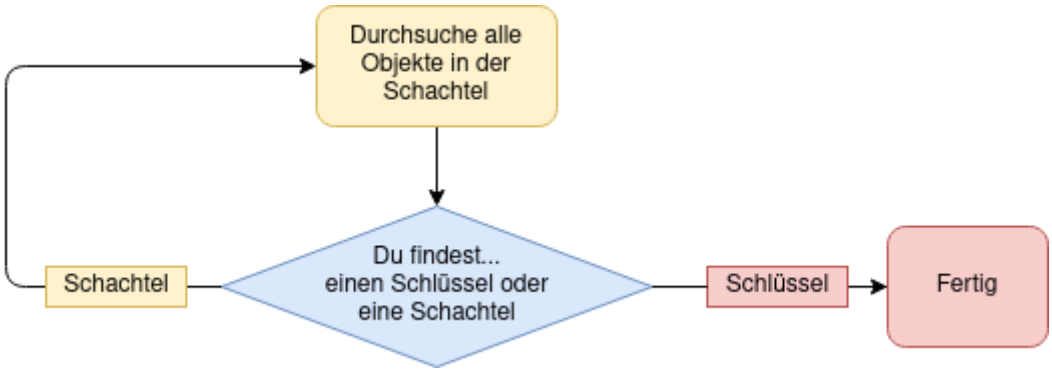
Rekursive Schachtelsuche

Die rekursive Denkweise macht sich zunutze, dass wir für jede Schachtel, wie wir finden, dasselbe tun müssen:

- Aufmachen.
- Wenn ein Schlüssel drin ist: Freuen!
- Wenn eine Schachtel drin ist: Das was wir mit jeder Schachtel machen...




```
funktion suche_schluessel(schachtel):
  für jeden gegenstand in schachtel:
    wenn gegenstand.istSchachtel():
      suche_schluessel(gegenstand)
    sonst wenn gegenstand.istSchlüssel:
      ausgeben "Schlüssel gefunden!"
```

Bei der Betrachtung des Pseudocodes fällt auf, dass sich die Funktion `suche_schlüssel` selbst aufruft – das ist der Ausdruck im Code des Denkprinzips "das was wir mit jeder Schachtel machen" von oben.

Wenn eine Funktion sich selbst aufruft spricht man von **Rekursion**.

Ein Wort zur Eleganz: Die rekursive Formulierung eines Algorithmus ist oft klarer als die iterative - sie bietet aber keine Performancevorteile – oft sind iterative Formulierungen sogar schneller.

Fallunterscheidung ist unbedingt notwendig

Die Funktion ruft sich aber nicht bedingungslos selbst auf, sondern nur dann, wenn eine Schachtel (und kein Schlüssel) gefunden wird. Wenn man diese Fallunterscheidung weglässt, erzeugt man eine "rekursive Endlosschleife": Die Funktion ruft sich bedingungslos immer wieder selbst auf, das Programm kommt zu keinem Ende.