Informatik I: Einführung in die Programmierung

4. Funktionen: Aufrufe und Definitionen

Albert-Ludwigs-Universität Freiburg



Prof. Dr. Peter Thiemann

28. Oktober 2026

1 Funktionsaufrufe

- Syntax
- Standardfunktionen
- Exkurs: Zeichenkodierung und Unicode

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

Funktionsaufrufe

- Funktionen sind Abbildungen von einem Definitionsbereich in einen Bildbereich.
- Eine Funktion erwartet Argumente aus dem Definitionsbereich und gibt einen Funktionswert (oder *Rückgabewert*) aus dem Bildbereich zurück.
- Eine Funktion kann Effekte haben, z.B.:
  - eine Ausgabe erzeugen,
  - eine Eingabe lesen,
  - uvam
- Viele Standardfunktionen sind in Python vordefiniert.

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

Standardfunktionen: Typ-Konversion

Die Funktionen `int`, `float` und `str` können „passende“ Werte in den jeweiligen Typ konvertieren.

Python-Interpreter

```
>>> int(-2.6) # Konversion nach int durch Abschneiden.
-2
>>> int('vier')
 File "<stdin>", line 1, in <module>
ValueError: invalid literal for int() ...
>>> float(4)
4.0
>>> str(42)
'42'
```

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

Standardfunktionen: Ein-/Ausgabe

- Bekannt: `print` gibt Werte aus.
- Die Funktion `input` gibt einen Wert aus (wie `print`) und liest dann einen String ein.

Python-Interpreter

```
>>> input("Gib mir einen Keks: ")
Gib mir einen Keks: Keks
'Keks'
>>> name = input("Wie heißt du? ")
Wie heißt du? Oskar
>>> print("Hallo,", name + "!")
Hallo, Oskar!
```

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

Standardfunktionen kombiniert: Eingabe von Zahlen

Da `input` nur Strings einliest, muss die Eingabe konvertiert werden!

Python-Interpreter

```
>>> CM_PER_INCH = 2.54
>>> länge = input("Länge in cm: ")
Länge in cm: 195
>>> länge # ein String
'195'
>>> länge_cm = float(länge)
>>> länge_inches = länge_cm / CM_PER_INCH
>>> print(länge + "cm", "=", str(länge_inches) + "in")
195cm = 76.77165354330708in
```

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

Standardfunktionen: Numerische Funktionen

- `abs` liefert den Absolutwert
- `round` rundet.

```
>>> abs(-2)
2
>>> abs(42)
42
>>> round(2.500001)
3
```

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

Standardfunktionen: Zeichenkonversion

- Die Funktion `chr` konvertiert ein `int` in ein Unicode-Zeichen.
- Die Funktion `ord` konvertiert in die umgekehrte Richtung.
- In Python werden Zeichen durch einbuchstabige Strings dargestellt.

```
>>> chr(42)
'*'
>>> chr(255)
'ÿ'
>>> ord('*')
42
>>> ord('**')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: ord() expected a character, but string of length 2 found
```

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

Kleiner Exkurs: Zeichenkodierung

- Computer können Berechnungen durchführen.
- Seit langem werden mit dem Computer auch Texte verarbeitet.
- Wie werden Texte im Computer dargestellt?
- Jedes Zeichen wird durch eine Zahl kodiert. Texte sind Sequenzen von solchen Kodezahlen.
- Damit wird auch die **Textverabeitung** zu einer Berechnung.

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

ASCII

- Einer der ersten Zeichenkodes war ASCII (American Standard Code for Information Interchange) – entwickelt für Fernschreiber und Lochstreifen.
- Benötigt 7 Bits und enthält alle druckbaren Zeichen der englischen Sprache sowie nicht-druckbare Steuerzeichen (z.B. Zeilenwechsel).

USASCII code chart

Bits b_7 b_6 b_5	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
b_4 b_3 b_2 b_1 (Row)	0	1	2	3	4	5	6	7
0 0 0 0	0 NUL	DLE	SP	0	@	P	`	p
0 0 0 1	1 SOH	DC1	!	1	A	Q	a	q
0 0 1 0	2 STX	DC2	"	2	B	R	b	r
0 0 1 1	3 ETX	DC3	#	3	C	S	c	s
0 1 0 0	4 EOT	DC4	$	4	D	T	d	t
0 1 0 1	5 ENQ	NAK	%	5	E	U	e	u
0 1 1 0	6 ACK	SYN	&	6	F	V	f	v
0 1 1 1	7 BEL	ETB	'	7	G	W	g	w
1 0 0 0	8 BS	CAN	(	8	H	X	h	x
1 0 0 1	9 HT	EM	)	9	I	Y	i	y
1 0 1 0	10 LF	SUB	*	:	J	Z	j	z
1 0 1 1	11 VT	ESC	+	;	K	[	k	{
1 1 0 0	12 FF	FS	,	<	L	\	l	|
1 1 0 1	13 CR	GS	-	=	M	]	m	}
1 1 1 0	14 SO	RS	.	>	N	^	n	~
1 1 1 1	15 SI	US	/	?	O	_	o	DEL

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

Erweitertes ASCII

- In anderen Sprachen wurden zusätzliche Zeichen benötigt.
- Da praktisch alle Rechner 8-Bit-Bytes als kleinste Speichereinheit nutzten, standen die Kodes mit gesetztem höchstwertigen Bit (128–255) für Erweiterungen zur Verfügung.
- Diverse Erweiterungen, z.B. ISO-Latin-1 (mit Umlauten, seit 1987).
- Auf dem IBM-PC gab es andere Erweiterungen, Windows-1252.
- Sprachen, die nicht auf dem lateinischen Alphabet basieren, haben große Probleme, ISO-2022-JP: 日本語を分かりますか

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

Unicode

- Um für alle Sprachräume eine einheitliche Kodierung für Zeichen zu haben, wurde Unicode entwickelt (Version 1.0 im Jahr 1991).
- Seit September 2025 (Version 17.0) unterstützt Unicode 172 Schriften mit 159,801 Codepoints, darunter 3,790 Emojis.
- 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.
- Zum Thema Emojis gibt es ein eigenes Subkomitee ...
- Im Mai 2019 gab es außerplanmäßig Version 12.1, bei der ein einziges Zeichen hinzugefügt wurde?!

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

UTF-32, UTF-16 und UTF-8

- Ein Unicode-Zeichen kann durch eine 32-Bit-Zahl dargestellt werden (UTF-32 oder UCS-4).
- Meist wird nur die Ebene 0 benötigt. Daher ist es effizienter, die Kodierung UTF-16 einzusetzen, bei der Zeichen der Ebene 0 direkt als 16-Bit-Zahl kodiert werden. Zeichen aus anderen Ebenen benötigen 32 Bit.
- Im WWW wird meist UTF-8 eingesetzt, eine Kodierung mit variabler Länge:

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 / in Emails zustande?
- Oft sind ISO-Latin-1/UTF-8 Verwechslungen der Grund!

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

2 Mathematische Funktionen

- `math`-Modul
- Direktimport

Funktions-Aufrufe
Mathematische Funktionen
math-Modul
Direktimport
Funktions-Definition
Sichtbarkeit

Mathematische Funktionen: Das Math-Modul

- Funktionen wie `sin` stehen nicht direkt zur Verfügung. Sie müssen durch Importieren des Mathematik-Moduls `math` bekannt gemacht werden.
- Werte aus dem Modul können durch Voranstellen von `math.` genutzt werden (Punktschreibweise):

```
>>> 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
```

Funktions-Aufrufe
Mathematische Funktionen
math-Modul
Direktimport
Funktions-Definition
Sichtbarkeit

Mathematische Funktionen: Direkt importieren

- Die Punktschreibweise verhindert Namenskollisionen, ist aber umständlich.
- Direkter Import eines Bezeichners:
  `from` *module* `import` *name*
- Direkter Import aller Bezeichner eines Moduls:
  `from` *module* `import *`

```
>>> pi = 3.14
>>> pi
3.14
>>> from math import *
>>> cos(pi)
-1.0
```

Funktions-Aufrufe
Mathematische Funktionen
math-Modul
Direktimport
Funktions-Definition
Sichtbarkeit

Mathematische Funktionen: isclose

- Gleitkommazahlen sollten nicht mit == verglichen werden.
- Besser: Prüfen, ob zwei Zahlen nah beieinander sind.
- Verwende `math.isclose`

```
>>> import math
>>> d : float = 2 - 2.1
>>> print(d)
-0.10000000000000009
>>> print(d == -0.1)
False
>>> print(math.isclose(d, -0.1))
True
```

Funktions-Aufrufe
Mathematische Funktionen
math-Modul
Direktimport
Funktions-Definition
Sichtbarkeit

3 Funktionsdefinitionen

- Definition
- Einrückungen
- Aufruf
- Argumente, Parameter, Rückgabewerte

Funktions-Aufrufe
Mathematische Funktionen
Funktions-Definition
Definition
Einrückungen
Aufruf
Argumente, Parameter, Rückgabewerte
Sichtbarkeit

Funktionen definieren

- Ein Python-Programm kann selbst neue Funktionen definieren.
- Hier: systematische Herangehensweise mit einer Konstruktionsanleitung
  - Kurzbeschreibung
  - Signatur
  - Tests
  - Gerüst
  - Rumpf
- In dieser Reihenfolge…

Funktions-Aufrufe
Mathematische Funktionen
Funktions-Definition
Definition
Einrückungen
Aufruf
Argumente, Parameter, Rückgabewerte
Sichtbarkeit

Beispiel: Temperaturumrechnung

Kurzbeschreibung

Ein Kommentar, der den Zweck kurz in einem Satz beschreibt.

```
# Temperatur von Celsius nach Fahrenheit umrechnen
```

Signatur

Direkt nach der Kurzbeschreibung. Aussagekräftige Namen. Typen der Ein- und Ausgaben möglichst präzise.

```
def fahrenheit_from_celsius (celsius : float) -> float: ...
```

Funktions-Aufrufe
Mathematische Funktionen
Funktions-Definition
Definition
Einrückungen
Aufruf
Argumente, Parameter, Rückgabewerte
Sichtbarkeit

Beispiel: Temperaturumrechnung

Möglichst einfache Beispiele, die interessante Fälle abdecken. Ergebnisse von Hand berechnen. Randfälle beachten!

Tests

```
def test_fahrenheit_from_celsius():
    assert math.isclose(fahrenheit_from_celsius(0.0), 32.0)
    assert math.isclose(fahrenheit_from_celsius(100.0), 212.0)
    assert math.isclose(fahrenheit_from_celsius(37.7777778), 100.0)
    assert math.isclose(fahrenheit_from_celsius (-40), -40)
```

Neue Anweisung: `assert` *Ausdruck*

- Stoppt mit Fehler, wenn der Wert von *Ausdruck* nicht true ist

Funktions-Aufrufe
Mathematische Funktionen
Funktions-Definition
Definition
Einrückungen
Aufruf
Argumente, Parameter, Rückgabewerte
Sichtbarkeit

Beispiel: Temperaturumrechnung

Funktions-Aufrufe
Mathematische Funktionen
Funktions-Definition
Definition
Einrückungen
Aufruf
Argumente, Parameter, Rückgabewerte
Sichtbarkeit

Gerüst / Rumpf

Hier nur: Funktionen liefern einen Wert zurück

```
def fahrenheit_from_celsius (celsius : float) -> float:
    return celsius * 9.0 / 5.0 + 32.0
```

Neue Anweisung: `return` *`Ausdruck`*

- Beendet eine Funktion und liefert den Wert von *`Ausdruck`* als Ergebnis des Funktionsaufrufs.

Beispiel: Temperaturumrechnung

Funktions-Aufrufe
Mathematische Funktionen
Funktions-Definition
Definition
Einrückungen
Aufruf
Argumente, Parameter, Rückgabewerte
Sichtbarkeit

Epilog

Zum Abschluss starten die Tests ...

```
test_fahrenheit_from_celsius()
```

Neue Funktionen definieren

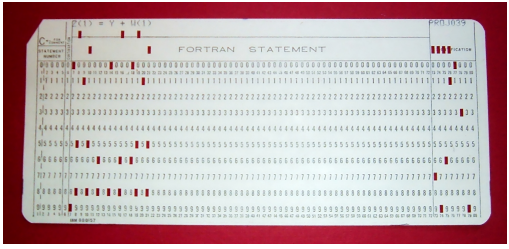
```
def bezeichner(formaler_parameter : typ) -> return_typ:
    Anweisungen      # Funktionsrumpf
    return Ausdruck
```

- Eine Funktionsdefinition beginnt mit dem Schlüsselwort `def`, danach kommt der Funktionsname gefolgt von der Liste der formalen Parameter und dem Rückgabetyp, dann ein Doppelpunkt.
- Der Funktionsrumpf ist ein Block von *gleich weit eingerückten* Anweisungen, z.B. Zuweisungen oder Funktionsaufrufe.
- Die Elemente der Parameterliste haben die Form *Bezeichner* : *Typ* in runden Klammern getrennt durch Kommas.
- Der Wert des Ausdrucks nach `return` wird zum Wert des Funktionsaufrufs.

Funktions-Aufrufe
Mathematische Funktionen
Funktions-Definition
Definition
Einrückungen
Aufruf
Argumente, Parameter, Rückgabewerte
Sichtbarkeit

Einrückungen in Python

- Einrückungen am Zeilenanfang sind bedeutungstragend. Vgl FORTRAN:



- Gleiche Einrückung = zusammengehöriger Block von Anweisungen
- In den meisten anderen Programmiersprachen durch Klammerung { } oder klammernde Schlüsselwörter.
- Wie viele Leerzeichen sollen verwendet werden?

→ PEP8: 4 Leerzeichen pro Ebene der Einrückung (*keine Tabs* verwenden!)

Funktions-Aufrufe
Mathematische Funktionen
Funktions-Definition
Definition
Einrückungen
Aufruf
Argumente, Parameter, Rückgabewerte
Sichtbarkeit

Selbst definierte Funktionen nutzen

- Funktionsnamen sind Variablennamen
- Funktionen haben einen speziellen Typ
- Selbstdefinierte Funktionen werden wie Standardfunktionen aufgerufen

Funktions-Aufrufe
Mathematische Funktionen
Funktions-Definition
Definition
Einrückungen
Aufruf
Argumente, Parameter, Rückgabewerte
Sichtbarkeit

Argumente, Parameter, Rückgabewerte

Beispiel

Entwickle eine Funktion, die Zentimeter in Inches umrechnet.

Konstruktionsanleitung

- Kurzbeschreibung
- Signatur
- Tests
- Gerüst
- Rumpf

Funktions-Aufrufe
Mathematische Funktionen
Funktions-Definition
Definition
Einrückungen
Aufruf
Argumente, Parameter, Rückgabewerte
Sichtbarkeit

4 Sichtbarkeit

- Lokale Variablen und Parameter
- Kellertabelle
- Traceback
- Globale Variable

Funktions-Aufrufe
Mathematische Funktionen
Funktions-Definition
Sichtbarkeit
Lokale Variablen und Parameter
Kellertabelle
Traceback
Globale Variable

Sichtbarkeit von lokalen Variablen und Parametern

- Parameter (`part1`, `part2`) sind nur innerhalb der Funktion sichtbar.
- Lokal (durch Zuweisung) eingeführte Variablen (`cat`) ebenfalls.
- Globale Variable sind weiterhin lesbar.

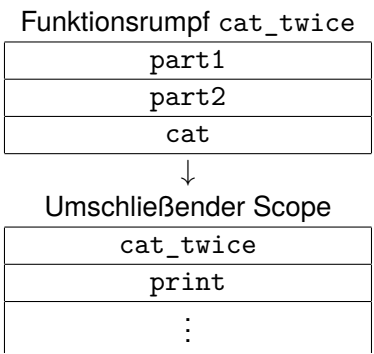
cat.py

```
def cat_twice (part1 : str, part2 : str) -> str:
    cat : str = part1 + part2
    print(cat)
    return cat * 2
```

Funktions-Aufrufe
Mathematische Funktionen
Funktions-Definition
Sichtbarkeit
Lokale Variablen und Parameter
Kellertabelle
Traceback
Globale Variable

Umgebung (Scope)

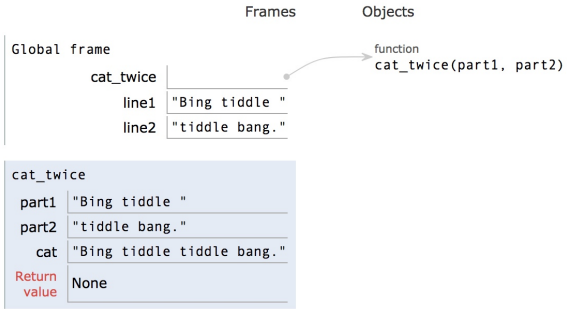
- Der Rumpf einer Funktion bildet eine Umgebung (Scope).
- Die Umgebung bindet die lokal definierten Variablen der Funktion (inkl. Parameter).
- Alle weiteren Variablen sind frei im Funktionsrumpf und beziehen sich auf einen umschließenden Scope.
- Beim Aufruf der Funktion wird für den Scope ein Kellerrahmen (stack frame) für die Werte der Variablen angelegt.



Funktions-Aufrufe
Mathematische Funktionen
Funktions-Definition
Sichtbarkeit
Lokale Variablen und Parameter
Kellertabelle
Traceback
Globale Variable

Kellertabelle

- Die Variablenbelegungen innerhalb von Funktionsaufrufen (also die Kellerrahmen) können durch eine Kellertabelle visualisiert werden (hier hilft `http://pythontutor.com/`). Ende von `cat_twice`



Funktions-Aufrufe
Mathematische Funktionen
Funktions-Definition
Sichtbarkeit
Lokale Variablen und Parameter
Kellertabelle
Traceback
Globale Variable

Globale Variable

- Funktionen sollen vorrangig lokale Variable und Parameter nutzen.
- Funktionen können globale Variablen lesen, falls diese nicht durch lokale Variable gleichen Namens verdeckt werden (shadowing).

Funktions-Aufrufe
Mathematische Funktionen
Funktions-Definition
Sichtbarkeit
Lokale Variablen und Parameter
Kellertabelle
Traceback
Globale Variable

Globale Variable (Beispiel)

globalvar.py

```
dude : int = 666
def depp() -> int:
    return dude
print("depp␣returns", depp())
def independent(dude : int) -> int:
    return dude
print("independent␣returns", independent(42))
def ignorant() -> int:
    dude : int = 333
    return dude
print("ignorant␣returns", ignorant(), "dude=", dude)
```

Funktions-Aufrufe
Mathematische Funktionen
Funktions-Definition
Sichtbarkeit
Lokale Variablen und Parameter
Kellertabelle
Traceback
Globale Variable

Zusammenfassung

- Funktionen sind benannte Programmstücke.
- Beim Aufruf einer Funktion müssen Argumente angegeben werden, die die formalen Parameter in der Definition mit Werten belegen.
- Funktionen geben einen Funktionswert zurück, der mit `return` festgelegt wird (sonst `None`).
- Jede Funktionsdefinition hat eine eigene Umgebung für die Parameter und lokalen Variablen (durch Zuweisung eingeführt).
- Globale Variablen können gelesen werden, falls sie nicht durch einen Parameter oder eine lokale Variable verdeckt werden.
- `pythontutor.com` visualisiert die Programmausführung mit Hilfe von Kellertabellen.

Funktions-Aufrufe
Mathematische Funktionen
Funktions-Definition
Sichtbarkeit
Lokale Variablen und Parameter
Kellertabelle
Traceback
Globale Variable