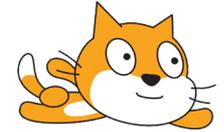


Scratch 3.0 Vertiefungskurs

Kurstag 1





Kurstag 1 (Mittwoch, 30.1.19, 14.00 - 17.00 Uhr)

- Scratch 3.0. Was ist neu?
- Programmierkonzepte
- Umsetzung von Projektideen

Kurstag 2 (Mittwoch, 13.3.19, 14.00 – 17.00 Uhr)

- Vorgehensweise bei der Fehlersuche
- Verbindung mit der physischen Welt (micro:bit, LEGO, MakeyMakey)
- Didaktische Hinweise

Optional aber empfohlen: Für die Wochen zwischen den Kurstagen kann an einem individuellen Projekt gearbeitet werden.



Wichtigste Neuerungen

- Benutzt HTML5+Javascript und nicht mehr Flash
- Läuft nun auch auf Tablets
- Modernere Benutzeroberfläche
- Auslagerung von Funktionen in Extensions (Erweiterungen)
- Schnittstelle zu externer Hardware (LEGO, micro:bit etc.)
- Verbesserte Sound und Bildeditoren

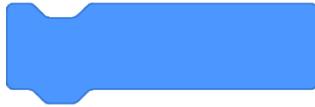
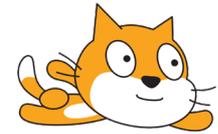
Wichtiges Bestehendes

- Konten wurden automatisch auf Scratch 3.0 migriert
- Alte Projekte funktionieren immer noch
- Grundprinzipien von Scratch bleiben bestehen

Programmierkonzepte



Programmierkonzepte – Anweisung



Ein Anweisungsblock.

Eine Anweisung ist ein **eindeutiger Befehl**,
welcher ein **Objekt** im Programm **manipuliert**.

Bewegung

- Bewegung: gehe 10 er Schritt
- Aussehen: drehe dich um 15 Grad
- Klang: drehe dich um 15 Grad
- Ereignisse: gehe zu Zufallsposition
- Steuerung: gehe zu x: 183 y: 74
- Fühlen: gleite in 1 Sek. zu Zufallsposition
- Operatoren

Anweisung für Figur:
Bewegung

Aussehen

- Bewegung: sage Hallo! für 2 Sekunden
- Aussehen: sage Hallo!
- Klang: denke Hmm... für 2 Sekunden
- Ereignisse: denke Hmm...
- Steuerung: wechsele zu Kostüm Kostüm 1
- Fühlen: wechsele zum nächsten Kostüm
- Operatoren

Anweisungen für Figur:
Aussehen

Klang

- Bewegung: spiele Klang Miau ganz
- Aussehen: spiele Klang Miau
- Klang: stoppe alle Klänge
- Ereignisse: ändere Effekt Höhe um 10
- Steuerung: setze Effekt Höhe auf 100
- Fühlen: schalte Klangeffekte aus
- Operatoren

Anweisungen für
Klangausgabe

Programmierkonzepte – Sequenz

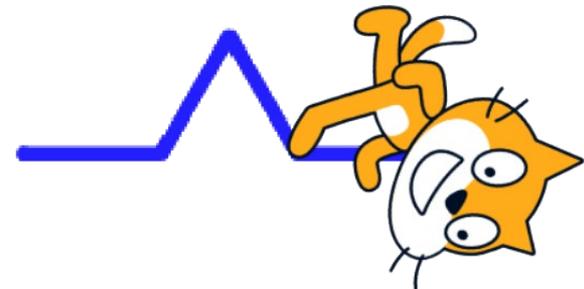


Sequenz von Anweisungen



Eine Sequenz ist eine Abfolge von Anweisungen, die untereinander zusammengesteckt werden können. Sie werden **nacheinander** (sequentiell) **von oben nach unten** ausgeführt.

Ergebnis der Sequenz von Anweisungen: Mit dem Malstift zusammen zeichnet die Bewegung der Figur eine Zickzacklinie im Spielfeld.



Passende Blöcke schnappen immer aneinander.

Programmierkonzepte – Schleife



Schleifenblöcke findet man im Menü Steuerung:



In einem Programmcode reihen sich Anweisungsblöcke in langen Sequenzen aneinander. Um den Programmcode **kompakter** zu gestalten, können **sich wiederholende** Anweisungssequenzen **in Schleifen** verpackt werden.

Es werden drei verschiedene Arten von Schleifen unterschieden:



Wiederhole x Mal: Diese Schleife wiederholt die Sequenz innerhalb der Schleife genau so oft, wie in dem Zahlenfeld eingegeben.



Wiederhole fortlaufend: Die Sequenz innerhalb der Schleife wird unendlich oft wiederholt.

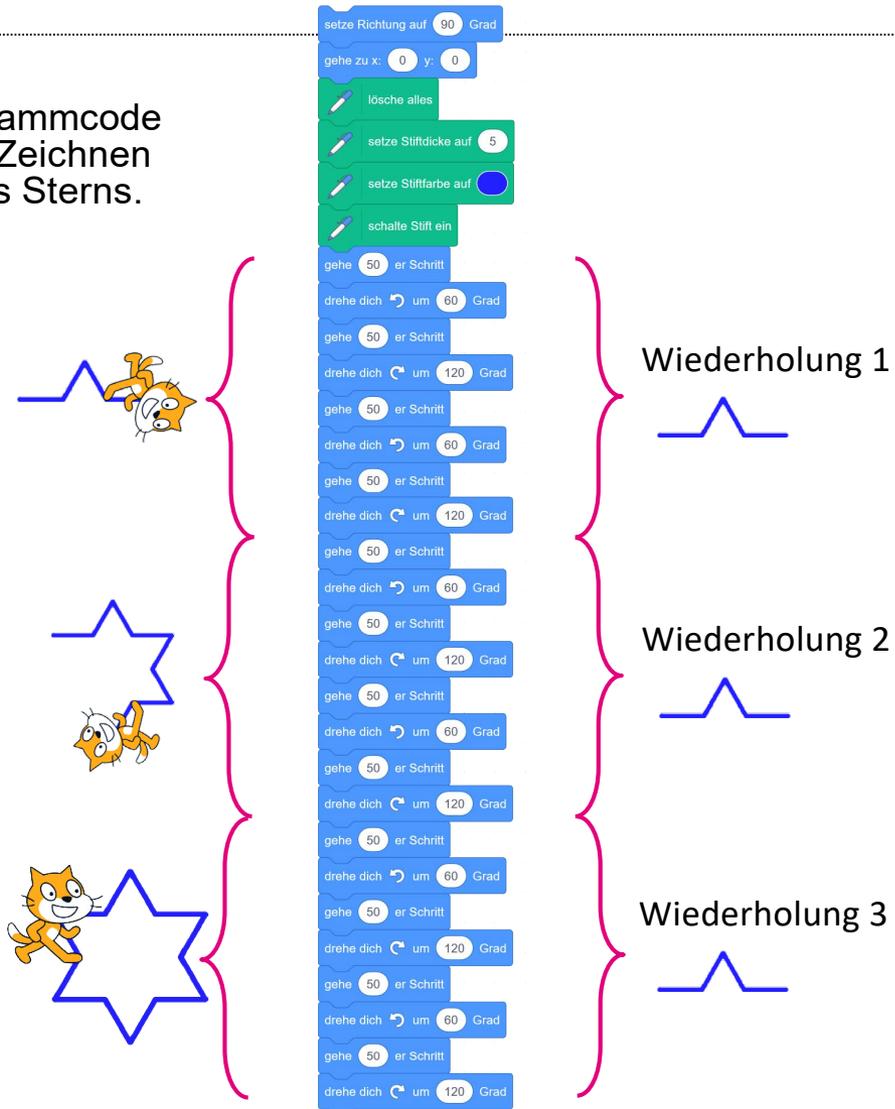


Wiederhole bis: Jeder Durchlauf prüft, ob die Abbruchbedingung erfüllt ist. Hier wird bspw. die Wiederholung beendet, sobald die Leertaste gedrückt wurde.

Programmierkonzepte – Schleife

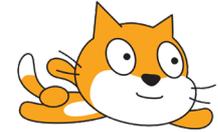


Programmcode zum Zeichnen eines Sterns.

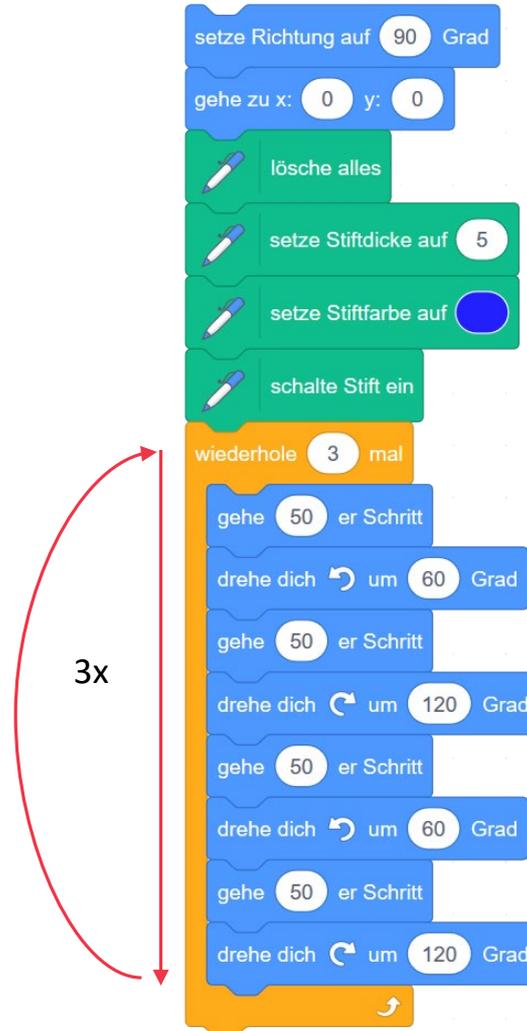
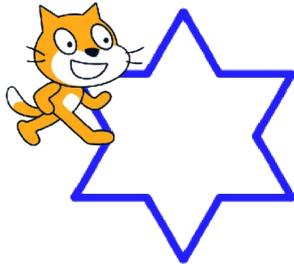


 Durch **Mustererkennung** identifiziert man Programmcode, der sich wiederholt. Sich wiederholende Programmblöcke können in Schleifen verpackt werden.

Programmierkonzepte – Schleife



Kompaktere Darstellung des Programmcodes zum Zeichnen eines Sterns mittels einer Schleife. Die Ausgabe ist bei beiden Programmen identisch.





Bedingungsblöcke findet man im Menü Steuerung:



Wenn man die Ausführung einer Anweisung oder einer Sequenz von **bestimmten Faktoren** abhängig machen möchte, dann eignen sich die **Bedingungsblöcke**.



Eine Bedingung prüft immer, ob eine Frage mit «**wahr**» (true) beantwortet werden kann. Falls ja, dann wird die Anweisung oder die Sequenz innerhalb des Bedingungsblocks ausgeführt. Wenn die Antwort auf die Frage «**nicht wahr**» ist (false), dann wird der Programmcode übersprungen. Unmittelbar nach dem Bedingungsblock wird das Programm wieder fortgesetzt.



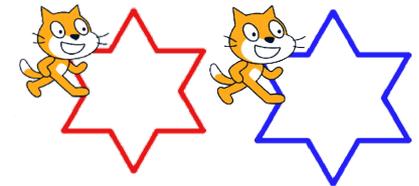
Neben **Falls/Dann**, gibt es noch die **Falls/Dann/Sonst** Bedingung. Der einzige Unterschied ist, dass hier bei einer nicht wahren Antwort die Anweisungen im Teil «sonst» ausgeführt werden. In diesem Block wird folglich immer entweder der «dann» Teil oder der «sonst» Teil ausgeführt.

Programmierkonzepte – Bedingung



```
setze Richtung auf 90 Grad
gehe zu x: 0 y: 0
lösche alles
setze Stiftdicke auf 5
setze Stiftfarbe auf [blau]
falls Zufallszahl von 1 bis 2 = 2, dann
  setze Stiftfarbe auf [rot]
schalte Stift ein
wiederhole 3 mal
  gehe 50 er Schritt
  drehe dich um 60 Grad
  gehe 50 er Schritt
  drehe dich um 120 Grad
  gehe 50 er Schritt
  drehe dich um 60 Grad
  gehe 50 er Schritt
  drehe dich um 120 Grad
```

```
setze Richtung auf 90 Grad
gehe zu x: 0 y: 0
lösche alles
setze Stiftdicke auf 5
falls Zufallszahl von 1 bis 2 = 2, dann
  setze Stiftfarbe auf [rot]
sonst
  setze Stiftfarbe auf [blau]
schalte Stift ein
wiederhole 3 mal
  gehe 50 er Schritt
  drehe dich um 60 Grad
  gehe 50 er Schritt
  drehe dich um 120 Grad
  gehe 50 er Schritt
  drehe dich um 60 Grad
  gehe 50 er Schritt
  drehe dich um 120 Grad
```



Jede Farbe hat eine Wahrscheinlichkeit von 1:2.



Ereignisblöcke findet man im Menü Ereignisse:



Wenn man die Ausführung einer Anweisung oder einer Sequenz von äusseren Faktoren abhängig machen möchte, dann eignet sich das Ereignis.

Egal, an welcher Position sich das Programm gerade zum Zeitpunkt des Eintreffens des Ereignisses befindet, der Ereignisblock wird immer sofort ausgeführt. Dieses Verhalten wird auch als «Interrupt» bezeichnet.

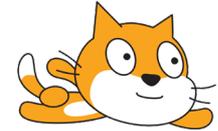


Sobald ein Ereignis eintrifft, wird die Sequenz unterhalb des Ereignisblocks sofort ausgeführt. Ereignisblöcke haben immer diese Form



Ereignisblöcke unterscheiden sich von Bedingungen dadurch, dass sie jederzeit ausgeführt werden können. Bei Bedingungen geschieht das nur zu dem Zeitpunkt, in dem das Programm den Bedingungsblock innerhalb einer Sequenz erreicht und die Frage prüft. Mehrere Ereignisblöcke können zudem parallel laufen. Ereignisblöcke, die nicht durch ein Ereignis ausgelöst werden, werden nie ausgeführt.

Programmierkonzepte – Ereignis



Hier gibt es zwei Ereignisblöcke «grüne Fahne wurde oberhalb des Spielfelds angeklickt». Beide Blöcke werden somit bei diesem Ereignis **gleichzeitig** ausgeführt. Wenn die Leertaste irgendwann im Verlauf des Programms vom Nutzer gedrückt wird, dann erscheint die Sprechblase «Hallo!» für 2 Sekunden neben dem Agenten im Spielfeld.

```
Wenn grüne Fahne angeklickt wird
  setze Richtung auf 90 Grad
  gehe zu x: 0 y: 0
  lösche alles
  setze Stiftdicke auf 5
  setze Stiftfarbe auf blau
  schalte Stift ein
  wiederhole 3 mal
    gehe 50 er Schritt
    drehe dich um 60 Grad
    gehe 50 er Schritt
    drehe dich um 120 Grad
    gehe 50 er Schritt
    drehe dich um 60 Grad
    gehe 50 er Schritt
    drehe dich um 120 Grad
```

```
Wenn grüne Fahne angeklickt wird
  spiele Klang Miau
```

```
Wenn Taste Leertaste gedrückt wird
  sage Hallo! für 2 Sekunden
```



Programmierkonzepte – Unterprogramm/Funktion



Unterprogramme findet man im Menü Weitere Blöcke:



Um eine bestimmte Sequenz mehrfach benutzen bzw. **wiederverwenden** zu können und um das Programm kompakter zu gestalten werden **Unterprogramme/Funktionen** verwendet.

Bei Unterprogrammen fasst man eine Sequenz in eine einzige Anweisung zusammen. Darum hat ein Unterprogramm immer zwei Blöcke:



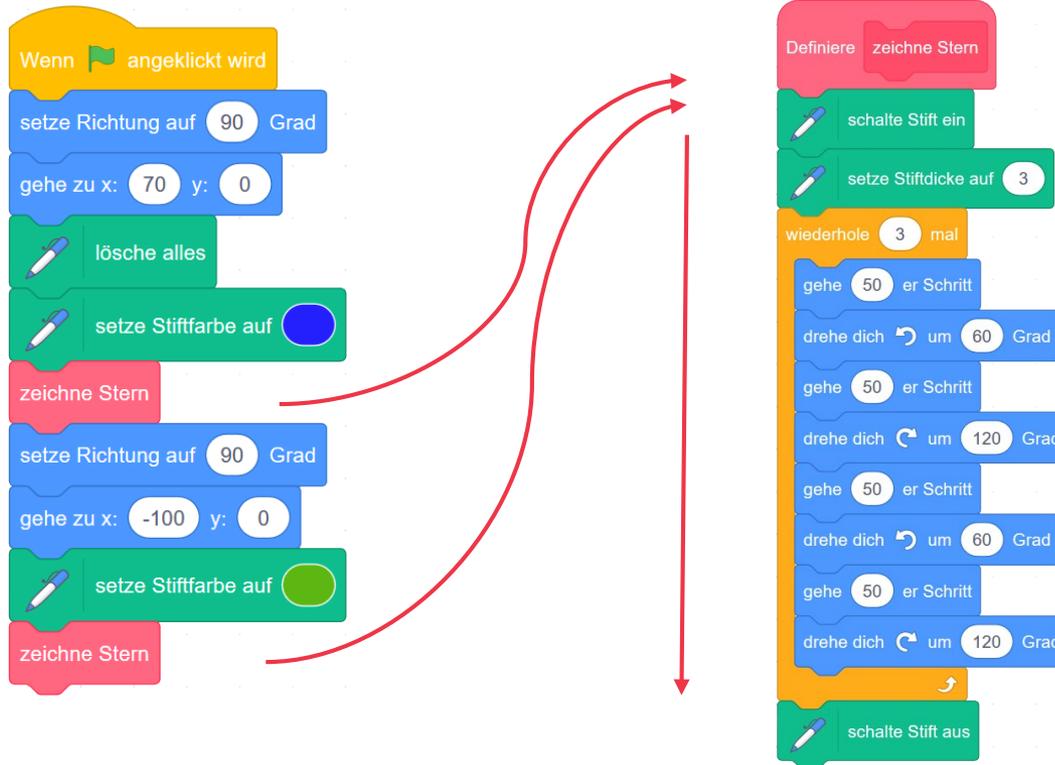
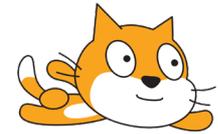
Unterhalb des **Definitionsblocks** werden die Anweisungen für das Unterprogramm definiert.



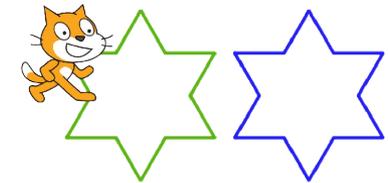
Der zweite Block ist der dazugehörige **Anweisungsblock des Unterprogramms**. Über diesen wird das Unterprogramm aufgerufen und ausgeführt.

Unterprogramme werden auch als «Methoden» oder «Funktionen» bezeichnet. Im Grunde führt jede Anweisung ein Unterprogramm aus. Existierenden Anweisungen sind bereits als Unterprogramme vorprogrammiert und stehen als «Bibliothek» zur Verfügung.

Programmierkonzepte – Unterprogramm/Funktion

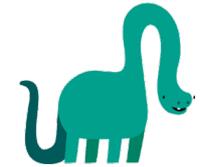


Dabei werden zwei Sterne im Spielfeld nebeneinander gezeichnet.



Der Programmcode zum Zeichnen eines Sterns wird in einem Unterprogramm definiert und «ausgelagert». Beim Hauptprogramm wird das Unterprogramm über die «zeichne Stern» Anweisung an zwei verschiedenen Stellen aufgerufen. Dies macht den Programmcode modularer und das Zeichnen des Sterns wiederverwendbar.

Programmierkonzepte – Parameter

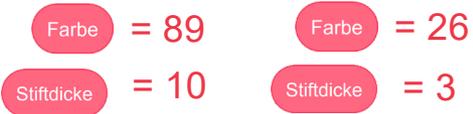


```
Wenn  angeklickt wird
  setze Richtung auf 90 Grad
  gehe zu x: 70 y: 0
  lösche alles
  zeichne Stern mit Farbe 26 und Stiftdicke 3
  setze Richtung auf 90 Grad
  gehe zu x: -100 y: 0
  zeichne Stern mit Farbe 89 und Stiftdicke 10
```



```
Definiere zeichne Stern mit Farbe Farbe und Stiftdicke Stiftdicke
  setze Stift Farbe auf Farbe
  setze Stiftdicke auf Stiftdicke
  schalte Stift ein
  wiederhole 3 mal
    gehe 50 er Schritt
    drehe dich um 60 Grad
    gehe 50 er Schritt
    drehe dich um 120 Grad
    gehe 50 er Schritt
    drehe dich um 60 Grad
    gehe 50 er Schritt
    drehe dich um 120 Grad
  schalte Stift aus
```

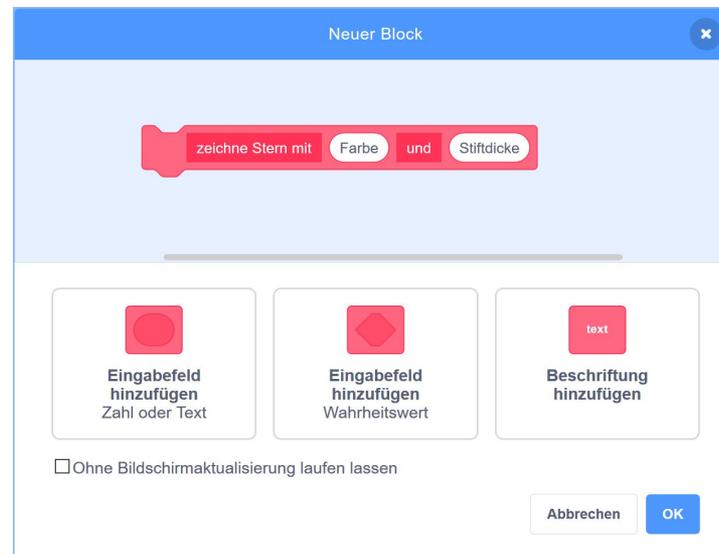
Durch **Abstraktion** identifiziert man eine Sequenz, die im Unterprogramm ausgelagert werden kann. Hier stellt sich die Frage, welche Anweisungen essentiell sind für das Zeichnen eines Sterns.



Oftmals möchte man die Ausführung eines Unterprogramms flexibel gestaltet. D.h. man möchte erst zum Zeitpunkt des Aufrufs der Anweisung gewisse Werte dem Unterprogramm mitgeben. Dafür gibt es **Parameter**.



Bei der Definition des eigenen Unterprogramms können ebenfalls mehrere Parameter definiert werden:



In Scratch haben Parameter drei verschiedene Datentypen: **Zahl** (rund), **Text** (rechteckig) und **Boolesch (Aussagenlogik)** (polygon). Der Beschriftungstext dient nur der Leserlichkeit der Anweisung.

Programmierkonzepte – Variable



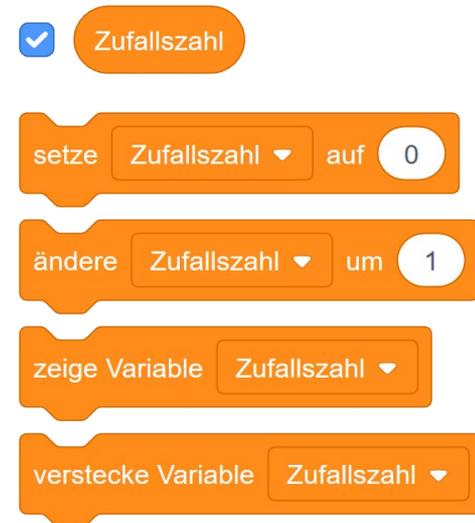
Variablen findet man im Menü **Daten**:



Oftmals möchte man gewisse Werte speichern und an verschiedenen Orten im Programm wiederverwenden. Dafür eignen sich Variablen.

Bei der Definition einer Variablen kann man angeben, ob diese nur **lokal** für eine bestimmte Figur oder **global** für alle Figuren zur Verfügung stehen soll:

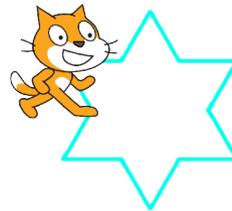
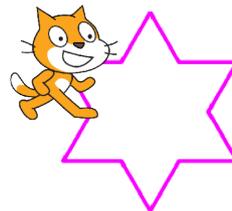
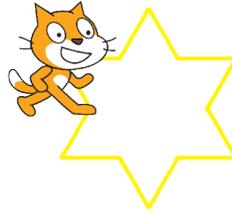
Sobald die Variable erstellt wurde, stehen verschiedene Blöcke zur Verfügung: Anweisungsblöcke, welche die Variable verändern können, sowie ein runder Block, der den Variablenwert speichert und somit als Parameter eingesetzt werden kann.



Programmierkonzepte – Variable



```
Wenn  angeklickt wird
  setze Richtung auf 90 Grad
  gehe zu x: 0 y: 0
  setze Türkis auf 49
  setze Gelb auf 16
  setze Pink auf 83
  setze Zufallszahl auf Zufallszahl von 1 bis 3
  lösche alles
  falls Zufallszahl = 1, dann
    setze Stift Farbe auf Gelb
  falls Zufallszahl = 2, dann
    setze Stift Farbe auf Pink
  falls Zufallszahl = 3, dann
    setze Stift Farbe auf Türkis
  zeichne Stern
```



```
Zufallszahl 1
Gelb 16
Pink 83
Türkis 49
```

```
Zufallszahl 2
Gelb 16
Pink 83
Türkis 49
```

```
Zufallszahl 3
Gelb 16
Pink 83
Türkis 49
```

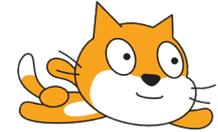
Programmcode zum Zeichnen eines Sterns mit zufällig gewählter Farbe. Die Zufallszahl von 1-3 wird über eine Variable gespeichert, die Farbenwerte Gelb, Pink und Türkis ebenso. Die Variablen der Farben werden als Parameter den «setze Stiftfarbe auf» Anweisungen übergeben.



Mit der Anweisung «setze Variablenname auf» wird der Variablen ein Wert zugewiesen. Dieser bleibt bestehen bis der Wert über dieselbe Anweisung später im Programm überschrieben wird.

Übung: All You Need Is Love





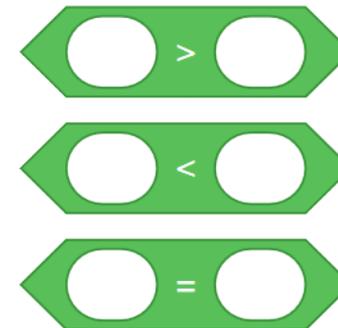
Boolesche Algebra findet man im Menü **Operatoren**:



Das Ergebnis einer Booleschen Algebra ist entweder **wahr** (true) oder **nicht wahr** (false). Diese Datentypen werden in der Form des Polygons dargestellt. In Scratch gibt es drei Arten von Aussagen. Dabei muss jeder Platzhalter der Aussage ebenso einen booleschen Wert enthalten (Polygonform).



Die Ergebnisse von Vergleichen in der Mathematik ergeben ebenso die Aussage «wahr» oder «nicht wahr». Darum haben diese Blöcke eine Polygonform und können als Eingaben im booleschen Aussagenblock dienen.



Programmierkonzepte – Boolesche Algebra



Bei einer **UND** (AND) Verknüpfung müssen immer **beide Seiten** der Aussage «**wahr**» sein, damit die gesamte Aussage «wahr» ist.



= wahr



= nicht wahr



= nicht wahr

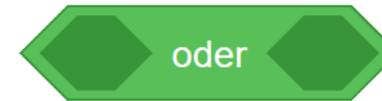


= nicht wahr

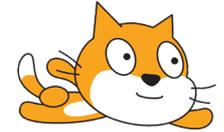
Programmierkonzepte – Boolesche Algebra



Bei einer **ODER** (OR) Verknüpfung muss mindestens eine Seite der Aussage «wahr» sein, damit die gesamte Aussage «wahr» ist.



Programmierkonzepte – Boolesche Algebra



Bei einer **NICHT** (NOT) Verknüpfung wird die Aussage immer ins Gegenteil gesetzt.



true

= wahr



false

= nicht wahr



false

= nicht wahr



true

= wahr



false

= nicht wahr



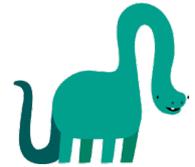
true

= wahr

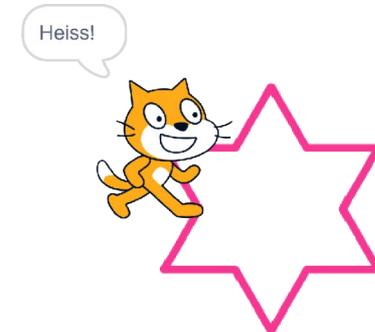
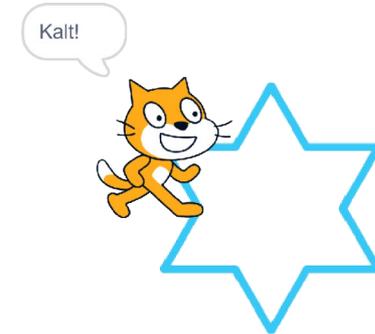
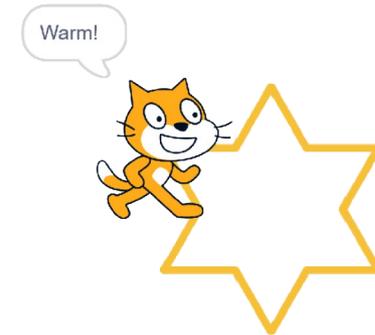


Wahr, wenn die Leertaste nicht gedrückt ist.
Nicht wahr, wenn sie gedrückt ist.

Programmierkonzepte – Anwendung von Boolescher Algebra



```
Wenn  angeklickt wird
  setze Richtung auf 90 Grad
  gehe zu x: 0 y: 0
  lösche alles
  setze Zufallszahl auf Zufallszahl von 1 bis 99
  falls 0 < Zufallszahl und Zufallszahl < 34, dann
    sage Warm!
  sonst
    falls 33 < Zufallszahl und Zufallszahl < 67, dann
      sage Kalt!
    sonst
      sage Heiss!
  zeichne Stern mit Zufallszahl und 5
```



Programmierkonzepte – Anwendung von Boolescher Algebra



Überall, wo in Scratch ein Datentyp in Polygonform verlangt wird, kann das Ergebnis einer beliebig komplex verschachtelten Booleschen Algebra verwendet werden.

Dies trifft bei Bedingungen zu:



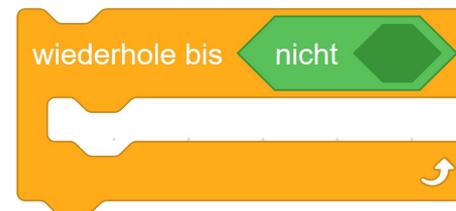
Bei der «warte bis» Anweisung: Hier wird der Programmcode nicht weiter ausgeführt, bis die Aussage «wahr» ist.



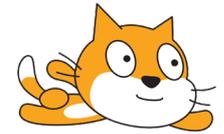
Bei der «wiederhole bis» Schleife wird diese solange ausgeführt, bis die Aussage «wahr» ist. Das ist die Abbruchbedingung.



Mit einem «nicht» Block kann die «wiederhole bis» Schleife zur «wiederhole solange» Schleife umfunktioniert werden.



Programmierkonzepte – Anwendung von Boolescher Algebra



```
Wenn [ ] angeklickt wird
  setze Richtung auf 90 Grad
  gehe zu x: 0 y: 0
  setze Stiftfarbe auf [red]
  lösche alles
  wiederhole bis [Taste Leertaste gedrückt? oder Maustaste gedrückt?]
  ändere Stift Farbe um 10
  zeichne Stern
  warte 0.2 Sekunden
```

```
Wenn [ ] angeklickt wird
  setze Richtung auf 90 Grad
  gehe zu x: 0 y: 0
  setze Stiftfarbe auf [red]
  lösche alles
  wiederhole bis nicht [Taste Leertaste gedrückt? oder Maustaste gedrückt?]
  ändere Stift Farbe um 10
  zeichne Stern
  warte 0.2 Sekunden
```

Programmierkonzepte – Datentypen



Datentypen beschreiben Eigenschaften von Daten, wie z.B. deren Wertebereich und Operationen, die man auf alle Daten dieses Typs anwenden kann. In Scratch existieren hauptsächlich drei Arten von Datentypen:

Zahl, dargestellt als rundes Textfeld

- Ganzzahl (Integer), z.B. [0, -4, 67, 200, ...]
- Gleitkommazahl (Float), z.B. [0.1, -27.8, ...] mit einer Genauigkeit von 6 Stellen hinter dem Komma.



Boolesch (Boolean), dargestellt als Polygon

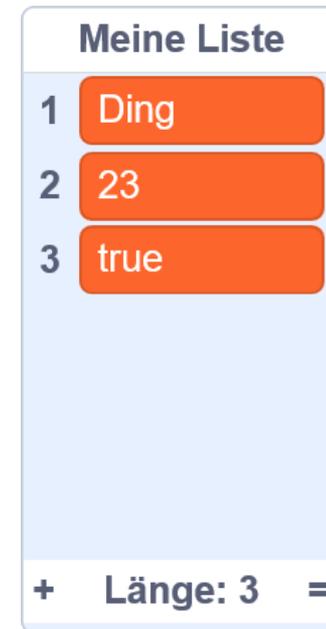


Zeichenkette (String), dargestellt als rundes Textfeld

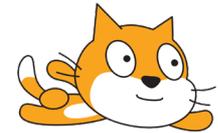




Es ist auch möglich, verschieden lange Listen (aus Menü «Variablen») mit unterschiedlichen Datentypen zusammenzustellen:



Programmierkonzepte – Datentypen



Zum Zeitpunkt der Zuordnung eines Wertes einer Variablen mittels der Anweisung «setze Variablenname auf» wird der Datentyp festgelegt. Dabei kann im rechteckigen Textfeld eine Zahl, eine Zeichenkette oder eine boolesche Aussagenlogik eingegeben werden.

setze Variable auf & * Zeichenkette mit Sonderzeichen 56 \$!!!

Variable & * Zeichenkette mit Sonderzeichen 56 \$!!!

setze Variable auf 567.6

Variable 567.6

setze Variable auf $5 = 5$ und $5 > 4$

Variable true

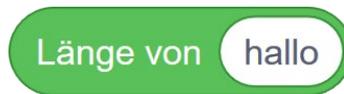
setze Variable auf 5.894
ändere Variable um 1

Variable 6.894

Programmierkonzepte – Datentypen



Beim Umgang mit unterschiedlichen Datentypen muss man sich immer überlegen, welcher Wertebereich der Datentyp besitzt und was für ein Datentyp als Ergebnis einer Operation erwartet wird.



Durch Doppelklick auf dem Operationsblock wird das Ergebnis in einer Sprechblase direkt angezeigt.

Lehrplan 21 – Kompetenzen im Programmieren



2					
	b	» können durch Probieren Lösungswege für einfache Problemstellungen suchen und auf Korrektheit prüfen (z.B. einen Weg suchen, eine Spielstrategie entwickeln). Sie können verschiedene Lösungswege vergleichen.			
	c	» können Abläufe mit Schleifen und Verzweigungen aus ihrer Umwelt erkennen, beschreiben und strukturiert darstellen (z.B. mittels Flussdiagrammen).			
	d	» können einfache Abläufe mit Schleifen, bedingten Anweisungen und Parametern lesen und manuell ausführen.			
	e	» verstehen, dass ein Computer nur vordefinierte Anweisungen ausführen kann und dass ein Programm eine Abfolge von solchen Anweisungen ist.			
	f	» können Programme mit Schleifen, bedingten Anweisungen und Parametern schreiben und testen.	MI - Produktion und Präsentation MA.2.C.2.g		
3	g	» können selbstentdeckte Lösungswege für einfache Probleme in Form von lauffähigen und korrekten Computerprogrammen mit Schleifen, bedingten Anweisungen und Parametern formulieren.			ole,
	h	» können selbstentwickelte Algorithmen in Form von lauffähigen und korrekten Computerprogrammen mit Variablen und Unterprogrammen formulieren.			l) und
	i	» können verschiedene Algorithmen zur Lösung desselben Problems vergleichen und beurteilen (z.B. lineare und binäre Suche, Sortierverfahren).			dem
	3	h	» können Dokumente so ablegen, dass auch andere sie wieder finden.		MI - Handhabung
		i	» können logische Operatoren verwenden (und, oder, nicht).		
		j	» können Daten in einer Datenbank strukturieren, erfassen, suchen und automatisiert auswerten.		
		k	» können Methoden zur Datenreplikation unterscheiden und anwenden (Backup, Synchronisation, Versionierung).		

The background features a stylized illustration of a person with their arms raised in a celebratory gesture. The person is rendered in shades of grey and green, with a red circular element on their chest. They are set against a vibrant purple background filled with faint, white outlines of various star and geometric shapes. A dark grey rectangular box is overlaid on the center of the image, containing the text.

Übung: Umsetzung einer Projektidee