Scratch 3.0 Vertiefungskurs
Kurstag 1

Kursüberblick

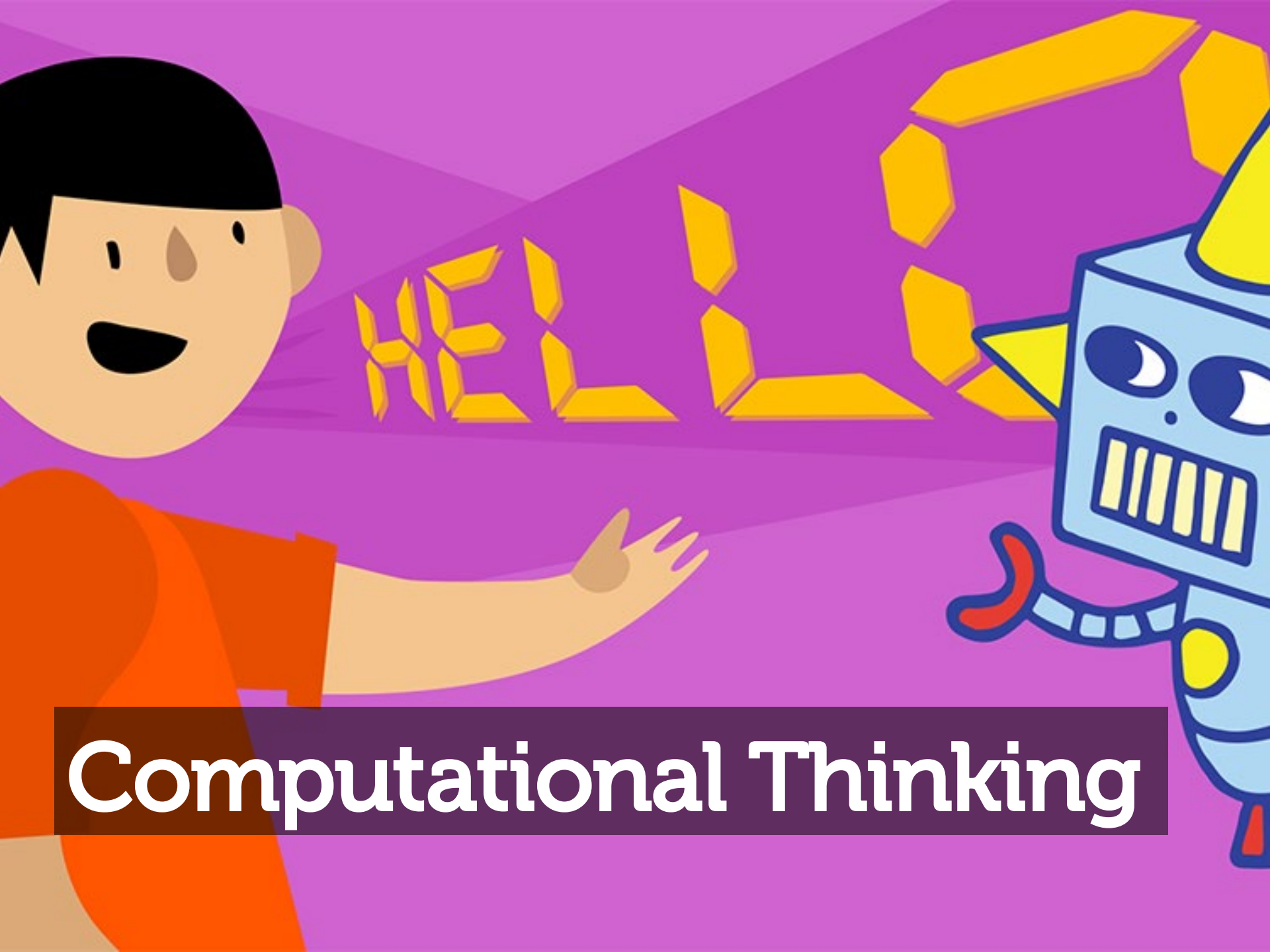
Kurstag 1 (Mittwoch, 24.2.21, 13.00 - 16.00 Uhr)

- Programmierkonzepte wie Schleifen, Bedingungen, Variablen, Unterprogramme usw. verstehen und anwenden können.
- Strukturiertes Vorgehen bei der Umsetzung einer Projektidee mittels Dekomposition, Mustererkennung und Abstraktion.

Kurstag 2 (Mittwoch, 24.3.21, 13.00 – 16.00 Uhr)

- Vorgehensweise bei der Fehlersuche.
- Scratch Erweiterungen mit LEGO Mindstorms, LEGO WeDo, MakeyMakey, micro:bit.
- Didaktische Hinweise (Kursaufbau, Best Practices, Materialien, Beurteilung usw.).
- Themen nach Wunsch.

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



Computational Thinking

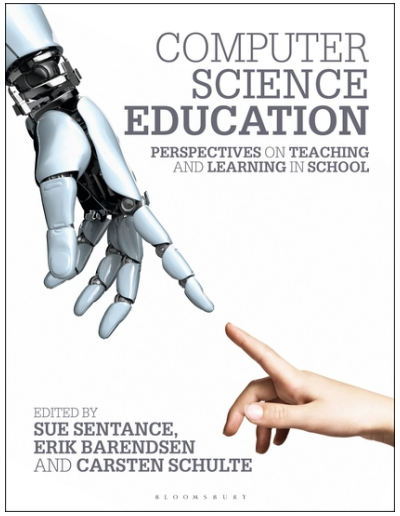
Computational Thinking (CT) – die Problemlösekompetenz der Informatik

CT Konzepte:

- Logik und logisches Denken
- Algorithmen und algorithmisches Denken
- Muster und Mustererkennung
- Abstaktion und Verallgemeinerung
- Evaluation
- Automation

CT Praktiken:

- Dekomposition eines Problems
- Entwicklung von rechnergestützten Artefakten
- Testen und Debuggen
- Iterative Verfeinerung/Verbesserung (inkrementielles Implementieren)
- Kollaboration & Kreativität (auch wichtige 21st Century Kompetenzen)



Computer Science Education (2018). *Perspectives on teaching and learning in school.*
Sue Sentance, Erik Barendsen, Carsten Schulte,
Bloomsbury
ISBN: 9781350057111

Computational Thinking (CT) – Verschiedene Stufen der Expertise

Umgebungen:

(1) Objekt-basiert, verkörpert (z.B. CS Unplugged Aktivitäten)

(2) Ausführbar, symbolisch (z.B. visuelle Block-Programmiersprachen wie Scratch)

(3) Ausführbar, formell (z.B. Textprogrammiersprachen wie Python)

Aktivitäten:

(1) Sich mit einer Problemstellung auseinandersetzen

(2) Einen Algorithmus implementieren

(3) Testen und Evaluieren

Definitionen noch in der Forschungs-phase!

Autonomie:

(1) Passiv (z.B. Abarbeiten von Schritt-für-Schritt Tutorials)

(2) Aktive Rolle in einer Gruppe oder Einzelarbeit mit Hilfestellung

(3) Aktive führende Rolle in einer Gruppe oder autonome Einzelarbeit

Quelle: gemäss aktuellem Forschungsprojekt "CTSkills" des SNF NFP77.

Lehrplan 21 – Computational Thinking Kompetenzen

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.2.0.2.g
3	g	» können selbstentdeckte Lösungswege für einfache Probleme in Form von lauffähigen und korrekten Computerprogrammen mit Schleifen, bedingten Anweisungen und Parametern formulieren.	
	h	» können selbstentwickelte Algorithmen in Form von lauffähigen und korrekten Computerprogrammen mit Variablen und Unterprogrammen formulieren.	
	i	» können verschiedene Algorithmen zur Lösung desselben Problems vergleichen und beurteilen (z.B. lineare und binäre Suche, Sortierverfahren).	

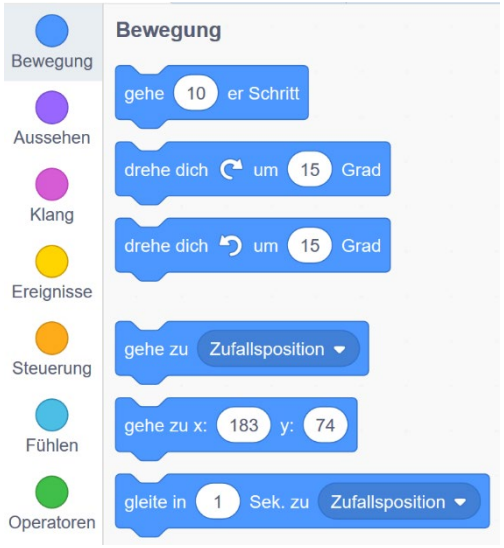
3	h	» können Dokumente so ablegen, dass auch andere sie wieder finden.	MI - Handhabung
	i	» können logische Operatoren verwenden (und, oder, nicht).	
	j	» können Daten in einer Datenbank strukturieren, erfassen, suchen und automatisiert auswerten.	
	k	» können Methoden zur Datenreplikation unterscheiden und anwenden (Backup, Synchronisation, Versionierung).	

Programmierkonzepte

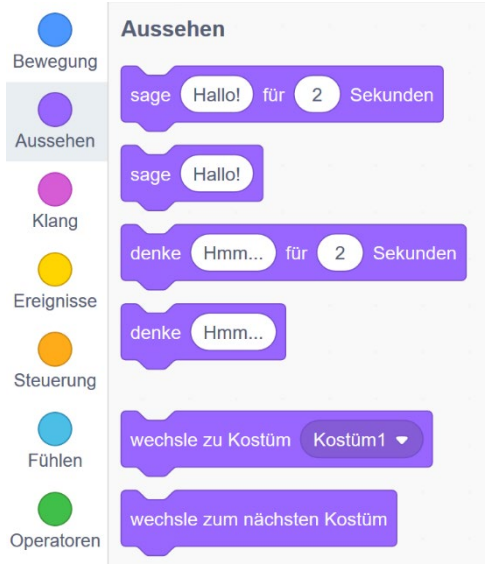
Programmierkonzepte – Anweisung

Ein Anweisungsblock.

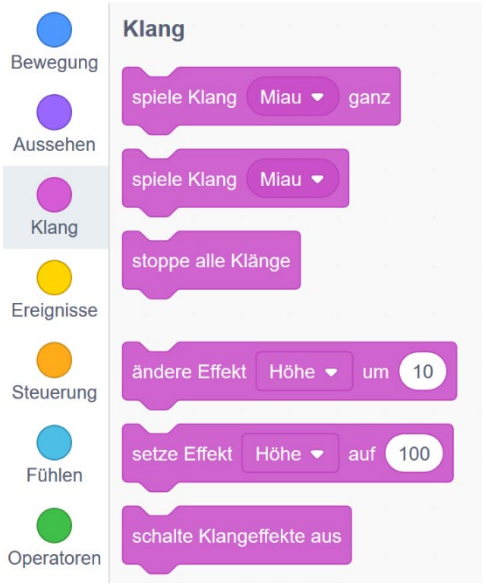
Eine Anweisung ist ein eindeutiger Befehl,
welcher ein Objekt im Programm manipuliert.



Anweisung für Figur:
Bewegung

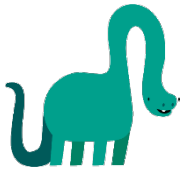


Anweisungen für Figur:
Aussehen



Anweisungen für
Klangausgabe

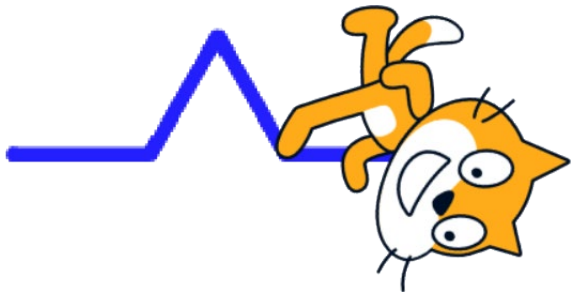
Programmierkonzepte – Sequenz



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.



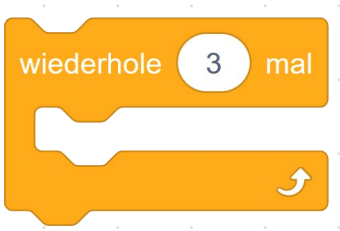
Passende Blöcke schnappen immer aneinander.

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:



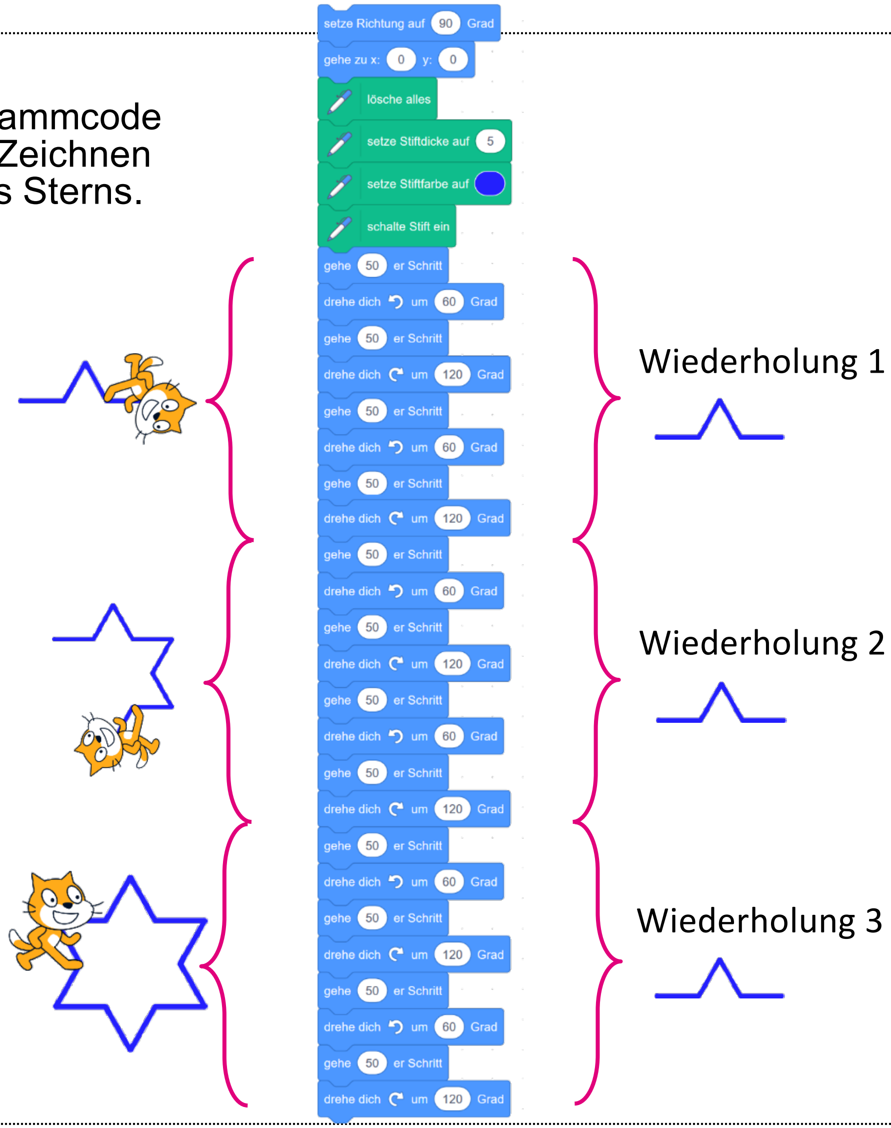
Wiederhole x Mal: Diese Schleife wiederholt die Sequenz innerhalb der Schleife genau so oft, wie in dem Zahlenfeld eingegeben.

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

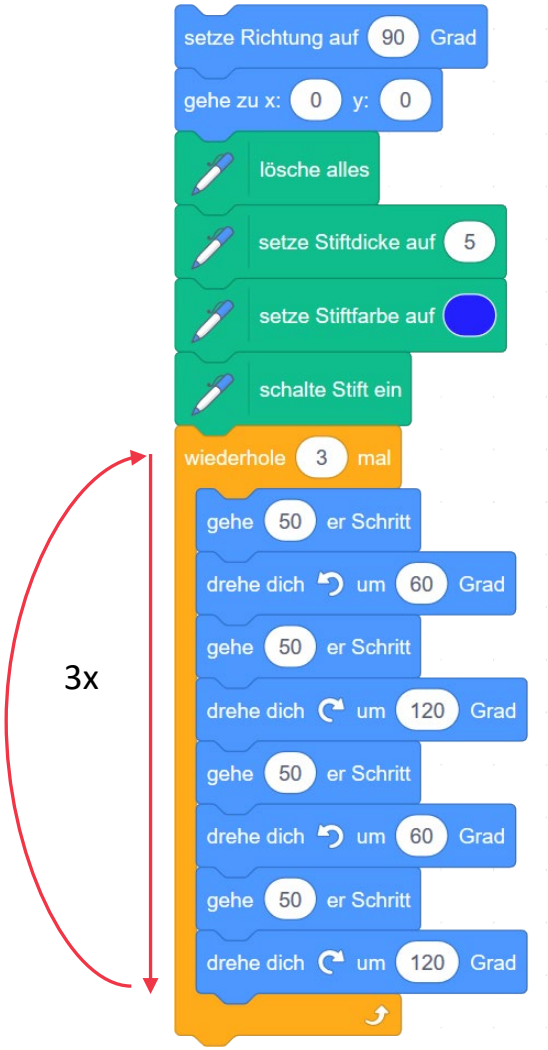
Programmcode
zum Zeichnen
eines Sterns.



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.

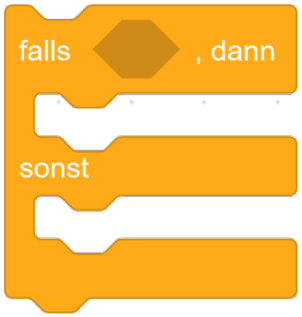


Programmierkonzepte – Bedingung

Bedingungsblöcke findet man im Menü Steuerung:

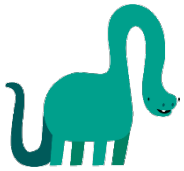
Wenn man die Ausführung einer Anweisung oder einer Sequenz von bestimmten Faktoren abhängig machen möchte, dann eignen sich die Bedingungsblöcke.

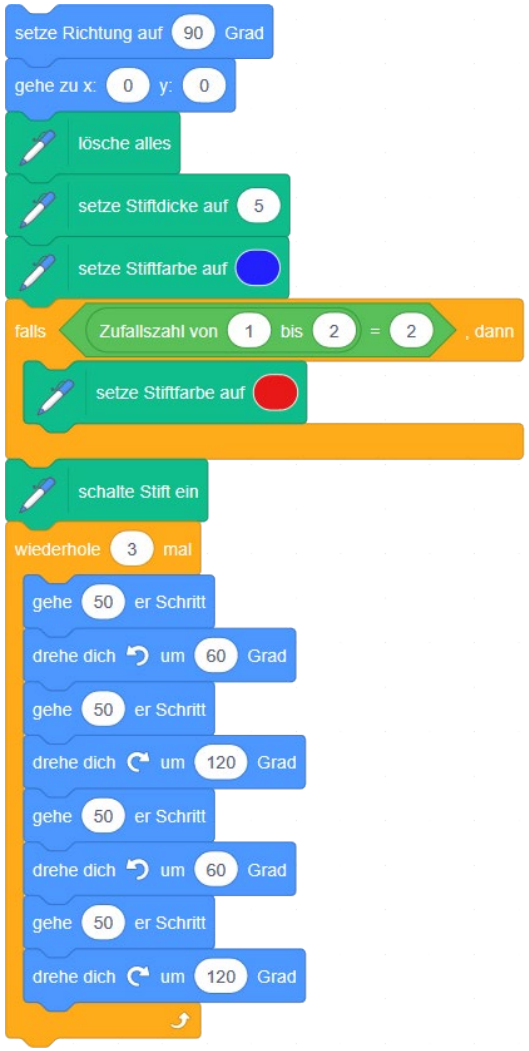
Eine Bedingung prüft immer, ob eine Frage mit «wahr» (true) beantwortet werden kann. Falls ja, dann wird die Anweisung oder die Sequenz innerhalb des Bedingungsblocks ausgeführt. Wenn die Antwort auf die Frage «nicht wahr» ist (false), dann wird der Programmcode übersprungen. Unmittelbar nach dem Bedingungsblock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

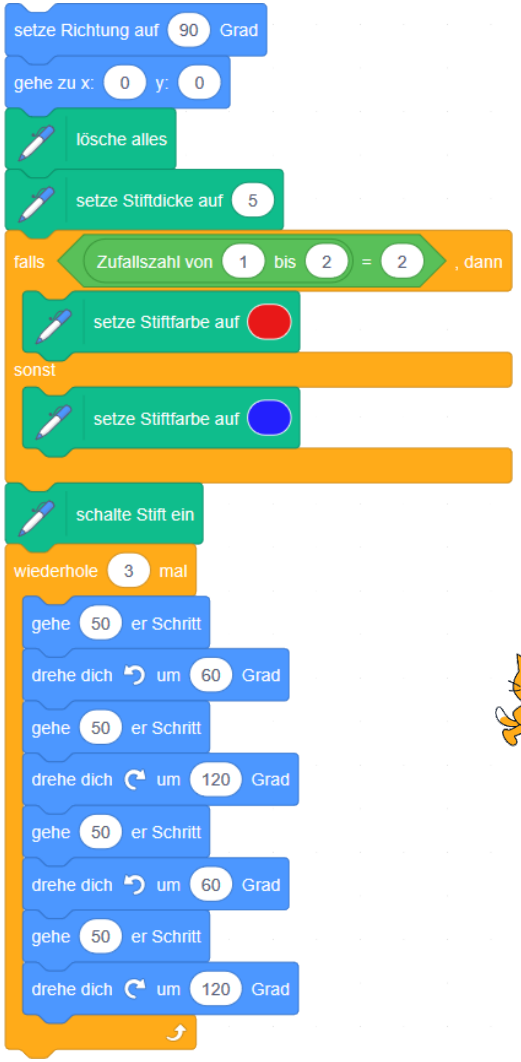


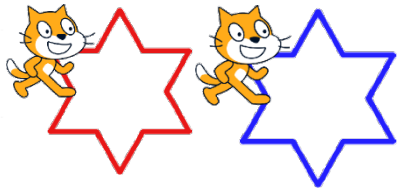
Neben Falls/Dann, gibt es noch die Falls/Dann/Sonst Bedingung. Der einzige Unterschied ist, dass hier bei einer nicht wahren Antwort die Anweisungen im Teil «sonst» ausgeführt werden. In diesem Block wird folglich immer entweder der «dann» Teil oder der «sonst» Teil ausgeführt.

Programmierkonzepte – Bedingung









Jede Farbe hat eine Wahrscheinlichkeit von 1:2.

Programmierkonzepte – Ereignis

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.

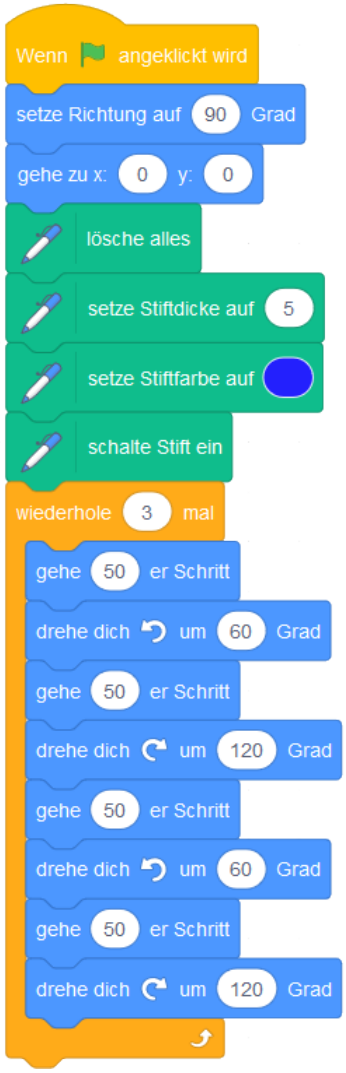
Egal, an welcher Position sich das Programm gerade zum Zeitpunkt des Eintreffens des Ereignisses befindet, der Ereignisblock wird immer sofort ausgeführt. Dieses Verhalten wird auch als «Interrupt» bezeichnet.

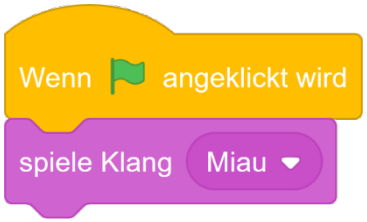
Sobald ein Ereignis eintrifft, wird die Sequenz unterhalb des Ereignisblocks sofort ausgeführt. Ereignisblöcke haben immer diese Form

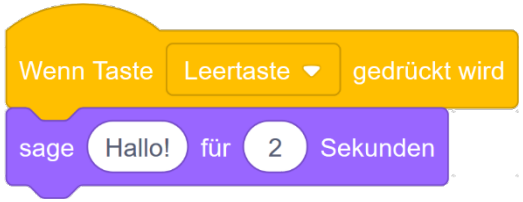
Ereignisblöcke unterscheiden sich von Bedingungen dadurch, dass sie jederzeit ausgeführt werden können. Bei Bedingungen geschieht das nur zu dem Zeitpunkt, in dem das Programm den Bedingungsblock innerhalb einer Sequenz erreicht und die Frage prüft. Mehrere Ereignisblöcke können zudem parallel laufen.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







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.

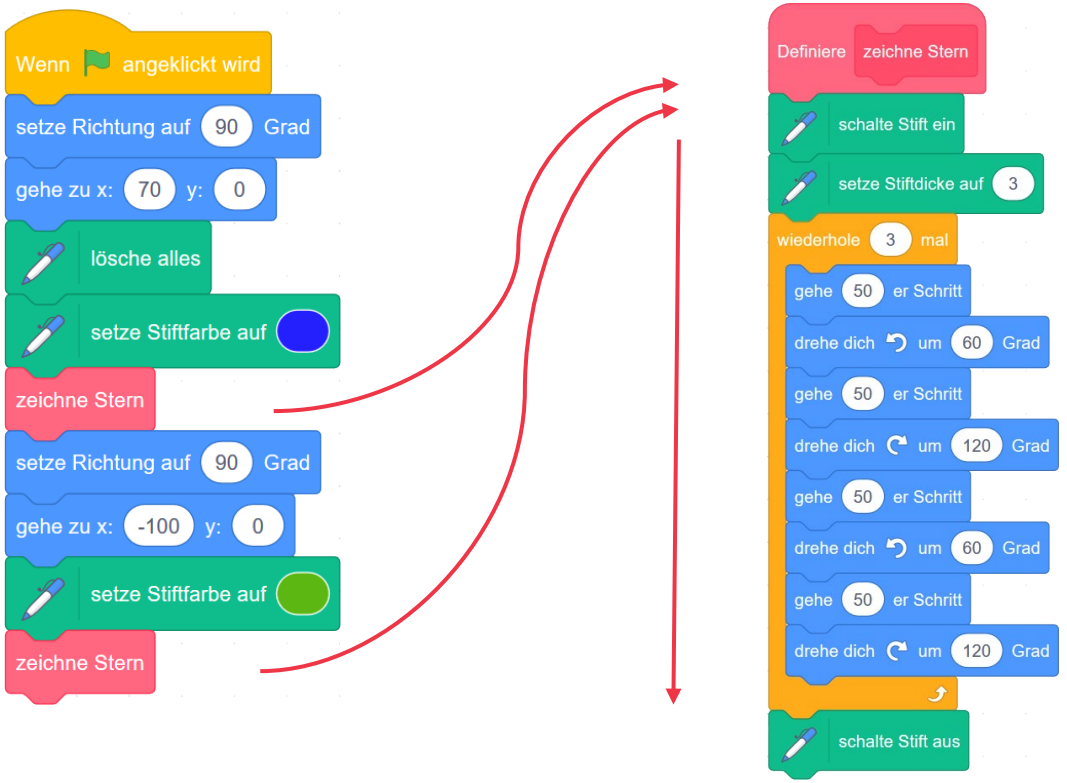
Bei Unterprogrammen fasst man eine Sequenz in eine einzige Anweisung zusammen. Darum hat ein Unterprogramm immer zwei Blöcke:

Unterhalb des Definitionsblocks werden die Anweisungen für das Unterprogramm definiert.

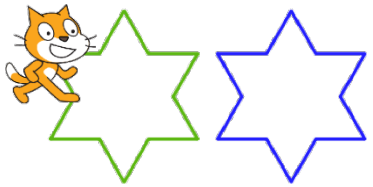
Der zweite Block ist der dazugehörige Anweisungsblock des Unterprogramms. Über diesen wird das Unterprogramm aufgerufen und ausgeführt.

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

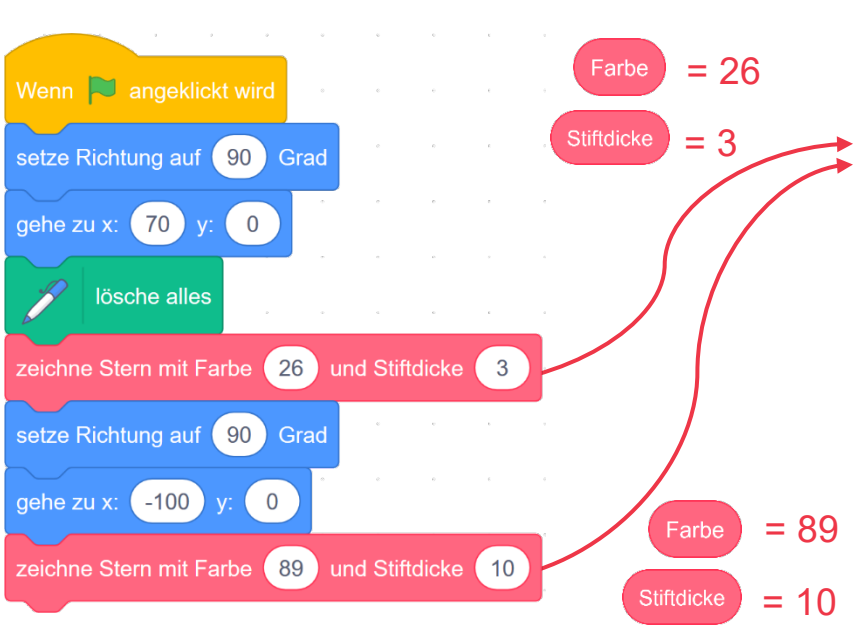


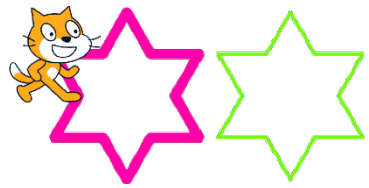
Dabei werden zwei Sterne im Spielfeld nebeneinander gezeichnet.



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.

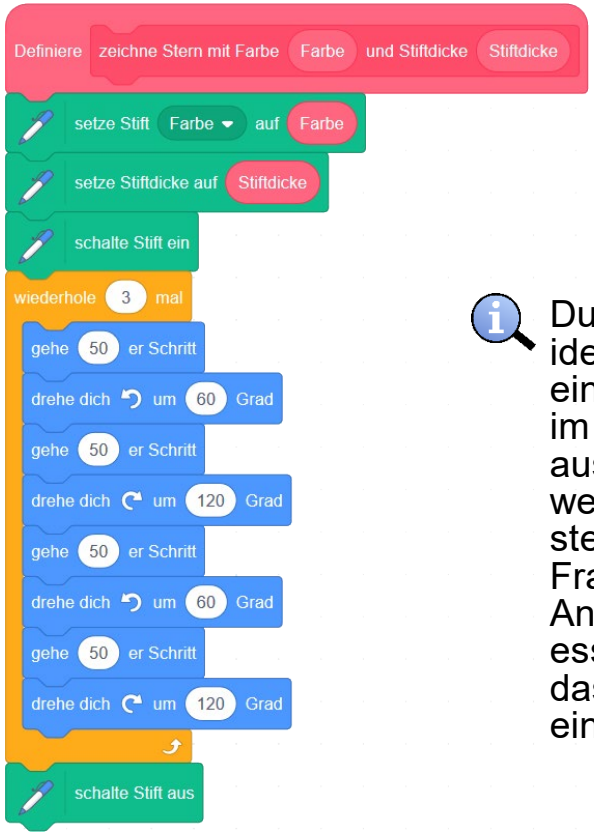
Programmierkonzepte – Parameter





Farbe = 89
Stiftdicke = 10

Farbe = 26
Stiftdicke = 3

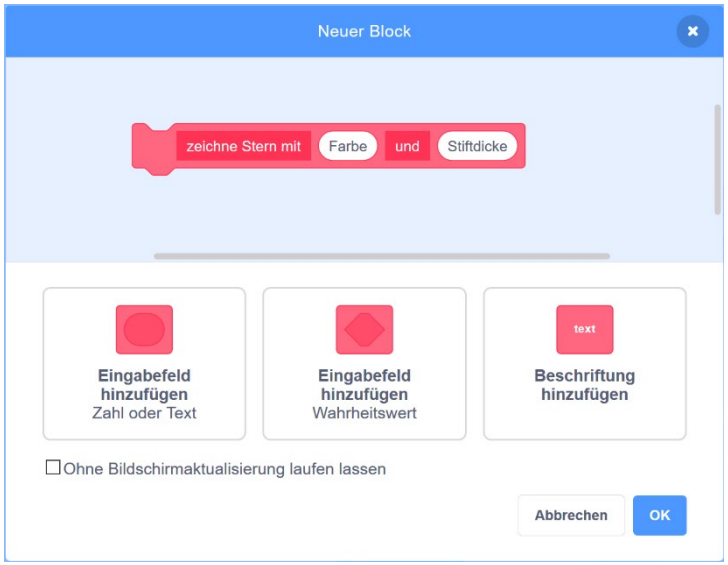


Durch Abstraktion identifiziert man eine Sequenz, die im Unterprogramm ausgelagert werden kann. Hier stellt sich die Frage, welche Anweisungen essentiell sind für das Zeichnen eines Sterns.

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.

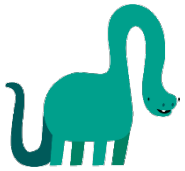
Programmierkonzepte – Parameter

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.

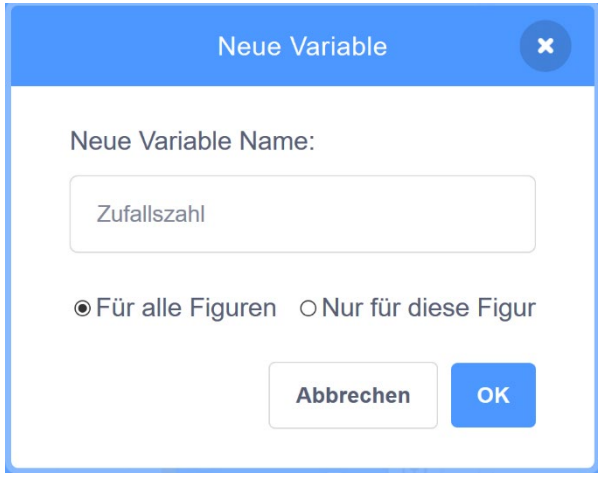
Programmierkonzepte – Variable



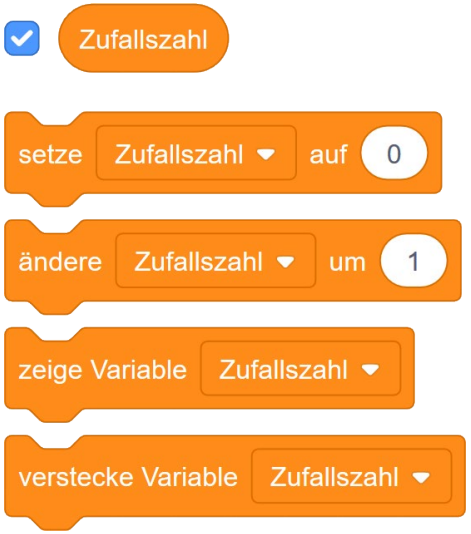
Variablen findet man im Menü Daten:

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

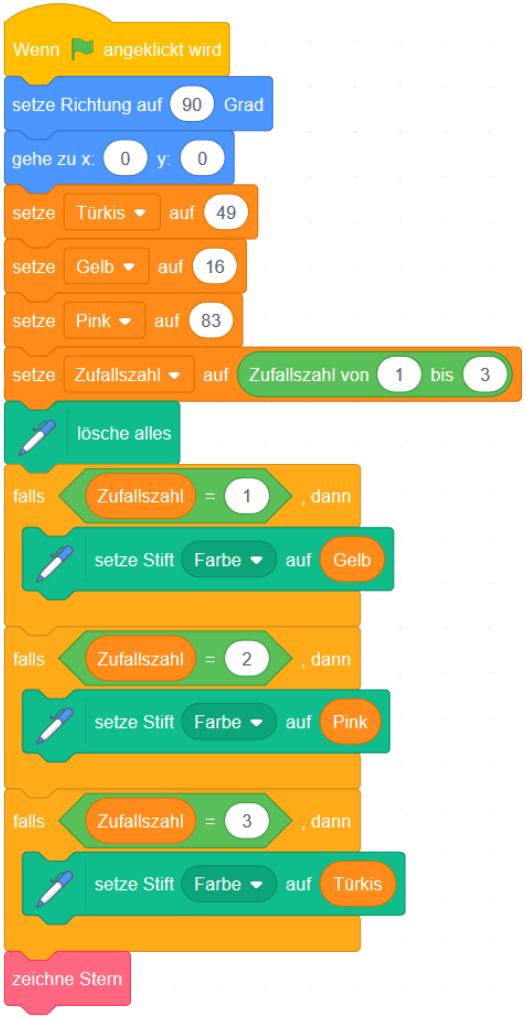
Bei der Definition einer Variablen kann man angeben, ob diese nur lokal für eine bestimmte Figur oder global für alle Figuren zur Verfügung stehen soll:



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

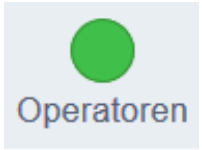


Programmcode zum Zeichnen eines Sterns mit zufällig gewählter Farbe. Die Zufallszahl von 1-3 wird über eine Variable gespeichert, die Farbenwerte Gelb, Pink und Türkis ebenso. Die Variablen der Farben werden als Parameter den «setze Stiftfarbe auf» Anweisungen übergeben.

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

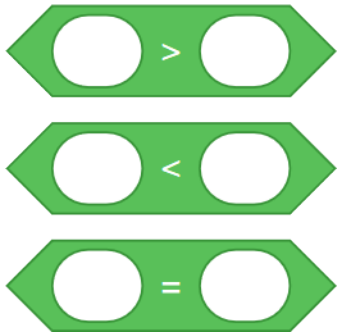
Programmierkonzepte – Boolesche Algebra

Boolesche Algebra findet man im Menü Operatoren:

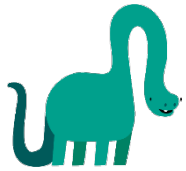


Das Ergebnis einer Booleschen Algebra ist entweder wahr (true) oder nicht wahr (false). Diese Datentypen werden in der Form des Polygons dargestellt. In Scratch gibt es drei Arten von Aussagen. Dabei muss jeder Platzhalter der Aussage ebenso einen booleschen Wert enthalten (Polygonform).

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

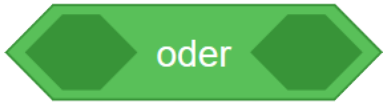
= nicht wahr

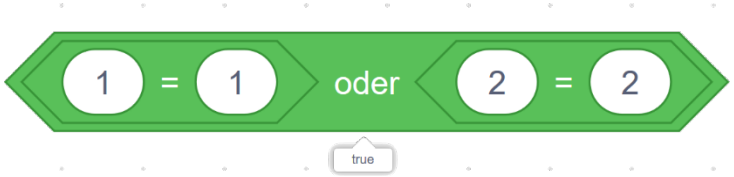
= nicht wahr

= nicht wahr

Programmierkonzepte – Boolesche Algebra

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





= wahr

= wahr

= wahr

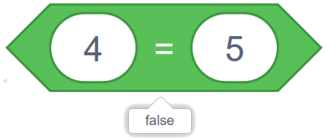
= nicht wahr

Programmierkonzepte – Boolesche Algebra

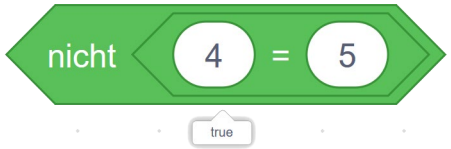
Bei einer NICHT (NOT) Verknüpfung wird die Aussage immer ins Gegenteil gesetzt.

= wahr

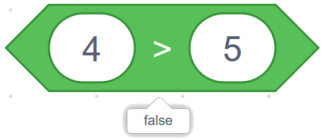
= nicht wahr



= nicht wahr

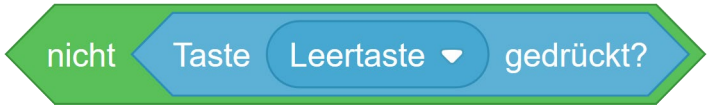


= wahr



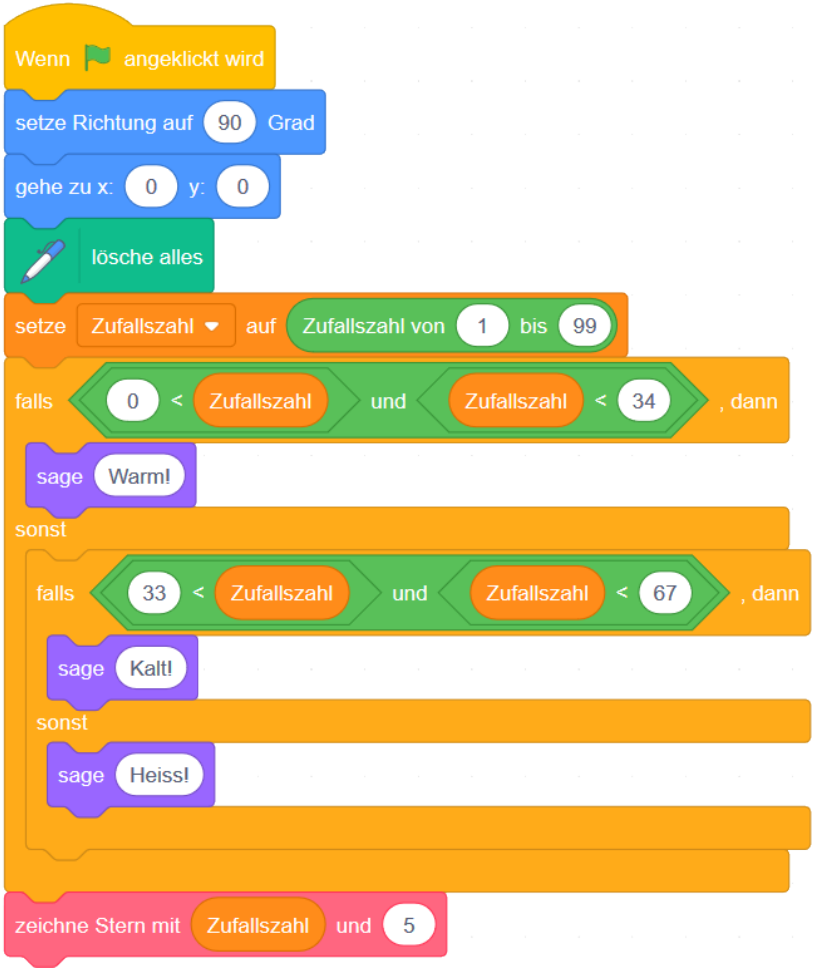
= nicht wahr

= wahr



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

Programmierkonzepte – Anwendung von Boolescher Algebra



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.

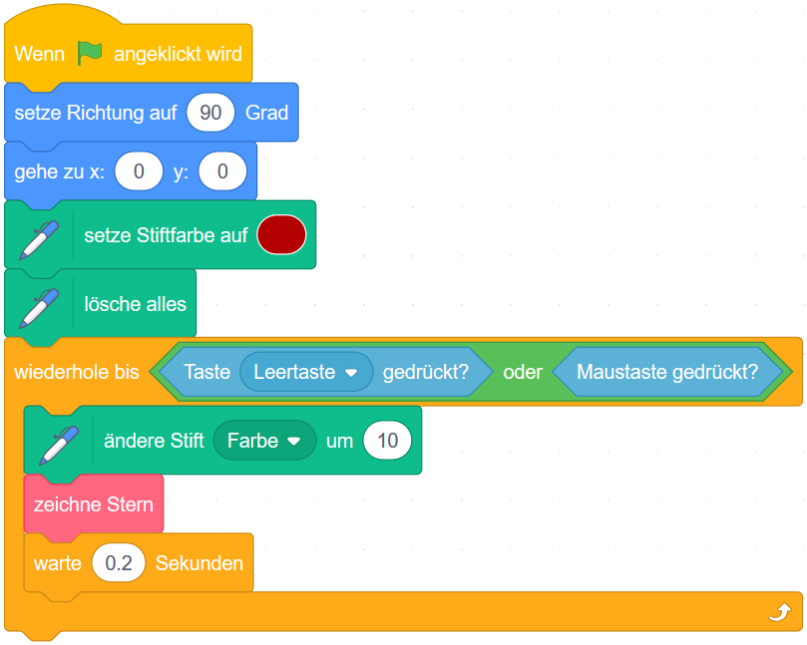
Dies trifft bei Bedingungen zu:

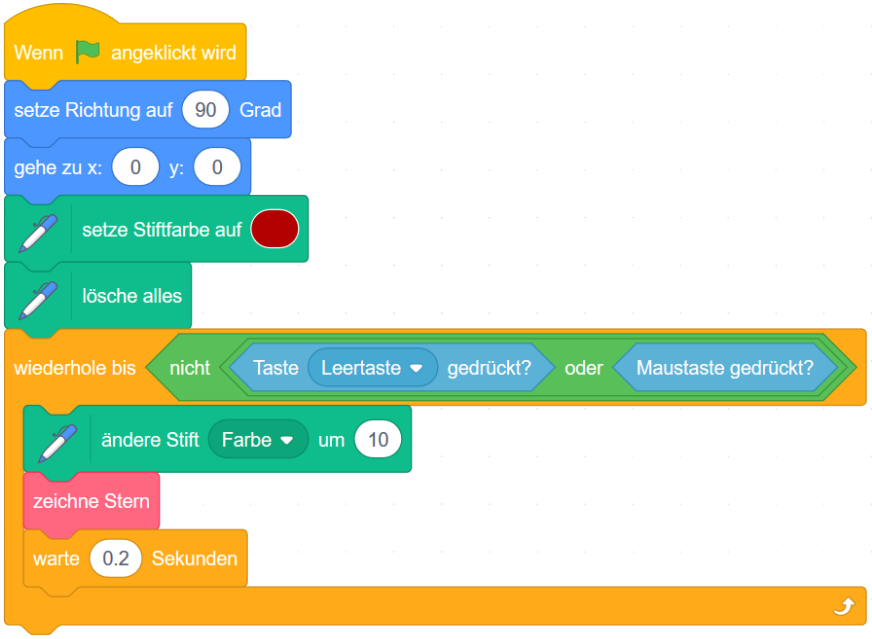
Bei der «warte bis» Anweisung: Hier wird der Programmcode nicht weiter ausgeführt, bis die Aussage «wahr» ist.

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

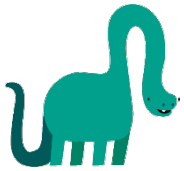
Mit einem «nicht» Block kann die «wiederhole bis» Schleife zur «wiederhole solange» Schleife umfunktioniert werden.

Programmierkonzepte – Anwendung von Boolescher Algebra





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

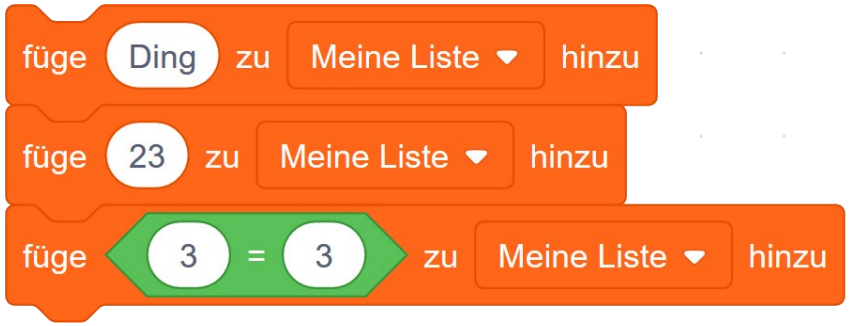
- Ganzzahl (Integer), z.B. [0, -4, 67, 200, …]
- Gleitkommazahl (Float), z.B. [0.1, -27.8, ...] mit einer Genauigkeit von 6 Stellen hinter dem Komma.

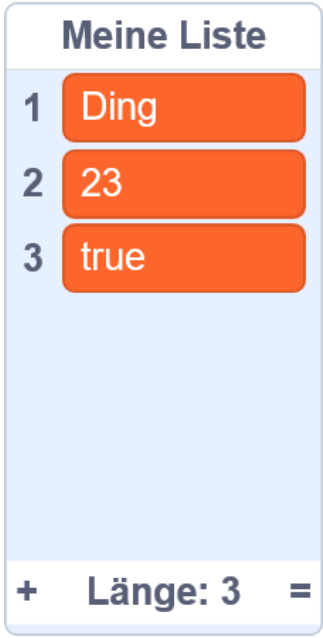
Boolesch (Boolean), dargestellt als Polygon

Zeichenkette (String), dargestellt als rundes Textfeld

Programmierkonzepte – Datentypen

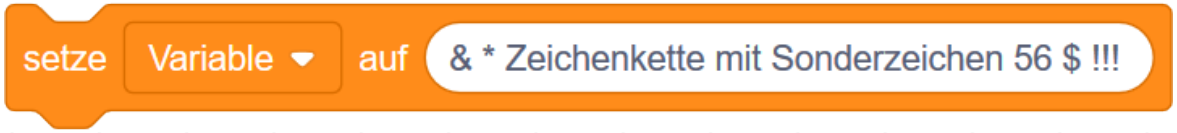
Es ist auch möglich, verschieden lange Listen (aus Menü «Variablen») mit unterschiedlichen Datentypen zusammenzustellen:

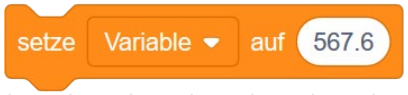


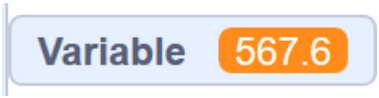


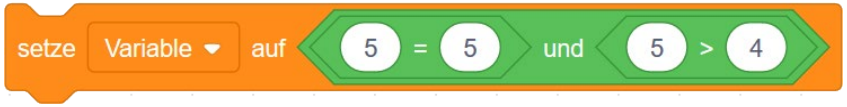
Programmierkonzepte – Datentypen

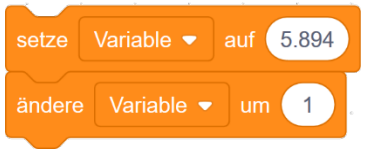
Zum Zeitpunkt der Zuordnung eines Wertes einer Variablen mittels der Anweisung «setze Variablenname auf» wird der Datentyp festgelegt. Dabei kann im rechteckigen Textfeld eine Zahl, eine Zeichenkette oder eine boolesche Aussagenlogik eingegeben werden.



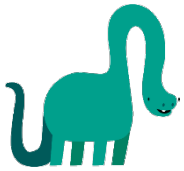




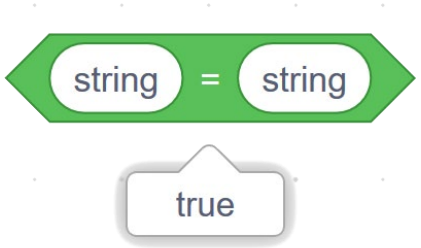


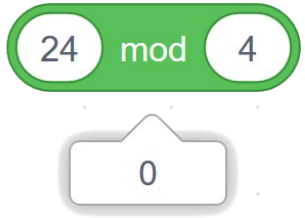


Programmierkonzepte – Datentypen



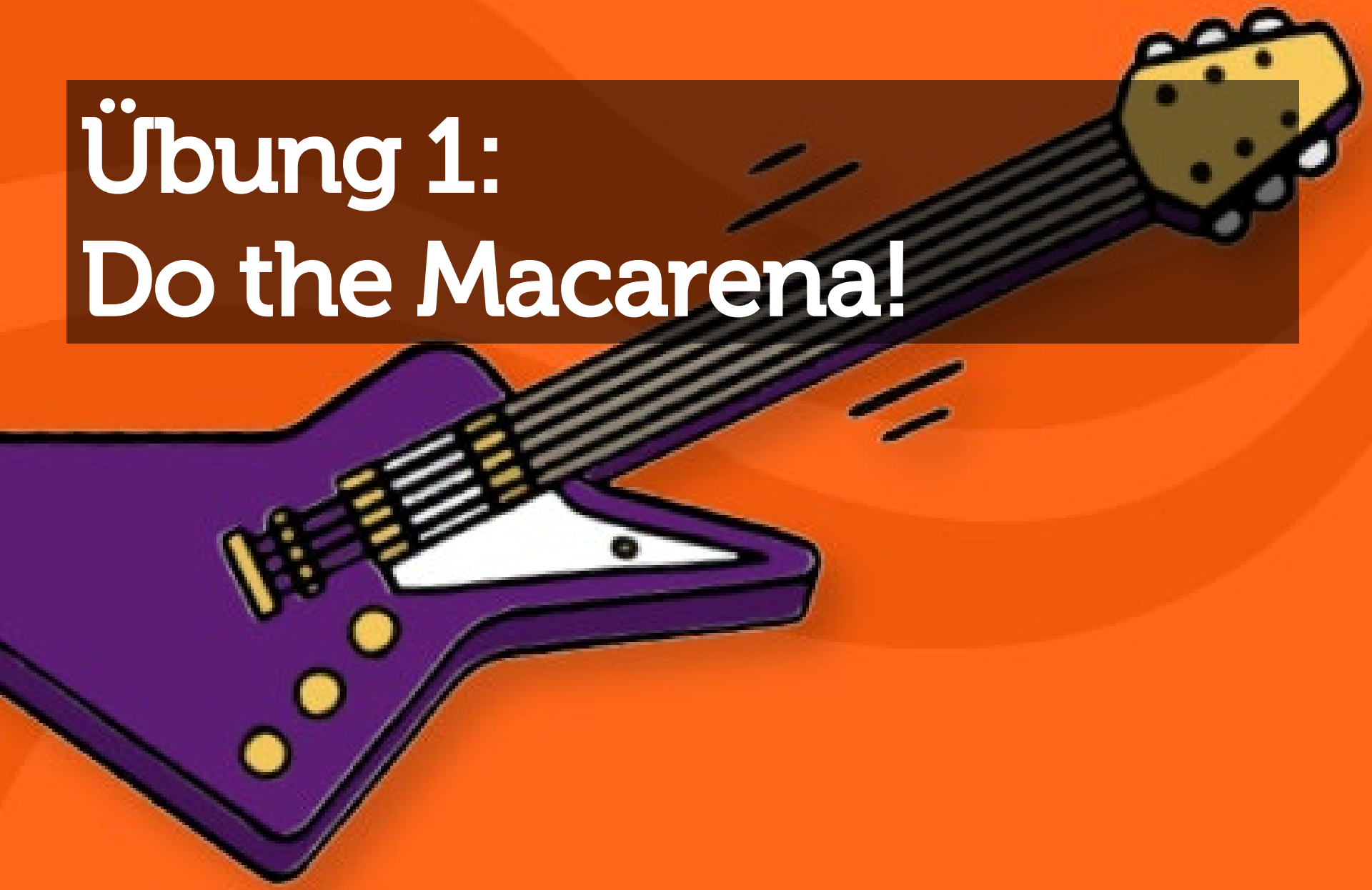
Beim Umgang mit unterschiedlichen Datentypen muss man sich immer überlegen, welcher Wertebereich der Datentyp besitzt und was für ein Datentyp als Ergebnis einer Operation erwartet wird.





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

Übung 1:
Do the Macarena!



Übung: Do the Macarena!

Auftrag: Programmiere in Scratch den Tanz Macarena. Nutze dabei möglichst Schleifen, Parameter, Variablen und Unterprogramme.

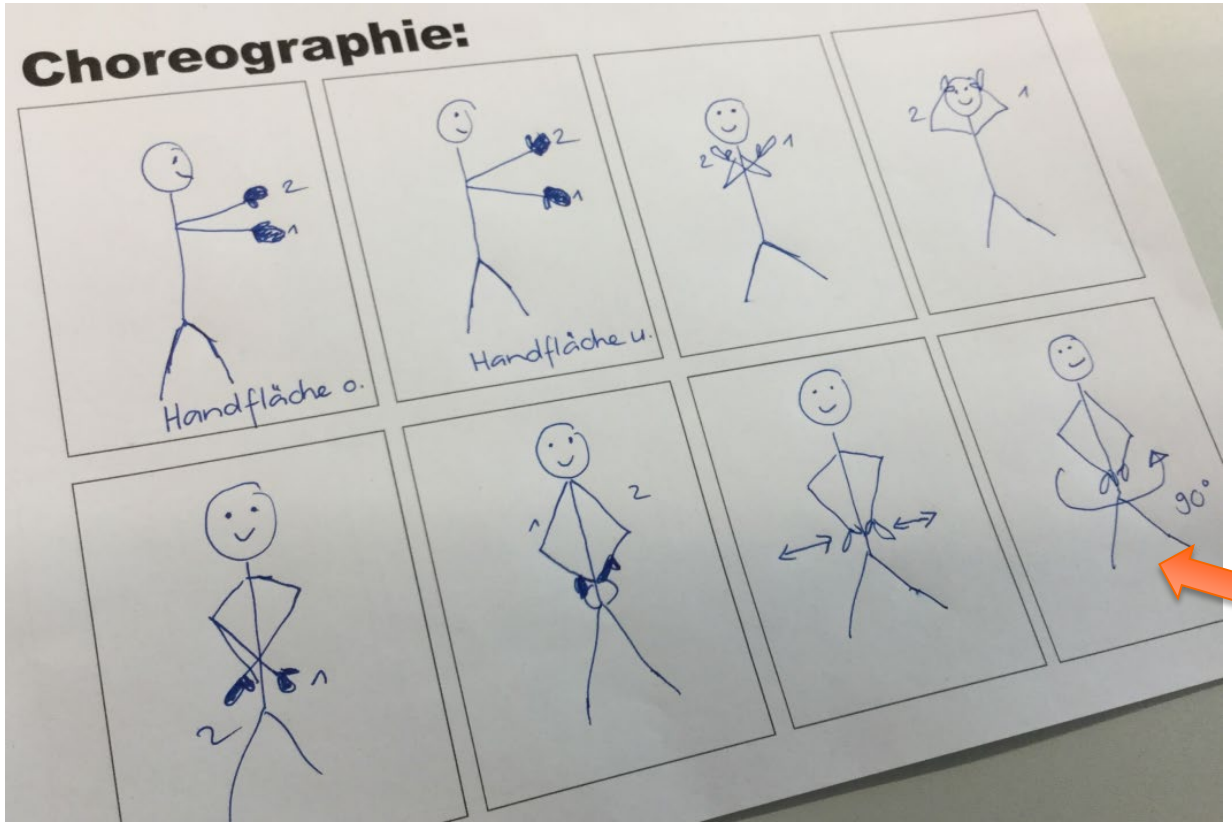
Idee: Strukturiertes Vorgehen bei der Umsetzung einer Projektidee mittels Dekomposition, Mustererkennung und Abstraktion am Beispiel Tanz.

Musikvideo: https://youtu.be/zWaymcVmJ-A

Tanztutorial: https://youtu.be/OzV63IRR8BQ

Dekomposition des Tanzes

Schritt 1: Studiere das Macarena Tanztutorial. Notiere die wichtigsten Tanzpositionen.



Dekomposition

in

Anweisung

Sequenz

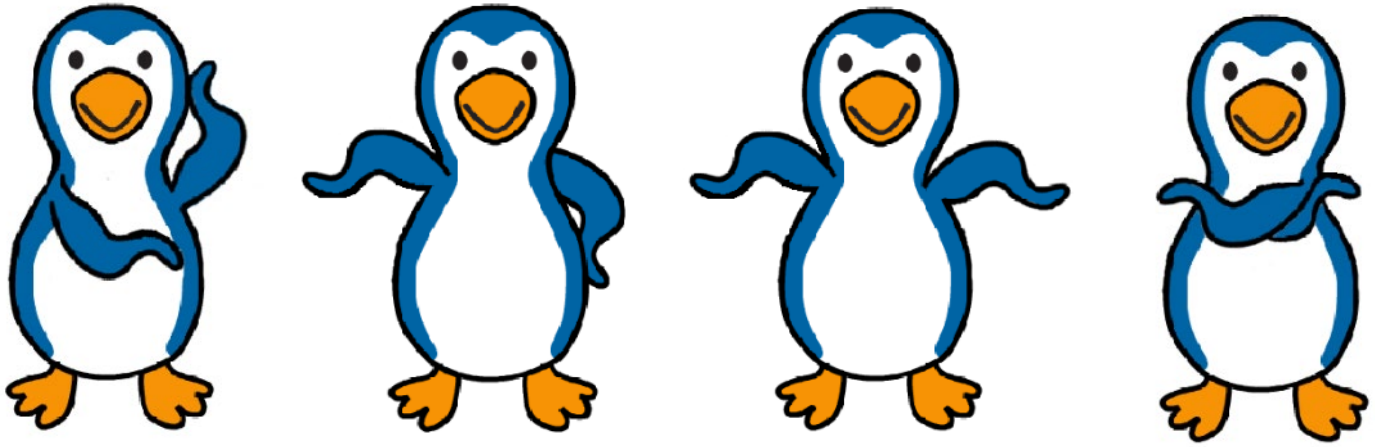
Diese Anweisungen sind mehrdeutig für den Menschen und nicht ausführbar für einen Computer!

Macarena als ausführbares Programm formulieren

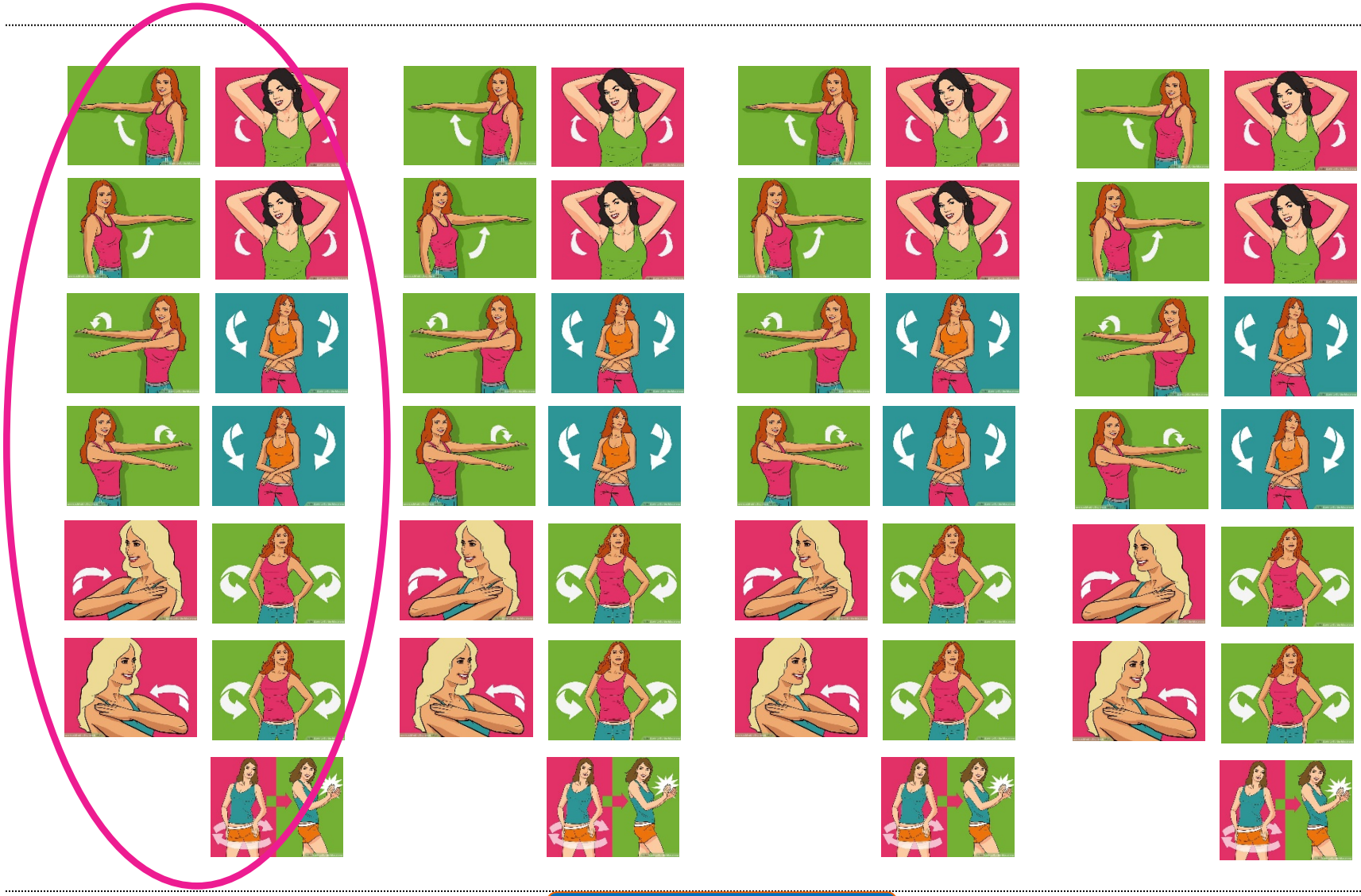
Schritt 2: Abstrahiere die wichtigsten Details der Tanzpositionen. Was haben die verschiedenen Tanzpositionen gemeinsam? Wo unterscheiden sie sich? Wie ist die Symmetrie? Suche Wiederholungen und Regelmässigkeiten und notiere diese.

Abstraktion

Mustererkennung



Den ganzen Macarena Tanz betrachten

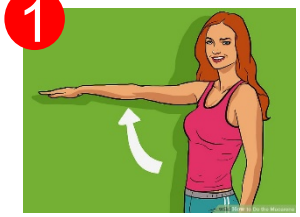


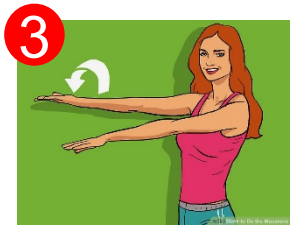
phsz

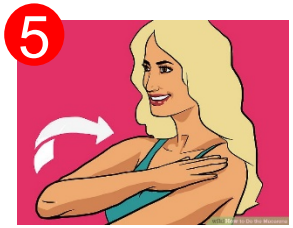
4x Wiederholung

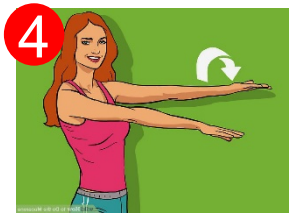
Mustererkennung

Erste Teilsequenz betrachten





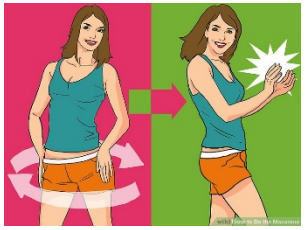




Rechts	Rechts	Rechts	Rechts	Rechts	Rechts
Links	Links	Links	Links	Links	Links
Parallel	Parallel	Gekreuzt	Parallel	Gekreuzt	Parallel

Abstraktion

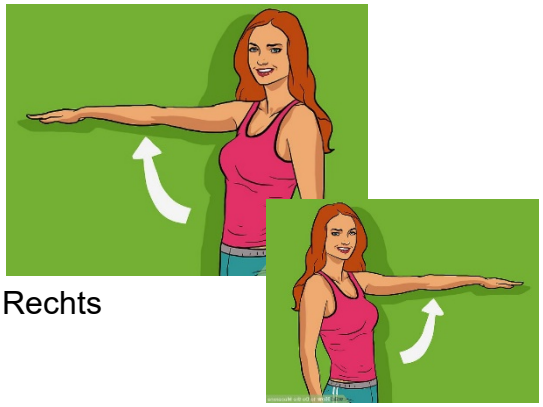
Mustererkennung



Spezialfall

Tanzpositionen in formellerer Sprache formulieren

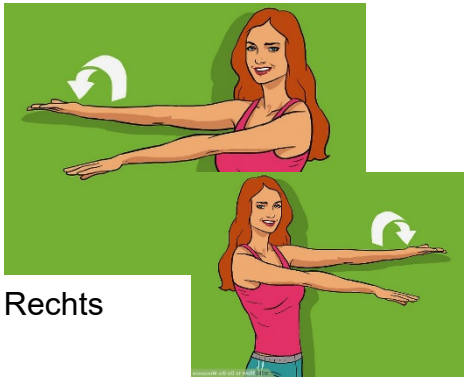
Schritt 3: Formuliere die Tanzpositionen in einer formelleren Sprache, sodass z.B. ein Mensch die Positionen durch mündliche Anweisungen nachmachen könnte. Grösste Gemeinsamkeit aller Tanzpositionen: die Handflächen zeigen immer in eine Richtung oder auf ein Körperteil. So wird auch formuliert:



Rechts

Links

Rechte/linke Handfläche zum Boden zeigen
Rechte/linke Handfläche auf Schulterhöhe
Rechte/linke Handfläche parallel zum Körper
Rechte/linke Handfläche vor Körper

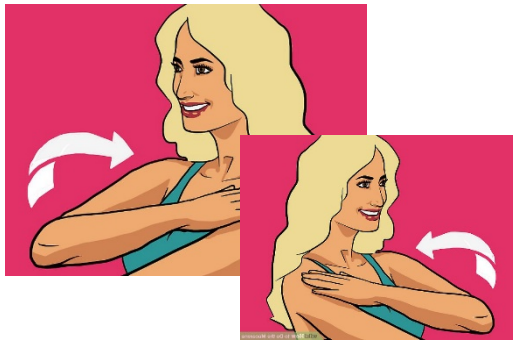


Rechts

Links

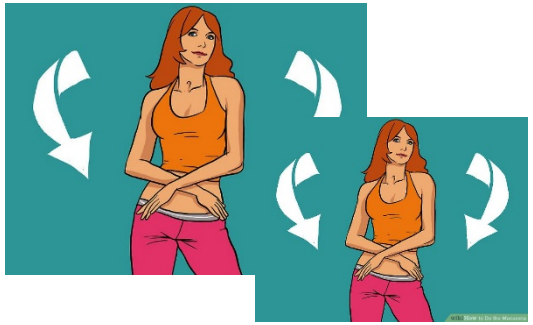
Rechte/linke Handfläche zum Himmel zeigen
Rechte/linke Handfläche auf Schulterhöhe
Rechte/linke Handfläche parallel zum Körper
Rechte/linke Handfläche vor Körper

Tanzpositionen in formellerer Sprache formulieren

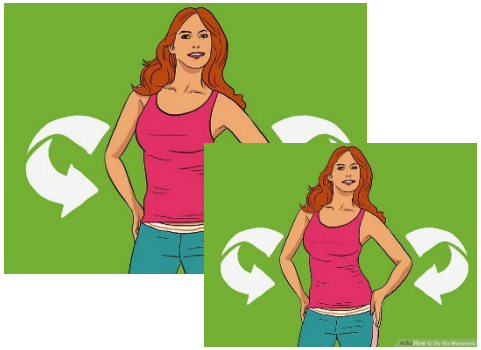


Rechte/linke Handfläche auf Schulter legen
Rechte/linke Handfläche gekreuzt zum Körper

Rechte/linke Handfläche zum Kopf
Rechte/linke Handfläche parallel zum Körper

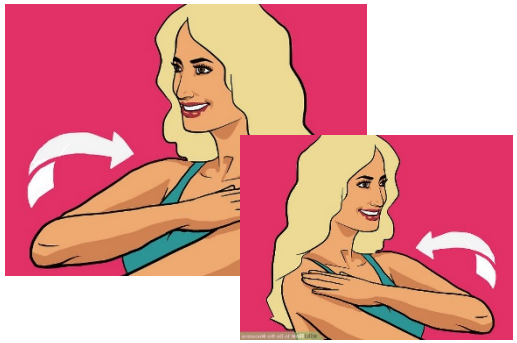


Rechte/linke Handfläche auf Hüfte legen
Rechte/linke Handfläche gekreuzt zum Körper



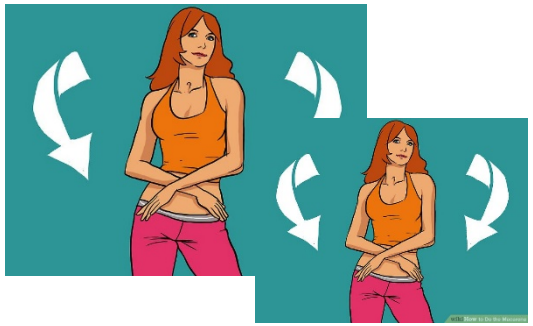
Rechte/linke Handfläche auf Po legen
Rechte/linke Handfläche parallel zum Körper

Anweisungen Formulieren

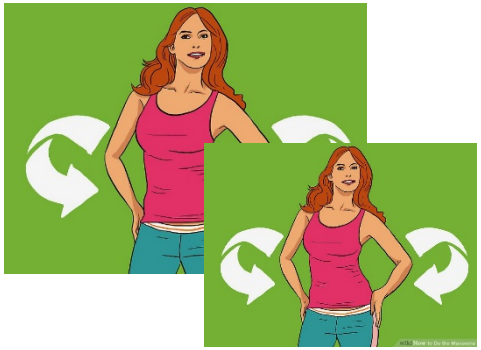


Rechte/linke Handfläche auf Schulter legen
Rechte/linke Handfläche gekreuzt zum Körper

Rechte/linke Handfläche zum Kopf
Rechte/linke Handfläche parallel zum Körper

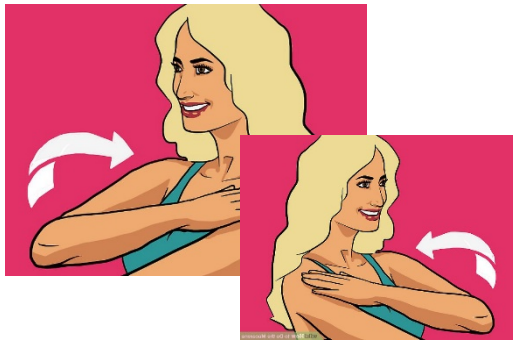


Rechte/linke Handfläche auf Hüfte legen
Rechte/linke Handfläche gekreuzt zum Körper



Rechte/linke Handfläche auf Po legen
Rechte/linke Handfläche parallel zum Körper

Variablen definieren



Rechts/Links Handfläche auf Ziel

Rechts/Links Handfläche gekreuzt/parallel

Rechts/Links Handfläche zum Ziel

Rechts/Links Handfläche gekreuzt/parallel

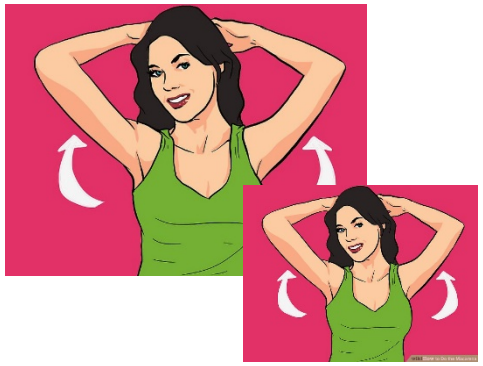
Rechts/Links Handfläche auf Ziel

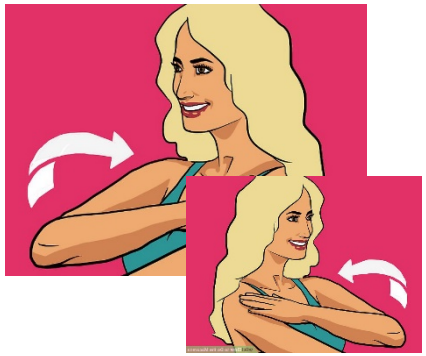
Rechts/Links Handfläche gekreuzt/parallel

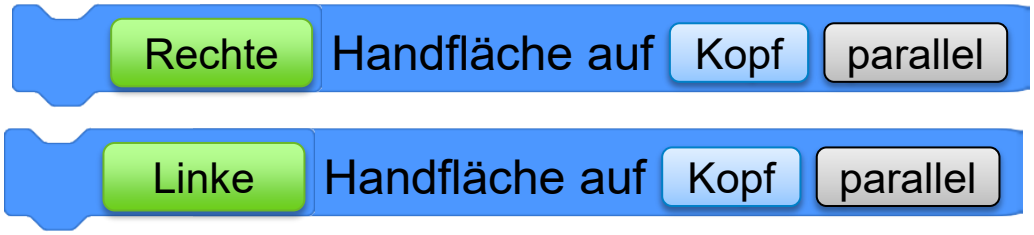
Rechts/Links Handfläche auf Ziel

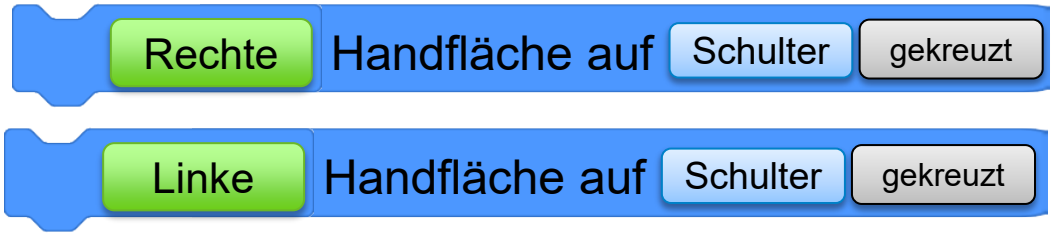
Rechts/Links Handfläche gekreuzt/parallel

Variablen definieren und als Parameter in einer Anweisung übergeben

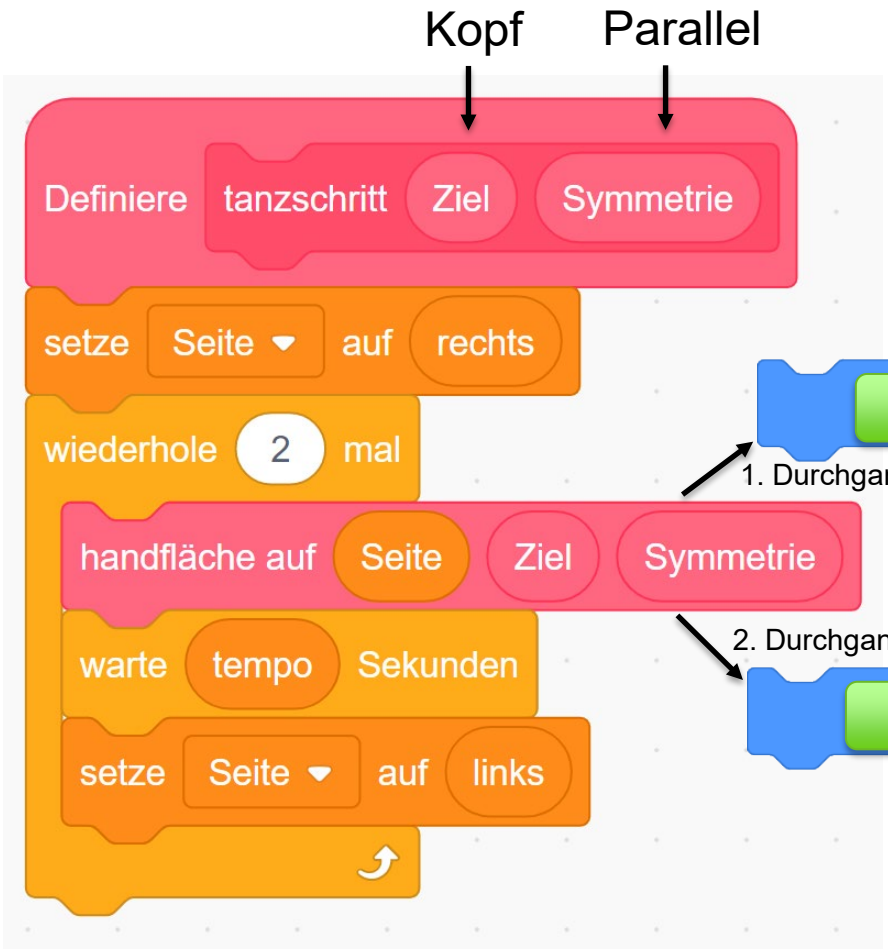




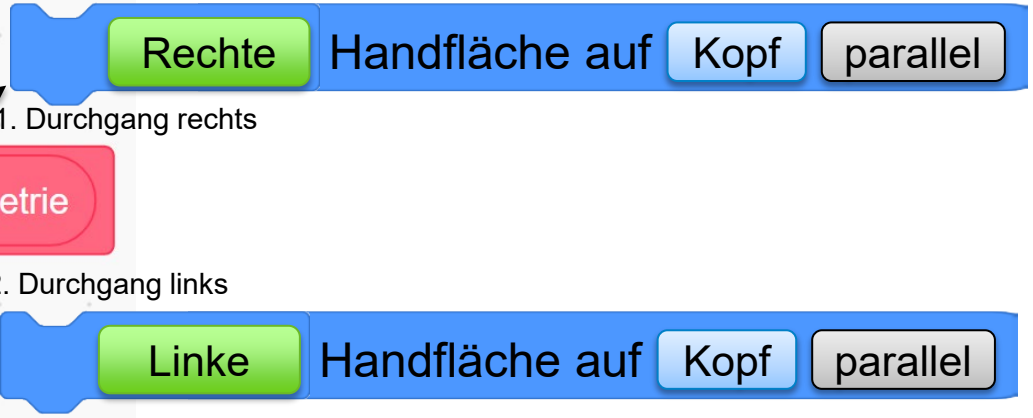




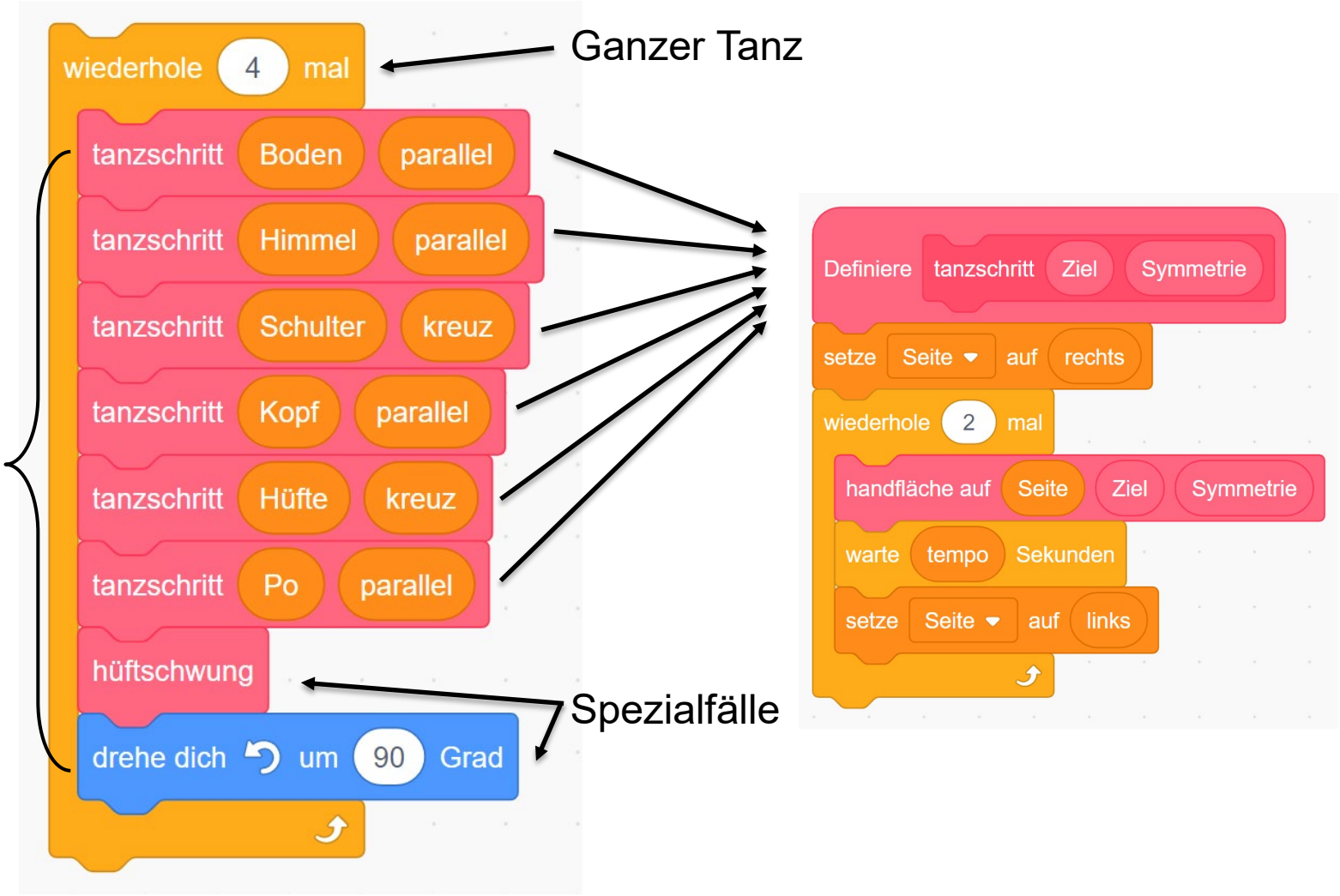
Implementiere den Tanz in Scratch



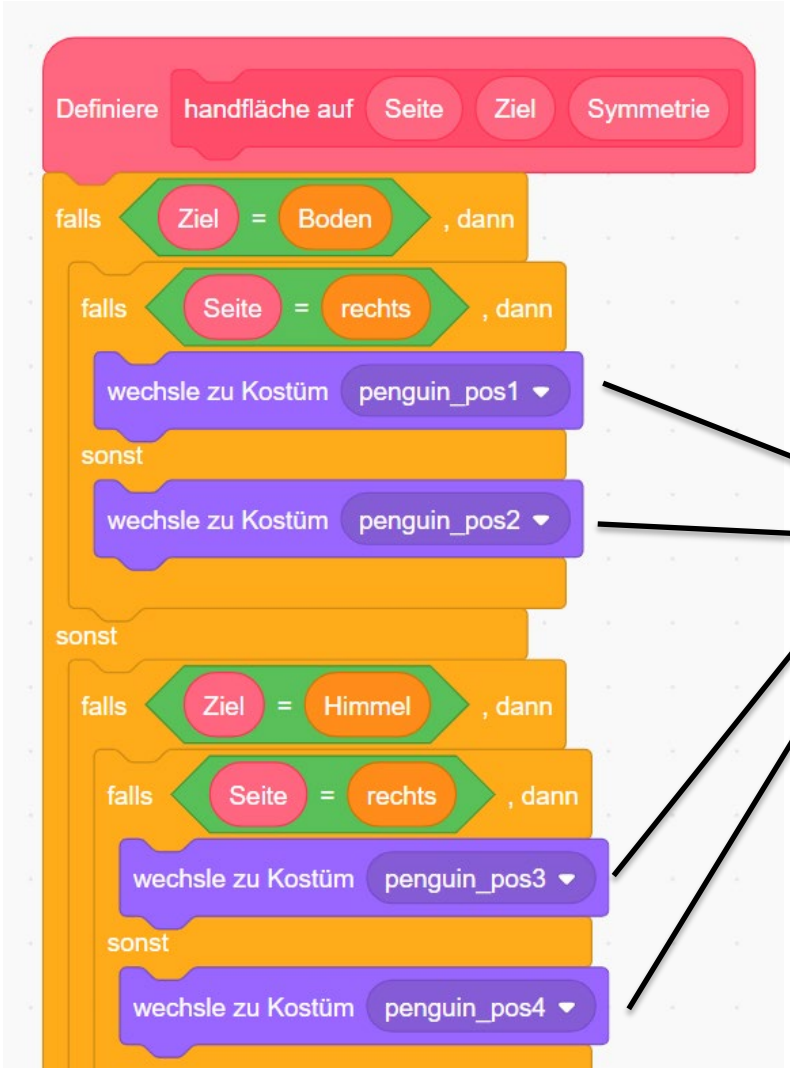
Jeder Tanzschritt hat eine rechte und linke Seite.
2 Durchgäge total.



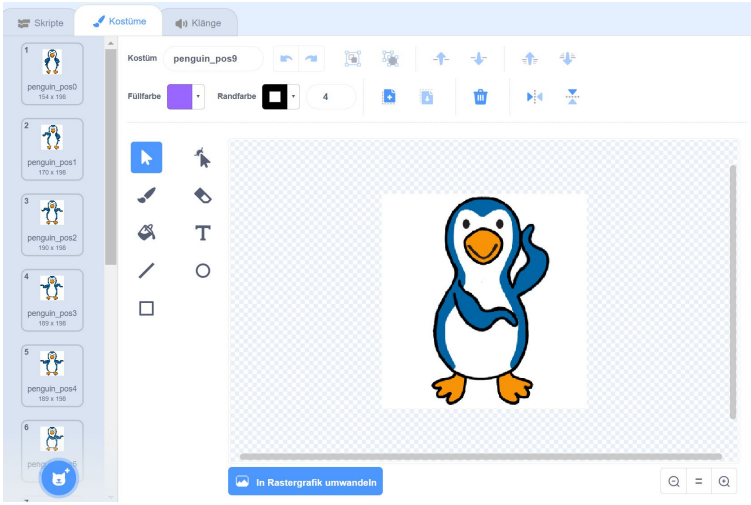
Implementiere den Tanz in Scratch



Implementiere den Tanz in Scratch

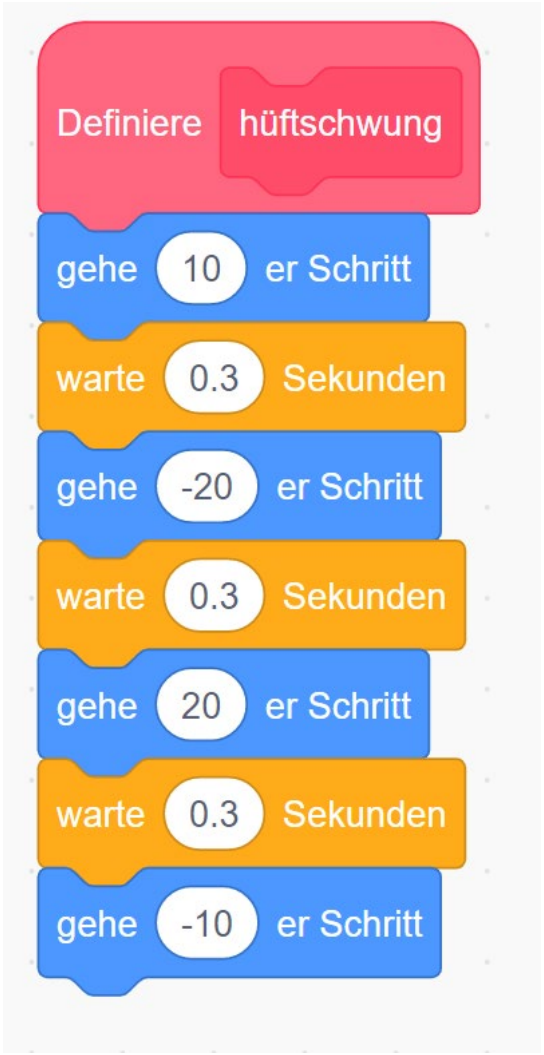


Die richtigen Kostüme je nach Tanzposition auswählen.



Etc.

Implementiere den Tanz in Scratch



Spezialfälle, wie Hüftschwung, separat implementieren.


Übung 2:
Umsetzung einer
eigenen Projektidee