Befehl gehört zur Kategorie „Zuweisung“ (eng. assignment). `a = 275` ist somit ein Beispiel für eine **Zuweisung**.



Wir können die Zuweisung auch lesen als „a soll 275 speichern“. Eine Zuweisung erkennen wir am **Zuweisungsoperator**[2] (eng. assignment operator). Da das Gleichheitszeichen (=) in Python für das

[2] Sie kennen den Begriff bereits aus der Mathematik. Bei der Addition von zwei Zahlen, z.B. $2+3$, stellt das „Pluszeichen“ einen mathematischen Operator dar. Oft wird in der Mathematik auch der Begriff Rechenzeichen verwendet.

Speichern von Werten in einer Variablen benutzt wird (und nicht wie in der Mathematik im Sinne einer Gleichung), verwenden wir für das Zeichen den Fachbegriff Zuweisungsoperator.

Wichtig! Bei einer **Zuweisung** muss auf der **linken** Seite des Zuweisungsoperators eine Variable stehen. ■

Clean Code 2 — Leerzeichen 1. Links und rechts eines **Zuweisungsoperators** fügen wir je ein Leerzeichen ein.

7.5 Wo wird der Inhalt einer Variablen gespeichert?

Eine Variable ist ein Speicherplatz mit einem Namen. Das Computerbauteil, in dem der Speicherplatz für eine Variable reserviert wird, heisst **Arbeitsspeicher**. Häufig wird das Bauteil auch als RAM[3] bezeichnet, da für den Arbeitsspeicher im Computer fast ausschliesslich dieser Speichertyp verwendet wird. Der Arbeitsspeicher für Standardnotebooks ist typischerweise zwischen 8 GB und 32 GB gross. Nur ein **Teil** davon wird zur Speicherung von Variablen verwendet.

7.6 Welche Variablennamen sind erlaubt?

Den Namen einer Variablen können Sie fast frei wählen. Sie können Gross- und Kleinbuchstaben verwenden, Ziffern $(0-9)$ und den Unterstrich[4] (eng. underscore) (_). Es gibt nur sehr wenige Einschränkungen:

- Der Name darf **nicht** mit einer **Ziffer beginnen**.
- Der Name darf **nicht** einem Schlüsselwort (eng. keyword) [5] von Python entsprechen.
- Der Name darf **keine** Leerzeichen beinhalten.

Die Namen `1farbe`, `import` und `meine Farbe` sind somit **nicht** erlaubt. `farbe1`, `warenimport` und `meineFarbe` sind hingegen erlaubt.

Wichtig! Python **unterscheidet** zwischen Gross- und Kleinbuchstaben. Wir sagen, die Programmiersprache ist **case sensitive**. ■

■ **Beispiel 7.3** Wenn Sie eine Variable `tmp`[6] verwenden und an einer anderen Stelle `TMP` schreiben, dann sind das für Python **zwei unterschiedliche** Variablen. ■

Übrigens: Die korrekte Verwendung der Gross- und Kleinbuchstaben gilt auch für Schlüsselwörter (wie `import`).

7.7 Wie wählen wir die Variablennamen?

Eigentlich können wir recht frei einen Namen wählen, zum Beispiel `Meine_LiebliNgs_figur` oder `birne`. Jedoch **sollten** wir einen Variablennamen so wählen, dass wir durch den Namen erkennen können, was für ein Wert darin gespeichert ist. Dies entspricht der **Clean-Code**-Idee.

Clean Code 3 — Sinnvolle Variablennamen. Wir wählen **Variablennamen** so, dass wir möglichst direkt verstehen, was darin gespeichert wird.

[3] Random Access Memory (dt. Direktzugriffsspeicher)
[4] Auch Bodenstrich oder Tiefstrich genannt.
[5] Auch reserviertes Wort genannt. Dies sind Wörter, die in Python eine bestimmte Bedeutung haben. Zum Beispiel ist `import` ein Schlüsselwort. Schlüsselwörter werden in einer IDE meist farblich hervorgehoben.
[6] Abkürzung für **tmp**orary.

■ **Beispiel 7.4** In Listing 7.3 haben wir den Namen `a` für die Variable gewählt. Der Name bezeichnet die Seitenlänge des Quadrats. Dies kann als ein sinnvoller Name betrachtet werden, da die Seitenlänge eines Quadrats typischerweise in der Geometrie mit *a* bezeichnet wird. Ausserdem wird das Programm unter dem Namen `quadrat_var.py` gespeichert. Daraus ist ersichtlich, dass es sich um die Seitenlänge handeln muss. Wir können es auch noch expliziter machen und den Namen `seitenlänge` verwenden.
Ein schlechter Name wäre `apfel`. Dieser Name sorgt für Verwirrung. Geht es hier um Obst? Was wird in einer Variablen mit dem Namen `apfel` gespeichert? Warum eine Zahl? Sollte es nicht eine Apfelsorte sein? ■

Clean Code 4 — Snake Case. In Python ist es üblich, für Variablennamen **nur** Kleinbuchstaben zu verwenden. Besteht der Name aus mehreren Wörtern, dann „trennen“ wir diese Wörter durch einen Unterstrich (_). Diese Schreibweise wird Snake Case genannt.

■ **Beispiel 7.5** Die Variablennamen `meine_farbe`, `zahl_1` oder `temperatur_in_celsius` sind in Snake Case notiert. ■

H Andere Programmiersprachen empfehlen andere Schreibweisen für Variablennamen. Abbildung 7.3 zeigt auf humorvolle Art die gebräuchlichsten Schreibweisen.

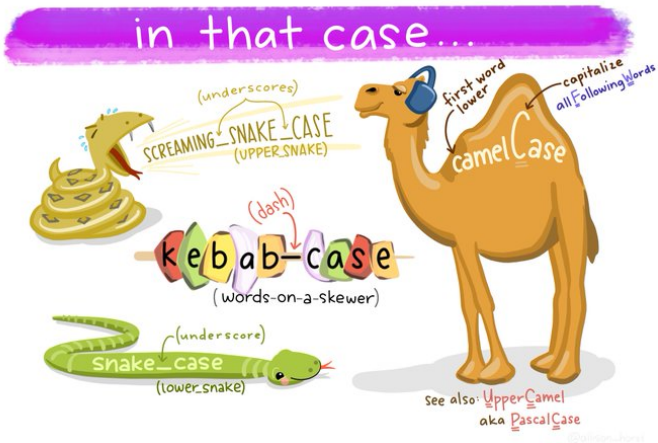


Abbildung 7.3: In Python ist „Snake Case“ üblich.

7.8 Wie benutzen wir den Inhalt einer Variablen?

Wir können jederzeit den Inhalt einer Variablen benutzen, in dem wir den entsprechenden **Variablennamen** notieren. Python ersetzt dann während der Ausführung den Variablennamen durch den gespeicherten Inhalt. Wir sagen, die Variable wird **ausgelesen**.

■ **Beispiel 7.6** Im Programm aus Listing 7.3 wird der Inhalt der Variablen `a` an vier Stellen benutzt. In Zeile vier wird für den Funktionsaufruf die Variable als Argument verwendet. Die Variable wird während der Ausführung durch die gespeicherte Zahl (100) ersetzt. Dies wiederholt sich für die Zeilen sechs, acht und zehn. ■

Wichtig! Das Auslesen einer Variable löscht den gespeicherten Inhalt der Variable **nicht**. ■

H Falls beim Auslesen die passende Variable nicht existiert (da zum Beispiel vorher kein Wert darin gespeichert wurde), dann gibt es eine **Fehlermeldung**. Typischerweise wird ein `NameError` erzeugt.

7.9 Aufgaben

In den folgenden Aufgaben setzen Sie sich mit Variablen auseinander.

7.9.1 Aufgabe 1

Entscheiden Sie für **jeden** Variablennamen, ob es sich um einen gültigen Namen handelt oder nicht. In dieser Aufgabe spielt der Clean-Code-Gedanke keine Rolle. Es geht nur darum, ob der Name gültig ist. Falls es sich um einen ungültigen Namen handelt, dann markieren Sie die ungültigen Zeichen. Ist es ein gültiger Name, dann markieren Sie dies mit ✓.

1. `Monty Python`
2. `monty python`
3. `monty_python`
4. `0ahnung`
5. `s109`
6. `meine_lieblingsfarbe`

7.9.2 Aufgabe 2

1. Es ist 13:37:42 Uhr. Die Anzahl der Stunden, die Anzahl der Minuten und die Anzahl der Sekunden soll jeweils in einer Variablen gespeichert werden. Notieren Sie das passende Programm und beachten Sie die **Clean-Code-Regeln** bzgl. Variablennamen.

2. Die Temperatur beträgt 27 °C. Notieren Sie ein Programm, welches diese Temperatur in einer Variablen speichert. Verwenden Sie einen sinnvollen Namen.

3. Die Latenz beträgt 21 ms. Notieren Sie ein Programm, welches diese Latenz in einer Variablen speichert. Verwenden Sie einen sinnvollen Namen.

7.9.3 Aufgabe 3

Studieren Sie das Listing 7.7. Schreiben Sie das Programm so um, dass drei Variablen mit sinnvollen Namen benutzt werden. Definieren Sie die Variablen nach der `import`-Anweisung.

Wichtig! Jeder **Wert** (z.B. `100` in `turtle.fd(100)`) kann durch eine **Variable** „ersetzt“ werden, die den gleichen Wert speichert (z.B. `a = 100` und dann `turtle.fd(a)`). ■

```
1  import turtle
2
3  turtle.fd(100)
4  turtle.lt(120)
5  turtle.fd(100)
6  turtle.lt(60)
7  turtle.fd(100)
8  turtle.lt(120)
9  turtle.fd(100)
10 turtle.lt(60)
11 turtle.done()
```

Listing 7.7: Ein Rhombus.