

2 Erste Programmelemente

2.1 Kommentare

Kommentarzeilen in Programmen sind Zeilen, welche vom Computer nicht beachtet werden. In allen Programmiersprachen beginnen diese Zeilen mit einem bestimmten Zeichen. Zeilen, welche mit dem Zeichen # beginnen, werden von Python als Kommentare betrachtet und ignoriert. Steht das Zeichen # nicht am Anfang einer Zeile, wird nur der Teil auf der Zeile ignoriert, welcher nach dem Zeichen kommt.

Beispiel:

```
1 # Dies ist eine Kommentarzeile, welche ignoriert wird
2 print("Hallo!") # Die Zeile wird erst ab hier ignoriert.
```

Nun fragt man sich, was denn der Sinn hinter Zeilen ist, welche von Python gar nicht beachtet werden. Diese erfüllen sehr wichtige Funktionen:

- Sie dienen dazu, dass andere Personen deine Programme einfacher lesen und verstehen können.
- Sie helfen dir, ein Programm oder einen Programmabschnitt zu planen.

Wenn du zum Beispiel ein Programm schreiben möchtest, welches den Benutzer nach zwei Zahlen fragt, diese addiert und dann die Summe auf dem Bildschirm ausgibt, kannst du dieses Programm schon planen, ohne eine Ahnung zu haben, wie die einzelnen Schritte funktionieren.


Dazu schreibst du das folgende Programm:

```
1 #####
2 ##                                     ##
3 ## Programm zur Addition von zwei Zahlen ##
4 ##                                     ##
5 #####
6 # Benutzer nach zwei Zahlen fragen und diese speichern
7 # Die erhaltenen Zahlen addieren
8 # Die Summe auf dem Bildschirm ausgeben
```

Wenn du Python dieses Programm ausführen lässt, passiert erst einmal gar nichts. Es besteht ja nur aus Zeilen, welche von Python nicht beachtet werden. Doch du hast nun eine ziemlich genaue Vorstellung, in welcher Reihenfolge du das Programm schreiben willst. Sobald du gelernt hast, wie man dies tut, kannst du die entsprechenden Blöcke zwischen die Kommentare programmieren.

Beachte zum Schreiben von Kommentaren folgende Regeln:

- Schreibe am Anfang des Programms einen ausführlicheren Kommentar, was das Programm tut.
- Schreibe zu jedem logisch zusammengehörenden Block einen Kommentar, der erklärt, was dieser Programmblock tut. Fällt dir keine passende Beschreibung dazu ein, besteht die Möglichkeit, dass der Block nicht tut, was du möchtest oder er gar überflüssig ist.
- Schreibe zu jeder Variable eine kurze Erklärung, wozu sie benutzt wird. Fällt dir keine gute Erklärung ein, brauchst du die entsprechende Variable vermutlich nicht. (Was Variablen sind, lernst du gleich im nächsten Abschnitt.)

	Name:	Klasse:	Datum:
	BS Scripte	BFS12 Python - 2	Imr

2.2 Variablen

Variablen in Python werden genau wie in jeder anderen Programmiersprache zum Abspeichern oder Zwischenspeichern von Werten (Zahlen, Text, Listen usw.) benutzt.

In Python stellt man sich Variablen jedoch besser als Namensschilder vor, welche man einem Objekt gibt. Wenn du zum Beispiel ein Zwischenergebnis einer Rechnung hast, gibst du ihm einen Namen, welcher erklärt, um was es sich handelt. In diesem Fall ist vermutlich der Name `zwischenenergebnis` angebracht.

Das Anhängen eines Namensschildes wird in Python mit „`=`“ ausgedrückt. Auf der linken Seite des Gleichzeichens steht der neue Name und auf der rechten Seite das Objekt, welches den Namen erhalten soll. Probiere folgendes Code in der Python Konsole aus:

```
>>> zwischenenergebnis = 3.14111222
>>> zwischenenergebnis
3.14111222
```

Wie du siehst steht der Name `zwischenenergebnis` jetzt für die Zahl `3.1411222`.

Die Wahl des Zeichens `=` ist aus mathematischer Sicht ungünstig. In der Mathematik hat das Zeichen eine andere Bedeutung. Es sagt einfach, dass zwei Sachen gleich sind. Dabei spielt es aber keine Rolle, was links und was rechts des Gleichzeichens steht.

Mathematisch gesehen ist der folgende Ausdruck eine Gleichung ohne Lösung:

$$x = x + 1$$

In Python macht es aber durchaus Sinn, die Befehlsabfolge

```
>>> x = 4
>>> x = x + 1
>>> x
5
```

auszuführen. Es wird nämlich einfach der Wert der Variable um eins erhöht.

Im letzten Kapitel hast du die Funktion `input()` kennen gelernt. Diese kannst du nun benutzen, um eine Eingabe von Benutzer einzulesen und anschließend in einer Variable abzuspeichern. Du musst dir immer auch überlegen, was du den Benutzer fragen möchtest, damit das Programm möglichst bedienerfreundlich wird.

So fragen dir den Benutzer zum Beispiel mit dem folgenden Befehl nach seinem Namen und speichern ihn in einer Variable mit der Bezeichnung `name` ab.

```
>>> name = input("Wie heisst du?")
Wie heisst du? Paula
>>> name
'Paula'
```

Wie du siehst, ist nun unter `name` der Wert `'Paula'` gespeichert. Warum die Zeichen `'` vorkommen, lernst du im nächsten Kapitel. Python kennzeichnet damit die Tatsache, dass es sich um Text handelt.

2.2.1 Aufgaben

1. Im Folgenden werden einige Zeilen Programmcode ausprobiert. Überlege dir, was nach dem Ausführen der Zeilen in den Variablen `erste_zahl`, `zweite_zahl` und `temp` gespeichert ist. Wozu dient der kurze Programm-Ausschnitt?

```
>>> erste_zahl = 7
>>> zweite_zahl = 9
>>> temp = erste_zahl
>>> erste_zahl = zweite_zahl
>>> zweite_zahl = temp
```

2. Probiere im Befehlsprompt die folgenden Variablennamen aus, in dem du eine beliebige Zahl abspeicherst: `zahl2`, `Zahl2`, `Zahl 2`, `2zahl`, `zahl.2`, `Zahl_2`, `2_Zahl`, `_Zahl2`, `2_Zahl`.

Dies machst du am einfachsten, in dem du versuchst, etwas unter der Variable abzuspeichern und anschliessend kontrollierst, ob es geklappt hat. Etwa so:

```
>>> variablen_name = 28
>>> variablen_name
28
```

- (a) Welche Variablennamen sind zulässig?
 - (b) Finde an Hand der obigen Beispiele und der Python-Dokumentation heraus, wie die Regeln für zulässige Variablennamen lauten.
3. Schreibe in einer neuen Datei mit dem Namen `papagei.py` ein Programm, welches vom Benutzer mit `input()` eine Eingabe verlangt, diese in einer Variable speichert und anschliessend mit `print()` wieder ausgibt.

2.3 Verwenden von Modulen

Gewisse Funktionalitäten wurden in Python in sogenannte Module ausgegliedert. Auf diese Art kann die Grundinstallation von Python schlank gehalten werden und Module nach bedarf nachgeladen werden. Wir schauen uns das ganze am Beispiel des math Moduls an. In diesem werden verschiedene mathematische Funktionen wie zum Beispiel `sqrt()` zur Berechnung der Quadratwurzel zur Verfügung gestellt. Fühst du die folgende Zeile in der Python-Konsole aus, wird dir Python einen Fehler ausgeben:

```
>>> sqrt(49)
```

Du musst zuerst die `sqrt()` Funktion aus dem math Modul laden. Dies geschieht über den Befehl `import`.

```
>>> from math import sqrt
>>> sqrt(49)
```

berechnet nun erfolgreich die Wurzel aus 49. Falls man alle Befehle von einem Modul importieren möchte, kann man das Modul als ganzes importieren. Dann muss man aber bei jedem Befehl festlegen, dass er aus dem besagten Modul kommt.

```
>>> import math
>>> math.sqrt(49)
```

Teilweise findet man in Beispielen auch die Form `from math import *`, welche alle Befehle aus dem math importiert. Dies gilt aber vor allem in grösseren Programmen als schlechter Programmierstil, da man nicht genau weiss, welche Befehle damit alles importiert werden.

Oft auch nützlich ist das random Paket, welches verschiedene Funktionen enthält, um Zufallszahlen¹ zu generieren, Listen zu mischen oder zufällige Elemente auszuwählen. Wir können zum Beispiel mit den folgenden Zeilen einen Würfel simulieren:

```
>>> from random import randint
>>> randint(1, 6)
```

Siehe auch

Einen Überblick über alle verfügbaren Pakete in der Python Standard Bibliothek findest du in der Python-Dokumentation unter:
<https://docs.python.org/3/library/>

2.3.1 Aufgaben

1. Schreibe ein Programm, welches die Länge der Diagonale eines Rechtecks mit den Seitenlängen a und b berechnet.
2. Wenn man die drei Seitenlängen eines Quaders hat, dann kann man die drei Flächendiagonalen (Längen der Diagonalen in einer Seitenfläche des Quaders) sowie die Raumdiagonale berechnen. Schreibe dazu ein Programm.
3. Drei Würfel werden miteinander geworfen. Schreibe ein Programm, welches drei Würfe simuliert und die Summe der geworfenen Zahlen ausgibt.

Abgewandelt von <https://pythonbuch.com/elemente.html>

¹Der Computer kann keine wirklichen Zufallszahlen generieren. Man spricht darum von Pseudozufallszahlen.