Python für Linguisten

Dozentin: Wiebke Petersen & Co-Dozent: Valentin Heinz

3. Foliensatz
Funktionsdefinitionen

Wiederholung: Funktionsaufruf

- Python bringt einige vordefinierte Funktionen mit: z.B. `round(number)`, `min(number1,number2)`, `type(object)`, `id(object)`.
- Ein Funktionsaufruf ist ein Ausdruck
  `function_name(arg1,arg2)`
  `arg1,arg2` sind die Argumente mit denen die Funktion `function_name` aufgerufen wird.
- Jede Funktion hat eine festgelegte Stelligkeit (Zahl der Argumente).
- Bei der Auswertung eines Funktionsaufrufs werden zunächst die Argumente ausgewertet und anschließend wird die Funktion mit den sich ergebenden Werten ausgewertet. Genaugenommen sind Operatoren auch Funktionen (mit einer anderen Syntax).
- Funktionsaufrufe können als Input für andere Funktionen eingesetzt werden (die Funktionen werden verschachtelt)
  Beispiel: `type(id("hello"))`
  Beispiel: `min(round(4.3),4.2)`
  Beispiel: `round(min(4.3,4.2))`

Definition von Funktionen

```
1  >>>def double(x):
2          y=2*x
3          print ("Das Doppelte von "+str(x)+" ist "+str(y))
4          return y
5
6  >>> n=double(4)
7  Das Doppelte von 4 ist 8
8  >>> n+5
9  13
```

- Zeile 1-4: Funktionsdefinition
- Zeile 6: Funktionsaufruf
- Die Funktionsdefinition ist eine Anweisung (statement) und bildet somit einen Block.
- Sie besteht aus dem Funktionskopf (Zeile 1) und dem Funktionskörper (Zeile 2-4).

Definition von Funktionen

```
1  >>>def double(x):
2          y=2*x
3          print ("Das Doppelte von "+str(x)+" ist "+str(y))
4          return y
```

- Der Funktionskopf hat immer die Form
  `def function_name(list of parameters)`
- Die Parameter sind Variablen, die beim Aufruf der Funktion mit Argumenten gefüllt werden.
- Der Körper besteht aus einer beliebigen Zahl von Anweisungen (> 0). Er wird eingerückt, da er zusammen mit dem Kopf eine Funktionsdefinition bildet.
- Beim Aufruf der Funktion werden die Parameter mit den Argumenten gefüllt und es werden schrittweise die Anweisungen im Funktionskörper ausgewertet, bis eine `return`-Anweisung ausgeführt wird. Diese gibt ihren Wert an die Stelle zurück, an der der Funktionsaufruf erfolgte. Vorsicht: Nach einer `return`-Anweisung werden keine weiteren Anweisungen des Funktionskörpers mehr ausgeführt.

Definition von Funktionen mit Dokumentation

```
1  from __future__ import division
2  def area(base, height):
3      ''' (number,number) -> float
4      returns the area of a triangle with base length 'base' and height length 'height'
5      >>> area(5,4)
6      10
7       '''
8      return base * height / 2
```

- Zeile 2: Funktionskopf
- Zeile 3: Festlegung der Datentypen
- Zeile 4: Beschreibung der Funktion
- Zeile 5: Beispielaufruf
- Zeile 6: Output für den Beispielaufruf
- Zeile 3-7: Docstring. Diese Information wird bei der Anfrage `help(area)` zusätzlich zu Zeile 2 angezeigt.
- Vergessen Sie nie bei der Definition einer Funktion den Docstring mit anzugeben.

Rezept für gute Funktionsdefinitionen

1. Beispiele für den Funktionsaufruf
   wählen Sie einen sprechenden Namen
2. Legen Sie die Datentypen fest (type contract)
3. Schreiben Sie den Funktionskopf
   Wählen Sie sprechende Namen für die Parameter
4. Schreiben Sie die Funktionsbeschreibung
   Vergessen Sie nicht zu schreiben, was die Funktion tut und was sie zurückgibt.
5. Testen Sie die Funktion ausgiebig

Aufgabe

- Schreiben Sie eine Funktion, die Fahrenheit in Celsius umrechnet.
- Schreiben Sie eine Funktion, die die Initialen einer Person zurückgibt.
- Schreiben Sie eine Funktion, die das Volumen einer Kiste berechnet.
- Schreiben Sie eine Funktion, die die Wandflächen eines Zimmers berechnet.
- Schreiben Sie eine Funktion, die die benötigte Menge von Farbeimern berechnet, die zum Streichen eines Zimmers benötigt werden.
- Schreiben Sie eine Funktion, die Minuten in Sekunden, Stunden in Sekunden und Tage in Sekunden umrechnet.