Kapitel 4: Algorithmische Grundstrukturen und ihre Darstellung

Bisher haben wir hauptsächlich Programme gesehen, bei denen der Programmcode Zeile für Zeile nacheinander ausgewertet wird. Mit Hilfe **algorithmischer Grundstrukturen** können wir diesem Verhalten abweichen.

Algorithmische Grundstrukturen steuern dabei den Ablauf eines Programms. Mit ihnen könnt ihr Anweisungsblöcke definieren, die nur unter einer bestimmten Bedingung ausgeführt werden oder auch solche, deren Ausführung mehrfach erfolgt. Hierfür unterscheiden wir **Sequenzen**, **Verzweigungen** und **Schleifen**, welche wir uns - inklusiv dreier **Darstellungsformen** - nachfolgend genauer anschauen.

4.6. Komplexe Bedingungen

Manchmal genügt nicht nur eine einzige Bedingung.

In machen Fällen reicht es uns, wenn eine von mehreren Bedingungen erfüllt ist oder es müssen mehrere Bedingungen gleichzeitig erfüllt sein.

Zur Umsetzung dessen gibt es **logische Operatoren**.

Wir unterscheiden das **logische UND (`&&` bzw. in Python `and`)** und das **logische ODER (`||` bzw. in Python `or`)**.

Beim **logischen UND** müssen beide Eingaben wahr sein, damit auch das Ergebnis der Verknüpfung wahr ist:

Eingabe 1 - `E1`	**Eingabe 2 - `E2`**	**Ausgabe - `E1 && E2`**
True	True	True
True	False	False
False	True	False
False	False	False

Beim **logischen ODER** muss nur eine der beiden Eingaben wahr sein, damit das Ergebnis der Verknüpfung wahr ist:

Eingabe 1 - `E1`	Eingabe 2 - `E2`	Ausgabe - `E1 || E2`
True	True	True
True	False	True
False	True	True
False	False	False

Das logische UND in Python:

In [ ]:
```
thema = 'Programmierung'
klasse = 10

# Beide Bedingungen müssen wahr (True) sein!
if thema == 'Programmierung' and klasse == 10:
    print('It\'s a Match!')
else:
    print('Bestimmt auch eine gute Kombination!')
```

Das logische ODER in Python:

In [ ]:
```
zahl1 = 69
zahl2 = 42
zahl3 = 187

# Nur eine der Bedingungen muss wahr (True) sein!
if zahl1 == 42 or zahl2 == 42 or zahl3 == 42:
    print('Juhu! Die Zahl 42 ist am Start!')
else:
    print('Leider keine 42 dabei ...')
```

4.7. Übungen zu komplexen Bedingungen

Aufgabe 1

Gib jeweils an, welchen Wert der gegebene boolesche Ausdruck (für $a = 4$ und $b = 1$) hat.

- `(a < 5) || (a > 6)` → True
- `(a == 3) || (2 == b)` → False
- `(b > 1) && (a > 3)` → False
- `(a + b < 19) && (b < 1)` → False
- `(3 < 5) && (3 > 5)` → False

Aufgabe 2

Implementiere ein Programm, welches in der Lage ist, nach der Eingabe der Seitenlängen der Seiten a, b und c zu entscheiden, ob ein Dreieck gleichschenklig ist.

Hinweis: Ein Dreieck ist gleichschenklig, wenn die Seiten a und b **ODER** die Seiten b und c **ODER** die Seiten c und a gleich lang sind.

In [2]:
```
a = input('Gib die Länge der Seite a ein: ')
a = float(a)

b = input('Gib die Länge der Seite b ein: ')
b = float(b)

c = input('Gib die Länge der Seite c ein: ')
c = float(c)

if a == b or b == c or c == a:
    print(f'Ein Dreieck mit a = {a}, b = {b} und c = {c} ist: Gleichschenklig!')
else:
    print(f'Ein Dreieck mit a = {a}, b = {b} und c = {c} ist: NICHT gleichschenklig
```

```
Ein Dreieck mit a = 5.0, b = 4.0 und c = 4.0 ist: Gleichschenklig!
```

Aufgabe 3

Implementiere ein Programm, welches in der Lage ist, nach der Eingabe der Seitenlängen der Seiten a, b und c zu entscheiden, ob ein Dreieck rechtwinklig ist.

Hinweis: Nutze den Satz des Pythagoras. Jede Seite kann die Hypotenuse sein.

In [3]:
```
a = input('Gib die Länge der Seite a ein: ')
a = float(a)

b = input('Gib die Länge der Seite b ein: ')
b = float(b)
```

```
c = input('Gib die Länge der Seite c ein: ')
c = float(c)

if a**2 + b**2 == c**2 or c**2 + a**2 == b**2 or b**2 + c**2 == a**2:
    print(f'Ein Dreieck mit a = {a}, b = {b} und c = {c} ist: Rechtwinklig!')
else:
    print(f'Ein Dreieck mit a = {a}, b = {b} und c = {c} ist: NICHT rechtwinklig!')
```

```
Ein Dreieck mit a = 5.0, b = 4.0 und c = 3.0 ist: Rechtwinklig!
```