

Python für Linguisten

Dozentin: Wiebke Petersen & Co-Dozent: Valentin Heinz

2. Foliensatz

Variablen, Datentypen, vordefinierte Operationen und Funktionen

Shell

- Die Python Shell meldet sich mit dem **Prompt**: `>>>`
- Jede **Anweisung**, die hinter den Prompt geschrieben wird, wird sofort ausgeführt.

Beispiel:

```
>>>name="Anna"
```

weist der Variablen `anna` den Wert "Anna" zu.

- Den Befehl `print` ergänzt Python automatisch.

Beispiel:

```
>>>4 + 3
```

ist äquivalent zu

```
>>>print(4 + 3)
```

und liefert den Output 7 auf dem Monitor.

Anweisungen

- Jede Anweisung startet in einem neuen Block.
- Ein neuer Block startet immer in einer neuen Zeile (bisher sind unsere Blöcke zumeist einzeilig).
- Diese Anweisungen kennen Sie bereits:
 - `variable_name = expression`
weist der Variablen `variable_name` das Ergebnis der Auswertung des **Ausdrucks** `expression` zu.
Beispiel: `>>> number = 5 + 7`
weist der Variablen `number` den Wert von `5 + 7` also `12` zu.
 - `print(expression)`
gibt das Ergebnis der Auswertung des Ausdrucks `expression` auf dem Bildschirm aus.
Beispiel: `>>> print(5 + 7)`
gibt `12` aus.

Objekte & Ausdrücke

Zurück zu `>>> print(5 + 7)`

- 5 und 7 sind **Objekte** in Python, sie erhalten einen festen Speicherplatz.
- Mit `>>> id(object)` können sie die interne ID des Objekts abfragen
- Testen Sie folgendes: `>>> id(2)`, `>>> x=2`, `>>> y=x`, `>>> id(x)`,
`>>> id(y)`.
- Objekte können von unterschiedlichem **Typ** sein. 5 und 7 sind beide vom Typ `int` (integer).
- Testen Sie: `>>> type(2)`, `>>> type(2.0)`, `>>> type("two")`
- Mit `str(x)`, `int(x)`, `float(x)`, `list(x)` können sie den Typ eines Objekts ändern
- `+` ist ein zweistelliger **Operator**. Die Typen der Inputobjekte müssen passen! Der Ausdruck `5 + 7` wird zu dem neuen Objekt 12 ausgewertet.
- `+` und `-` können auch als einstellige Operatoren verwendet werden.
- Sie können auf Klammern in Ausdrücken mit mehreren Operatoren verzichten, wenn sie einen Ausdruck gemäß der üblichen Operatorenpräzedenz auswerten lassen wollen.
Beispiel: `>>> 5 + 7 * -3` wird zu `-16` ausgewertet.

Übersicht über die gängigen numerischen Operatoren

+ Addition ($3 + 2 = 5$)

- Subtraktion ($3 - 2 = 1$)

* Multiplikation ($3 * 2 = 6$)

/ Division ($7/2 = 3.5$), Vorsicht: bei Python 2.7 müssen sie zunächst die Anweisung `from __future__ import division` erteilen.

// ganzzahlige Division ($7/2 = 3$)

% Modulo, Rest bei ganzzahliger Division ($7\%2 = 1$)

** Exponent ($2 ** 3 = 8$)

Operatorpräzedenz: $+, - < *, /, //, \% < ** < -x, +x$

Strings

- Strings sind Zeichenketten und müssen in einfache oder doppelte Anführungszeichen gesetzt werden.

```
>>> this_string = "hello", >>> this_string = 'hello'.
```

Diese können auch mehrfach verwendet werden und somit andere Anführungszeichen in einem String einschließen. Beispiel:

```
1 >>> x= ''' dies ist eine 'ziemlich' lange "Anweisung"
2 mit 'Zeilenumbruch' '''
3 >>> print(x)
```

- Sonderzeichen müssen escaped werden:
Tabulator: `\t`
Zeilenumbruch: `\n`
Backslash: `\\`
einfache oder doppelte Anführungszeichen: `\'`, `\"`

Stringoperationen

- Zuweisung von Strings zu Variablen: `>>> this_string = "hello"`
- Stringkonkatenation: Strings werden mit dem Operator `+` konkateniert (`"some" + "value"`)
- mehrfache Konkatenation mit Operator `*`: `string * int, int * string`
- Substringtest: `>>>'ell' in this_string, >>>'le' in this_string`
- Ein String wird intern über einen Index repräsentiert.

h	e	l	l	o
0	1	2	3	4
-5	-4	-3	-2	-1

- Auf den Index kann mit einer Integer in eckigen Klammern zugegriffen werden. Beispiel: `>>> this_string[0]` gibt 'h' aus
- Es kann auch ein Indexintervall angegeben werden (Slicing):
`>>> this_string[0:2]` Wichtig: **bis** drittes Element, also Index 2
- dies kann auch als `>>> this_string[:2]` geschrieben werden.
- Testen Sie verschiedene Intervalle:
`>>> this_string[2:], this_string[:], this_string[-4:]...`

Datentypen – Listen

- Eine leere Liste wird durch `>>> this_list = []` erstellt, alternativ kann auch `>>> this_list = list()` verwendet werden
- Eine Liste mit Elementen wird durch `>>> this_list = ['a','b','c']` erstellt. Alternative?
- Wie Strings verfügen Listen über einen Index: `>>> this_list[0]` gibt 'a' zurück.

Syntax- & Semantikfehler

- Syntaxfehler treten auf, wenn die Anweisungen syntaktisch nicht wohlgeformt sind.
Beispiel: `>>> 5*/3` Beispiel: `>>> 5+3) *2`
- Der Python-Interpreter meldet Ihnen genau wo der Fehler auftritt.
- Semantikfehler treten auf, wenn die Anweisungen zwar syntaktisch wohlgeformt sind, aber aufgrund von z.B. Objekten falschen Typs oder aufgrund einer falschen Zahl von Argumenten nicht interpretiert werden können (TypeError).
Beispiel: `>>> 5/"hello"`
Beispiel: `>>> type(4,5)`
- Der Python-Interpreter meldet Ihnen genau wo und welche Art von Fehler auftritt.
- Weiteres Beispiel für einen Semantikfehler: `>>> 5/(4 % 2)` (was passiert hier?)
- Noch ein Beispiel für einen Semantikfehler: `>>> 'hello'[6]` (was passiert hier?)
- **Lesen Sie bitte die Meldungen des Python-Interpreters immer und gründlich!**

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))`

vordefinierte Funktionen

- Mit `dir(__builtins__)` können Sie sich die vordefinierten Funktionen anzeigen lassen (es werden nicht nur Funktionen angezeigt).
- Mit `help(function_name)` erhalten Sie Hilfsinformationen zu der Funktion `function_name`

Schauen Sie sich bitte die Definitionen der folgenden Funktionen an:

`len`, `round`, `abs`, `set`