LÖSUNGSVORSCHLÄGE ZU DEN FINGERÜBUNGEN

Bei Fragen wenden Sie sich bitte an Ihre Tutoren
(z. B. über das **Piazza**-Forum oder *s.berktold@tum.de*)

Datentyp `boolean`

(1) Typkonvertierung

Zu **Fehlern** führen bei `int()` und `float()` bspw. Strings, die ungültige Zeichen wie `'a'` oder `','` enthalten.

Die Funktion `bool()` wertet für den Ganzzahl (*int*) `0`, den String (*str*) `""` bzw. `''` und die Gleitkommazahl (*float*) `0.0` zu `False` aus. Alle anderen Werte dieser Typen (wie z. B. das Leerzeichen `' '`) liefern `True`!

Auswertung der **Beispiele**:

- `int(2.8)`: **`2`**
- `int()`: **`0`**
- `float(True)`: **`1.0`**
- `int(False)`: **`0`**
- `bool(-1)`: **`True`**
- `bool("")`: **`False`**

(2) Boolesche Operationen

Die *Auswertungsreihenfolge* der booleschen Operatoren ist: `not` vor `and` vor `or`
Daher gilt bspw. `not a and b or c == ((not a) and b) or c`

Auswertung der Beispiele:

- `not not True` == **`True`**
- `"a" >= "A"` == **`True`**
- `"ABCD" >= "AD"` == **`False`**
- `bool(2*7.7 - 5*3) and True` == `True and True` == **`True`**
- `not 5 > 2 or (2/2 == 1)` == `False or True` == **`True`**
- `5%2 < 1 or (3 != 4 and 2**3 < 7)` == `False or (True and False)` == `False or False` == **`False`**

Bei der Anwendung von Vergleichsoperatoren auf Strings wird der ASCII-Code eines jeden Zeichens Zeichen für Zeichen verglichen. I. d. R. kann gesagt werden, dass der Vergleich lexikographisch erfolgt, wobei z. B. trotzdem `"A" < "a"` gilt. Der ASCII-Code eines Zeichens kann mit der Funktion `ord()` ermittelt werden.

`if`-Abfragen

(1) **Sortieren** (minimal)

Entscheidend ist folgende Abfrage. Davor müssen `a` und `b` initialisiert werden.

```
if a > b:
    a, b = b, a
```

bzw.

```
if a > b:
    tmp = a
    a = b
    b = a
```

(2) **Altersgruppen**

Mehrere Lösungen sind möglich, wobei darauf zu achten ist, dass auch negative Eingabewerte (< 0) und Eingaben, die größer sind als 130, behandelt werden müssen. Entsprechend könnte dann eine Ausgabe der Art „Sie sind kein Mensch." erfolgen. Außerdem muss der durch die Nutzereingabe erhalten Wert zu einem Integer konvertiert werden, damit die Anwendung von Vergleichsoperatoren Sinn macht, weil `input()` immer Strings liefert.

Beispiellösung:

```
alter = int(input("Wie alt bist du? "))

gruppe = ""  # hier nicht notwendig aber Konvention

if alter == 0:
    gruppe = "Säugling"
elif alter >= 1 and alter <= 13:
    gruppe = "Kind"
elif alter >= 14 and alter <= 17:
    gruppe = "Jugendliche(r)"
elif alter >= 18 and alter <= 130:
    gruppe = "Erwachsene(r)"
else:
    gruppe = "übermenschliche Kreatur"

print("Du wurdest als " + gruppe + " identifiziert.")
```

(3) [*Zusatzaufgabe*] **Intervallvergleiche**

In der obigen Lösung ersetze …

… „`alter >= 1 and alter <= 13`" durch „`1 <= alter <= 13`"
… „`alter >= 14 and alter <= 17`" durch „`14 <= alter <= 17`"
… „`alter >= 18 and alter <= 130`" durch „`18 <= alter <= 130`"

`while`-Schleifen

(1) Erste Versuche

```
i = 0 # Zählervariable

while i < 10: # bzw. <= 9
    print(i)
    i += 1
```

(2) Fencepost-Problem

```
n = 6  # beliebiger positiver Integer
i = 1  # Zählervariable (Startwert)
string = ""  # Hier wird angefügt

while i < n:
    string += str(i) + ", "
    i += 1
else: # optional (siehe (3))
    string += str(i)

print(string)
```

(3) Die Versionen sind identisch, falls kein **`break`** im `while`-Teil vorkommt. Anderenfalls kann es vorkommen, dass die gesamte Schleife durch das `break`-Statement verlassen wird, was bedeutet, dass der `else`-Teil übersprungen wird. Der Teil unterhalb der (gesamten) Schleife wird in beiden Fällen (also immer) ausgeführt und unterscheidet sich insofern von dem `else`-Teil.

(4) [*Zusatzaufgabe*] Potenz a^b

Da `a` und `b` verändert werden dürfen, kann `b` als Zählervariable benutzt werden. Man könnte allerdings ebenso eine weitere Variable verwenden.

```
a = 2  # Basis
b = 4  # Exponent
c = 1  # Ergebnis (wobei a⁰ == 1)

while b > 0:
    c = c * a
    b -= 1  # Zähler dekrementieren

print(c)  # optionale Ausgabe
```