

Grundlagen

Operator	Bedeutung
+	Addition
-	Subtraktion
*	Multiplikation
/	Division
//	ganzzahlige Division
%	Modulo (Rest der Division)
**	Potenzierung

```
print(3 + 2)
print(3 - 2)
print(3 * 2)
print(3 / 2)
print(3 // 2)
print(3 % 2)
print(3**2)
```

```
5
1
6
1.5
1
1
9
```

Erzeugung von Leerzeilen: Die Zeichenkette `\n` in einem String erzeugt eine neue Zeile.

```
print("Text\nText")
print("\n")
print("Text")
```

Text
Text

Text

Reihenfolge der Operatoren: - Klammern haben Vorrang - Multiplikation vor Addition - Potenzierung vor Multiplikation vor Addition

```
print(3 + 2 * 5) # Multiplikation vor Addition  
print((3 + 2) * 5) # Klammern haben Vorrang  
print(3 + 2 * 5**2) # Potenzierung vor Multiplikation vor Addition
```

13
25
53

Datentypen

Typname	Art der Daten
integer	ganze Zahlen
float	Gleitkommazahlen
String	Zeichenketten
Boolean	Wahrheitswert (True oder False)
NoneType	Wert None

```
print(type(42)) # Integer (Ganzzahl)  
print(type(3.14)) # Float (Gleitkommazahl)  
print(type("Hallo")) # String (Text)  
print(type(True), type(False)) # Boolean (Wahrheitswert)  
print(type(None)) # NoneType (kein Wert)
```

```
<class 'int'>  
<class 'float'>  
<class 'str'>  
<class 'bool'> <class 'bool'>  
<class 'NoneType'>
```

Umgang mit Strings

Operation	Code
Konkatenation (Verkettung)	"Alice" + "Bob"
Wiederholung	"Alice" * 3
Neue Zeile	"\n"
Umwandeln in Großbuchstaben	"...".upper()
Umwandeln in Kleinbuchstaben	"...".lower()
Leerzeichen entfernen am Rand	"...".strip()

```
print("Alice" + "Bob")
print("Alice " * 3)
print("Hallo\nWelt")
print("mach mich groß".upper())
print("MACH MICH KLEIN".lower())
print("  zuschneiden bitte ".strip())
```

```
AliceBob
Alice Alice Alice
Hallo
Welt
MACH MICH GROSS
mach mich klein
zuschneiden bitte
```

Mit f-Strings kann man Python-Ausdrücke in Strings einbetten. Alle Teile, die zwischen {...} eingeschlossen sind, werden als Python-Ausdrücke ausgewertet und das Ergebnis der Auswertung an dieser Stelle in den String eingefügt.

```
print(f"2 + 2 = {2 + 2}")
```

```
2 + 2 = 4
```

Mit `len(...)` kann man die Länge eines Strings ermitteln.

```
s = "Guten Morgen!"
print(len(s))
```

Variablen verwenden

Eine Variable ist wie eine “Schachtel” im Computerspeicher, wo man Daten ablegen kann. Weißt man einem Variablennamen mit = erstmals einen Wert zu, wird die entsprechende Variable im Computerspeicher angelegt.

Eine Variable mit Namen `alter` wird angelegt und mit dem Wert 15 belegt:

```
alter = 15
```

Verwendet man im Folgenden den Namen `alter`, wird auf den gespeicherten Wert zugegriffen:

```
print(alter)
print(alter + 5)
```

```
15
20
```

Variablen haben einen Typ, der ihrem Inhalt entspricht:

```
print(type(alter))
```

```
<class 'int'>
```

Der Variablentyp kann sich ändern, wenn die Variable mit einem neuen Wert belegt wird:

```
alter = 2.5
print(type(alter))

alter = "fünfzehn"
print(type(alter))
```

```
<class 'float'>
```

```
<class 'str'>
```

Für Variablennamen gelten folgende Regeln: - Dürfen nur Buchstaben, Zahlen und Unterstriche (__) enthalten - Dürfen nicht mit einer Zahl beginnen - Dürfen nur aus einem Wort bestehen (keine Leerzeichen)

erlaubt z.B.: `mein_name`, `alter2`, `_preis`

nicht erlaubt z.B.: `2alter`, `mein name`, `mein-name`

Input-Befehl

Der Befehl `input("...")` gibt den angegebenen String aus und wartet anschließend auf eine Eingabe des Benutzers. Die Eingabe wird als String-Wert dem Programm zurückgegeben und kann z.B. in einer Variable abgelegt werden.

Hier wird die Eingabe des Benutzers in der Variable `name` abgelegt:

```
name = input("Wie heißt du?")
```

```
name = input("Wie heißt du? ") # Eingabe: Dornröschen
print("Hallo " + name + "!")
```

Hallo Dornröschen!

Achtung: `input()` gibt immer einen String zurück.

```
eingabe = input("Gib eine Zahl ein: ") # Eingabe: 42
print(type(eingabe))
```

```
<class 'str'>
```

Der eine Zahl enthaltende String kann aber in einen Zahltyp (`int` oder `float`) umgewandelt werden mit `int(...)` oder `float(...)`.

```
eingabe = input("Gib eine Zahl ein: ") # Eingabe: 42
zahl = int(eingabe)
print(type(zahl))
print(zahl + 5)
```

```
<class 'int'>
```

```
47
```

Analog kann man auch in `float` oder `String` umwandeln:

```
var = float("42.5")
print(type(var))
zahl = 17
print(type(str(zahl)))
```

```
<class 'float'>
```

```
<class 'str'>
```

Formatierte Ausgabe mit f-Strings

Bei der Ausgabe mit f-Strings kann man Formatierungsvorgaben machen.

```
name1 = "Alice"
name2 = "Bob"
name3 = "Charlie"
alter1 = 15
alter2 = 16
alter3 = 18
groesse1 = 1.70
groesse2 = 1.85
groesse3 = 1.7382

print("Name      Alter   Größe")
print(f"{name1:8} {alter1:5d} {groesse1:7.2f}")
print(f"{name2:8} {alter2:5d} {groesse2:7.2f}")
print(f"{name3:8} {alter3:5d} {groesse3:7.2f}")
```

Name	Alter	Größe
Alice	15	1.70
Bob	16	1.85
Charlie	18	1.74

Möglichkeiten der Formatierung:

Format	Bedeutung
:8	mindestens 8 Zeichen breit, rechtsbündig (Standard)
:5d	mindestens 5 Zeichen breit, rechtsbündig, Integer (d = decimal)
:7.2f	mindestens 7 Zeichen breit, rechtsbündig, Fließkommazahl (f = float) mit 2 Nachkommastellen

Es gibt viele weitere Formatierungsmöglichkeiten. Eine vollständige Übersicht findet man in der Python-Dokumentation: <https://docs.python.org/3/library/string.html#formatspec>