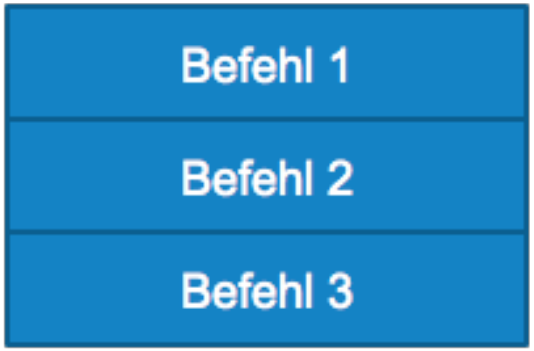
Algorithmen in der Sekundarstufe I

Am Ende der Primarstufe sollte Schüler/innen diese Grundkonzepte kennen und anwenden können:

Sequenz



Abläufe planen und ausführen

Schleife

Bedingung

wenn Wand berührt, dann Spiel verloren

Vom “Problem“ zum Programm in der Sekundarstufe

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 logische Operatoren verwenden (und, oder, nicht).

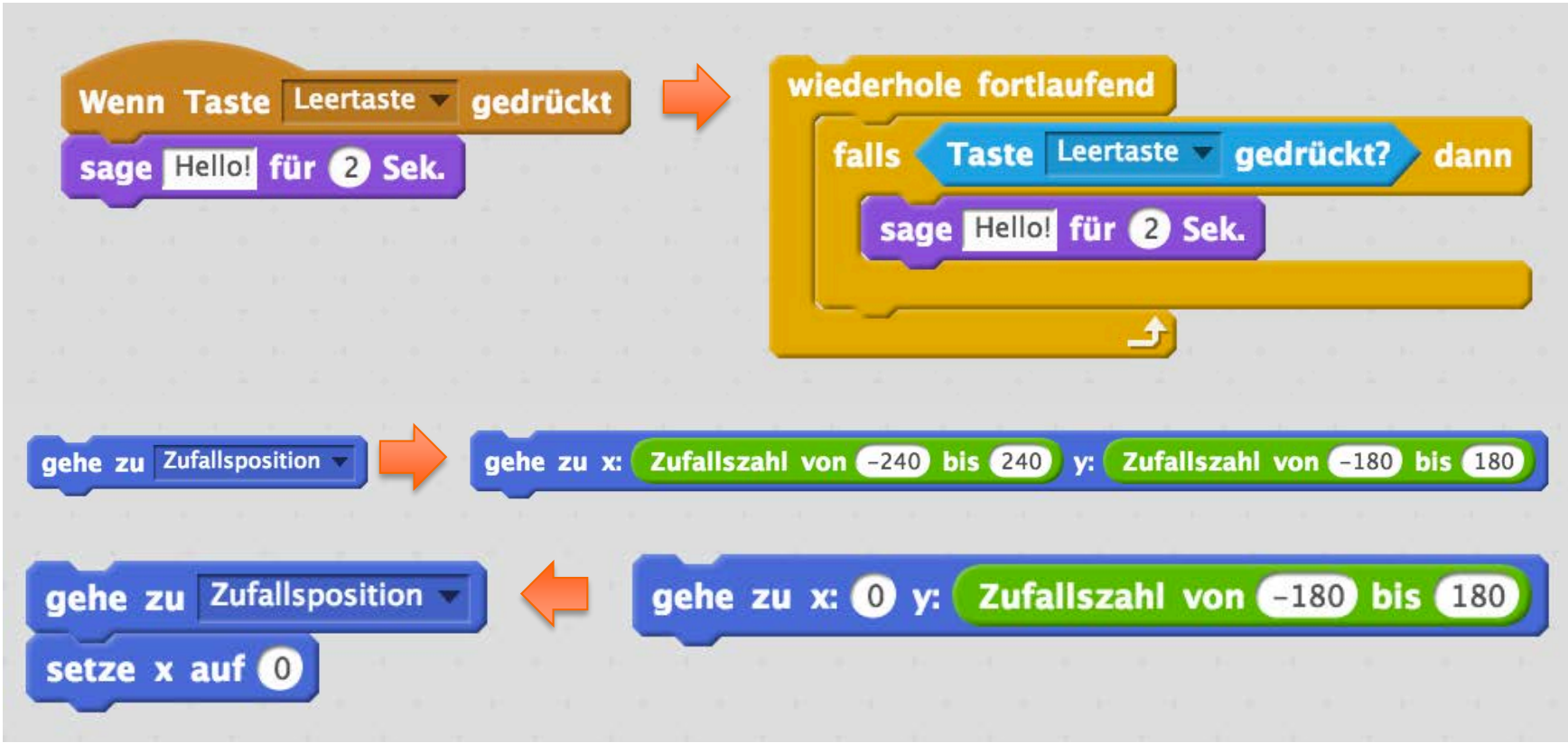
Probleme können sehr vielfältig sein:

- Wie berechne ich mit einem Programm die Fläche eines Dreiecks?
- Wie kann ich eine Figur am Bildschirm mit der Maus steuern?
- ...

→ eigenen Weg wählen, probieren und entdecken → Scratch

Abstraktionsstufen - Level of Abstraction

Scratch bietet bereits einige Blöcke an, die theoretisch aus anderen Blöcken bestehen könnten. Diese Blöcke erleichtern den Einstieg und können später durch „mächtigere“ Konstrukte ersetzt werden:



Variable - Abstraktionsstufen

Primarstufe

Variable wird implizit verwendet, aber nicht selbst definiert:

Die Variable „Antwort“ wird nicht selbst gesetzt, sondern automatisch über den „frage“-Block. Die „Antwort“ ist ein Block, das Prinzip der Variable wird aber nur gestreift.

Variable - Abstraktionsstufen

Primarstufe

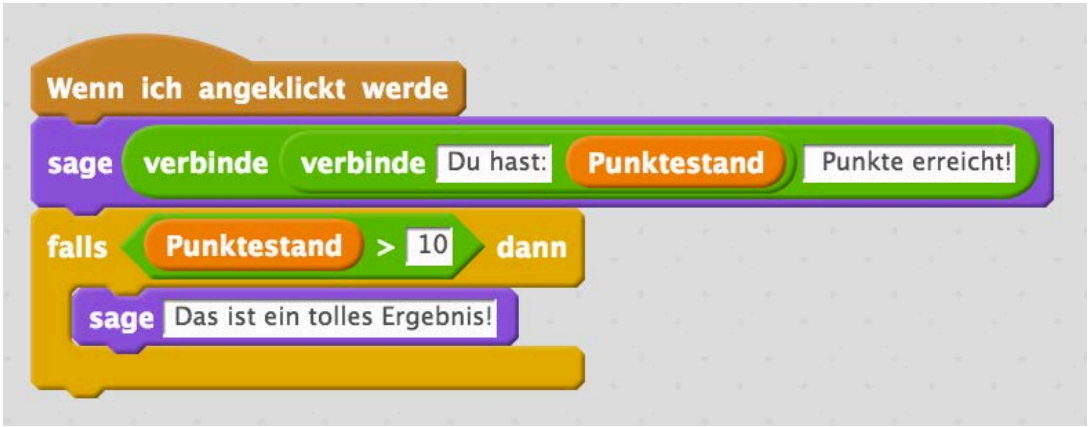
Variable wird selbst definiert und als einfacher Zähler verwendet:

Die Variable „Punktestand“ wird selbst gesetzt und verändert. Sie wird jedoch noch nicht als Platzhalter in andere Blöcke eingesetzt.

Variable - Abstraktionsstufen

Sekundarstufe I

Variable wird selbst definiert und als Platzhalter in andere Blöcke eingesetzt:

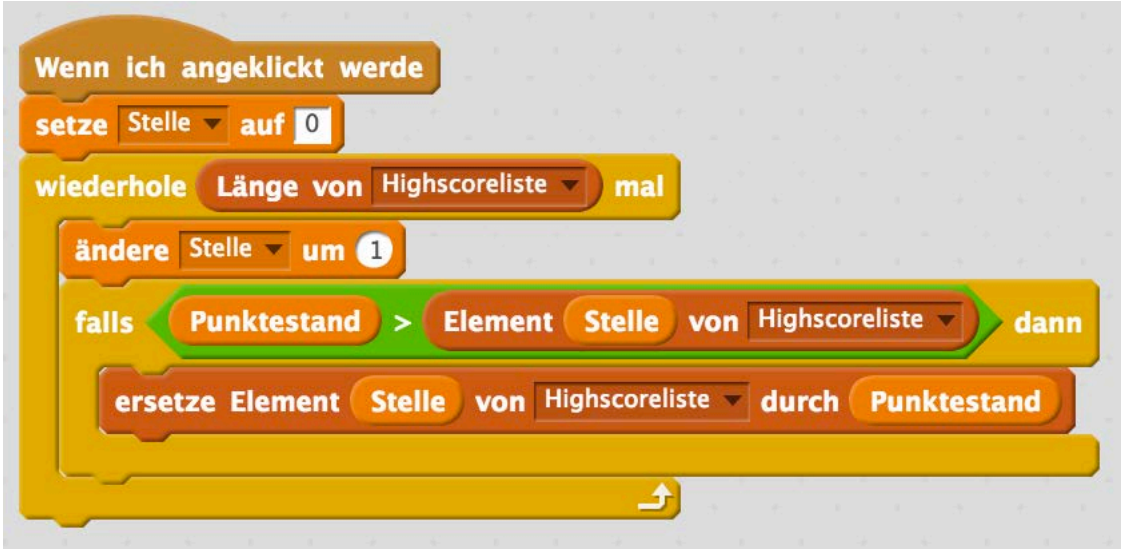


Hierfür braucht man ein Verständnis von der Variable als „Schachtel“, die zu unterschiedlichen Zeitpunkten einen unterschiedlichen Wert haben kann.

Variable - Abstraktionsstufen

Sekundarstufe I

Listen = Sammlung mehrerer Variablen

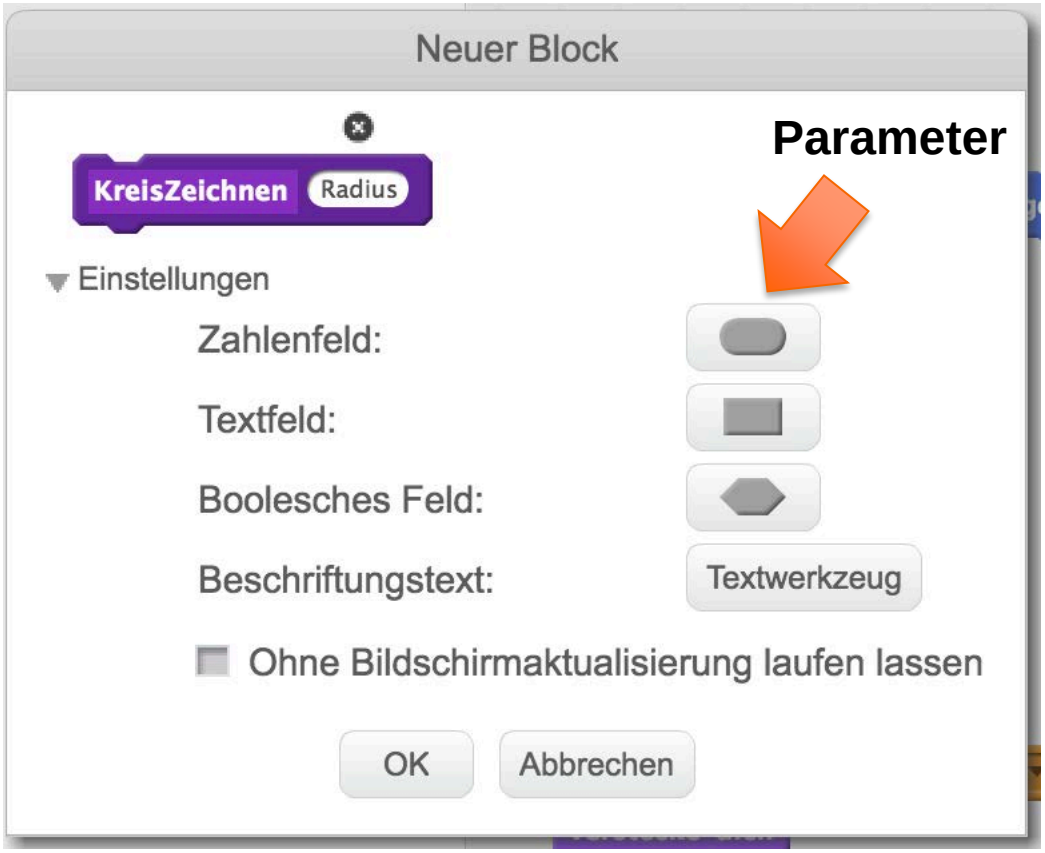


Variable als Laufindex zum Suchen von Einträgen in der Liste. Zum Beispiel zum Ersetzen eines Eintrags an einer Stelle in der Liste.

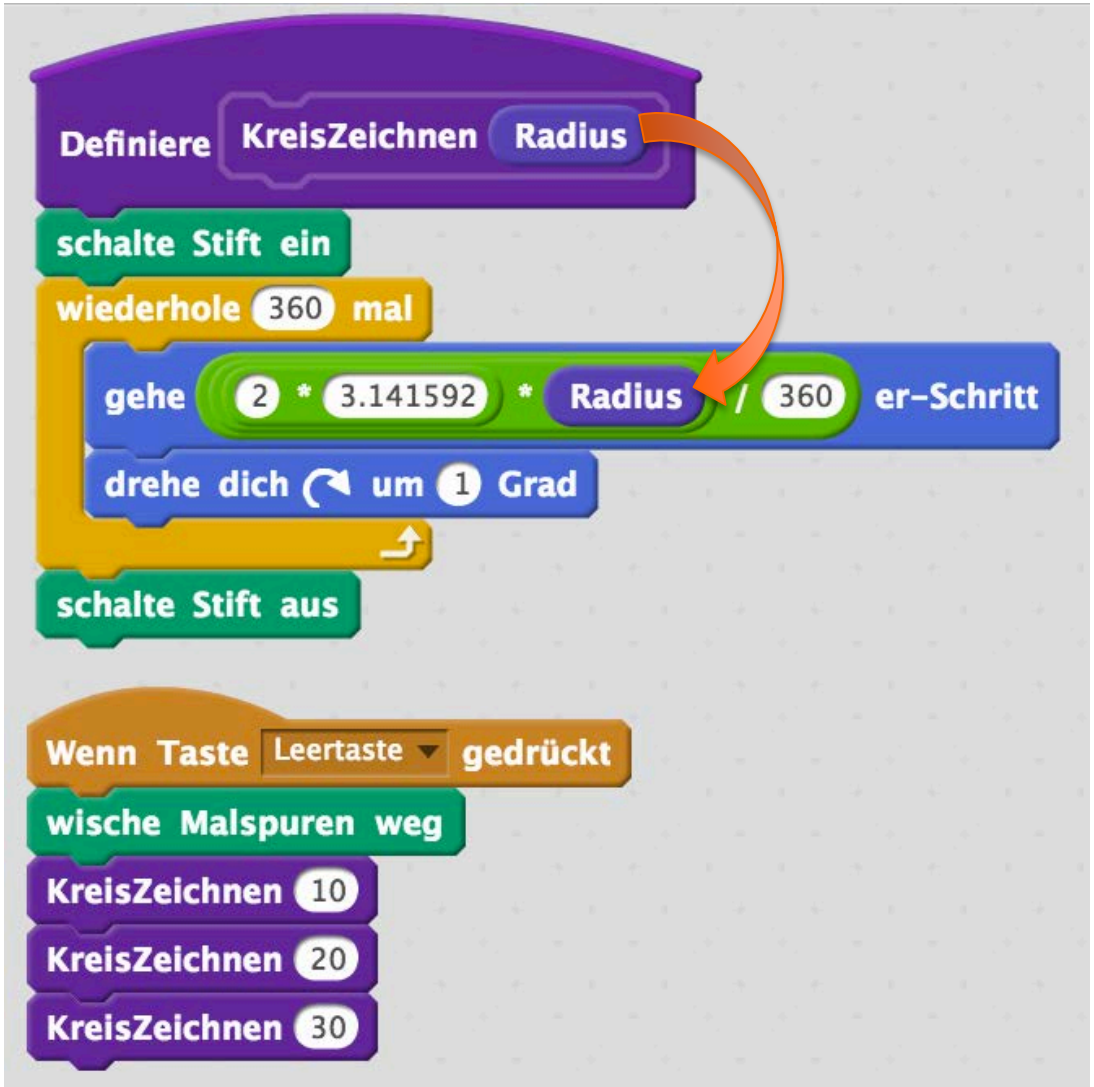
Konzept vom Unterprogramm

- Prozeduren, Funktionen, **Unterprogramme** sind ein **Mittel der Abstraktion**:
  - *Wiederverwendung* – Es gibt im Projekt mehrere Stellen, wo die gleichen Befehle genutzt werden. An einer Stelle zusammenfassen = nur einmal korrigieren/ändern müssen.
  - *Schnittstelle* – Implementation unabhängig vom Rest möglich, solange Schnittstelle eingehalten wird (Blackbox-Prinzip).
  - *Parametrisierung* – ein Unterprogramm definiert mit Hilfe von Parametern eine ganze Klasse von Problemen und nicht nur einen spezifischen Fall.

Unterprogramm = eigener Block in Scratch



Eigene Blöcke - Beispiel

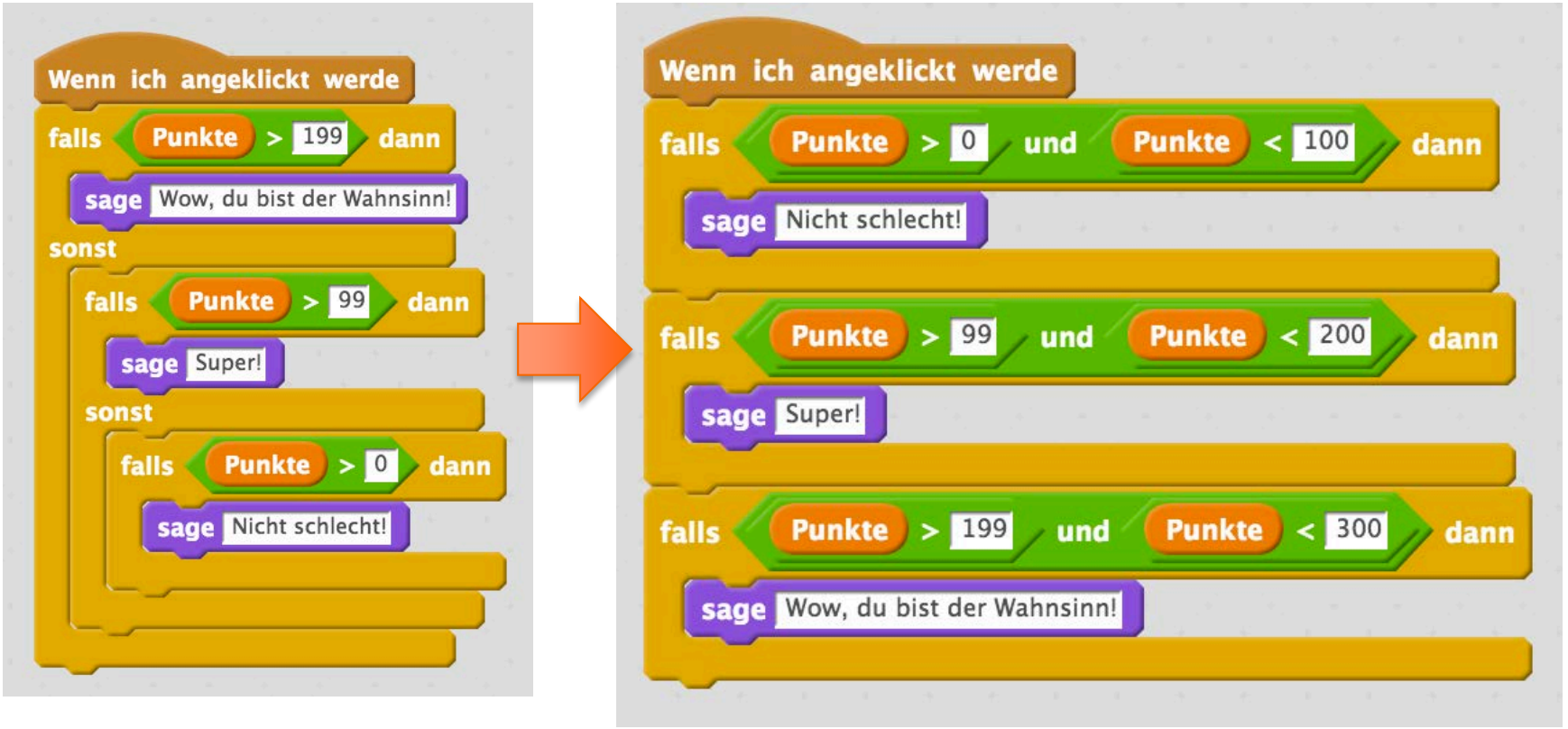


lokale Variable „Radius“

$$U = 2 \cdot \pi \cdot r$$

Logische Operatoren UND, ODER, NICHT verwenden

Logische Operatoren verbinden Wahrheitswerte.



Logische Operatoren UND, ODER, NICHT verwenden

falls (Leertaste gedrückt UND (NICHT (wird Farbe ... berührt))), dann:

(NICHT ((...) ODER (...)))

Logische Operatoren UND, ODER, NICHT verwenden

Bei sehr viel Kombination wird die Erstellung des Programms fehleranfällig und schwer zu lesen

Beispiel ist keine Zielstellung
für die Sekundarstufe I

```
Wenn ich UPDATE PLAYER empfange
falls die = false und Lives > 0 dann
    falls nicht Friction = 0 und nicht Taste Pfeil nach oben gedrückt? dann
        setze VelX auf VelX * Friction
        setze VelY auf VelY * Friction
    falls Taste Pfeil nach links gedrückt? dann
        ändere VelR um Turn Speed * -1
        falls -9 > VelR dann
            setze VelR auf -9
```

https://scratch.mit.edu/projects/151213207

Algorithmen vergleichen: Sortieren und Finden

» können verschiedene Algorithmen zur Lösung desselben Problems vergleichen und beurteilen (z.B. lineare und binäre Suche, Sortierverfahren).

Beispiel: Lineare Suche vs. Binär-Suche unplugged

Wörterbuch oder Lexikon (= vorsortierte Einträge) mitbringen.

Beispiel: Lineare Suche vs. Binär-Suche unplugged

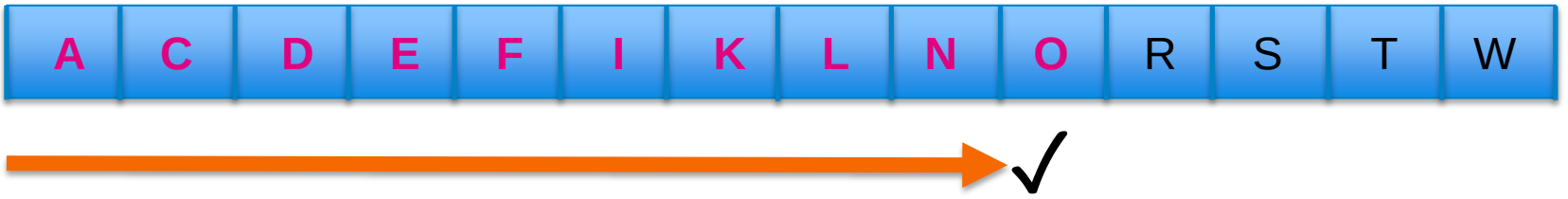
Vorgehen – unplugged Beispiel:

- SuS nennen einen zufälligen Begriff – zum Beispiel *Sonne*.
- LP fängt auf der ersten Seite im Buch an und fährt mit dem Finger über die Seite (zeilenweise, jeden Eintrag abfahren). Etwas Ausdauer zeigen ☺
- SuS sollen merken, dass dieses Suchverfahren ineffizient ist und viel zu lange dauert.
- Vorschläge sammeln, wie man das gesuchte Wort schneller finden könnte. → Sortierung hilft – wir können bereits zum Anfangsbuchstaben S springen.
- Bei Anfangsbuchstaben S erneut beginnen mit dem Finger zeilenweise abzufahren. Erneut ineffizient, der zweite Buchstabe ist aber nicht direkt über ein Register ablesbar im Buch
- → die Kinder sollen auf die Idee kommen, in die Mitte von „S“ zu blättern und dann zu schauen ob sie vorher oder nachher weitersuchen müssen.

Beispiel – lineare Suche

Bei der **linearen Suche** beginnt man am Anfang der Liste und schaut jedes Element an, bis man das gesuchte gefunden hat.

Beispiel Such nach dem Eintrag „O“



10 Einträge wurden angeschaut und verglichen

Eine Liste mit 10‘000 Einträgen:

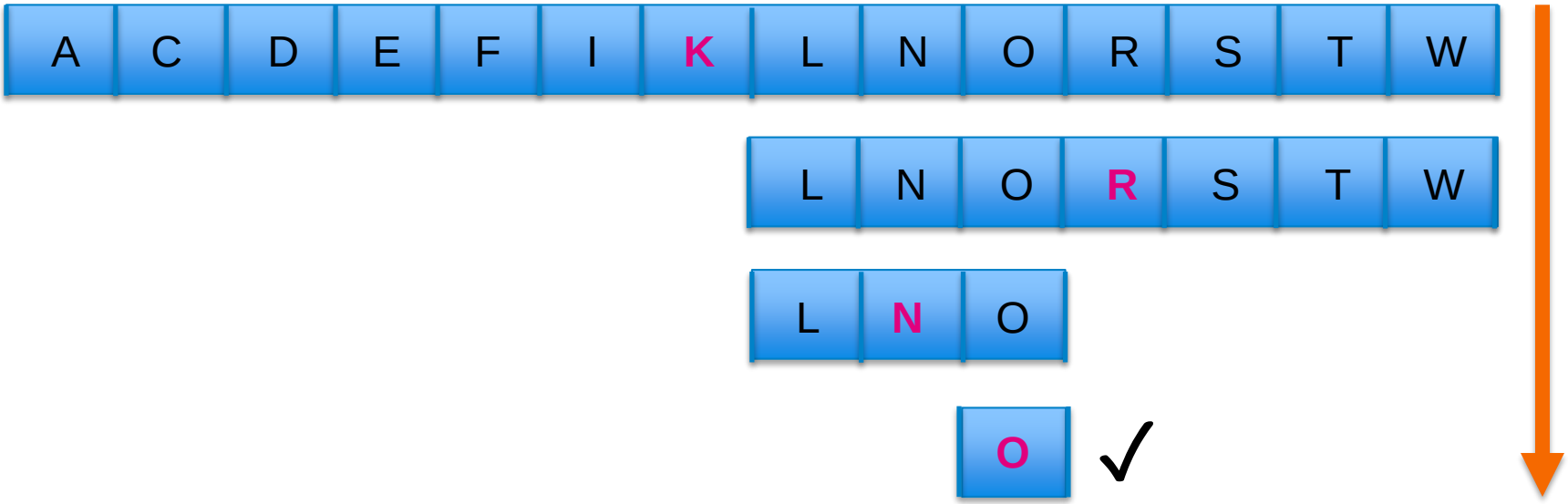
- Wie viele Einträge müssten wir im schlimmsten Fall anschauen?
- Wie viele Einträge müssten wir im besten Fall anschauen?
- Wie viele Einträge müssten wir durchschnittlich anschauen?

Beispiel – Binärsuche

Bei der **Binärsuche** teilt man die vorsortierte Liste immer in zwei Hälften.

→ binäre Entscheidung, ob man mit der einen oder anderen Hälfte vorsetzt.

Beispiel Such nach dem Eintrag „O“



4 Einträge wurden angeschaut und verglichen

Eine Liste mit 10‘000 Einträgen? schlimmsten Fall, bester Fall, durchschnittlich?

Beispiel – lineare Suche vs. Binärsuche

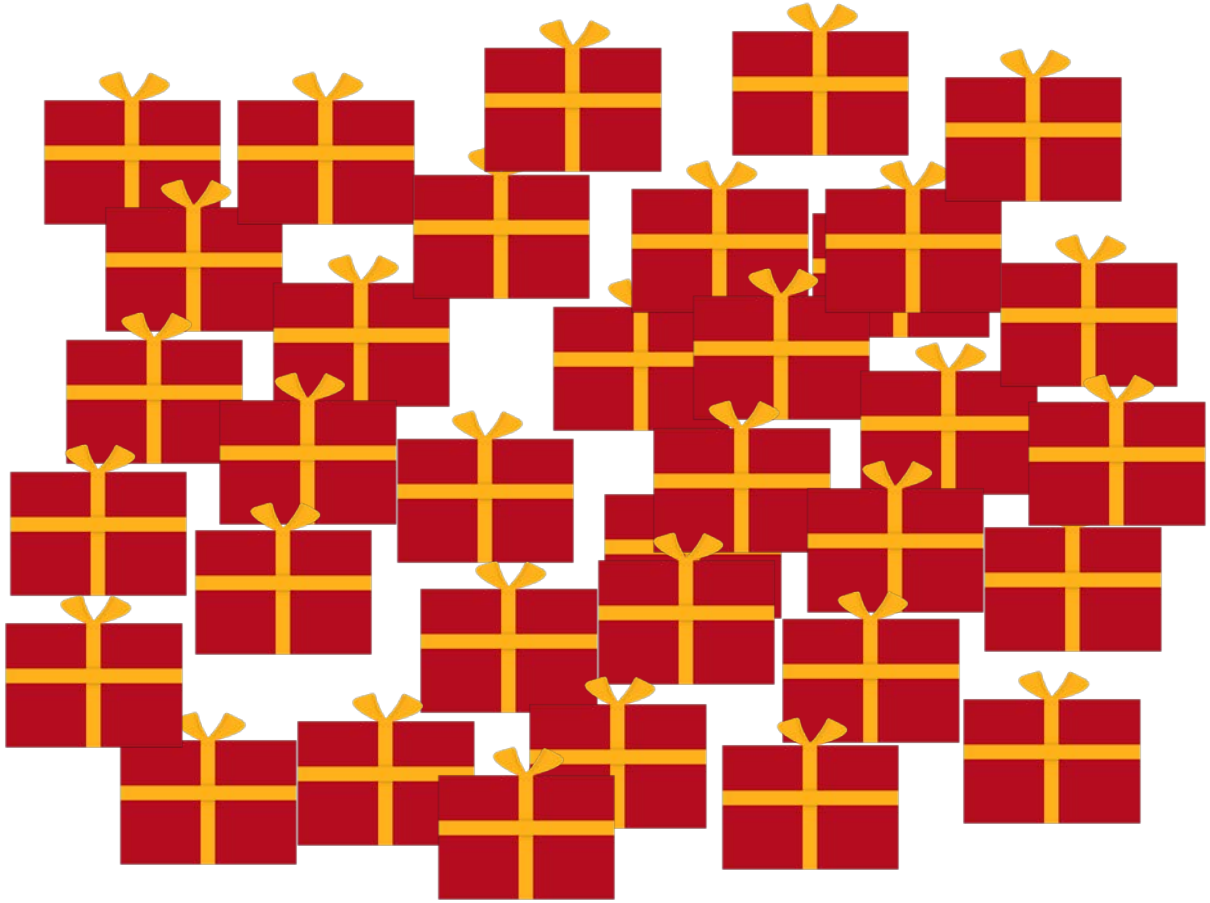
Erkenntnisse:

- Obwohl beide Algorithmen genau das gleiche Ergebnis (den Eintrag „O“) liefern, sind sie unterschiedlich schnell → effizient
- Als Nutzer sehen wir einem Programm in der Regel nicht an, welche Algorithmen eingesetzt werden, wir sehen nur das Ergebnis „O“.

Algorithmen können auch in der realen Welt helfen → Beispiel

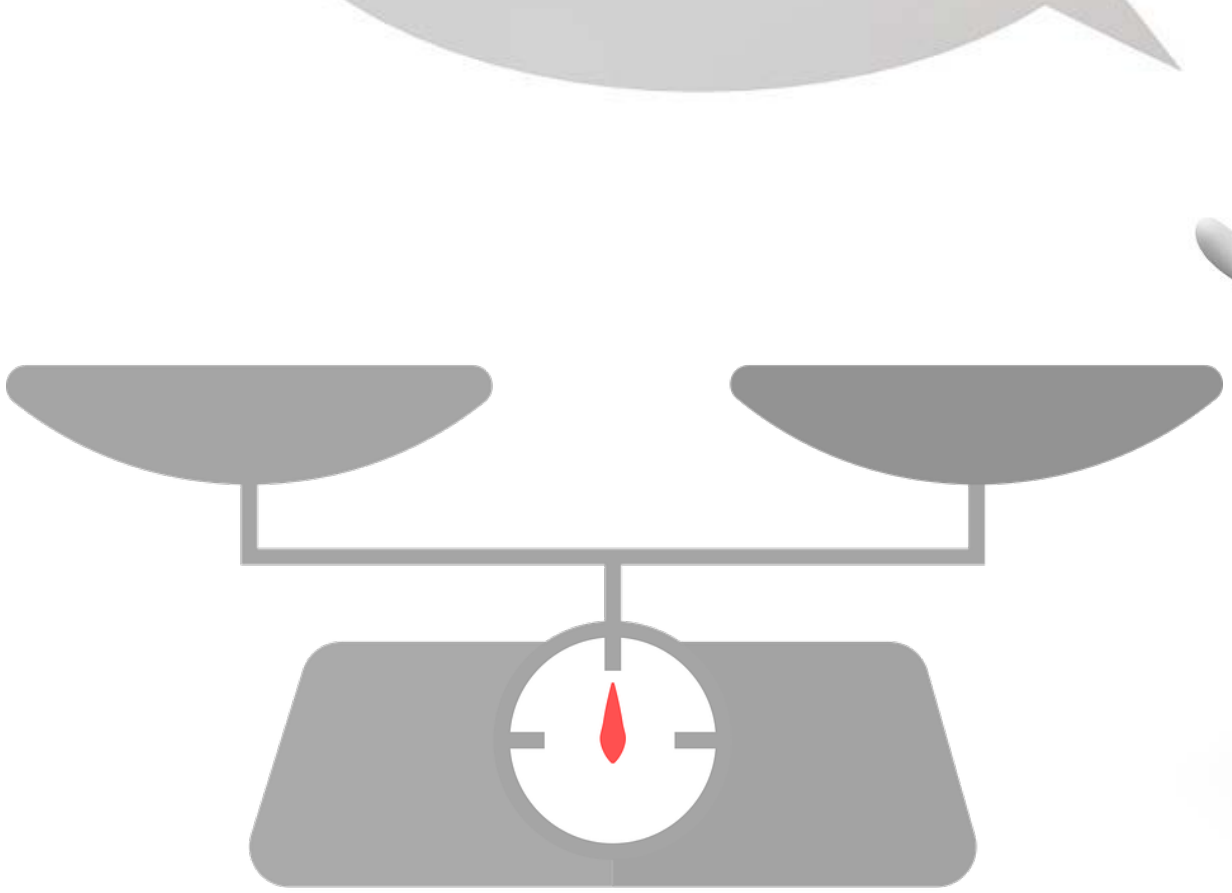
Binärsuche - Anwendungsbeispiel

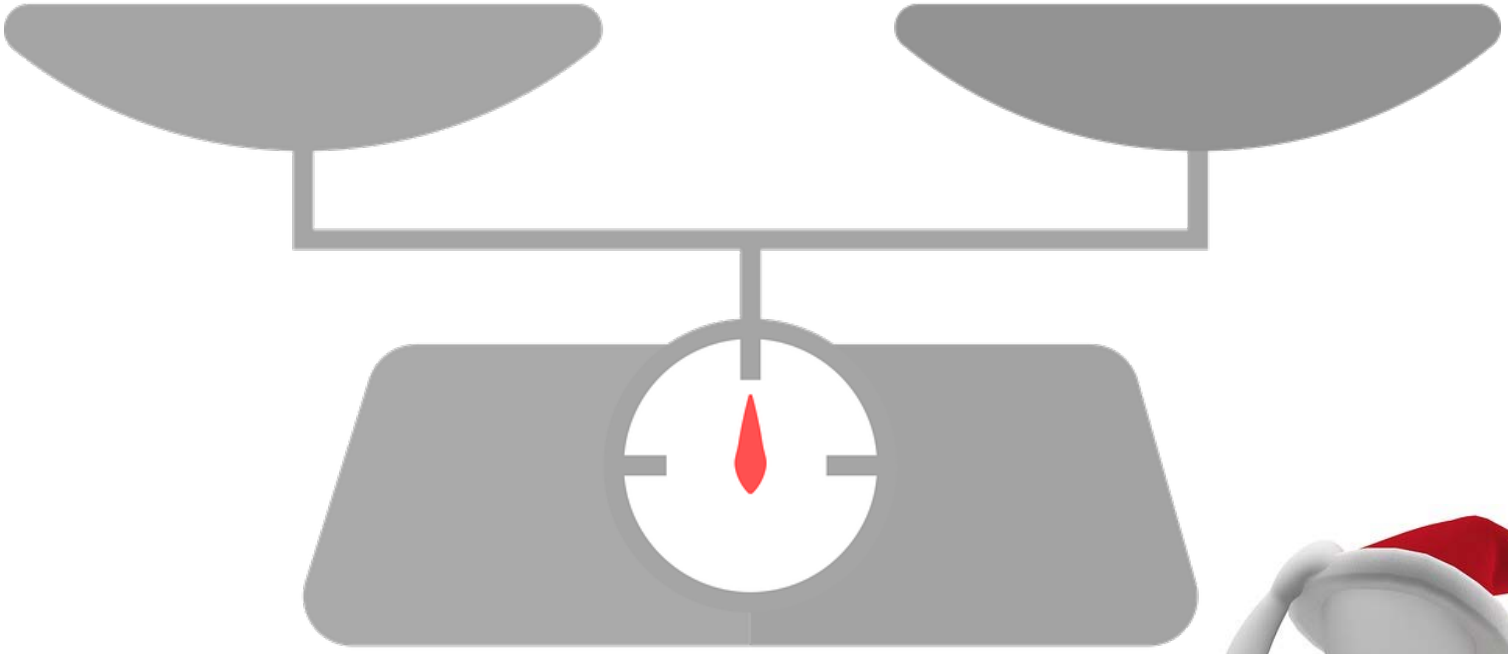
1000x  mit dem gleichen Spielzeugauto  darin



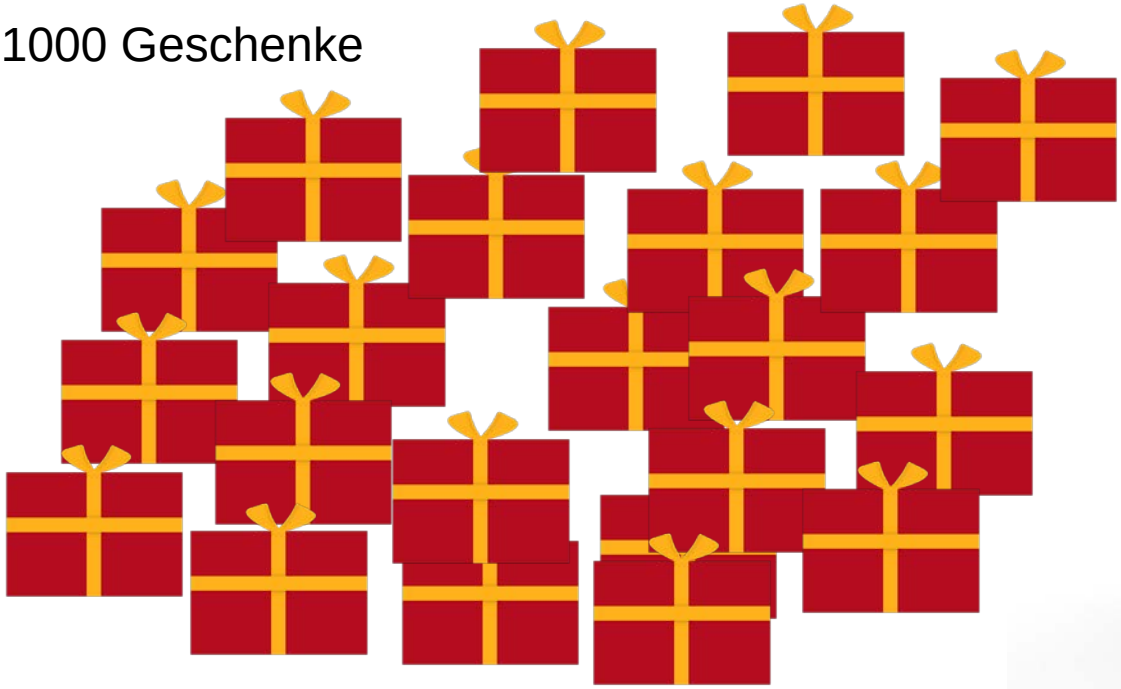
Oh, wo kommt denn der her?
Der fehlt jetzt in einem Geschenk!
Aber in welchem ???

Kein Problem, wir müssen
die Geschenke doch nur **10 Mal** mit
unserer riesigen Waage abwiegen,
dann wissen wir in welchem
der Schlüssel fehlt !



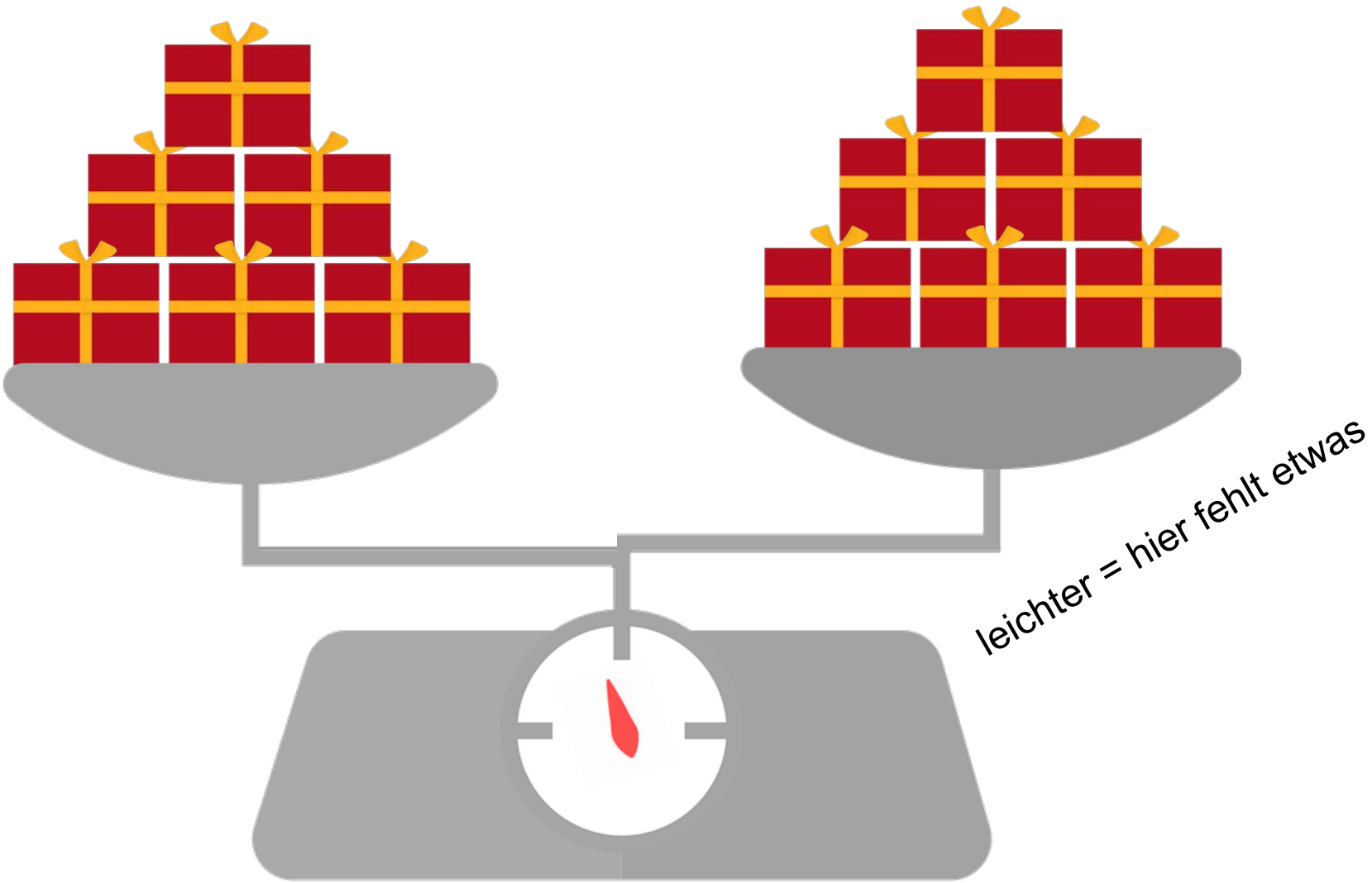


1000 Geschenke



500 Geschenke

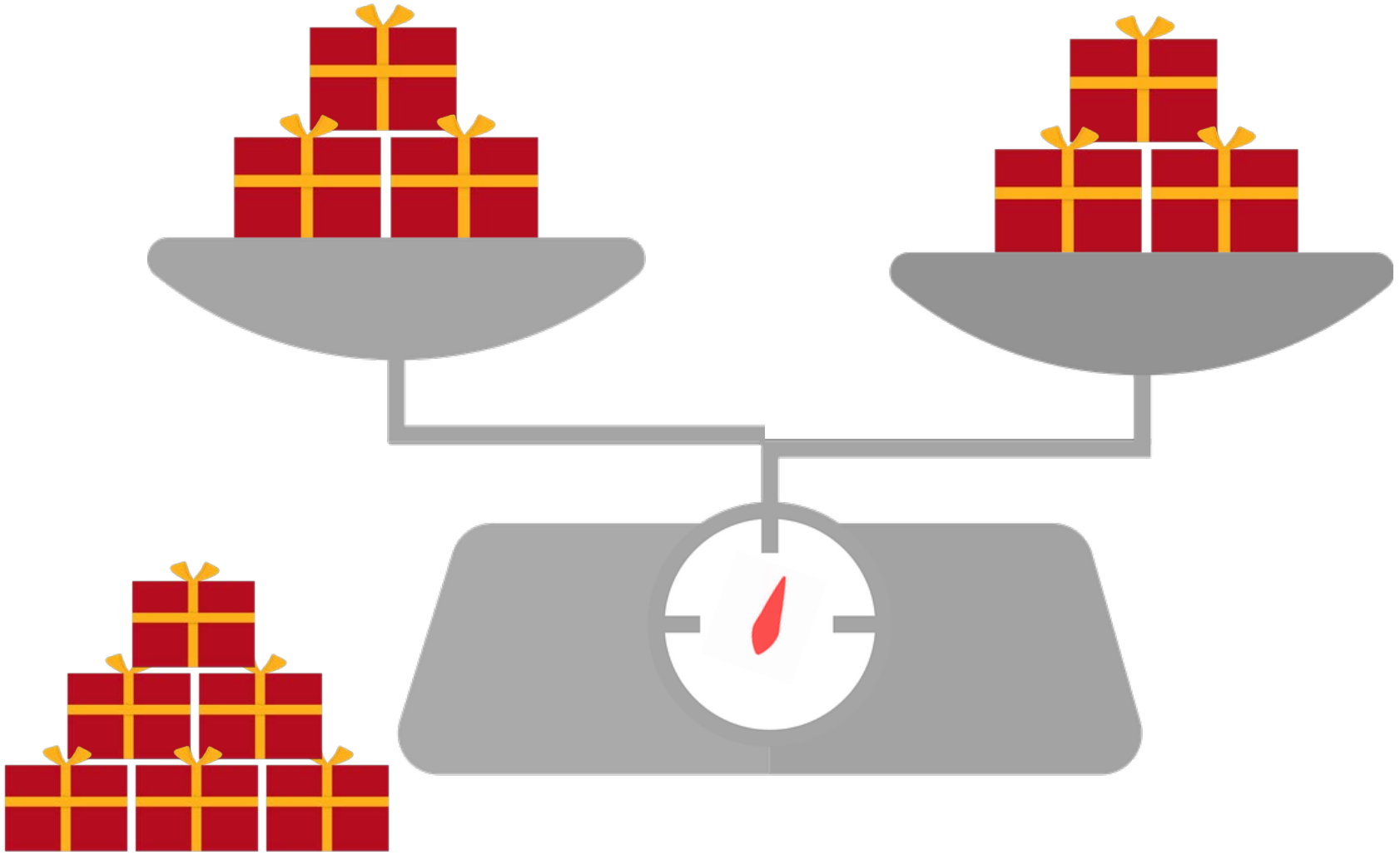
500 Geschenke



leichter = hier fehlt etwas

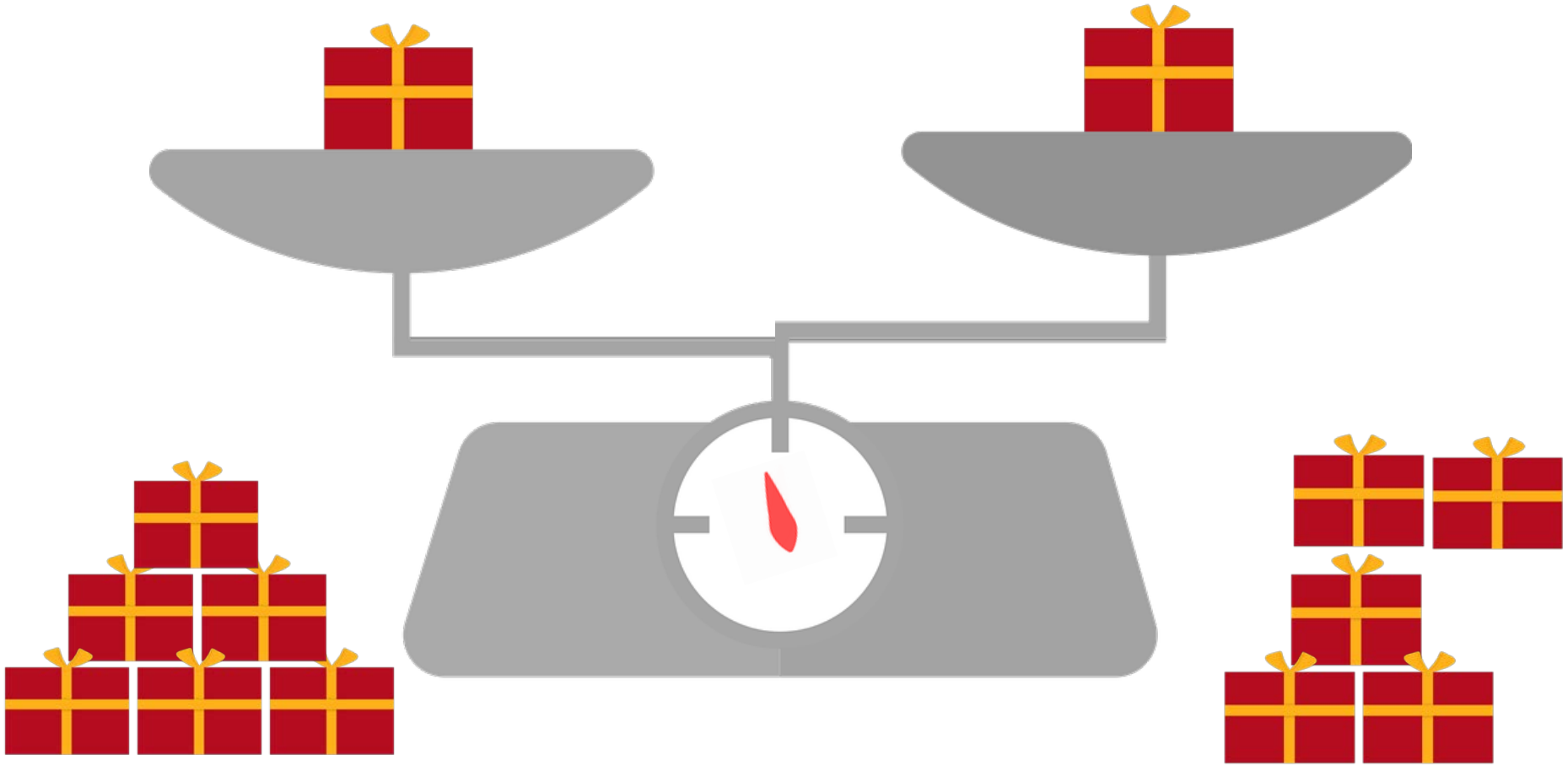
250 Geschenke

250 Geschenke



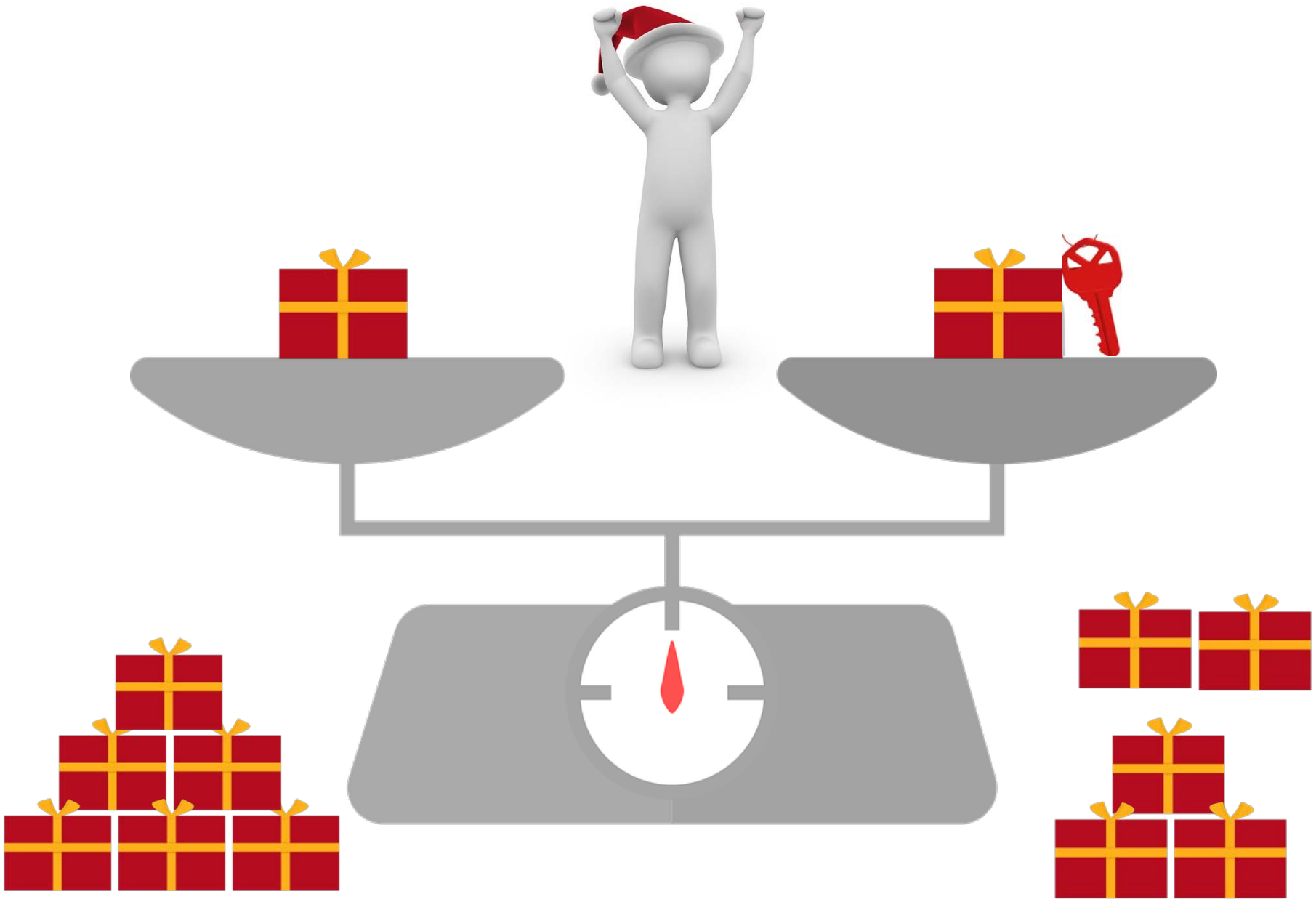
1 Geschenk

1 Geschenk



1 Geschenk

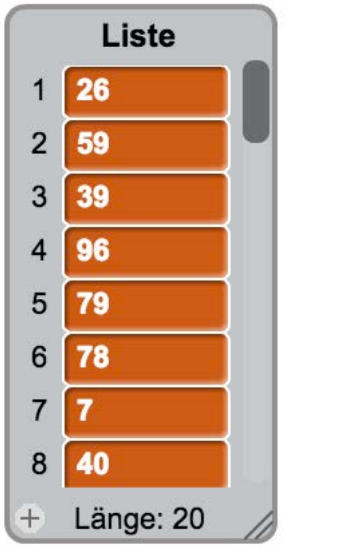
1 Geschenk

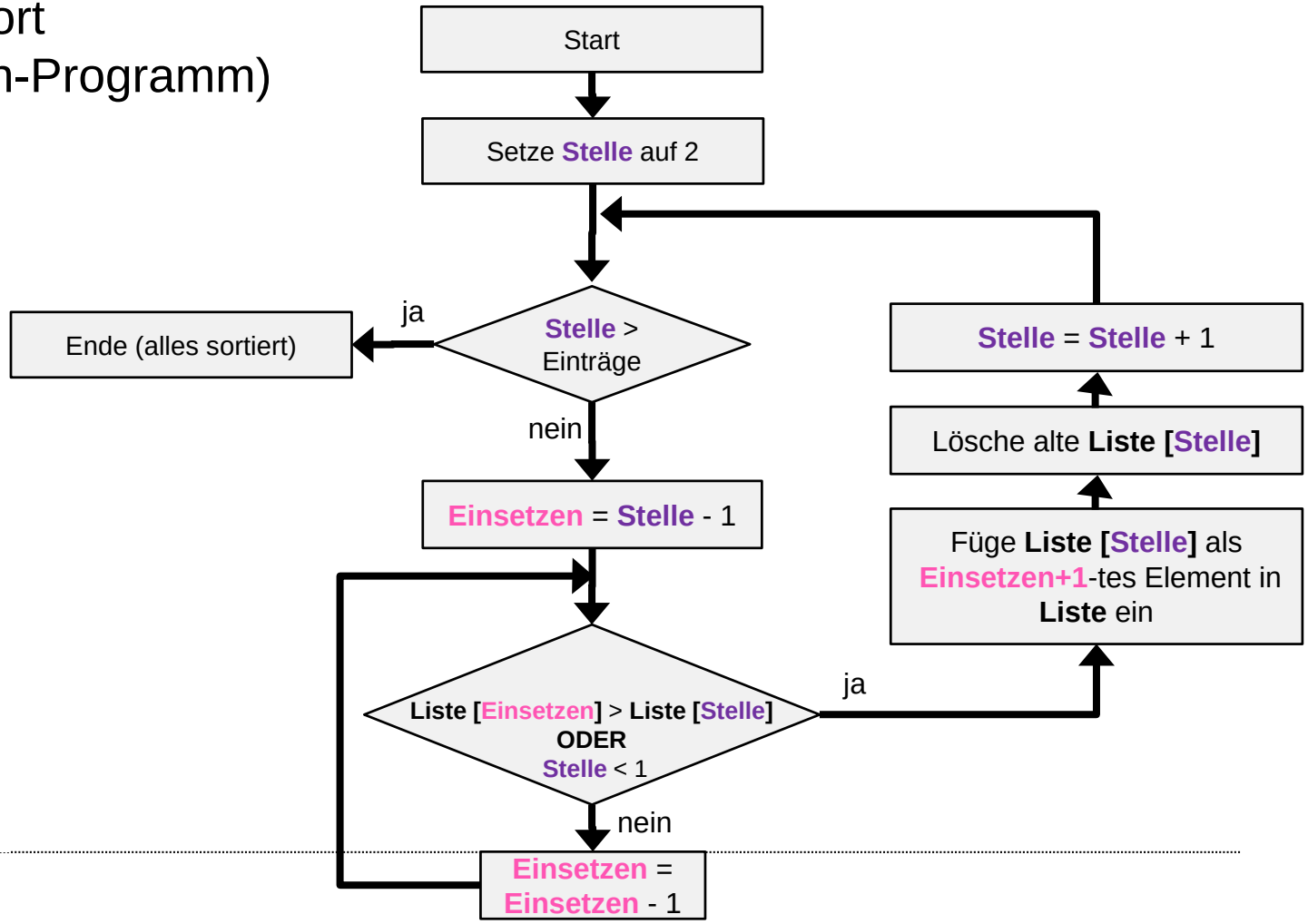


Algorithmen Vergleichen - Beispiel Sortieren und Suchen

» können verschiedene Algorithmen zur Lösung desselben Problems vergleichen und beurteilen (z.B. lineare und binäre Suche, Sortierverfahren).

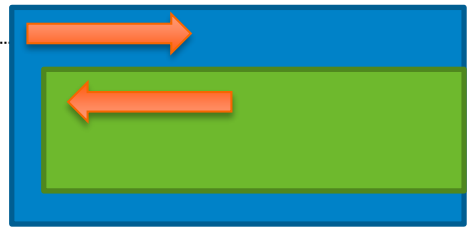
INSERTION-Sort
(kurzes Scratch-Programm)





Sortieren in Scratch – INSERTION Sort

[3,45,23,77,34,5,19]

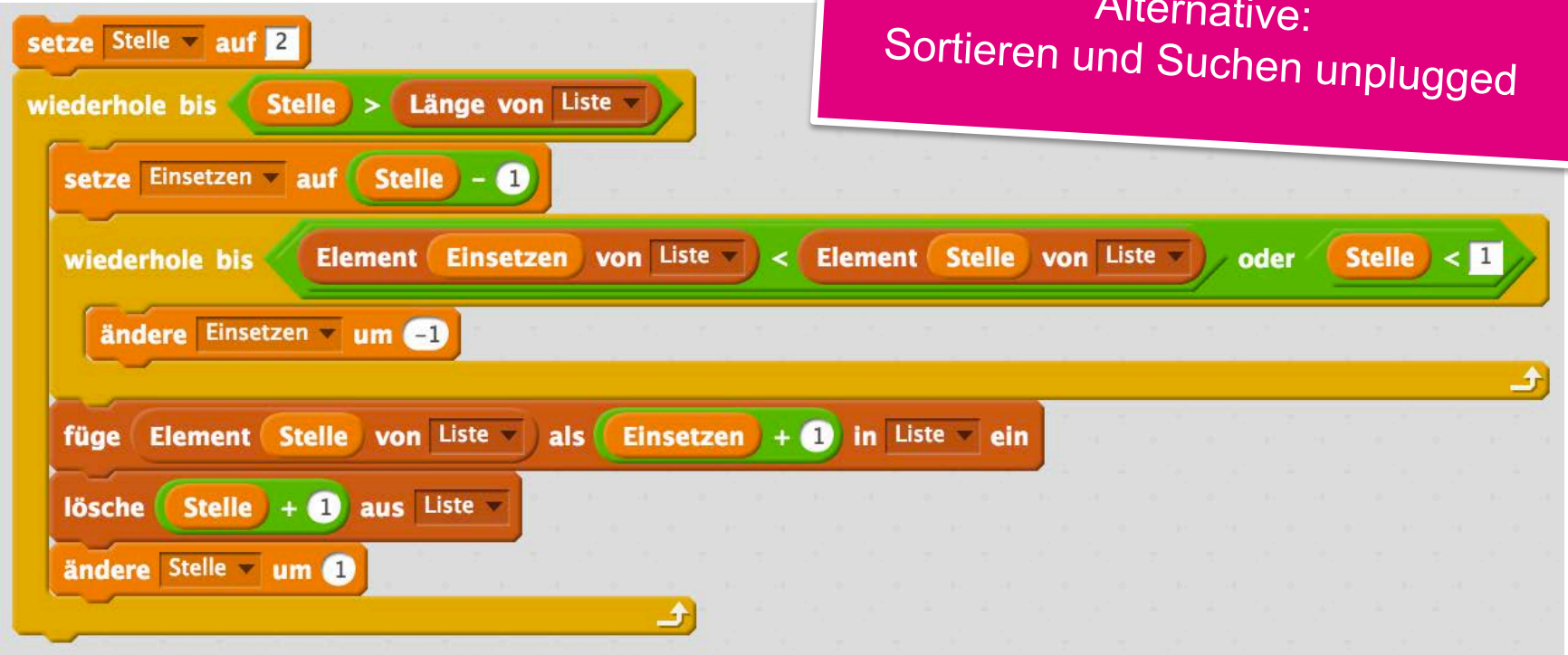


In Worten:

- Schritt 1:
  Wir vergleichen den Wert an der aktuelle Stelle mit allen vorherigen, bis wir einen kleineren Wert gefunden haben, oder am Anfang angekommen sind.
- Schritt 2:
  Wurde ein kleineren Wert gefunden, wir das aktuelle Element in die Liste NACH dieser Stelle eingefügt. Anschliessend wird das aktuelle Element aus der Liste gelöscht (wir haben es ja an die andere Stelle „kopiert“).
- Schritt 3:
  die aktuelle Stelle wird um +1 erhöht und mit Schritt 1 fortgesetzt, solange bis wir am Ende der Liste angekommen sind.

Sortieren in Scratch – INSERTION Sort

Alternative:
Sortieren und Suchen unplugged



Dieses Verfahren umzusetzen ist eventuell nicht zielstufengerecht. Man könnte es als vorgegebenen Block oder Scratch-Programm bereitstellen.
Scratch ist mit seiner Darstellung wird mit wachsender Komplexität schwieriger zu lesen und zu verstehen.

Sortieren und Finden – unplugged
Verkehrshaus (iFactory)

» können verschiedene Algorithmen zur Lösung desselben Problems vergleichen und beurteilen (z.B. lineare und binäre Suche, Sortierverfahren).