LU02e - Rechnen mit Zahlen

Mit Variablen vom Typ `int` oder `float` kannst du Berechnungen durchführen. Dabei muss das Resultat immer in einer Variable gespeichert werden.

Einführung

Die mathematischen Grundoperationen sind bekannt und einfach: Addition +, Subtraktion `-`, Multiplikation * und Division `/`. Auch die Rangfolge ist bekannt: Die Operationen werden von links nach rechts durchgeführt, wobei die Klammern berücksichtigt werden. Ausdrücke mit * und `/` werden vor denen mit + und `-` berechnet, wie es in der Grundschulmathematik üblich ist.



```
first = 2
print(first) # prints 2
second = 4
print(second) # prints 4

sum = first + second # The sum of the values
of the variables first and second is assigned
to the variable sum
print(sum) # prints 6
```

Vorrang und Klammern

Du kannst die Reihenfolge der Operationen durch die Verwendung von Klammern beeinflussen. Operationen innerhalb von Klammern werden vor den Operationen ausserhalb von Klammern ausgeführt.



```
calculation_with_parentheses = (1 + 1) + 3 *
(2 + 5)
print(calculation_with_parentheses) # prints
23

calculation_without_parentheses = 1 + 1 + 3 *
2 + 5
print(calculation_without_parentheses) #
prints 13
```

Das obige Beispiel kann auch in Schritte unterteilt werden.

```
calculation_with_parentheses = (1 + 1)
print(calculation_with_parentheses) # prints
2
calculation_with_parentheses =
calculation_with_parentheses + 3 * (2 + 5)
print(calculation_with_parentheses) # prints
23

calculation_without_parentheses = 1 + 1
calculation_without_parentheses =
calculation_without_parentheses + 3 * 2
calculation_without_parentheses =
calculation_without_parentheses + 5
print(calculation_without_parentheses) #
prints 13
```

Ausdruck und Anweisung Ein Ausdruck ist eine Kombination von Werten, die durch eine Berechnung oder Auswertung in einen anderen Wert umgewandelt wird.

Die Auswertung eines Ausdrucks wird immer durchgeführt, bevor sein Wert einer Variablen zugewiesen wird. So wird die Berechnung „1 + 1 + 3 * 2 + 5“ in dem obigen Beispielwert = 1 + 1 + 3 * 2 + 5 durchgeführt, bevor das Ergebnis der Variablen zugewiesen wird.

Ein Ausdruck wird dort ausgewertet, wo er im Quellcode des Programms vorkommt. So kann die Auswertung innerhalb einer Druckanweisung erfolgen, wenn der Ausdruck bei der Berechnung des Wertes des Parameters der Druckanweisung verwendet wird.

```
first = 2
second = 4

print(first + second)  # prints 6
print(2 + second - first - second)  # prints
0
```

Ein Ausdruck ändert den in einer Variablen gespeicherten Wert nicht, es sei denn, das Ergebnis des Ausdrucks wird einer Variablen wieder zugewiesen.

```
first = 2
second = 4

first = first + second # first = 6
```

Berechnen und Drucken

Der Befehl `print` gibt den Wert einer Variablen aus. Das zu druckende String-Literal, das durch Anführungszeichen gekennzeichnet ist, kann mit der Operation + mit anderen Inhalten ergänzt werden.



```
length = 42
print('Length ' + str(length))
```

```
Length 42
```

Wenn einer der Operanden der Operation + eine Zeichenkette ist, muss der andere Operand während der Programmausführung ebenfalls in einen String umgewandelt werden. Im folgenden Beispiel wird die Ganzzahl 2 in die Zeichenkette „2“ umgewandelt und eine Zeichenkette daran angehängt.



```
print('here is an integer --> ' + str(2))
print(str(2) + ' <-- here is an integer')
```

```
here is an integer –> 2
2 <– here is an integer
```

Auch hier gilt die zuvor eingeführte Rangfolge:



```
print('Four: ' + str(2 + 2))
print('But! Twenty-two: ' + str(2) + str(2))
```

```
Four: 4
But! Twenty-two: 22
```

Mit diesem Wissen können wir einen Ausdruck erstellen, der aus einem Text und einer Variablen besteht und im Zusammenhang mit dem Druck ausgewertet wird:



```
x = 10

print('The value of the variable x is: ' +
str(x))

y = 5
z = 6

print('y is ' + str(y) + ' and z is ' +
str(z))
```

```
The value of the variable x is: 10
y is 5 and z is 6
```

f-Strings: Einfacher Zahlen in Text einbinden

Das Konkatenieren (Zusammensetzen) von mehreren Variablen und Konstanten wird schnell unübersichtlich. Einfacher wird es mit Hilfe der **F-Strings**.

F-Strings werden auch „formatierte Stringliterale“ genannt und werden mit einem `f` am Anfang gekennzeichnet. Im Gegensatz zu einfachen Textkonstanten dürfen **F-Strings** Ausdrücke zwischen geschweiften Klammern enthalten.

Hier sind einige der Möglichkeiten, wie f-Strings Ihr Leben einfacher machen können:

Ohne F-Strings	**Mit F-Strings**
`x = 10` `print('The value of the variable x is: ' + str(x)')`	`x = 10` `print(f'The value of the variable x is: {x}')`

`y = 5`
`z = 6`
`print('y is ' + str(y) + ' and z is ' + str(z))`
	`y = 5`
	`z = 6`
	`print(f'y is {y} and z is {z})`

`bags = 3`
`apples_in_bag = 12`
 	
`print('There is a total of ' + int(bags * apples_in_bag) + ' apples')`
	`bags = 3`
	`apples_in_bag = 12`
	
	`print(f'There is a total of {bags * apples_in_bag} apples')`

Wie du siehst, ersparen dir die **F-Strings** das Konkatenieren der einzelnen Teile und das Konvertieren der Zahlenvariablen in Strings.

Auf Youtube findest du ein gutes Video dazu. Relevant ist der Anfang bis Minute 5:30

Missverständnisse in Bezug auf den Wert einer Variablen

Wenn ein Computer Programmcode ausführt, tut er dies einen Befehl nach dem anderen, wobei er immer genau so vorgeht, wie es der Code vorgibt. Wenn einer Variablen ein Wert zugewiesen wird, läuft immer dieselbe Kette von Ereignissen ab: Der Wert auf der rechten Seite des Gleichheitszeichens wird kopiert und der Variablen auf der linken Seite zugewiesen (d. h. an die von dieser Variable angegebene Stelle kopiert).

Für einen Programmierer ist es wichtig zu verstehen, dass die Zuweisung eines Wertes an eine Variable immer dasselbe bewirkt.

Im Folgenden werden drei häufige Missverständnisse im Zusammenhang mit der Zuweisung eines Wertes an eine Variable erläutert:

- Betrachtung der Wertzuweisung als Übertragung statt als Kopiervorgang: Nach der Ausführung von `first = second` wird oft angenommen, dass der Wert der Variablen second in den Wert der Variablen `first` verschoben wurde und dass die Variable `second` keinen Wert mehr enthält oder dass ihr Wert z. B. 0 ist. Dies ist falsch, denn die Ausführung von `first = second` bedeutet lediglich, dass der Wert an der durch `second` angegebenen Stelle an die durch die Variable `first` angegebene Stelle kopiert wird. Folglich wird die Variable `second` nicht verändert.
- Betrachtung der Wertzuweisung als Schaffung einer Abhängigkeit statt als Kopiervorgang: Nach der Ausführung von `first = second` wird oft angenommen, dass sich jede Änderung des Wertes der Variablen `second` automatisch auch auf den Wert der Variablen `first` auswirkt. Das ist falsch; die Zuweisung - also das Kopieren - ist ein einmaliger Vorgang. Sie findet nur statt, wenn der Programmcode `first = second` ausgeführt wird.
- Das dritte Missverständnis betrifft die Richtung des Kopierens: Häufig wird angenommen, dass bei der Ausführung von `first = second` der Wert der Variablen `first` als Wert der Variablen `second` gesetzt wird. Die Verwirrung manifestiert sich auch in Situationen, in denen der Programmierer versehentlich z. B. `42 = Wert` schreibt - glücklicherweise bieten IDEs auch in diesem Punkt Unterstützung.

Der beste Weg, den Ablauf eines Programms zu verstehen, ist vielleicht die Verwendung von Stift und Papier. Während du das Programm liest, schreibst du die Namen aller neuen Variablen auf und notierst, wie sich die Werte der Variablen im Code Zeile für Zeile ändern. Lass mich die Ausführung anhand des folgenden Programmcodes demonstrieren:

```
Zeile 1: first = (1 + 1)
Zeile 2: second = first + 3 * (2 + 5)
Zeile 3:
Zeile 4: first = 5
Zeile 5:
Zeile 6: third = first + second
Zeile 7: print(first)
Zeile 8: print(second)
Zeile 9: print(third)
```

```
Zeile 1: zuerst eine Variable einrichten
```

```
Zeile 1: Kopiere das Ergebnis der Berechnung
1 + 1 als Wert der Variablen first
Zeile 1: Der Wert der ersten Variablen ist 2
Zeile 2: Erzeuge die Variable second
Zeile 2: rechne 2 + 5, 2 + 5 -> 7
Zeile 2: berechne 3 * 7, 3 * 7 -> 21
Zeile 2: berechne zuerst + 21
Zeile 2: Kopiere den Wert der Variablen first
in die Berechnung, ihr Wert ist 2
Zeile 2: berechne 2 + 21, 2 + 21 -> 23
Zeile 2: Kopiere 23 in den Wert der Variablen
second
Zeile 2: der Wert der zweiten Variablen ist
23
Zeile 3: (leer, nichts tun)
Zeile 4: kopiere 5 in den Wert der Variablen
first
Zeile 4: Der Wert der Variablen first ist 5
Zeile 5: (leer, nichts tun)
Zeile 6: erzeuge Variable dritte
Zeile 6: berechne erste + zweite
Zeile 6: kopiere den Wert der Variablen first
in die Berechnung, ihr Wert ist 5
Zeile 6: berechne 5 + zweite
Zeile 6: Kopiere den Wert der Variablen
second in die Berechnung, ihr Wert ist 23
Zeile 6: berechne 5 + 23 -> 28
Zeile 6: kopiere 28 in den Wert der Variablen
third
Zeile 6: der Wert der Variable dritte ist 28
Zeile 7: Drucke die Variable first
Zeile 7: kopiere den Wert der Variablen first
für die Druckoperation, ihr Wert ist 5
Zeile 7: drucke den Wert 5
Zeile 8: drucke die Variable zweite
Zeile 8: Kopiere den Wert der Variablen
second für die Druckoperation, ihr Wert ist
23
Zeile 8: Drucke den Wert 23
Zeile 9: Drucke die dritte Variable
Zeile 9: Kopiere den Wert der dritten
Variablen für den Druckvorgang, ihr Wert ist
28
Zeile 9: wir drucken den Wert 28
```

M319-LU01, M319-B1F, M319-C1G, M319-C1F

Kevin Maurizi, Marcel Suter

Diese Theorieseite ist eine übersetzte und bearbeitete Theorieseite von Scott Morgan, verwendet unter CC BY NC SA.

From:
https://wiki.bzz.ch/ - **BZZ - Modulwiki**

Permanent link:
https://wiki.bzz.ch/de/modul/m319/learningunits/lu02/rechnen?rev=1750657503

Last update: **2025/06/23 07:45**