

# Logik

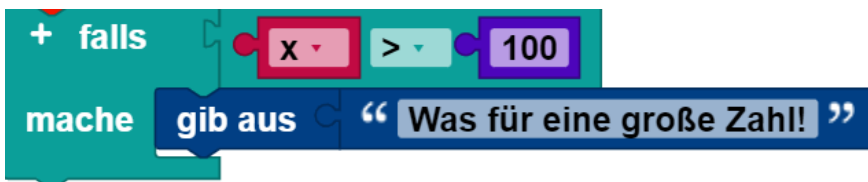
## Bedingte Anweisungen

Bedingte Anweisungen sind zentral für die Programmierung. Sie machen es möglich, Fallunterscheidungen zu formulieren wie:

- Wenn es einen Weg nach links gibt, biege links ab.
- Wenn Punktzahl = 100, drucke "Gut gemacht!".

### "wenn"-Block

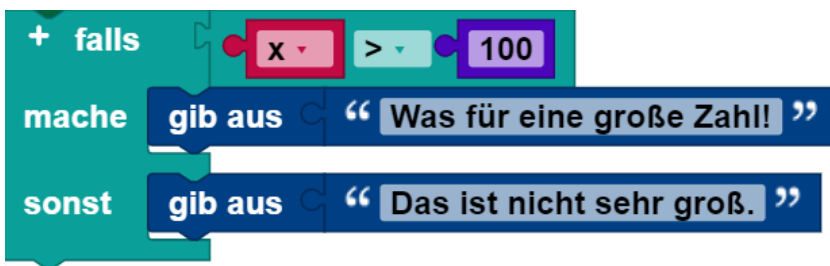
Die einfachste Bedingung ist ein "wenn"-Block:



Wenn dieser ausgeführt wird, wird der Wert der Variable x mit 100 verglichen. Wenn er größer ist, wird "Was für eine große Zahl!" ausgegeben. Andernfalls passiert nichts.

### "wenn-sonst"-Block

Es ist auch möglich, anzugeben, dass etwas passieren soll, wenn die Bedingung nicht wahr ist, wie in diesem Beispiel:

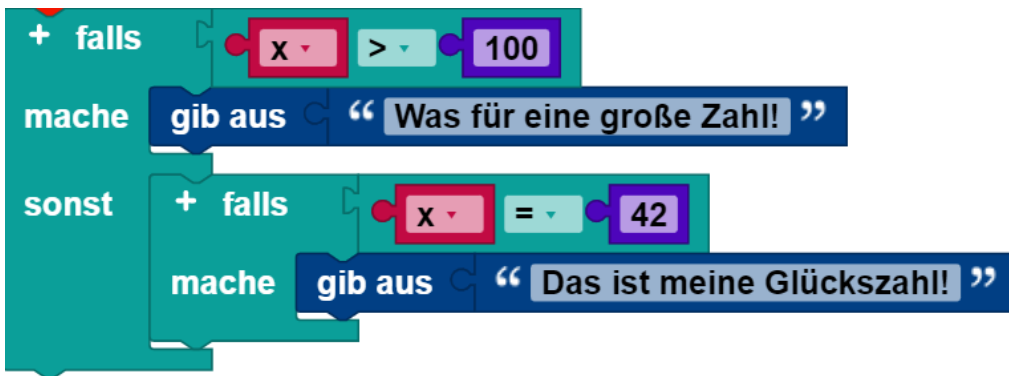


Wie beim vorherigen Block wird "Was für eine große Zahl!" ausgegeben, wenn  $x > 100$  ist. Andernfalls wird "Das ist nicht sehr groß." angegeben.

Ein "wenn"-Block kann einen "sonst"-Abschnitt haben, aber nicht mehr als einen.

### "wenn-sonst-wenn"-Block

Es ist auch möglich, mehrere Bedingungen mit einem einzigen "wenn"-Block zu testen, indem "sonst-wenn"-Klauseln hinzugefügt werden:



Der Block prüft zuerst, ob  $x > 100$  ist, und gibt "Was für eine große Zahl!" aus, wenn das der Fall ist. Ist dies nicht der Fall, prüft er weiter, ob  $x = 42$  ist. Wenn ja, gibt er "Das ist meine Glückszahl!" aus. Andernfalls passiert nichts.

Ein "wenn"-Block kann eine beliebige Anzahl von "sonst-wenn"-Abschnitten haben. Die Bedingungen werden von oben nach unten ausgewertet, bis eine erfüllt ist oder bis keine Bedingung mehr übrig sind.

## "wenn-sonst-wenn-sonst"-Block

"wenn"-Blöcke können sowohl "sonst-wenn" als auch "sonst"-Abschnitte haben:

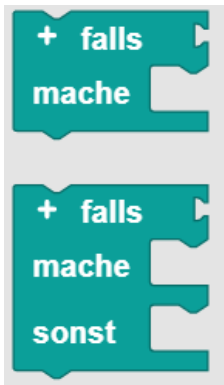


Der "sonst"-Abschnitt garantiert, dass eine Aktion ausgeführt wird, auch wenn keine der vorherigen Bedingungen wahr ist.

Ein "sonst"-Abschnitt kann nach einer beliebigen Anzahl von "sonst-wenn"-Abschnitten auftreten, einschließlich Null, dann erhält man einen ganz normalen "wenn-sonst"-Block.

## Blockmodifikation

In der Werkzeugleiste erscheint nur der einfache "wenn"-Block und der "wenn-sonst"-Block:



Um "sonst-wenn" - und "sonst"-Klauseln hinzuzufügen, kann man auf das (+) Symbol klicken. Mit (-) Symbol lassen sich "sonst-wenn" -Klauseln wieder entfernen:



Beachte, dass die Formen der Blöcke das Hinzufügen einer beliebigen Anzahl von "sonst-wenn"-Unterblöcken erlauben, aber nur bis zu einem "wenn"-Block.

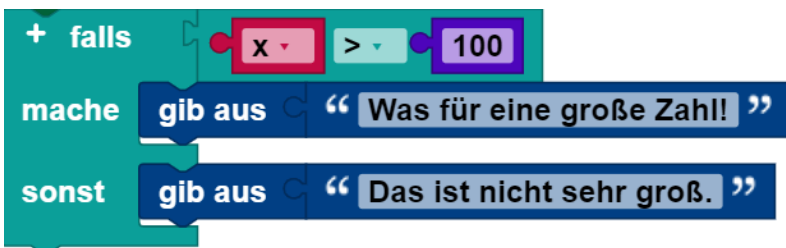
## Boolesche Logik

Boolesche Logik bildet die Grundlage für die Implementierung bedingter Anweisungen. Sie basiert auf einem einfachen mathematischen System mit zwei Zuständen:

- wahr
- falsch

Logikblöcke in ROBO Pro Coding sind in der Regel dafür da, Bedingungen und Schleifen zu kontrollieren.

Hier ein Beispiel:



Wenn der Wert der Variable x größer als 100 ist, gilt die Bedingung als wahr und "Was für eine große Zahl!" wird ausgegeben. Ist der Wert nicht größer als 100, ist die Bedingung falsch und es wird "Das ist nicht sehr groß." ausgegeben. Boolesche Werte können in Variablen gespeichert und an Funktionen übergeben werden, ähnlich wie Zahlen, Texte und Listenwerte.

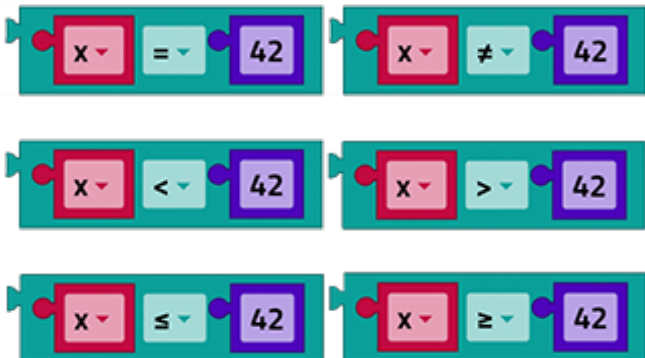
Wenn ein Block einen Booleschen Wert als Eingabe erwartet und keine Eingabe erfolgt, wird dies als falsch interpretiert. Es ist technisch möglich, jedoch nicht empfohlen, nicht boolesche Werte in Bedingungsangaben zu verwenden. Diese Methode wird nicht empfohlen, und ihr Verhalten kann sich in zukünftigen Versionen von ROBO Pro Coding ändern.

## Werte

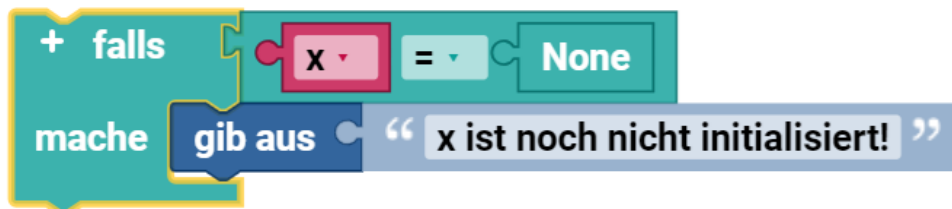
Ein einzelner Block mit einer Dropdown-Liste, die entweder wahr oder falsch angibt, kann verwendet werden, um einen Booleschen Wert abzurufen:

## Vergleichsoperatoren

Es gibt sechs Vergleichsoperatoren. Jedem werden zwei Eingaben (normalerweise zwei Zahlen) übergeben und der Vergleichsoperator gibt wahr oder falsch zurück, je nachdem, wie die Eingaben miteinander verglichen werden.

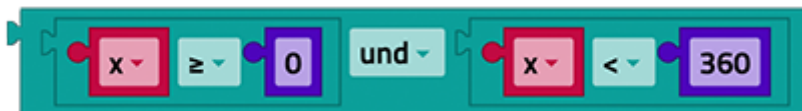


Die sechs Operatoren sind: gleich, nicht gleich, kleiner als, größer als, kleiner als oder gleich, größer als oder gleich.

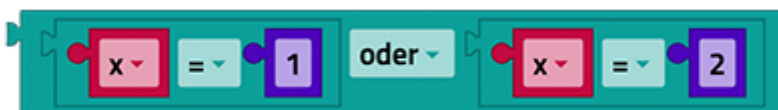


## Logische Operatoren

Der "und"-Block gibt dann und nur dann wahr zurück, wenn seine beiden Eingangswerte wahr sind.



Der "oder"-Block gibt wahr zurück, wenn mindestens einer seiner beiden Eingangswerte wahr ist.



## "nicht"-Block

Der "nicht"-Block wandelt eine boolesche Eingabe in ihr Gegenteil um. Zum Beispiel ist das Ergebnis von:



falsch.

Wenn keine Eingabe erfolgt, wird der Wert wahr angenommen, sodass der folgende Block den Wert falsch erzeugt:



Es wird jedoch nicht empfohlen, eine Eingabe leer zu lassen.

## "dreier Operator"-Block

Der dreier Operator verhält sich wie ein Miniatur-"wenn-sonst"-Block. Er nimmt drei Eingangswerte entgegen, der erste Eingangswert ist die zu testende boolesche Bedingung, der zweite Eingangswert ist der Wert, der zurückgegeben werden soll, wenn der Test wahr ergibt, der dritte Eingangswert ist der Wert, der zurückgegeben werden soll, wenn der Test falsch ergibt. Im folgenden Beispiel wird die Variable Farbe auf Rot gesetzt, wenn die Variable x kleiner als 10 ist, andernfalls wird die Variable Farbe auf Grün gesetzt.



Ein dreier Block kann immer durch einen "wenn-sonst"-Block ersetzt werden. Die folgenden zwei Beispiele sind genau gleich

