LU05d - Verknüpfte Bedingungen

Eine Bedingung kann aus mehreren Teilbedingungen bestehen, z.B. `alter < 18 UND groesse >= 1.25`. Die Teilbedinungen werden mit logischen Operatoren AND, OR, NOT, XOR verknüpft.

Einige Selektionen lassen sich nicht mit einer einzigen Bedingung ausdrücken.

Zum Beispiel: Das Alter einer Person muss zwischen 18 und 27 sein.

Diese Bedingung kann in Form von zwei Selektionen dargestellt werden:

Programmablaufplan	Pseudocode	Python	Java

	falls (alter >= 18)
	  falls (alter <= 27)
	  Befehle
	  ende
	ende
	
	Befehle
		if age >= 18:
		  if age <= 27:
		#Commands
		#Commands
			if (alter >= 18) {
			  if (alter <= 27) {
			  // Befehle
			  }
			}
			
			// Befehle

Da die zweite Selektion abhängig von der ersten Selektion ist, spricht man von **verschachtelten** Selektionen. Eine andere Möglichkeit um die Bedingung auszudrücken, ist eine **verknüpfte** Bedingung.

Programmablaufplan	Pseudocode	Python	Java

	falls ((alter >= 18) UND (alter <= 27))
	  Befehle
	ende
	
	Befehle
		if (age >= 18) and (age <= 27):
		  #Commands
		#Commands
		oder
		if 18 < age <= 27:
		  #Commands
		#Commands
			if ((alter >= 18) && (alter <= 27)) {
			  //
			Befehle
			}
			
			// Befehle

Die beiden Teilbedingungen werden mittels `UND` verknüpft. Nur wenn beide Teilbedingungen erfüllt sind, ist die Bedingung als Ganzes erfüllt. In den meisten Programmiersprachen musst du beide Teilbedingungen vollständig codieren, also `Variable Operator Konstante`.

Nur wenige Programmiersprachen wie Python kennen eine verkürzte Schreibweise `Falls (18 <`

```
Alter < = 27)
```

Beim Programmieren empfiehlt es sich, die Teilbedingungen jeweils in Klammern zu setzen. Dadurch ist es einfacher den Überblick zu behalten.

Logische Operatoren

Logische Operationen benötigen wir für Bedingungen bei Selektionen und Iterationen. Sobald eine Bedingung aus zwei oder mehr Teilbedingungen besteht, müssen wir diese mit logischen Operatoren `and`, `or` und `xor` verknüpfen.

- Ein Ausdruck, der aus zwei Ausdrücken besteht, die mit dem **Und**-Operator kombiniert werden, ist wahr, wenn beide kombinierten Ausdrücke wahr sind.
- Ein Ausdruck, der aus zwei Ausdrücken besteht, die mit dem **Oder**-Operator kombiniert werden, ist wahr, wenn entweder einer oder beide der kombinierten Ausdrücke als wahr gewertet werden.
- Ein Ausdruck, der aus zwei Ausdrücken besteht, die mit dem **XOR**-Operator kombiniert werden, ist wahr, wenn genau einer der kombinierten Ausdrücke als wahr gewertet werden.

Mengenlehre

Siehe auch
https://de.wikibooks.org/wiki/Mathe_f%C3%BCr_Nicht-Freaks:_Verkn%C3%BCpfungen_zwischen_Mengen

Angelehnt an die Mengenlehre stellen wir das Resultat einer Verknüpfung grafisch dar. Jede Teilbedingung wird mit einem Oval dargestellt.

- Innerhalb des Ovals liegen alle Werte, für die die Bedingung erfüllt ist.
- Ausserhalb des Ovals sind Werte, für die die Bedingung nicht erfüllt ist.

Wenn wir mehrere Bedingungen kombinieren, können wir die logischen Operationen grafisch darstellen.

Übersicht der logischen Operatoren

Für die folgenden Beispiele verwenden wir eine Gruppe von Personen:

Name	Alter	Geschlecht
Anna	17	W
Beat	17	M
Carla	18	W
David	18	M
Emma	19	W

Name	Alter	Geschlecht
Frank	19	M

Ausserdem verwenden wir zwei Teilbedingungen:

- (alter > 18)
- (geschlecht = 'W')

Betrachten wir zunächst die Teilbedingungen einzeln, so ergibt sich folgendes Bild:

alter > 18	geschlecht = 'W'

Legen wir beide Abbildungen übereinander, so haben wir folgendes Bild:

AND

```
Falls ( (alter > 18) AND (geschlecht = 'W') )   // Pseudocode
if alter > 18 and geschlecht == 'W':            #Python
if ( (alter > 18) && (geschlecht == 'W') ) {    // Java
```

Die Bedingung ist nur erfüllt, wenn beide Teilbedingungen erfüllt sind.

OR

```
Falls ( (alter > 18) OR (geschlecht = 'W') )    // Pseudocode
if alter > 18 or geschlecht == 'W':             #Python
if ( (alter > 18) || (geschlecht == 'W') ) {    // Java
```

Die Bedingung ist erfüllt, wenn eine oder beide Teilbedingungen erfüllt sind.

Auswerten von verknüpften Bedingungen

Der Computer muss jede Bedingung auf einen bool'schen [1)] Wert reduzieren.

Er kann streng genommen nur Bedingungen in der Form `Falls TRUE` bzw. `Falls FALSE` überhaupt verarbeiten. Um eine verknüpfte Bedingung auszuwerten muss der Computer daher diese Bedingung schrittweise vereinfachen:

1. Die Teilbedingung in den innersten Klammern werden ausgewertet. Dies ergibt für jede Teilbedingung den Wert TRUE oder FALSE.
2. Das Ergebnis der Auswertung in Schritt 1 werden verknüpft und ebenfalls ausgewertet. Dadurch

entsteht die gewünschte, verinfachte Form der Bedingung.

Die folgende Grafik zeigt dieses Vorgehen beim Vereinfachen einer Bedingung. Das Beispiel geht davon aus, dass das `Alter = 25` ist.

Im nächsten Beispiel kombinieren wir zwei einzelne Bedingungen mit `and`. Der Code wird verwendet, um zu prüfen, ob die Zahl in der Variablen größer oder gleich 5 und kleiner oder gleich 10 ist. Mit anderen Worten, ob sie innerhalb des Bereichs von 5-10 liegt:

```
print('Is the number within the range 5-10:
')
number = 7

if (number >= 4 and number <= 10):
    print('It is! :)')
else:
    print('It is not :(')

Is the number within the range 5-10:
It is! :)
```

Im nächsten Schritt werden zwei Bedingungen mit dem `or`-Operator angegeben: Ist die Zahl kleiner als Null oder größer als 100. Die Bedingung ist erfüllt, wenn die Zahl eine der beiden Bedingungen erfüllt:

```
print('Is the number less than 0 or greater
than 100')
number = 145

if (number < 0 or number > 100):
    print('It is! :)')
else:
    print('It is not :(')

Is the number less than 0 or greater than 100
It is! :)
```

In diesem Beispiel vertauschen wir das Ergebnis des Ausdrucks Zahl > 4 mit `not`, d. h. dem `not`-Operator. Der `not`-Operator ist so geschrieben, dass der zu kippende Ausdruck in Klammern eingeschlossen ist und der `not`-Operator vor den Klammern steht.

```
number = 7

if (not (number > 4)):
    print('The number is not greater than
4.')
else:
    print('The number is greater than or
equal to 4.')

The number is greater than or equal to 4.
```

M319-LU05, M319-E1G, M319-E1F

 Marcel Suter, Kevin Maurizi

1)
Ein boolscher Wert kennt nur die beiden Zustände TRUE oder FALSE