Boolsche Algebra

1.1 Entscheidungen und Bedingungen

An vielen Stellen im täglichen Leben, sowie auch beim Programmieren müssen Entscheidungen getroffen werden: "Soll ich früher aufstehen, um ihn Ruhe zu früchstücken oder lass ich das Frühstück ausfallen, damit ich länger schlafen kann?"
Um nun abends den Wecker zu stellen muss man sich Folgendes überlegen und ausführen:

```
Falls <Will ich frühstücken?>:
    stelle_wecker(6)
Ansonsten:
    stelle_wecker(7)
```

Die Weckzeit ist also abhängig von der Bedingung: "Will ich frühstücken?".

Bedingung Eine Bedingung beim Programmieren ist eine Aussage oder ein Ausdruck, die mit "Ja" oder "Nein" zu beantworten ist. Komplizierte Antworten gibt es nicht! Eine Antwort muss immer "Ja" oder "Nein" sein, "Vielleicht" oder "Morgen" sind nicht erlaubt.
Um genau zu sein heißen diese Werte beim Programmieren meistens `True` (engl. für "Wahr") und `False` (engl. für "Falsch").
Vielleicht stellst Du Dir jetzt die Frage, wie eine Bedingung zu formulieren ist, da "Will ich frühstücken?" für Menschen einfach zu verstehen ist, für einen Computer allerdings nicht.

Die einfachste Bedinung ist `True` ("Wahr") oder `False` ("Falsch") zu schreiben.

```
Falls <Wahr>:
    sage("Die Bedingung ist wahr")
```

Zugebenermaßen ist dies nicht sinnvoll, da man in dem obigen Beispiel die Bedingung einfach weglassen kann.

Vergleichsoperationen sind eine Möglichkeit um Bedingungen zu beschreiben. Wie in der Schule können z.B. Operationen wie `>` (größer), `<` (kleiner), `>=` (größer gleich), `<=` (kleiner gleich) verwendet werden um Zahlen zu vergleichen.

Ein Ausdruck wie `30 > 20` ergibt `True` und `30 < 20` ergibt `False`. Diese können also als Bedingung verwendet werden kann.

```
# 'alter' ist eine zuvor definierte Variable
Falls alter > 10:
    sage("Du bist älter als 10")
```

Operation	Befehl	Beispiel	Ausgabe
Größer	>	20 > 20	Falsch
Größer Gleich	>=	20 >= 20	Wahr
Kleiner	<	10 < 30	Wahr
Kleiner Gleich	<=	10 <= 25	Wahr
Gleichheit	==	10 == 10	Wahr
Ungleichheit	!=	10 != 10	Falsch

Table 1.1: Vorhandene Vergleichsoperationen

1.2 Aufgaben

- Was (`True` oder `False`) ergeben die folgenden Ausdrücke jeweils:
  - 10 < 12:
  - 12 < 10
  - 15 != 15
  - 20 >= 15
  - 12 == 12
- In der Variablen `Schuhgröße` steht die Schuhgröße einer Person.
  Schreibe eine Falls-Ansonsten-Abfrage, die ausgibt (benutze "sage"), ob diese Person große Füße hat oder nicht. Große Füße bedeutet, dass die Schuhgröße größer als 42 ist.

1.3 if-Abfrage (Falls-Abfrage)

Bisher haben wir in den Beispielen keinen 'echten' Code verwendet. Dies wollen wir jetzt ändern. Eine "Falls"-Abfrage sieht in Python so aus:

```
1 # 'wert' ist eine zuvor definierte beliebige Zahl-Varible
2 if wert < 10:
3     print("Kleiner 10.")
4
5 print("Nach if-Abfrage.")
```

Listing 1.1: 'Kleiner'-Vergleich mit einer Variablen

Dazu schreibt man `if` (engl. für "falls), dann die Bedingung gefolgt von einem Doppelpunkt ':'. Darunter eine oder mehrere eingerückte Zeilen.
Die Einrückung (mit 4 Leerzeichen oder Tabulator-Taste) bedeutet, dass alles was eingerückt ist **genau dann** ausgeführt wird, wenn die Bedingung wahr ist.

Führt man dies nun aus mit den Werten `wert = 5` und `wert = 15` aus, verhält sich die Ausgabe unterschiedlich:

```
1 # wert = 5
2 Kleiner 10.
3 Nach if-Abfrage.
```

Listing 1.2: Ausgabe für kleineren Wert

```
1 # wert = 15
2 Nach if-Abfrage.
```

Listing 1.3: Beispiel Ausgabe für größeren Wert

Man sieht an den Beispielen, dass "Kleiner 10." nur dann ausgegeben wird, wenn der Wert kleiner als 10 ist. Die nichteingerückte Zeile darunter wird in beiden Fällen ausgeführt und dies führt dazu, dass "Nach if-Abfrage." immer ausgeben wird.

Falls-Ansonsten-Abfrage (if-else-Abfrage) Ist die Bedingung in der "Falls"-Abfrage falsch, so kann eine "Ansonsten"-Abfrage (engl. "else") hinzugefügt werden, die **genau dann** ausgeführt wird, wenn die Bedingung falsch ist.

```
1 # 'wert' ist eine zuvor definierte beliebige Zahl-Varible
2 if wert < 10:
3     print("Kleiner 10.")
4 else:
5     print("Größer oder gleich 10.")
6
7 print("Nach if-Abfrage.")
```

Listing 1.4: 'Kleiner'-Vergleich mit if-else-Abfrage

Führt man dies nun aus mit den Werten `wert = 5` und `wert = 15` aus, verhält sich die Ausgabe wie folgt:

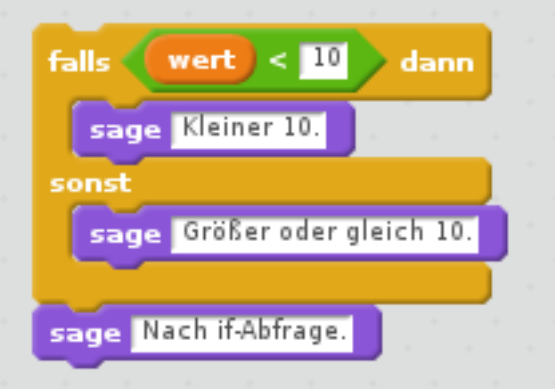
```
# wert = 5
Kleiner 10.
Nach if-Abfrage.
```

Listing 1.5: Ausgabe für kleineren Wert

```
# wert = 15
Größer oder gleich 10.
Nach if-Abfrage.
```

Listing 1.6: Beispiel Ausgabe für größeren Wert

In Scratch würde das obige Beispiel dann so aussehen:



Man sieht hier sehr schön die Einrückungen der "sage"-Blöcke, während der grüne sptize Block die Bedingung darstellt.

1.4 Aufgaben

- Du hast eine Variable 'zufall', die einen zufälligen Wert hat:

```
import random
zufall = random.randint(1, 100)
```

Listing 1.7: Python Code für eine zufällige Variable zwischen 1-100

Schreibe eine if-else Abfrage die bestimmt, ob der Wert der Variablen größer oder gleich 50 ist oder nicht. Gib dies dann mithilfe des print() aus.