Bedingte Anweisungen

if, elif, else

Der Datentyp Boolean

Der Datentyp Boolean ist der Wahrheitswert, er kennt nur zwei Zustände:

True

oder

False

Ein beliebiger Wert kann mit der built-in Funktion bool zu einem Boolean umgeformt werden:

value = bool(3)

Was ist unwahr?

Folgende Werte, die an die Funktion bool() übergeben werden, werden von Python als falsch interpretiert:

0 => Ganzahl 0

0.0 => Float 0.0

"" => leerer String

[] => leere Liste

() => leerer Tupel

{} => leeres Dictionary

None

False => der boolsche Wert False

Das if-Statement

Die if-Anweisung wird benutzt, um den Programmablauf in einem Python-Programm zu steuern.

Damit wird es möglich zur Laufzeit zu entscheiden, ob bestimmte Programmteile ausgeführt werden sollen oder nicht.

Die einfachste Form einer if-Anweisung in einem Python-Programm sieht folgendermaßen aus

```
if bedingung:
        anweisung
        #...weitere Anweisungen, falls nötig
```

Der eingerückte Block wird nur dann ausgeführt, wenn die Bedingung zutrifft.

D.h., wenn sie logisch wahr ist.

Operatoren, die uns einen Wahrheitswert liefern

Folgende Operatoren liefern uns einen boolschen Wert (true oder false):

1) Arithmetische Vergleichsoperatoren, zb. >=
2) Logische Operatoren (Boolsche Operatoren) z.b. and
3) Identitätsoperatoren, zb. is
4) Membership-Operator in

Arithmetische Vergleichsoperatoren

Vergleiche mit den arithmetischen Vergleichsoperatoren wie == oder != erzeugen ebenfalls ein Boolean Value

a == a: True

a != a: False

a != b: True

Arithmetische Vergleichsoperatoren

< ist kleiner

<= ist kleiner gleich

> ist größer

>= ist größer gleich

!= ist ungleich

== ist gleich

```
if "Hamburg" == "Hamburg":
    print("der String ist gleich")
```

```
4 != 3
```

liefert True

Boolsche Operatoren

In Python stehen uns folgende logische Operatoren zur Verfügung:

Boolsches oder (or)

Boolsches UND (and)

Boolsches Nicht (not) => dient zur Negation

```
if True and False:
    print("Programmer's nightmare")
```

Boolsche Operatoren

```
if green or blue:
        print("green or blue")

if green and blue:
        print("green and blue")

if not not True:
        print(„Nicht nicht wahr ist wahr")
```

Operatoren-Rangfolge:

not

and

or

Logisch nicht wahr

Erinnerung: folgende Werte gelten in Python u.a. als falsch:

numerische Null-Werte(0, 0.0)
der boolschen Wert False
leere Zeichenketten ""
leere Listen, leere Tupel, leeres set [], (), set()
leere Dictionaries {}
sowie der spezielle Wert None

Vorsicht: negative Zahlen gelten als wahr!

Membership-Operator

Das Schlüsselwort in trifft eine Aussage darüber, ob ein Wert in einem anderen sequentiellen Wert (String, Liste, Tupel) vorkommt

```
a = "San Francisco"
if "San" in a:
        print("San ist in San Francisco")

numbers = [2, 3, 6, 9]
If 4 in numbers:
    print("die 4 ist nicht enthalten")
```

Identitätsoperator

Die Identitätsoperatoren is und is not treffen eine Aussage darüber, ob ein Element auf den gleichen Speicherbereich referenziert wie ein anderes und somit die gleiche Identität hat.

a = 2

b = 3

a is b

False

a is not b

True

Der Speicherbereich kann mit der Funktion id geprüft werden: id(a).

Der Else-Zweig

```
if bedingung:
        anweisung
else:
        andere anweisung
```

Beispiel

```
if user_name == "Georg":
        do_something()
else:
        do_something_else()
```

Elif

Wird eine weitere Bedingung benötigt, nutzen wir den elif-Zweig. In anderen Sprachen auch als else if bekannt.

```
if user_name == "Georg":
        do_something()
elif user_name == "Paul":
         do_something_special()
else:
        do_something_else()
```