Informelle Grundlagen

Es folgt eine kurze Einleitung in einige Teile der Programmiersprache Python.

Ausgabe von Text

Wie wir im vorigen Kapitel gesehen haben, gibt der Programmcode

```
print("Text Hier Einfügen")
```

den eingefügten Text über die Standardausgabe (CMD, Linux-Terminal etc.) aus. Dies funktioniert, indem die Funktion `print` mit einem Text als einzigen Parameter aufgerufen wird. `print` tut folgendes:

- Jeder Parameter wird, sofern er keiner ist, in einen Text umgewandelt
- Jeder Parameter wird der Reihe nach über die Standardausgabe ausgegeben
- Zwichen den Parametern wird je ein Leerzeichen ausgegeben
- Am Schluss wird eine Neuzeile ausgegeben

Somit gibt das Programm

```
print("jedes", "Wort", "ist", "getrennt")
```

den Text `jedes Wort ist getrennt` aus.
Die CMD erwartet, dass von jedem Programm als letztes Zeichen die Neuzeile ausgegeben wird. Dieser Punkt spielt somit bei einem einzigen print-Aufruf keine Rolle, wird jedoch auffällig, sobald wir mehrere davon haben.

Anders als in Sprachen wie C und Java muss eine Anweisung nicht mit einem Semikolon (`;`) abgeschlossen werden, es reicht hierfür das Zeilenende aus. Somit kann man mehrere Zeilen Text der Reihe nach ausgeben, indem man mehrere `print`-Aufrufe in aufeinanderfolgende Zeilen schreibt:

```
print("Dieser Text kommt in Zeile 1")
print("Dieser Text kommt in Zeile 2")
```

Python ignoriert Leerzeilen bei der Ausführung. Ebenso wird, sobald es auf das `#`-Zeichen stößt, der Rest der Zeile (bis zum Zeilenende) als Kommentar gelesen und ignoriert. Somit verhalten sich folgende (durch `# ---` getrennte) Programme vollkommen gleich:

```
print("abc")
print("def")
# ---


print("abc")


print("def")


# ---
# diese Anweisung gibt "abc" aus
print("abc")
# diese Anweisung gibt "def" aus
print("def")
# ---
print("abc") # diese Anweisung gibt "abc" aus
print("def") # diese Anweisung gibt "def" aus
```

Variablen und Zuweisungen

Will ich beispielsweise dreimal den Text `Dieser Text wird dreimal ausgegeben` ausgeben, so kann ich dies durch das Programm

```
print("Dieser Text wird dreimal ausgegeben")
print("Dieser Text wird dreimal ausgegeben")
print("Dieser Text wird dreimal ausgegeben")
```

erreichen. Ist der Text jedoch deutlich länger (beispielsweise `Es ist Ziel dieses Programms, diesen Text siebenmal, also einmal für jede Zahl von 1 bis 7, auf die Standardausgabe auszugeben`), wird das Programm unnötig lang, und wünscht man, den Text zu ändern, so muss man das mehrmals tun. Python erlaubt es, dies zu vereinfachen: der Text kann einer Variable zugewiesen werden, welche daraufhin als Platzhalter für den Text verwendet werden kann:

```
my_text = "Es ist Ziel dieses Programms, diesen Text siebenmal, also einmal für jede Zahl von
1 bis 7, auf die Standardausgabe auszugeben"
print(my_text)
print(my_text)
```

```
print(my_text)
print(my_text)
print(my_text)
print(my_text)
print(my_text)
```

Es ist auch möglich, den Wert der Variable im Verlauf des Programmes zu ändern:

```
mein_text = "Es ist Ziel dieses Programms, diesen Text siebenmal, also einmal für jede Zahl
von 1 bis 7, auf die Standardausgabe auszugeben"
print(my_text)
print(my_text)
print(my_text)
print(my_text)
print(my_text)
print(my_text)
print(my_text)
mein_text = "Es ist Ziel dieses Programms, diesen Text dreimal, also einmal für jede Zahl von
1 bis 3, auf die Standardausgabe auszugeben"
print(my_text)
print(my_text)
print(my_text)
```

Ebenso können mehrere Variablen definiert werden.

```
even = "ist eine gerade Zahl"
uneven = "ist eine ungerade Zahl"
print("0", even)
print("1", uneven)
print("2", even)
print("3", uneven)
print("4", even)
print("5", uneven)
print("6", even)
```

Auch können Variablen bei Bedarf gelöscht werden

```
x = "Hallo!"
print(x)
del x
```

(Wird auf eine Variable zugegriffen, die nicht definiert oder schon gelöscht wurde, so stoppt Python die Ausführung des Programms und wirft einen Fehler)

Der Name einer Variable darf aus

- Groß- und
- Kleinbuchstaben
- Ziffern
- Unterstrichen (`_`)

bestehen, und nicht mit einer Ziffer anfangen. Ebenso gibt es bestimmte sog. "keywords", welche besondere Bedeutung haben und nicht zur Benennung von Variablen verwendet werden dürfen. Bisher ist uns nur `del` bekannt, es gibt jedoch sehr viele solcher Worte in Python (beispielsweise `in`, `if`, `while`, `match`, `class`, `def`, `from`). Es ist üblich, innerhalb eines Programmes dasselbe Format für Variablennamen zu verwenden: so kann ich eine Variable, die die Position des Spielers in einem Videospiel speichert, `spielerposition`, aber auch `spieler_position`, `SpielerPosition` oder `_Z7spieler8position` nennen. Python lässt diese Namen allesamt zu, jedoch führt Verwendung von mehr als einem Format oft zu Verwirrung und schlecht lesbarem Code. Es ist daher üblich, sich an den offiziellen Python-Styleguide PEP 8 zu halten, nach welchem Variablen Namen im `snake_case` (Worte klein geschrieben und mit Unterstrichen getrennt) tragen sollen. "Konstante Variablen", deren Wert im Laufe des Programms nicht geändert wird, werden im `SCREAMING_SNAKE_CASE` benannt, also großgeschrieben und mit Unterstrichen getrennt.

Es wird auch zumeist auf Englisch programmiert. Alle eingebauten Funktionen und Klassen sind beispielsweise auf Englisch benannt und dokumentiert. Auch ist davon abzuraten, nicht-ASCII-Buchstaben und Zeichen (wie `äöü`, aber auch `î` und `á`) in Variablennamen zu verwenden. In allem folgenden Programmcode werden Variablen somit auf Englisch benannt (Kommentare bleiben jedoch deutsch).

Benutzereingaben

Für etwas Interaktvität kann es nützlich sein, von dem Benutzer des Programms eine Eingabe zo fordern. Hierfür wird die Python-Funktion `input` verwendet. Das einfachste hiermit realisierbare Programm lautet

```
print(input())
```

Es wartet, bis der Benutzer im Terminalfenster eine Zeile Text eingibt und gibt diese Zeile sofort wieder aus. Natürlich kann dieser Wert auch in einer Variable gespeichert werden.

```
print("Wie heißt du?")
name = input()
print("Hallo,", name)
```

```
print("Ich weiß jetzt, dass du", name, "heißt!")
```

Optional kann dieser Funktion auch ein Parameter übergeben werden. Wird zum Beispiel `input("Wie heißt du?")` aufgerufen, so wird die Zeile mit diesem Text begonnen, wobei der Nutzer unmittelbar darauf seinen Namen eingeben kann. Das Ergebnis liest sich dann als `Wie heißt du?Max Mustermann` (wobei nur der Teil nach dem Fragezeichen eingegeben werden musste). Generell wird empfohlen, an das Ende dieses Textes ein Leerzeichen zu hängen. Die Benutzererfahrung kann dazu deutlich verbessert werden, wenn alle 'leeren Aufrufe' von `input()` durch eine einfache Eingabeaufforderung wie `input("> ")` ersetzt werden.

Version #17
Erstellt: 2025-11-16 19:39:38 UTC von Fabian Schiener
Zuletzt aktualisiert: 2026-05-10 18:21:50 UTC von Fabian Schiener