Die `if`-Anweisung (`if`-Block)

Python zeichnet sich dadurch aus, **dass NUR die Einrückung einen Programmblock kennzeichnet**. Einrücken dient bei Python nicht bloß der Übersichtlichkeit sondern ist auch Teil der Syntax! Alle Zeilen derselben Ebene müssen gleich eingerückt sein! Direkt vor dem eingerückten Block steht (fast immer?) ein **Doppelpunkt** :

```
if Bedingung:
     abhängige Anweisung;
     abhängige Anweisung;
     ...
keine abhängige Anweisung;
```

```
x = 3
if x > 4:
     x = 7
     print("x wurde auf 7 gesetzt")
```

```
x = 3
if x == 3:          # == ist Abfrage auf Gleichheit
     x = 7
print("x wurde auf 7 gesetzt")  # wird immer ausgeführt??
```

Ist die Bedingung nicht erfüllt, wird der ganze abhängige Block übersprungen und es geht mit der ersten darauffolgenden nichtabhängigen Anweisung weiter.

Die `if-else`-Anweisung (`if-else`-Block)

Hier folgt auf den `if`-Block **direkt** ein `else`-Block. Ist die Bedingung `True`, wird nur der if-Block ausgeführt, andernfalls nur der `else`-Block. Auch nach dem `else` steht der Doppelpunkt:

```
if Bedingung:
     abhängige Anweisung;
     abhängige Anweisung;
     ...
else:
     abhängige Anweisung;
     abhängige Anweisung;
```

```
     ...
keine abhängige Anweisung;

x = 3
if x > 4:
     x = 7
     print("x wurde auf 7 gesetzt")
else:
     print("x wurde nicht geändert")
```

Die `if-elif-else`-Anweisung (`if-elif-else`-Block)

Hier folgt auf den `if`-Block **direkt** ein oder mehrer `elif`-Blöcke. Nach dem letzten kann (muss aber nicht) ein `else`-Block folgen.

```
if Bedingung:
     abhängige Anweisung;
     abhängige Anweisung;
     ...
elif andere_Bedingung:
     abhängige Anweisung;
     abhängige Anweisung;
     ...
elif noch_eine_andere_Bedingung:
     abhängige Anweisung;
     abhängige Anweisung;
     ...
else:
     abhängige Anweisung;
     abhängige Anweisung;
     ...
keine abhängige Anweisung;

x = 3
if x > 4:
     x = 7
     print("x wurde auf 7 gesetzt")
elif x > 2:
     x = x+1
     print("x wurde um 1 erhöht")
else:
     print("x wurde nicht geändert")
```

Bei dieser Konstruktion werden der Reihe nach alle Bedingungen getestet. Trifft eine der Bedingungen zu, wird deren abhängiger Block ausgeführt und der ganze Rest übersprungen. Ist keine der Bedingungen wahr, so wird nur der `else`-Block ausgeführt, falls dieser vorhanden ist. Es wird also maximal ein einziger der vorhandenen Blöcke ausgeführt.

Man kann durchaus mehrere `if`-Konstruktionen verschachteln, muss dann entsprechend tiefer einrücken:

```
if x > 4:
     if x > 7:
          x = 9
     elif x > 10:
          x = 101
     print("x wurde auf", x, "gesetzt")
elif x > 2:
     x = x+1
     print("x wurde um 1 erhöht")
else:
     print("x wurde nicht geändert")
```

Welche Anweisungen werden hier für `x = 1, 3, 7, 8, 30` ausgeführt?

logische Operatoren

Mit den logischen Operatoren

`==`	(ist gleich),
`!=`	(ist nicht gleich),
`<`	(ist kleiner als),
`<=`	(ist kleiner oder gleich als),
`>`	(ist größer als),
`>=`	(ist größer oder gleich als)

können Sie einfache Bedingungen programmieren.

Mittels `not, or, and` können aus einfachen Bedingungen komplexere Bedingungen gebaut werden:

```
if  x > 0 and not y < 5:
     ...
```