

Informatik I: Einführung in die Programmierung

4. Funktionen: Aufrufe und Definitionen

Albert-Ludwigs-Universität Freiburg



Bernhard Nebel
20./24. Oktober 2017

1 Funktionsaufrufe



- Syntax
- Standardfunktionen
- Exkurs: Zeichenkodierung und Unicode

- Funktions-Aufrufe
 - Syntax
 - Standardfunktionen
 - Exkurs: Zeichenkodierung und Unicode
- Mathematische Funktionen
- Funktions-Definition
- Namensraum
- Rückgabewerte

20./24. Oktober 2017

B. Nebel – Info I

3 / 36

Funktionsaufrufe



- Innerhalb der Mathematik sind Funktionen **Abbildungen** von einem Definitionsbereich in einen Bildbereich.
- Innerhalb von Programmiersprachen ist eine Funktion ein **Programmstück** (meistens mit einem Namen versehen).
- Normalerweise erwartet eine Funktion **Argumente** und gibt einen **Funktionswert** (oder **Rückgabewert**) zurück, und berechnet also eine Abbildung – aber **Seiteneffekte** Abhängigkeit von **globalen Variablen** sind möglich.
- type-Funktion:

Python-Interpreter

```
>>> type(42)  
<class 'int'>
```

- Funktion mit variabler Anzahl von Argumenten und ohne Rückgabewert (aber mit **Seiteneffekt**): `print`
- Funktion ohne Argumente und ohne Rückgabewert: `exit`

- Funktions-Aufrufe
 - Syntax
 - Standardfunktionen
 - Exkurs: Zeichenkodierung und Unicode
- Mathematische Funktionen
- Funktions-Definition
- Namensraum
- Rückgabewerte

20./24. Oktober 2017

B. Nebel – Info I

4 / 36

Standardfunktionen: Typen-Konversion



Mit den Funktionen `int`, `float`, `complex`, `str` kann man „passende“ Werte in den jeweiligen Typ umwandeln. Umwandlung nach `int` durch „Abschneiden“.

Python-Interpreter

```
>>> int(-2.6)  
-2  
>>> int('vier')  
File "<stdin>", line 1, in <module>  
ValueError: invalid literal for int() ...  
>>> complex('42')  
(42+0j)  
>>> float(4)  
4.0  
>>> str(42)  
'42'
```

- Funktions-Aufrufe
 - Syntax
 - Standardfunktionen
 - Exkurs: Zeichenkodierung und Unicode
- Mathematische Funktionen
- Funktions-Definition
- Namensraum
- Rückgabewerte

20./24. Oktober 2017

B. Nebel – Info I

5 / 36

abs liefert den Absolutwert (auch bei complex), round rundet, und pow berechnet die Exponentiation bei zwei Argumenten oder die Exponentiation modulo dem dritten Argument.

Python-Interpreter

```
>>> abs(-2)
2
>>> abs(1+1j)
1.4142135623730951
>>> round(2.500001)
3
>>> pow(2, 3)
8
>>> pow(2, 3, 4)
0
```

Funktions-Aufrufe
Syntax
Standardfunktionen
Exkurs: Zeichenkodierung und Unicode
Mathematische Funktionen
Funktions-Definition
Namensraum
Rückgabewerte

Mit den Funktionen chr und ord kann man Zahlen in **Unicode-Zeichen** und umgekehrt umwandeln, wobei in Python Zeichen identisch mit einbuchstabigen Strings sind:

Python-Interpreter

```
>>> chr(42)
'*'
>>> chr(255)
'ÿ'
>>> ord('*')
42
>>> ord('**')
Traceback (most recent call last): ...
TypeError: ord() expected a character, but string of length 2 found
```

Funktions-Aufrufe
Syntax
Standardfunktionen
Exkurs: Zeichenkodierung und Unicode
Mathematische Funktionen
Funktions-Definition
Namensraum
Rückgabewerte

- Computer kann man dafür nutzen, **Berechnungen** durchzuführen.
- Sehr früh hat man aber auch begonnen, mit dem Computer **Texte zu verarbeiten**
- Wie stellt man Texte im Computer dar?
- Man weist jedem Buchstaben einen Zahlenwert zu. Texte sind dann Sequenzen von solchen Codezahlen.
- Damit wird dann auch **Textverarbeitung** zu einer Berechnung.

Funktions-Aufrufe
Syntax
Standardfunktionen
Exkurs: Zeichenkodierung und Unicode
Mathematische Funktionen
Funktions-Definition
Namensraum
Rückgabewerte

- Einer der ersten Zeichenkodes war **ASCII** (American Standard Code for Information Interchange) – entwickelt für Fernschreiber.

USASCII code chart

Code	0	1	2	3	4	5	6	7
0	NUL	DLE	SP	@	P	^	_	~
1	SOH	DC1	!	A	Q	e	q	
2	STX	DC2	"	B	R	b	r	
3	ETX	DC3	#	C	S	c	s	
4	EOF	DC4	\$	D	T	d	t	
5	ENO	NAK	%	E	U	e	u	
6	ACK	SYN	&	F	V	f	v	
7	BEL	ETB	'	G	W	g	w	
8	BS	CAN	(H	X	h	x	
9	HT	EM)	I	Y	i	y	
10	LF	SUB	*	:	J	z	~	
11	VT	ESC	+	;	K	[^	
12	FF	FS	,	<	L	\		
13	CR	GS	-	=	M]	_	
14	SO	RS	.	>	N	^	~	
15	SI	US	/	?	O	~	DEL	

- Benötigt 7 Bits und enthält alle druckbaren Zeichen der englischen Sprache sowie nicht-druckbare Steuerzeichen (z.B. Zeilenwechsel).

Funktions-Aufrufe
Syntax
Standardfunktionen
Exkurs: Zeichenkodierung und Unicode
Mathematische Funktionen
Funktions-Definition
Namensraum
Rückgabewerte

- In anderen Sprachen wurden **zusätzliche Zeichen** benötigt.
- Da mittlerweile praktisch alle Rechner **8-Bit-Bytes** als kleinste Speichereinheit nutzen, kann man die höherwertigen Codes (128–255) für Erweiterungen nutzen.
- Diverse Erweiterungen, z.B. **ISO-Latin-1** (mit Umlauten) usw.
- Auf dem IBM-PC gab es andere Erweiterungen.
- Sprachen, die nicht auf dem lateinischen Alphabet basieren, haben große Probleme.

- Funktions-Aufrufe
- Syntax
- Standardfunktionen
- Exkurs: Zeichenkodierung und Unicode
- Mathematische Funktionen
- Funktions-Definition
- Namensraum
- Rückgabewerte

- Um für alle Sprachräume einen einheitlichen Zeichencode zu haben, wurde **Unicode** entwickelt (Version 1.0 im Jahr 1991).
- Mittlerweile (Juni 2015, Version 8.0) enthält Unicode **120737** Codepoints.
- Organisiert in 17 Ebenen mit jeweils 2^{16} Codepoints (manche allerdings ungenutzt)
- Die ersten 128 Codepoints stimmen mit ASCII überein, die ersten 256 mit ISO-Latin-1.

- Funktions-Aufrufe
- Syntax
- Standardfunktionen
- Exkurs: Zeichenkodierung und Unicode
- Mathematische Funktionen
- Funktions-Definition
- Namensraum
- Rückgabewerte

- Man kann Unicode-Zeichen als eine 32-Bit-Zahl darstellen (**UTF-32** oder UCS-4).
- Da man meist nur die Ebene 0 benötigt, ist es effizienter, die Kodierung **UTF-16** einzusetzen, bei der die Ebene 0 direkt als 16-Bit-Zahl kodiert wird. Zeichen aus anderen Ebenen benötigen 32 Bit.
- Im WWW wird meist **UTF-8** eingesetzt:

Unicode	UTF-8 binär
0–127	0xxxxxxx
128–2047	110xxxxx 10xxxxxx
2048–65535	1110xxxx 10xxxxxx 10xxxxxx
65536–1114111	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx

- Wie kommen die **komischen Zeichen** auf Webseiten zustande?
- Oft sind ISO-Latin-1/UTF-8 Verwechslungen der Grund!

- Funktions-Aufrufe
- Syntax
- Standardfunktionen
- Exkurs: Zeichenkodierung und Unicode
- Mathematische Funktionen
- Funktions-Definition
- Namensraum
- Rückgabewerte

- `math`-Modul
- Direktimport

- Funktions-Aufrufe
- Mathematische Funktionen
- `math`-Modul
- Direktimport
- Funktions-Definition
- Namensraum
- Rückgabewerte



- Natürlich wollen wir Funktionen wie `sin` verwenden. Die muss man in Python aber erst durch **Importieren** des **Mathematik-Moduls** bekannt machen.
- Danach können wir die Teile des Moduls durch **Voranstellen** von `math.` nutzen (**Punkt Schreibweise**):

Funktions-
Aufrufe
Mathemati-
sche
Funktionen
math-Modul
Direktimport
Funktions-
Definition
Namens-
raum
Rückgabe-
werte

Python-Interpreter

```
>>> import math
>>> math.pi
3.141592653589793
>>> math.sin(1/4*math.pi)
0.7071067811865475
>>> math.sin(math.pi)
1.2246467991473532e-16
>>> math.exp(math.log(2))
2.0
```



- Die Punkt Schreibweise verhindert **Namenskollisionen**, ist aber umständlich
- Mit `from module import name` kann ein Name direkt importiert werden.
- `from module import *` werden alle Namen direkt importiert.

Funktions-
Aufrufe
Mathemati-
sche
Funktionen
math-Modul
Direktimport
Funktions-
Definition
Namens-
raum
Rückgabe-
werte

Python-Interpreter

```
>>> from math import pi
>>> pi
3.141592653589793
>>> from math import *
>>> cos(pi)
-1.0
```



- Definition
- Einrückungen
- Aufruf
- Argumente und Parameter

Funktions-
Aufrufe
Mathemati-
sche
Funktionen
Funktions-
Definition
Definition
Einrückungen
Aufruf
Argumente und
Parameter
Namens-
raum
Rückgabe-
werte



- Mit dem Schlüsselwort `def` kann man eine neue Funktion einführen.
- Nach `def` kommt der **Funktionsname** gefolgt von der Parameterliste und dann ein Doppelpunkt.
- Nach dem **Funktionskopf** gibt der Python-Interpreter das **Funktionsprompt**-Zeichen `...` aus.
- Dann folgt der **Funktionsrumpf**: *Gleich weit eingerückte Anweisungen*, z.B. Zuweisungen oder Funktionsaufrufe:

Funktions-
Aufrufe
Mathemati-
sche
Funktionen
Funktions-
Definition
Definition
Einrückungen
Aufruf
Argumente und
Parameter
Namens-
raum
Rückgabe-
werte

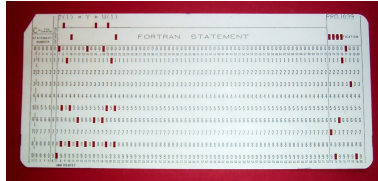
Python-Interpreter

```
>>> def print_lyrics():
...     print("I'm a lumberjack, and I'm okay")
...     print("I sleep all night and I work all day")
...
>>>
```

Einrückungen in Python



- Im Gegensatz zu fast allen anderen Programmiersprachen (außer z.B. FORTRAN, Miranda, Haskell), sind **Einrückungen** am Zeilenanfang bedeutungstragend.



- In Python ist gleiche Einrückung = zusammen gehöriger Block von Anweisungen
- In den meisten anderen Programmiersprachen durch Klammerung { } oder klammernde Schlüsselwörter.
- Wie viele Leerzeichen sollte man machen?
→ **PEP8**: 4 Leerzeichen pro Ebene (keine Tabs nutzen!)

Funktions-
Aufrufe
Mathemati-
sche
Funktionen
Funktions-
Definition
Definition
Einrückungen
Aufruf
Argumente und
Parameter
Namens-
raum
Rückgabe-
werte

Selbst definierte Funktionen nutzen



- Funktionsnamen müssen den gleichen Regeln folgen wie Variablennamen.
- Tatsächlich verhalten sich Funktionsnamen wie Variablennamen und haben einen entsprechenden Typ.
- Man kann eigene Funktionen wie Standardfunktionen aufrufen

Python-Interpreter

```
>>> print(print_lyrics)
<function print_lyrics at 0x100520560>
>>> type(print_lyrics)
<class 'function'>
>>> print_lyrics()
I'm a lumberjack, and I'm okay
I sleep all night and I work all day
>>> print_lyrics = 42
```

Funktions-
Aufrufe
Mathemati-
sche
Funktionen
Funktions-
Definition
Definition
Einrückungen
Aufruf
Argumente und
Parameter
Namens-
raum
Rückgabe-
werte

Definierte Funktionen in Funktionsdefinitionen



Was passiert hier?

Python-Interpreter

```
>>> def print_lyrics():
...     print("I'm a lumberjack, and I'm okay")
...     print("I sleep all night and I work all day")
...
>>>
>>> def repeat_lyrics():
...     print_lyrics()
...     print_lyrics()
...
>>> repeat_lyrics()
I'm a lumberjack ...
```

Was wird hier exakt ausgeführt?

Funktions-
Aufrufe
Mathemati-
sche
Funktionen
Funktions-
Definition
Definition
Einrückungen
Aufruf
Argumente und
Parameter
Namens-
raum
Rückgabe-
werte

Argumente und Parameter



- Auch definierte Funktionen benötigen oft *Argumente*.
- Bei der Definition gibt man *Parameter* an, die beim Aufruf durch die *Argumente* ersetzt werden.

Python-Interpreter

```
>>> michael = 'baldwin'
>>> def print_twice(bruce):
...     print(bruce)
...     print(bruce)
...
>>> print_twice(michael)
baldwin
baldwin
>>> print_twice('Spam ' * 3)
Spam Spam Spam
Spam Spam Spam
```

Funktions-
Aufrufe
Mathemati-
sche
Funktionen
Funktions-
Definition
Definition
Einrückungen
Aufruf
Argumente und
Parameter
Namens-
raum
Rückgabe-
werte

Funktionen als Argumente



- Wir können Funktionen wie andere Werte als Argumente übergeben.

Python-Interpreter

```
>>> def do_twice(f):
...     f()
...     f()
...
>>> do_twice(print_lyrics)
I'm a lumberjack, and I'm okay
I sleep all night and I work all day
I'm a lumberjack, and I'm okay
I sleep all night and I work all day
```

- Das schauen wir uns in der 2. Hälfte des Semesters noch genauer an!

Funktions-Aufrufe
 Mathematische Funktionen
 Funktions-Definition
 Definition
 Einrückungen
 Aufruf
 Argumente und Parameter
 Namensraum
 Rückgabewerte

4 Namensraum



- Lokale Variablen und Parameter
- Stapeldiagramm
- Traceback
- Globale Variablen

Funktions-Aufrufe
 Mathematische Funktionen
 Funktions-Definition
 Namensraum
 Lokale Variablen und Parameter
 Stapeldiagramm
 Traceback
 Globale Variablen
 Rückgabewerte

Namensraum von lokalen Variablen und Parametern



- Parameter sind nur innerhalb der Funktion sichtbar.
- Lokal (durch Zuweisung) eingeführte Variablen ebenfalls.

Python-Interpreter

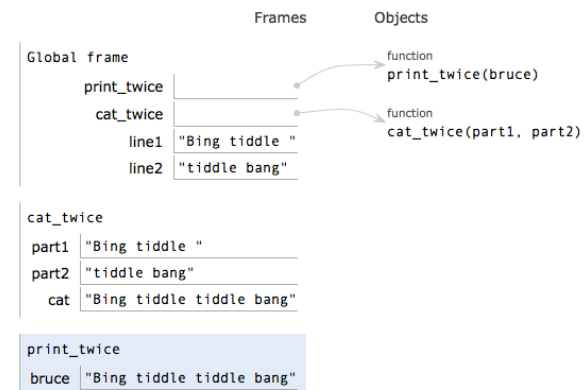
```
>>> def cat_twice(part1, part2):
...     cat = part1 + part2
...     print_twice(cat)
...
>>> line1 = 'Bing tiddle '
>>> line2 = 'tiddle bang.'
>>> cat_twice(line1, line2)
Bing tiddle tiddle bang.
Bing tiddle tiddle bang.
>>> cat
NameError: name 'cat' is not defined
```

Funktions-Aufrufe
 Mathematische Funktionen
 Funktions-Definition
 Namensraum
 Lokale Variablen und Parameter
 Stapeldiagramm
 Traceback
 Globale Variablen
 Rückgabewerte

Stapeldiagramm



- Entsprechend zu den Zustandsdiagrammen kann man die Variablenbelegungen in **Stapeldiagrammen** visualisieren (hier hilft pythontutor.com).
 Innerhalb von `print_twice`:



Funktions-Aufrufe
 Mathematische Funktionen
 Funktions-Definition
 Namensraum
 Lokale Variablen und Parameter
 Stapeldiagramm
 Traceback
 Globale Variablen
 Rückgabewerte

- Tritt bei der Ausführung einer Funktion ein Fehler auf, z.B. Zugriff auf die nicht vorhandene Variable `cat` in `print_twice`, dann gibt es ein **Traceback** (entsprechend zu unserem Stapeldiagramm):

Python-Interpreter

```
>>> cat_twice(line1, line2)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 3, in cat_twice
  File "<stdin>", line 3, in print_twice
NameError: global name 'cat' is not defined
```

- Funktions-Aufrufe
- Mathematische Funktionen
- Funktions-Definition
- Namensraum
- Lokale Variablen und Parameter Stapeldiagramm
- Traceback
- Globale Variablen
- Rückgabewerte

- Man sollte nur **lokale Variable** und Parameter nutzen.
- Man kann **lesend** auf globale Variablen zugreifen, falls es nicht eine lokale Variable gleichen Namens gibt.
- Manchmal möchte man aber auch **globale Variablen** ändern (z.B. zur globalen Moduseinstellung oder für Zähler): Schlüsselwort `global`.

Python-Interpreter

```
>>> counter = 0
>>> def inc():
...     global counter
...     counter = counter + 1
...
>>> inc()
>>> counter
1
```

- Funktions-Aufrufe
- Mathematische Funktionen
- Funktions-Definition
- Namensraum
- Lokale Variablen und Parameter Stapeldiagramm
- Traceback
- Globale Variablen
- Rückgabewerte

- Funktions-Aufrufe
- Mathematische Funktionen
- Funktions-Definition
- Namensraum
- Rückgabewerte

- Funktionen können einen Wert zurückgeben, wie z.B. `chr` oder `sin`.
- Einige Funktionen haben keinen Rückgabewert, weil sie nur einen (Seiten-)Effekt verursachen sollen, wie z.B. `inc` und `print`.
- Tatsächlich geben diese den speziellen Wert `None` zurück.

Python-Interpreter

```
>>> result = print('Bruce')
Bruce
>>> result
>>> print(result)
None [≠ der String 'None!']
```

- `None` ist der einzige Wert des Typs `NoneType`.

- Funktions-Aufrufe
- Mathematische Funktionen
- Funktions-Definition
- Namensraum
- Rückgabewerte

- Soll die Funktion einen Wert zurück geben, müssen wir das Schlüsselwort **return** benutzen.

Python-Interpreter

```
>>> def sum3(a, b, c):  
...     return a + b + c  
...  
>>> sum3(1, 2, 3)  
6
```

- Können wir nicht auch `print(·)` benutzen, um einen Funktionswert zurück zu geben?

Python-Interpreter

```
>>> def printsum3(a, b, c):  
...     print(a + b + c)  
...  
>>> sum3(1, 2, 3)  
6  
>>> sum3(1, 2, 3) + 4  
10  
>>> printsum3(1, 2, 3) + 4  
6  
TypeError: unsupported operand type(s) for +:  
'NoneType' and 'int'
```

- **Funktionen** sind benannte vorgegebene Programmstücke (Standardfunktionen) oder selbst definierte Funktionen.
- Beim Aufruf einer Funktion müssen **Argumente** angegeben werden, die die formalen **Parameter** mit Werten belegen.
- Funktionen geben normalerweise einen **Funktionswert** zurück: `return`.
- Funktionen führen einen neuen **Namensraum** ein für die Parameter und **lokalen** Variablen (durch Zuweisung eingeführt).
- Lesend kann man immer auf **globale** Variablen zugreifen, schreibend mit Hilfe des `global`-Schlüsselworts.
- pythontutor.com visualisiert die Programmausführung mit Hilfe von Zustands-/Stapeldiagrammen