# Aufgabenblatt 3

Bevor wir nun dazu kommen, „echte“ Programme zu schreiben, benötigen wir noch eine weitere Fähigkeit der Programmiersprache Ruby. Diese finden Sie aber auch in anderen Programmiersprachen wieder.

## Bedingte Anweisungen / Verzweigungen

Bei dem Kartensortieren haben Sie damals das „Programm“ in umgangssprachliche Form niedergeschrieben. Dabei trat häufiger die Formulierung „wenn, dann“, „wenn nicht, dann“. In Ruby kann man dies wie folgt realisieren:

```
if 3 * 2 > 3
  puts "3*2 ist größer als 3"
end
```

beziehungsweise auch

```
unless 3 * 2 > 3
  puts "3*2 ist nicht größer als als 3"
end
```

Probieren Sie diese Programme aus.

### Weitere Vergleiche

Insbesondere für Zahlen gibt es in Ruby eine Menge Vergleichsmöglichkeiten. Versuchen Sie herauszufinden (in dem Sie die Programme von eben verändern) was die folgenden sog. „Vergleichsoperatoren“ mathematisch darstellen sollen.

```
==    !=    <=    =>    <    >
```

### Aufgepasst!

Ein häufiger Fehler der auch erfahrenen Programmierern hin und wieder mal passiert, ist der folgende:

```
x = 3
if x = 5
  puts "x ist fünf!"
end
```

Was ist hier passiert? Wird der Text ausgegeben? Wofür haben wir das einfache = bisher immer genutzt?  Welchen Vergleichsoperator wollte der Programmierer wohl eigentlich verwenden?

### Unterscheiden können

Die Formulierung „Wenn …, dann …, sonst …“ tritt auch gelegentlich in unserem Sprachgebrauch auf. Dies kann man im Ruby wie folgt lösen.

```
x = 3
if x > 5
  puts "größer 5"
else
  puts "kleiner oder gleich 5"
end
```

Versuchen Sie ein Programm zu schreiben, was ohne diese Konstruktion „else“ auskommt. Tipp: Verwenden Sie „if“ häufiger als einmal.

## Jetzt ist die Kreativität gefragt

Sie haben nun eine Menge gelernt. Mit diesem Wissen können Sie nun eigene Programme entwickeln, die bestimmte Aufgaben für sie durchführen sollen.

### Das Kartensortieren

Wir wollen hierbei in  kleinen Schritten vorgehen. Sie erinnern sich vielleicht daran, dass bei dem Kartensortieren „die Karte, die am dichtesten an der zuletzt einsortierten Karte ist, finden“ eine Teilaufgabe war. Wir werden dieses Problem nun lösen, allerdings erst einmal nur mit Zahlen.

```
karten = [7,9,2,33,8,4]
vergleich = 9

# Hier Ihr Programm
```

Das Programm soll die Zahl aus Karten, die am Dichtesten an der Vergleichszahl liegt finden und ausgeben. Verwenden Sie dazu Ihre Kenntnisse von „each“, „if“ usw. Zögern Sie nicht, bei Problemen nach Hilfe zu fragen.

### Viel Erfolg!