LU02c - Variablen in Python

Lernziele:

- Sie haben das Konzept der Variablen verstanden. Sie wissen, was Variablentypen, -namen und -werte sind.
- Sie wissen, wie man String-, Integer-, Gleitkomma- und boolesche Variablen erstellt und verwendet.

Einleitung

Wir haben uns bereits ein wenig mit `Strings` vertraut gemacht, als wir mit Benutzereingaben zu tun hatten. Wenden wir uns nun anderen Variablentypen zu, die in Python häufig verwendet werden.

Eine Variable kann man sich als einen Behälter vorstellen, in dem Informationen eines bestimmten Typs gespeichert werden können. Beispiele für diese verschiedenen Typen in Python sind Text (`string`), ganze Zahlen (`int`), Fließkommazahlen (`float`) und ob etwas wahr oder falsch ist (`bool`). Ein Wert wird einer Variablen mit dem Gleichheitszeichen (=) zugewiesen.

```
months = 12
```

In der obigen Anweisung wird der Wert 12 einer Variablen namens `months` zugewiesen. Die Anweisung könnte wie folgt gelesen werden: „der Variablen `months` wird der Wert 12 zugewiesen“.

Beachten Sie, dass die Typdeklaration in Python implizit ist und Sie den Typ nicht unbedingt in der Zuweisung deklarieren müssen.

Der Wert einer Variablen kann mit Hilfe des +-Zeichens mit einer Zeichenkette verbunden werden, wie im folgenden Beispiel zu sehen ist.

```
text = 'contains text'
whole_number = 123
floating_point = 3.141592653
true_or_false = True

print('Text variable: ' + text)
print('Integer variable: ' +
```

```
str(whole_number))
print('Floating-point variable: ' +
str(floating_point))
print('Boolean: ' + str(true_or_false))

Text variable: contains text
Integer variable: 123
Floating-point variable: 3.141592653
Booolean: true
```

Um Variablen mit den Datentypen `int`, `float` und `bool` in der Ausgabe `print()` mit einem String zu verknüpfen, müssen sie entweder vor oder während der Ausgabe mit dem Befehl str() in einen String umgewandelt werden.

Ändern eines einer Variable zugewiesenen Wertes

Eine Variable existiert ab dem Zeitpunkt ihrer Deklaration, und ihr Anfangswert bleibt erhalten, bis ihr ein anderer Wert zugewiesen wird. Sie können den Wert einer Variablen mit einer Anweisung ändern, die den Variablennamen, ein Gleichheitszeichen und den neu zuzuweisenden Wert enthält.

```
number = 123
print('The value of the variable is ' +
str(number))

number = 42
print('The value of the variable is ' +
str(number))

The value of the variable is 123
The value of the variable is 42
```

Schauen wir uns die Ausführung des vorangegangenen Programms Schritt für Schritt an:

1. Wenn eine Variable zum ersten Mal im Programm auftaucht, erstellt der Computer einen „benannten Container“ für die Variable. Dann wird der Wert auf der rechten Seite des Gleichheitszeichens in diesen benannten Container kopiert.
2. Wann immer eine Variable in einem Programm durch ihren Namen referenziert wird - hier wollen wir die Zeichenkette „The value of the variable is “ gefolgt vom Wert der Zahlenvariablen ausgeben - wird ihr Wert aus einem Container mit dem entsprechenden Namen

abgerufen.
3. Die Variable wird dann wieder über ihren Namen im Programm referenziert - es wird ein neuer Wert in den Container gelegt
4. Die Variable wird dann wieder über ihren Namen im Programm referenziert - wir wollen wieder die Zeichenkette „The value of the variable is “, gefolgt vom Wert der Zahlenvariablen, ausgeben. Wir fahren wie gewohnt fort, indem wir den Wert von number aus einem Container mit seinem Namen abrufen.

Am Ende des Programms werden Sie feststellen, dass der ursprüngliche Wert der Variablen verschwunden ist. **Eine Variable kann immer nur einen Wert enthalten.**

Im Gegensatz zu anderen Programmiersprachen können Sie eine Variable auch umdefinieren, um einen anderen impliziten Typ zu erhalten:

```
number = 123
print('The value of the variable is ' +
str(number))

number = 42.6
print('The value of the variable is ' +
str(number))

The value of the variable is 123
The value of the variable is 42.6
```

Im ersten Beispiel hatte die Variable `number` in Zeile 1 den Typ `int` und wurde in Zeile 4 einem anderen Typ `int` neu zugewiesen. Im zweiten Beispiel hatte die Variable `number` in Zeile 1 den Typ `int` und wurde in Zeile 4 dem Typ `float` neu zugewiesen. Das ist in Python völlig in Ordnung, kann aber seltsam wirken, wenn man aus anderen Sprachen wie z.B. Java kommt.

Implizite Typumwandlung in Python

Wenn Sie an bestimmte andere Programmiersprachen wie Java gewöhnt sind, haben Sie vielleicht schon explizite Typendeklarationen gesehen, wie z.B.:

```
boolean myBool = false;
int value = 10;
```

In Python ist dies nicht notwendig, und Python-Variablen müssen nicht explizit deklariert werden, um Speicherplatz zu reservieren. Die Deklaration erfolgt automatisch, wenn Sie einer Variablen einen

Wert zuweisen. Python ist intelligent genug, um zu verstehen, welche Operationen Sie erreichen wollen, und wird entsprechend handeln.

Ein Beispiel ist das folgende Programm:

```
number1 = 1
number2 = 2
print(number1/number2)
```

ergibt:

```
0.5
```

in Java würde es folgendermassen aussehen:

```
int number1 = 1;
int number2 = 2;
System.out.println(number1/number2);
```

ergibt:

```
0
```

da beide Datentypen `Integer` sind.

Wir werden hier an dieser Stelle nicht zu sehr auf die Mechanismen der Typendeklaration in Python eingehen, aber Sie können hier mehr darüber lesen.

Benennung von Variablen in Python

Die Benennung von Variablen ist ein grundlegender Aspekt bei der Beschreibung eines Programms. Schauen wir uns zwei Beispiele an.

```
a = 3.14
b = 22.0
c = a * b * b

print(c)
```

```
1519.76
```

```
pi = 3.14
radius = 22
surface_area = pi * radius * radius

print(surface_area)

1519.76
```

Die beiden vorangegangenen Beispiele funktionieren auf die gleiche Weise und geben das gleiche Ergebnis aus. Eines davon ist jedoch viel verständlicher. Hier geht es darum, den Flächeninhalt eines Kreises zu berechnen. In der ersten Zeile wird der Wert von pi definiert, in der zweiten der Radius des Kreises und in der dritten die Fläche berechnet.

Die Benennung von Variablen unterliegt bestimmten Beschränkungen. Variablennamen dürfen bestimmte Sonderzeichen, wie Ausrufezeichen (!), nicht enthalten. Leerzeichen sind ebenfalls nicht erlaubt, da sie dazu dienen, Teile von Befehlen zu trennen. Anstelle von Leerzeichen wird in vielen Programmiersprachen zur Abgrenzung von Wörtern entweder eine als `snake_case` oder `camelCase` bekannte Schreibweise verwendet.

Python-Variablennamen werden immer in `snake_case` geschrieben.

```
camelCaseVariable = 7 # will work, but not convention in Python
snake_case_variable = 7 # this is the format recommended by Python
```

Python hat Community-Richtlinien, wie Code formatiert werden sollte. Sie können den Leitfaden hier einsehen. Auch hier am BZZ haben wir BZZ-Codingstandards entwickeln nach denen wir unsere Programme schreiben möchten. Wir werden diese Richtlinien in diesem Kurs so weit wie möglich befolgen.

Folgende Punkte sind aus den BZZ-Codingstandards kopiert und für die Variablenbezeichner wichtig:

Sprechende Bezeichner

Ein sprechender Bezeichner sagt etwas über den Sinn und Zweck einer Komponente aus. Durch den Einsatz von sprechenden Bezeichnern werden viele Kommentare überflüssig.

	Die Bezeichner (Namen) von Klassen, Funktionen, Attributen, Variablen und Konstanten müssen sprechend sein.
	Alle Bezeichner (Namen) werden in Englisch geschrieben.
	Schleifenzähler innerhalb einer kurzen Iteration (max. 5 Zeilen) dürfen auch kürzere Variablennamen wie *i*, *j* verwenden.

Beispiel

```
total_marks = 0
i = 0
while i < 3:
    mark = input('Note >')
    total_marks = total_marks + int(mark)
    i = i + 1
average = total_marks / 3
print('Durchschnitt: ' + str(average))
```

Schreibweise

Durch eine einheitliche Schreibweise verbessern Sie die Lesbarkeit Ihres Sourcecodes.

	Vermeide Sonderzeichen, Leerzeichen und Umlaute in allen Bezeichnern, sowie in Datei- und Ordnernamen.

Variablen, Konstanten und Funktionen

	Die Bezeichner von Funktionen und Variablen bestehen aus Kleinbuchstaben.
	Zusammengesetzte Bezeichner werden in snake_case geschrieben (z. B. *get_fullname*)
	Konstanten werden nur mit Grossbuchstaben geschrieben (z.B. *PI*).

Beispiel

```
def main():
    PI=3.14159

    radius=float(input('Radius >'))
    circumference=2*radius*PI
    circular_area = radius*radius*PI
```

M319-B3F, M319-G1G, M319-G1E