Python für Linguisten

Dozentin: Wiebke Petersen & Co-Dozentin: Esther Seyffarth

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).
- Beim Aufruf von Funktionen wird für jedes Argument ein konkreter Wert angegeben, der dann bei der Auswertung der Funktion verwendet wird. 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(parameter_1, parameter_2, ... , parameter_n):`
- Die Parameter sind Variablen, die beim Aufruf der Funktion mit den übergebenen 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        # an Rechnern mit Python 2.x
2  def area(base,height):
3      '''(number,number) -> float
4      returns the area of a triangle with base length 'base' and
5      height length 'height'
6      >>> area(5,4)
7      10
8      '''
9      return base * height / 2
```

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

Rezept für gute Funktionsdefinitionen

1. Wählen Sie einen sprechenden Namen. Schlecht: `func1()`. Gut: `cube_volume()`.
2. Wählen Sie sprechende Namen für die Parameter.
3. Schreiben Sie den Funktionskopf.
4. Legen Sie die Datentypen fest (type contract).
5. Schreiben Sie die Funktionsbeschreibung: Was tut die Funktion, was gibt sie zurück?
6. Ergänzen Sie Beispiele für den Funktionsaufruf (ggf. reicht ein Beispiel).
7. Testen Sie die Funktion ausgiebig: Verhält sie sich so wie geplant? Treten unerwartete Fehler auf? Sind die Rückgabewerte nützlich? Entsprechen die Ergebnisse (Rückgabewerte) Ihrer Intuition/Rechnung? (Zur Fehlerbehandlung wird es noch eine separate Sitzung geben.)

Tipps zur Verwendung von Funktionen

- Sie können eine Funktion in der gleichen Datei, in der sie definiert ist, aufrufen.
- Allerdings muss die Funktionsdefinition vor (also oberhalb von) dem Aufruf der Funktion stehen.
- Eine Funktion kann man als eingebettetes Programm betrachten. Sobald die Funktion ausgewertet wurde und der Python-Interpreter sich wieder außerhalb der Funktion befindet, kann er nur noch auf die mit `return` zurückgegebenen Werte zugreifen!
- Sie können auch innerhalb eines Funktionskörpers neue Variablen erstellen, belegen und verwenden, aber diese Variablen sind nur innerhalb des Funktionskörpers bekannt.
- Überlegen Sie sich: Welche Werte brauche ich auch nach Beendigung der Funktion noch? Welche Werte dienen nur zum Berechnen und werden danach nicht mehr gebraucht?
- Falls Sie mehr als einen Rückgabewert brauchen, können Sie beliebig viele mit Komma getrennte Werte zurückgeben.

```
return original_value, modified_value, "additional value"
```

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 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.