Einführung in die Programmierung

Ronja Düffel
WS2016/17

07. Oktober 2016

WICHTIG!!!

Vorlesung ab Montag im

H 16
Hörsaalgebäude 4.Stock

Rückblick

- Datentypen
  - `bool`
  - Zahlen (`int` und `float`)
  - `string`
- Variablen
- Kontrollstrukturen
  - Verzweigungen (`if...:` und `if...else:`)
  - Schleifen (`while...:` und `for...:`)

aufsteigend sortierte Zahlenfolge

- überprüfen, ob Zahlenfolge aufsteigend sortiert ist ✓
- Zahlenfolge sortieren ???
- brauchen dynamische Anzahl an Variablen

Listen

Listen

- mehrere Werte unter einem Namen zusammengefasst
- Länge der Liste ist nicht festgelegt (dynamisch)
- auf einzelne Werte kann zugegriffen werden
- **Achtung!** In der Informatik fängt man bei 0 an zu zählen!
  `<liste>[0]` liefert das erste Element der Liste

Listen

Operator/ Funktion	Beschreibung
`<list>[x]`	Zugriff auf Element mit Index `x`
`<list>[x:y]`	Zugriff auf Teilliste von Index `x` bis `y`
`<list> + <list>`	zusammenfügen von Listen
`<list>.append(x)`	hinzufügen von x
`del <list>[x]`	löschen von Element mit Index `x`
`<list>.remove(x)`	löschen von Element x
`len(<list>)`	Länge der Liste

Funktionen

für Operationen die immer wieder gebraucht werden

- + Wiederverwertbarkeit
- + leichte Wartbarkeit
- + nur einmal schreiben
- + leicht auszutauschen
- + Übersichtlichkeit

Funktionen

File Edit Format Run Options Windows Help

```
#Eine Funktion

def add(a,b):
    '''Addiere die Zahlen a und b'''
    return a+b
```

Gültigkeitsbereiche

File Edit Debug Options Windows Help

```
>>> def sowas():
        x = 0
        print('x = 0')
        return

>>> x = 5
>>> sowas()
x = 0
>>> 5/x
```

Gültigkeitsbereiche

File Edit Debug Options Windows Help

```
>>> def sowas():
        x = 0
        print('x = 0')
        return

>>> x = 5
>>> sowas()
x = 0
>>> 5/x
1.0
>>>
```

Gültigkeitsbereiche

- Variablenname ist in dem Anweisungsblock gültig, in dem er definiert wird.
- unterscheide zwischen *lokalen* (innerhalb Block/Funktion) und *globalen* (auch außerhalb) Variablen
- Verwendung globaler Variablen innerhalb von Funktionen mit `global`

Module

Wiederverwendung von Funktionen in anderen Programmen :

- `import`
  - `import <Modulname>` (Dateiname ohne `.py`)
    Verwendung durch `<Modulname>.<Funktionsname>`
    (kein Namenskonflikt)
  - `from <Modulname> import <Funktionsname(n)>`
    Verwendung durch `<Funktionsname>`
    (!gleichnamige Funktionen werden überschrieben!)
  - `from <Modulname> import *`
    Alles wird importiert, gefährlich aber "bequem"

Dateien lesen und schreiben

open() : öffnet eine Datei in angegebenem Modus

- `'r'`: Lesemodus
- `'w'`: Schreibmodus **!**Datei wird überschreiben **!**
- `'a'`: Schreibmodus, neue Daten werden am Ende hinzugefügt

read() : Lese den Inhalt der Datei; komplett, oder die angegebene Anzahl an Bytes

write() : Schreibt Daten in Datei. Zeilenumbruch muss explizit angegeben werden

close() : schließt Datei.

Rekursion

Um Rekursion zu verstehen,
muss man erstmal Rekursion verstehen

- Methode etwas durch sich selbst zu definieren

Beispiel (Summe)

Die Funktion $f : \mathbb{N} \to \mathbb{N}$ *sei gegeben durch*

$$f(n) := \begin{cases} 0, & \textit{falls } n = 0 \quad \textit{Rekursionsanfang} \\ n + f(n-1), & \textit{sonst.} \quad \textit{Rekursionsschritt} \end{cases}$$

rekursive Programmierung

- Funktionen die sich selbst aufrufen (auch verschachtelt)
- Abbruchbedingung muss auch erreicht werden (Gefahr der Endlosschleife)

File Edit Format Run Options Windows Help

```
# Summe rekursiv

def sum_rek(n):
    if(n == 0):
        return 0
    else:
        return n + sum_rek(n-1)
```

ideale Kaninchen

- Ein Kaninchenpaar (m,w)
- ist nach einem Monat geschlechtsreif
- gebären nach einem Monat Tragzeit ein weiteres Kaninchenpaar
- Kaninchen sterben nie
- Wie viele Kaninchenpaare nach n Monaten (KP_n)?

$$KP_n = KP_{n-1} + KP_{n-2}$$

WICHTIG!!!

Vorlesung ab Montag im

H 16
Hörsaalgebäude 4.Stock