Unterabschnitt 5

Kontrollstrukturen

Grundlegende Bemerkungen

- Bisher: das Programm wird von oben nach unten, Zeile für Zeile abgearbeitet
- Was wir noch nicht können: Fallunterscheidungen, Wiederholung von Vorgängen eine bestimmte Anzahl von Malen (wesentliche Bausteine der Algorithmik)
- **Kontrollstrukturen** erlauben es den Programmfluss abhängig von der Eingabe zu ändern
- Wichtig für die Übung: `if/else`, `for`-Schleifen, `while`-Schleifen

if/else — Beispiel zum Einstieg

```
1  def groesser_3(zahl):
2      if zahl > 3:
3          return True
4      elif zahl == 3:
5          print("Eingabe war 3!")
6          return False
7      else:
8          return False
```

if/else

- Verzweigung des Programmflusses abhängig von einer Bedingung (`boolean`)
- Aufbau:
  - Beginnt mit Kontrollwort `if <Bedingung>:`
  - Es folgt ein eingerückter Block von Anweisungen. Wird nur ausgeführt, wenn Bedingung wahr.
  - Optional beliebig viele `elif <Bedingung>:`-Blöcke, die jeweils nur dann ausgeführt werden, wenn die Bedingung wahr ist und keiner der vorherigen Blöcke ausgeführt wurde.
  - Abschließend optional ein `else:`-Block, der nur ausgeführt wird, wenn keiner der Blöcke darüber ausgeführt wurde.
- **Also**, es kann immer nur höchstens einer der Blöcke ausgeführt werden

if/else — Weiteres Beispiel

```
1  def foo(x):
2      if x > 1000:
3          print("Sehr gross")
4      elif x > 100:
5          print("Immer noch gross")
6      elif x > 10:
7          print("Nicht mehr ganz so gross")
8      # Achtung: kein else!
```

Übung zu if/else

Schreiben Sie eine Funktion, die einen String entgegennimmt und überprüft, ob dieser länger als 10 Zeichen ist.

Übung zu if/else (Lösung)

```
1 def laenger_10(string):
2     if len(string) > 10:
3         return True
4     else:
5         return False
6
7 print(laenger_10("Stundenplan"))
```

for-Schleife

- Ermöglicht die Wiederholung eines Programmblocks (**Schleifenkörper**) eine bestimmte Anzahl von Malen
- Der **Schleifenkopf** gibt die **Schleifenvariable** und einen listenartigen Wert an
- Die Schleifenvariable nimmt beim Durchlaufen des Schleifenkörpers jeweils den nächsten (bzw. den ersten) Wert der Liste an

```
1  ...
2  for SCHLEIFENVARIABLE in LISTE:  # Schleifenkopf
3      ANWEISEUNG_1 # Anfang Schleifenkoerper
4      ANWEISUNG_2
5      ...
6      ANWEISUNG_N  # Ende Schleifenkoerper
7  ...
```

for-Schleife — Beispiele

```
1  for x in [1,2,3]:
2      print x
3
4
5  for x in ["eine", "die"]:
6      for y in ["Ampel", "Strasse"]:
7          print x + " " + y
```

Range-Funktion

- Kann genutzt werden um eine Folge von Zahlen darzustellen
- Wird nur ein Argument angegeben, umfasst `range` alle Zahlen von 0 bis exklusive dem Argument
- Werden zwei Argumente verwendet, gibt das erste den Startwert an (inklusive)

```
1  for x in range (3):
2      print(x)   # 0, 1, 2
3
4  for x in range(1, 5):
5      print(x)   # 1, 2, 3, 4
```

Übung: Summen-Funktion

Schreiben Sie ein Funktion, die zu einer angegebenen natürlichen Zahl, die Summe aller Zahlen von 1 bis inklusive der angebenen Zahl zurückgibt.

Übung: Summen-Funktion (Lösung)

```
1  def summe(x):
2      teilergebnis = 0
3      for i in range(x+1):
4          teilergebnis = teilergebnis + i
5      return teilergebnis
```

While-Schleife

- Wiederholt einen Programmblock, solange eine Bedingung erfüllt ist (bzw. bleibt)
- Nachdem der Schleifenkörper durchlaufe wurde, wird erneut geprüft, ob die Bedingung wahr ist
- Mächtiger, aber “risikobehafteter” als for-Schleife
- Programmierer muss sicherstellen, dass die Schleifenbedingung falsch werden kann

While-Schleife — Beispiele

```
 1  def countdown(x)
 2      while x >= 0:
 3          print(x)
 4          x = x-1
 5
 6  # Ueber Liste iterieren (aehnlich for-Schleife)
 7  some_list = [1, 2, 3]
 8  pointer = 0
 9  while pointer < len(some_list):
10      print(some_list[pointer])
11      pointer +=1
```

While-Schleife — Negativbeispiele

```
 1  # Beispiel 1
 2  x = 10
 3  while x >= 0
 4      print(x)
 5
 6  # Beispiel 2
 7  some_var = 17
 8  while some_var > 10:
 9     print("Immernoch groesser 10!")
10     some_var = some_var+1
11
12  # Beispiel 3
13  def countdown(x):
14      while x <= 0:
15          print(x)
16          x = x - 1
```

Übung: Zahlen mit while-Schleife aufsummieren

Schreiben Sie eine Funktion, die eine Liste von Zahlen entgegennimmt und diese mithilfe einer `while`-Schleife aufsummiert.

Übung: Zahlen mit while-Schleife aufsummieren (Lösung)

```
1  def summe(l):
2      teilergebnis = 0
3      pointer = 0
4      while pointer < len(l):
5          teilergebnis = teilergebnis + l[pointer]
6          pointer = pointer + 1
7      return teilergebnis
```