Merkblatt if - elif - else

[ ]:

1 if - Anweisung

Die `if`-Anweisung wird benutzt, um Entscheidungen zu treffen. Sie besteht aus den Bestandteilen `if…elif…else`. Die letzte beiden Bestandteile müsse nicht vorkommen, `elif` kann mehrfach vorkommen. Zwingend erforderlich ist der Beginn mit `if` gefolgt von einem Ausdruck, der einen Wahrheitswert (`boolean`) liefert:

[1]:
```
a = 2
```

[2]:
```
if a > 0:
    print("Die Zahl ist positiv.")
```

```
Die Zahl ist positiv.
```

Beachte unbedingt den Doppelpunkt : nach dem Wahrheitswert/ Vergleich!

Mit `elif` kann eine weitere Entscheidung getroffen werden, falls die erste nicht zutraf.

[3]:
```
a = 0
if a > 0:
    print("Die Zahl ist positiv.")
elif a == 0:
    print("Die Zahl ist Null.")
```

```
Die Zahl ist Null.
```

Mit `else` kann angegeben werden, was passiert, wenn keine der Bedingungen zuvor zutraf.

[4]:
```
a = -3
if a > 0:
    print("Die Zahl ist positiv.")
elif a == 0:
    print("Die Zahl ist Null.")
else:
    print("Die Zahl ist negativ.")
```

```
Die Zahl ist negativ.
```

Beachte unbedingt den Doppelpunkt :, der auch nach `else` erforderlich ist!

1.1 Vergleiche

Die if-Anweisung erfordert einen Wahrheitswert, um zu entscheiden, wie das Programm fortgesetzt wird. Dieser Wahrheitswert kann einfach eine Variable vom Typ boolean sein. In aller Regel wird jedoch ein Vergleich direkt angegeben, der sich zu einem Wahrheitswert auswertet. Nachfolgend die wichtigsten Vergleichsoperatoren:

Operator	umgangssprachlich	Der Vergleich ist wahr (True), wenn …
==	ist gleich	… beide Werte gleich sind.
>	ist größer als	… der linke Wert größer als der rechte ist.
<	ist kleiner als	… der linke Wert kleiner als der rechte ist.
>=	ist größer als oder gleich	… der linke Wert größer oder gleich dem rechten ist.
<=	ist kleiner als oder gleich	… der linke Werte kleiner oder gleich dem rechten ist.

Oft ist es erforderlich. mehrere Vergleiche zu verküpfen, zum Beispiel so:

1.1.1 or

[5]:
```
a = 0
if a<0 or a>0:
    print("Die Zahl ist nicht Null.")
```

Der Operator `or` liefert wahr, wenn mindestens einer der beiden Vergleiche wahr ist.

[6]:
```
1>0 or 1<0
```

[6]:
```
True
```

Hier eine Übersicht über die Verknüpfung, die der logische Operator `or` vornimmt:

a	b	a or b
True	True	True
True	False	True
False	True	True
False	False	False

1.1.2 and

Der Operator `and` liefert nur dann wahr, wenn die Ausdrücke auf beiden Seiten wahr sind.

[7]:
```
1>0 and 1<0
```

[7]: `False`

Hier eine Übersicht über die Verknüpfung, die der logische Operator `and` vornimmt:

a	b	a and b
True	True	True
True	False	False
Flase	True	False
False	False	False

1.1.3 not

Weiterhin gibt es Operator `not`. Er negiert einen einzelnen Wahrheitswert.

[8]:
```
not True
```

[8]: `False`

a	not a
True	False
False	True

Auch Klammern `( )` können benutzt werden, um die Reihenfolge der Verarbeitung von Vergleichen zu bestimmen.