Einführung in die Programmiertechnik

Erweiterte Kontrollflusskonzepte

Erweiterungen der Kernsprache

- Alle Kontrollflußkonzepte in einer imperativen Sprache lassen sich auf die Kernsprache zurückführen:
  - sequentielle Komposition
  - Alternative
  - while-Schleife
- theoretisch: syntaktischer Zucker (*syntactic sugar*)
  - praktisch: Verständlichkeit des Programms wird durch kompaktere Notationen verbessert; Formulierung von Algorithmen wird vereinfacht

Bedingte Anweisung

- if_stmt ::= "if" expression ":" suite
  ("elif" expression ":" suite)*
  ["else" ":" suite]
- suite: Codeblock
  - in gleicher Zeile, oder eingerückt in Folgezeilen
  - if x > 5: print x
  - Konvention: eine Zeile nur, wenn es keinen else-Teil gibt
- expression: beliebiger Ausdruck
  - Klammern um Gesamtausdruck syntaktisch nicht nötig, per Konvention vermieden
- if-Blöcke können verschachtelt werden
  - Algorithmen sollten aber auf „wenige“ Ebenen der Einrückung beschränkt werden

Umformung von if-Anweisung

- if-Anweisung im if-Teil: Verwendung von and-Operator

Statt

```
if cond1:
  if cond2:
    aktion
```

schreibt man

```
if cond1 and cond2:
  aktion
```

Umformung von if-Anweisung

- if-Anweisung im else-Teil: Umformulierung als elif

Aus

```
if cond1:
    aktion1
else:
    if cond2:
        aktion2
    else:
        aktion3
```

wird

```
if cond1:
    aktion1
elif cond2:
    aktion2
else:
    aktion3
```

Spracherweiterung: Schleifen

- While-Schleife ist Spezialfall der „allgemeinen Schleife“:

  LOOP

  $A_1;A_2;\ldots;A_k;$

  IF B **THEN EXIT**;

  $A_{k+1};A_{k+2};\ldots;A_n$

  END

- Üblich:
  - While-Schleife
  - Do-While-Schleife
  - Repeat-Until-Schleife
- Orthogonalitätsprinzip: Für jeden Aufgabentyp sollte es in einer Programmiersprache nur ein Konstrukt geben

Schleifen in Python

- While-Schleife
- For-Schleife: Iterieren über eine Folge von Werten
  - Beispiel: Explizite Aufzählung von Werten in einer Liste

```
Werte = [ 3, 1, 4, 1, 5, 9, 2]
for ziffer in Werte:
    print ziffer
```

  - Zuweisung an Laufvariable erlaubt; Iteration setzt dann mit nächstem Wert fort
- Traditionelle „Zählschleife“: Funktion range() generiert Folge von Zahlen
  - range(10): Zahlen von 0..9
  - range(2, 20): Zahlen von 2..19
  - range(2, 20, 3): Zahlen von 2..19 in 3er-Schritten