

Informatik I: Einführung in die Programmierung

4. Funktionen: Aufrufe und Definitionen

Albert-Ludwigs-Universität Freiburg



UNI
FREIBURG

Peter Thiemann

5. November 2019



Funktionsaufrufe

Funktions- Aufrufe

Syntax

Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati- sche Funktionen

Funktions- Definition

Namens- raum

Rückgabe- werte



- Funktionen sind **Abbildungen** von einem **Definitionsbereich** in einen **Bildbereich**.
- Definitionsbereich und Bildbereich werden durch Datentypen und ggf weitere Einschränkungen beschrieben.
- Eine Funktion erwartet **Argumente** aus dem Definitionsbereich und gibt immer 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

Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



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

Python-Interpreter

```
>>> int(-2.6) # Umwandlung nach int durch Abschneiden.
```

Funktions-
Aufrufe

Syntax

Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



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

Python-Interpreter

```
>>> int(-2.6) # Umwandlung nach int durch Abschneiden.  
-2  
>>>
```

Funktions-
Aufrufe

Syntax

Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



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

Python-Interpreter

```
>>> int(-2.6) # Umwandlung nach int durch Abschneiden.  
-2  
>>> int('vier')
```

Funktions-
Aufrufe

Syntax

Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



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

Python-Interpreter

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

Funktions-
Aufrufe

Syntax

Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



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

Python-Interpreter

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

Funktions-
Aufrufe

Syntax

Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



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

Python-Interpreter

```
>>> int(-2.6) # Umwandlung nach int durch Abschneiden.  
-2  
>>> int('vier')  
File "<stdin>", line 1, in <module>  
ValueError: invalid literal for int() ...  
>>> complex('42')  
(42+0j)  
>>>
```

Funktions-
Aufrufe

Syntax

Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



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

Python-Interpreter

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

Funktions-
Aufrufe

Syntax

Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



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

Python-Interpreter

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

Funktions-
Aufrufe

Syntax

Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



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

Python-Interpreter

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

Funktions-
Aufrufe

Syntax
Standardfunktio-
nen
Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



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

Python-Interpreter

```
>>> int(-2.6) # Umwandlung nach int durch Abschneiden.
-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

Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- Bekannt: `print` kann Werte ausgeben.
- Zur Eingabe kann die Funktion `input` **Strings** einlesen.

Python-Interpreter

```
>>> input("Gib mir einen Keks: ")
```

Funktions-
Aufrufe

Syntax

Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- Bekannt: `print` kann Werte ausgeben.
- Zur Eingabe kann die Funktion `input` **Strings** einlesen.

Python-Interpreter

```
>>> input("Gib mir einen Keks: ")  
Gib mir einen Keks:
```

Funktions-
Aufrufe

Syntax

Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- Bekannt: `print` kann Werte ausgeben.
- Zur Eingabe kann die Funktion `input` **Strings** einlesen.

Python-Interpreter

```
>>> input("Gib mir einen Keks: ")
Gib mir einen Keks: Keks
'Keks'
```

Funktions-
Aufrufe

Syntax

Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- Bekannt: print kann Werte ausgeben.
- Zur Eingabe kann die Funktion input **Strings** einlesen.

Python-Interpreter

```
>>> input("Gib mir einen Keks: ")
Gib mir einen Keks: Keks
'Keks'
>>> name = input("Wie heißt du? ")
Wie heißt du?
```

Funktions-
Aufrufe

Syntax

Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- Bekannt: print kann Werte ausgeben.
- Zur Eingabe kann die Funktion input **Strings** einlesen.

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

Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte

Standardfunktionen kombiniert: Eingabe von Zahlen



Zur Eingabe von Zahlen muss der von `input` eingelesene String konvertiert werden!

Python-Interpreter

```
>>> CM_PER_INCH = 2.54
>>> länge = input("Länge in cm: ")
```

Funktions-
Aufrufe

Syntax

Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte

Standardfunktionen kombiniert: Eingabe von Zahlen



Zur Eingabe von Zahlen muss der von `input` eingelesene String konvertiert werden!

Python-Interpreter

```
>>> CM_PER_INCH = 2.54
>>> länge = input("Länge in cm: ")
Länge in cm:
```

Funktions-
Aufrufe

Syntax

Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte

Standardfunktionen kombiniert: Eingabe von Zahlen



Zur Eingabe von Zahlen muss der von `input` eingelesene String konvertiert werden!

Python-Interpreter

```
>>> CM_PER_INCH = 2.54
>>> länge = input("Länge in cm: ")
Länge in cm: 195
>>> länge # ein String
```

Funktions-
Aufrufe

Syntax

Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte

Standardfunktionen kombiniert: Eingabe von Zahlen



Zur Eingabe von Zahlen muss der von `input` eingelesene String 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

Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- `abs` liefert den Absolutwert (auch bei `complex`)
- `round` rundet.

Python-Interpreter

```
>>> abs(-2)
```

Funktions-
Aufrufe

Syntax

Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- `abs` liefert den Absolutwert (auch bei `complex`)
- `round` rundet.

Python-Interpreter

```
>>> abs(-2)
2
>>>
```

Funktions-
Aufrufe

Syntax

Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- `abs` liefert den Absolutwert (auch bei `complex`)
- `round` rundet.

Python-Interpreter

```
>>> abs(-2)
2
>>> abs(1+1j)
```

Funktions-
Aufrufe

Syntax

Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- `abs` liefert den Absolutwert (auch bei `complex`)
- `round` rundet.

Python-Interpreter

```
>>> abs(-2)
2
>>> abs(1+1j)
1.4142135623730951
>>>
```

Funktions-
Aufrufe

Syntax

Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- `abs` liefert den Absolutwert (auch bei `complex`)
- `round` rundet.

Python-Interpreter

```
>>> abs(-2)
2
>>> abs(1+1j)
1.4142135623730951
>>> round(2.500001)
```

Funktions-
Aufrufe

Syntax

Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- `abs` liefert den Absolutwert (auch bei `complex`)
- `round` rundet.

Python-Interpreter

```
>>> abs(-2)
2
>>> abs(1+1j)
1.4142135623730951
>>> round(2.500001)
3
```

Funktions-
Aufrufe

Syntax

Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



Die Funktionen `chr` und `ord` wandeln Zahlen in **Unicode-Zeichen** um (und umgekehrt), wobei in Python Zeichen identisch mit einbuchstabigen Strings sind:

Python-Interpreter

```
>>> chr(42)
```

Funktions-
Aufrufe

Syntax

Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



Die Funktionen `chr` und `ord` wandeln Zahlen in **Unicode-Zeichen** um (und umgekehrt), wobei in Python Zeichen identisch mit einbuchstabigen Strings sind:

Python-Interpreter

```
>>> chr(42)
'*'
>>>
```

Funktions-
Aufrufe

Syntax

Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



Die Funktionen `chr` und `ord` wandeln Zahlen in **Unicode-Zeichen** um (und umgekehrt), wobei in Python Zeichen identisch mit einbuchstabigen Strings sind:

Python-Interpreter

```
>>> chr(42)
'*'
>>> chr(255)
```

Funktions-
Aufrufe

Syntax

Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



Die Funktionen `chr` und `ord` wandeln Zahlen in **Unicode-Zeichen** um (und umgekehrt), wobei in Python Zeichen identisch mit einbuchstabigen Strings sind:

Python-Interpreter

```
>>> chr(42)
'*'
>>> chr(255)
'ÿ'
>>>
```

Funktions-
Aufrufe

Syntax

Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



Die Funktionen `chr` und `ord` wandeln Zahlen in **Unicode-Zeichen** um (und umgekehrt), wobei in Python Zeichen identisch mit einbuchstabigen Strings sind:

Python-Interpreter

```
>>> chr(42)
'*'
>>> chr(255)
'ÿ'
>>> ord('*')
```

Funktions-
Aufrufe

Syntax

Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



Die Funktionen `chr` und `ord` wandeln Zahlen in **Unicode-Zeichen** um (und umgekehrt), wobei in Python Zeichen identisch mit einbuchstabigen Strings sind:

Python-Interpreter

```
>>> chr(42)
'*'
>>> chr(255)
'ÿ'
>>> ord('*')
42
>>>
```

Funktions-
Aufrufe

Syntax

Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



Die Funktionen `chr` und `ord` wandeln Zahlen in **Unicode-Zeichen** um (und umgekehrt), wobei in Python Zeichen identisch mit einbuchstabigen Strings sind:

Python-Interpreter

```
>>> chr(42)
'*'
>>> chr(255)
'ÿ'
>>> ord('*')
42
>>> ord('**')
```

Funktions-
Aufrufe

Syntax

Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



Die Funktionen `chr` und `ord` wandeln Zahlen in **Unicode-Zeichen** um (und umgekehrt), 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
Standardfunktio-
nen
Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- Computer können **Berechnungen** durchführen.

Funktions-
Aufrufe

Syntax
Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- Computer können **Berechnungen** durchführen.
- Seit langem werden mit dem Computer auch **Texte verarbeitet**.

Funktions-
Aufrufe

Syntax
Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- Computer können **Berechnungen** durchführen.
- Seit langem werden mit dem Computer auch **Texte verarbeitet**.
- Wie werden Texte im Computer dargestellt?

Funktions-
Aufrufe

Syntax
Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- Computer können **Berechnungen** durchführen.
- Seit langem werden mit dem Computer auch **Texte verarbeitet**.
- Wie werden Texte im Computer dargestellt?
- Jeder Buchstabe wird durch einen Zahlenwert codiert.

Funktions-
Aufrufe

Syntax
Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- Computer können **Berechnungen** durchführen.
- Seit langem werden mit dem Computer auch **Texte verarbeitet**.
- Wie werden Texte im Computer dargestellt?
- Jeder Buchstabe wird durch einen Zahlenwert codiert.
- In erster Näherung sind Texte (Strings) Sequenzen von solchen Codezahlen.

Funktions-
Aufrufe

Syntax
Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- Computer können **Berechnungen** durchführen.
- Seit langem werden mit dem Computer auch **Texte verarbeitet**.
- Wie werden Texte im Computer dargestellt?
- Jeder Buchstabe wird durch einen Zahlenwert codiert.
- In erster Näherung sind Texte (Strings) Sequenzen von solchen Codezahlen.
- Damit wird auch die **Textverarbeitung** zu einer Berechnung.

Funktions-
Aufrufe

Syntax
Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



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

USASCII code chart

b7 b6 b5 b4 b3 b2 b1					Column									
b7	b6	b5	b4	b3	b2	b1	Row							
							0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	NUL	DLE	SP	@	P	\	p	
0	0	0	0	1	1	1	SOH	DC1	!	A	Q	o	q	
0	0	0	1	0	2	2	STX	DC2	"	B	R	b	r	
0	0	1	1	1	3	3	ETX	DC3	#	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
Standardfunktion-
nen
Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



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

USASCII code chart

bits		b7 b6 b5 b4				Columns							
		b3	b2	b1	Row#	0	1	2	3	4	5	6	7
0	0	0	0	0	0	NUL	DLE	SP	0	@	P	\	p
0	0	0	0	1	1	SOH	DC1	!	1	A	Q	o	q
0	0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	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	

- Benötigt 7 Bits und enthält alle druckbaren Zeichen der englischen Sprache

Funktions-
Aufrufe

Syntax
Standardfunktionen
Exkurs:
Zeichenkodierung
und Unicode

Mathematische
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- In anderen Sprachen wurden **zusätzliche Zeichen** benötigt.

Funktions-
Aufrufe

Syntax
Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- In anderen Sprachen wurden **zusätzliche Zeichen** benötigt.
- Da praktisch alle modernen Rechner **8-Bit-Bytes** als kleinste Speichereinheit nutzten, standen die höherwertigen Codes (128–255) für Erweiterungen zur Verfügung.

Funktions-
Aufrufe

Syntax
Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- In anderen Sprachen wurden **zusätzliche Zeichen** benötigt.
- Da praktisch alle modernen Rechner **8-Bit-Bytes** als kleinste Speichereinheit nutzten, standen die höherwertigen Codes (128–255) für Erweiterungen zur Verfügung.
- Diverse Erweiterungen, z.B. **ISO-Latin-1** (mit Umlauten).

Funktions-
Aufrufe

Syntax
Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- In anderen Sprachen wurden **zusätzliche Zeichen** benötigt.
- Da praktisch alle modernen Rechner **8-Bit-Bytes** als kleinste Speichereinheit nutzten, standen die höherwertigen Codes (128–255) für Erweiterungen zur Verfügung.
- Diverse Erweiterungen, z.B. **ISO-Latin-1** (mit Umlauten).
- Auf dem IBM-PC gab es andere Erweiterungen, **Windows-1252**.

Funktions-
Aufrufe

Syntax
Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- In anderen Sprachen wurden **zusätzliche Zeichen** benötigt.
- Da praktisch alle modernen Rechner **8-Bit-Bytes** als kleinste Speichereinheit nutzten, standen die höherwertigen Codes (128–255) für Erweiterungen zur Verfügung.
- Diverse Erweiterungen, z.B. **ISO-Latin-1** (mit Umlauten).
- Auf dem IBM-PC gab es andere Erweiterungen, **Windows-1252**.
- Sprachen, die nicht auf dem lateinischen Alphabet basieren, haben große Probleme, **ISO-2022-JP**.

Funktions-
Aufrufe

Syntax
Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- In anderen Sprachen wurden **zusätzliche Zeichen** benötigt.
- Da praktisch alle modernen Rechner **8-Bit-Bytes** als kleinste Speichereinheit nutzten, standen die höherwertigen Codes (128–255) für Erweiterungen zur Verfügung.
- Diverse Erweiterungen, z.B. **ISO-Latin-1** (mit Umlauten).
- 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
Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- Um für alle Sprachräume einen einheitlichen Zeichencode zu haben, wurde **Unicode** entwickelt (Version 1.0 im Jahr 1991).

Funktions-
Aufrufe

Syntax
Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- Um für alle Sprachräume einen einheitlichen Zeichencode zu haben, wurde **Unicode** entwickelt (Version 1.0 im Jahr 1991).
- Im Mai 2019 (Version 12.1) unterstützt Unicode **150 Schriften** mit **137993 Codepoints**, darunter **1273 Emojis**.

Funktions-
Aufrufe

Syntax
Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- Um für alle Sprachräume einen einheitlichen Zeichencode zu haben, wurde **Unicode** entwickelt (Version 1.0 im Jahr 1991).
- Im Mai 2019 (Version 12.1) unterstützt Unicode **150 Schriften** mit **137993 Codepoints**, darunter **1273 Emojis**.
- “Adds a single character at U+32FF for the square ligature form of the name of the Reiwa era.”

Funktions-
Aufrufe

Syntax
Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- Um für alle Sprachräume einen einheitlichen Zeichencode zu haben, wurde **Unicode** entwickelt (Version 1.0 im Jahr 1991).
- Im Mai 2019 (Version 12.1) unterstützt Unicode **150 Schriften** mit **137993 Codepoints**, darunter **1273 Emojis**.
- “Adds a single character at U+32FF for the square ligature form of the name of the Reiwa era.”
- Organisiert in 17 Ebenen mit jeweils 2^{16} Codepoints (manche ungenutzt)

Funktions-
Aufrufe

Syntax
Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- Um für alle Sprachräume einen einheitlichen Zeichencode zu haben, wurde **Unicode** entwickelt (Version 1.0 im Jahr 1991).
- Im Mai 2019 (Version 12.1) unterstützt Unicode **150 Schriften** mit **137993 Codepoints**, darunter **1273 Emojis**.
- “Adds a single character at U+32FF for the square ligature form of the name of the Reiwa era.”
- Organisiert in 17 Ebenen mit jeweils 2^{16} Codepoints (manche ungenutzt)
- Die ersten 128 Codepoints stimmen mit ASCII überein, die ersten 256 mit ISO-Latin-1.

Funktions-
Aufrufe

Syntax
Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- Um für alle Sprachräume einen einheitlichen Zeichencode zu haben, wurde **Unicode** entwickelt (Version 1.0 im Jahr 1991).
- Im Mai 2019 (Version 12.1) unterstützt Unicode **150 Schriften** mit **137993 Codepoints**, darunter **1273 Emojis**.
- “Adds a single character at U+32FF for the square ligature form of the name of the Reiwa era.”
- Organisiert in 17 Ebenen mit jeweils 2^{16} Codepoints (manche ungenutzt)
- Die ersten 128 Codepoints stimmen mit ASCII überein, die ersten 256 mit ISO-Latin-1.
- Zum Thema Emojis gibt es ein eigenes **Subkomitee** ...

Funktions-
Aufrufe

Syntax
Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- Jedes Unicode-Zeichen kann durch eine 32-Bit-Zahl dargestellt werden (UTF-32 oder UCS-4).

Funktions-
Aufrufe

Syntax
Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- Jedes 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 die Ebene 0 direkt als 16-Bit-Zahl kodiert wird. Zeichen aus anderen Ebenen benötigen 32 Bit.

Funktions-
Aufrufe

Syntax
Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- Jedes 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 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

Funktions-
Aufrufe

Syntax
Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- Jedes 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 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?

Funktions-
Aufrufe

Syntax
Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- Jedes 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 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
Standardfunktio-
nen

Exkurs:
Zeichenkodierung
und Unicode

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



Mathematische Funktionen

Funktions-
Aufrufe

**Mathemati-
sche
Funktionen**

math-Modul
Direktimport

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- 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**):

Python-Interpreter

```
>>> import math
>>> math.pi
```

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

`math`-Modul
Direktimport

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- 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**):

Python-Interpreter

```
>>> import math
>>> math.pi
3.141592653589793
>>>
```

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

`math`-Modul
Direktimport

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- 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**):

Python-Interpreter

```
>>> import math
>>> math.pi
3.141592653589793
>>> math.sin(1/4*math.pi)
```

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

`math`-Modul
Direktimport

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- 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**):

Python-Interpreter

```
>>> import math
>>> math.pi
3.141592653589793
>>> math.sin(1/4*math.pi)
0.7071067811865475
>>>
```

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

`math`-Modul
Direktimport

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- 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**):

Python-Interpreter

```
>>> import math
>>> math.pi
3.141592653589793
>>> math.sin(1/4*math.pi)
0.7071067811865475
>>> math.sin(math.pi)
```

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

`math`-Modul
Direktimport

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- 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**):

Python-Interpreter

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

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

`math`-Modul
Direktimport

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- 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**):

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))
```

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

`math`-Modul
Direktimport

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- 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**):

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
```

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

`math`-Modul
Direktimport

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- Die Punktschreibweise verhindert **Namenskollisionen**, ist aber umständlich
- Direkter Import eines Bezeichners (überschreibt vorherige Definition):
`from module import name`
- Direkter Import *aller* Bezeichner eines Moduls (überschreibt vorherige Def):
`from module import *`

Python-Interpreter

```
>>> pi = 42
```

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

math-Modul
Direktimport

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- Die Punktschreibweise verhindert **Namenskollisionen**, ist aber umständlich
- Direkter Import eines Bezeichners (überschreibt vorherige Definition):
`from module import name`
- Direkter Import *aller* Bezeichner eines Moduls (überschreibt vorherige Def):
`from module import *`

Python-Interpreter

```
>>> pi = 42
>>> pi
```

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

math-Modul
Direktimport

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- Die Punktschreibweise verhindert **Namenskollisionen**, ist aber umständlich
- Direkter Import eines Bezeichners (überschreibt vorherige Definition):
`from module import name`
- Direkter Import *aller* Bezeichner eines Moduls (überschreibt vorherige Def):
`from module import *`

Python-Interpreter

```
>>> pi = 42
>>> pi
42
>>> from math import pi
>>> pi
```

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

math-Modul
Direktimport

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- Die Punktschreibweise verhindert **Namenskollisionen**, ist aber umständlich
- Direkter Import eines Bezeichners (überschreibt vorherige Definition):
`from module import name`
- Direkter Import *aller* Bezeichner eines Moduls (überschreibt vorherige Def):
`from module import *`

Python-Interpreter

```
>>> pi = 42
>>> pi
42
>>> from math import pi
>>> pi
3.141592653589793
>>>
```

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

math-Modul
Direktimport

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- Die Punktschreibweise verhindert **Namenskollisionen**, ist aber umständlich
- Direkter Import eines Bezeichners (überschreibt vorherige Definition):
`from module import name`
- Direkter Import *aller* Bezeichner eines Moduls (überschreibt vorherige Def):
`from module import *`

Python-Interpreter

```
>>> pi = 42
>>> pi
42
>>> from math import pi
>>> pi
3.141592653589793
>>> from math import *
>>> cos(pi)
```

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

math-Modul
Direktimport

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- Die Punktschreibweise verhindert **Namenskollisionen**, ist aber umständlich
- Direkter Import eines Bezeichners (überschreibt vorherige Definition):
`from module import name`
- Direkter Import *aller* Bezeichner eines Moduls (überschreibt vorherige Def):
`from module import *`

Python-Interpreter

```
>>> pi = 42
>>> pi
42
>>> from math import pi
>>> pi
3.141592653589793
>>> from math import *
>>> cos(pi)
-1.0
```

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

math-Modul
Direktimport

Funktions-
Definition

Namens-
raum

Rückgabe-
werte

Sollte ein Programmierer `import *` verwenden?



Ja

Nein

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

`math`-Modul

`Direktimport`

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



Funktionsdefinitionen

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

**Funktions-
Definition**

Definition
Einrückungen
Aufruf
Argumente,
Parameter,
Rückgabewerte

Namens-
raum

Rückgabe-
werte



- Ein Python-Programm kann Funktionsdefinitionen enthalten.
- Eine Funktionsdefinition gibt einer Folge von Anweisungen einen Namen, damit die Anweisungen unter diesem Namen abgerufen werden können.
- Eine Funktionsdefinition beginnt mit dem **Schlüsselwort `def`**, danach kommt der **Funktionsname** gefolgt von der **Parameterliste** und dann ein Doppelpunkt.
- Dann folgt der **Funktionsrumpf** bestehend aus *gleich weit eingerückten Anweisungen*, z.B. Zuweisungen oder Funktionsaufrufe:

```
def print_lyrics():  
    print ("I'm_a_lumberjack_and_I'm_OK")  
    print ("I_sleep_all_night_and_I_work_all_day")
```

- Abspeichern in Datei `lyrics.py` (**wichtig** Endung `.py` für Python-Code)

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

Funktions-
Definition

Definition
Einrückungen
Aufruf
Argumente,
Parameter,
Rückgabewerte

Namens-
raum

Rückgabe-
werte



- **Einrückungen** am Zeilenanfang sind bedeutungstragend.
 - Gleiche Einrückung = zusammengehöriger 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,
Parameter,
Rückgabewerte

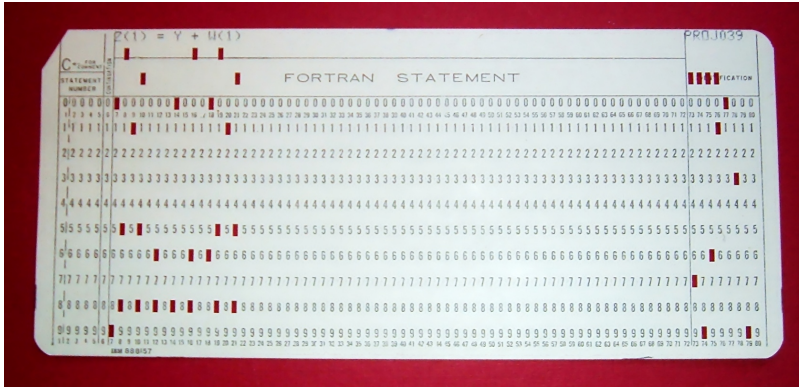
Namens-
raum

Rückgabe-
werte

Intermezzo: Einrückungen in FORTRAN



Einrückungen am Zeilenanfang sind bedeutungstragend.



Funktions-
Aufrufe

Mathemati-
sche
Funktionen

Funktions-
Definition

Definition
Einrückungen
Aufruf
Argumente,
Parameter,
Rückgabewerte

Namens-
raum

Rückgabe-
werte

Intermezzo: play the lumberjack



Python-Interpreter

```
>>> import webbrowser
>>> webbrowser.open('https://www.youtube.com/watch?v=89LfQU1cNFk')
True
```

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

Funktions-
Definition

Definition
Einrückungen
Aufruf
Argumente,
Parameter,
Rückgabewerte

Namens-
raum

Rückgabe-
werte



- Funktionsnamen verhalten sich wie Variablennamen.
- Funktionen haben einen speziellen Typ.
- Selbstdefinierte Funktionen werden wie Standardfunktionen aufgerufen

Python-Interpreter

```
>>> print(print_lyrics)
```

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

Funktions-
Definition

Definition
Einrückungen

Aufruf
Argumente,
Parameter,
Rückgabewerte

Namens-
raum

Rückgabe-
werte



- Funktionsnamen verhalten sich wie Variablennamen.
- Funktionen haben einen speziellen Typ.
- Selbstdefinierte Funktionen werden wie Standardfunktionen aufgerufen

Python-Interpreter

```
>>> print(print_lyrics)
<function print_lyrics at 0x100520560>
```

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

Funktions-
Definition

Definition
Einrückungen

Aufruf
Argumente,
Parameter,
Rückgabewerte

Namens-
raum

Rückgabe-
werte



- Funktionsnamen verhalten sich wie Variablennamen.
- Funktionen haben einen speziellen Typ.
- Selbstdefinierte Funktionen werden wie Standardfunktionen aufgerufen

Python-Interpreter

```
>>> print(print_lyrics)
<function print_lyrics at 0x100520560>
>>>
```

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

Funktions-
Definition

Definition
Einrückungen

Aufruf
Argumente,
Parameter,
Rückgabewerte

Namens-
raum

Rückgabe-
werte



- Funktionsnamen verhalten sich wie Variablennamen.
- Funktionen haben einen speziellen Typ.
- Selbstdefinierte Funktionen werden wie Standardfunktionen aufgerufen

Python-Interpreter

```
>>> print(print_lyrics)
<function print_lyrics at 0x100520560>
>>> type(print_lyrics)
```

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

Funktions-
Definition

Definition
Einrückungen

Aufruf
Argumente,
Parameter,
Rückgabewerte

Namens-
raum

Rückgabe-
werte



- Funktionsnamen verhalten sich wie Variablennamen.
- Funktionen haben einen speziellen Typ.
- Selbstdefinierte Funktionen werden wie Standardfunktionen aufgerufen

Python-Interpreter

```
>>> print(print_lyrics)
<function print_lyrics at 0x100520560>
>>> type(print_lyrics)
<class 'function'>
```

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

Funktions-
Definition

Definition
Einrückungen

Aufruf
Argumente,
Parameter,
Rückgabewerte

Namens-
raum

Rückgabe-
werte



- Funktionsnamen verhalten sich wie Variablennamen.
- Funktionen haben einen speziellen Typ.
- Selbstdefinierte Funktionen werden wie Standardfunktionen aufgerufen

Python-Interpreter

```
>>> print(print_lyrics)
<function print_lyrics at 0x100520560>
>>> type(print_lyrics)
<class 'function'>
>>>
```

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

Funktions-
Definition

Definition
Einrückungen

Aufruf
Argumente,
Parameter,
Rückgabewerte

Namens-
raum

Rückgabe-
werte



- Funktionsnamen verhalten sich wie Variablennamen.
- Funktionen haben einen speziellen Typ.
- Selbstdefinierte Funktionen werden wie Standardfunktionen aufgerufen

Python-Interpreter

```
>>> print(print_lyrics)
<function print_lyrics at 0x100520560>
>>> type(print_lyrics)
<class 'function'>
>>> print_lyrics()
```

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

Funktions-
Definition

Definition
Einrückungen

Aufruf
Argumente,
Parameter,
Rückgabewerte

Namens-
raum

Rückgabe-
werte



- Funktionsnamen verhalten sich wie Variablennamen.
- Funktionen haben einen speziellen Typ.
- Selbstdefinierte Funktionen werden wie Standardfunktionen aufgerufen

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
```

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

Funktions-
Definition

Definition
Einrückungen

Aufruf
Argumente,
Parameter,
Rückgabewerte

Namens-
raum

Rückgabe-
werte



- Funktionsnamen verhalten sich wie Variablennamen.
- Funktionen haben einen speziellen Typ.
- Selbstdefinierte Funktionen werden wie Standardfunktionen aufgerufen

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
>>>
```

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

Funktions-
Definition

Definition
Einrückungen

Aufruf
Argumente,
Parameter,
Rückgabewerte

Namens-
raum

Rückgabe-
werte



- Funktionsnamen verhalten sich wie Variablennamen.
- Funktionen haben einen speziellen Typ.
- Selbstdefinierte Funktionen werden wie Standardfunktionen aufgerufen

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,
Parameter,
Rückgabewerte

Namens-
raum

Rückgabe-
werte



Was passiert hier?

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

Was wird beim Aufruf von `repeat_lyrics()` ausgeführt?

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

Funktions-
Definition

Definition
Einrückungen

Aufruf

Argumente,
Parameter,
Rückgabewerte

Namens-
raum

Rückgabe-
werte



- Selbst definierte Funktionen benötigen oft *Argumente*.
- Die Funktionsdefinition verwendet hierfür *formale Parameter* (Variablennamen), an die beim Aufruf die *Argumentwerte* zugewiesen werden.
- *return Ausdruck* beendet die Ausführung der Funktion.
- Der Wert des Ausdrucks hinter return wird zum *Wert des Funktionsaufrufs*.

```
CM_PER_INCH = 2.54
def cm_to_inches (l_cm):
    return l_cm / CM_PER_INCH

print(cm_to_inches(195))
```

Druckt 76.77165354330708 aus

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

Funktions-
Definition

Definition
Einrückungen
Aufruf

Argumente,
Parameter,
Rückgabewerte

Namens-
raum

Rückgabe-
werte



Namensraum

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

Funktions-
Definition

**Namens-
raum**

Lokale Variablen
und Parameter

Kellertabelle

Traceback

Globale Variablen

Rückgabe-
werte



- Parameter (part1, part2) sind nur innerhalb der Funktion **sichtbar**.
- Eine **lokale Variable** (cat) wird im Funktionsrumpf durch Zuweisung für die Dauer der Funktionsausführung eingeführt.

```
def cat_twice (part1, part2):  
    cat = part1 + part2  
    print (cat)  
    print (cat)  
  
line1 = "Bing_tiddle_"  
line2 = "tiddle_bang."  
cat_twice(line1, line2)  
# output:  
# Bing tiddle tiddle bang.  
# Bing tiddle tiddle bang.  
cat  
# output:  
# NameError: name 'cat' is not defined
```

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Lokale Variablen
und Parameter

Kellertabelle

Traceback

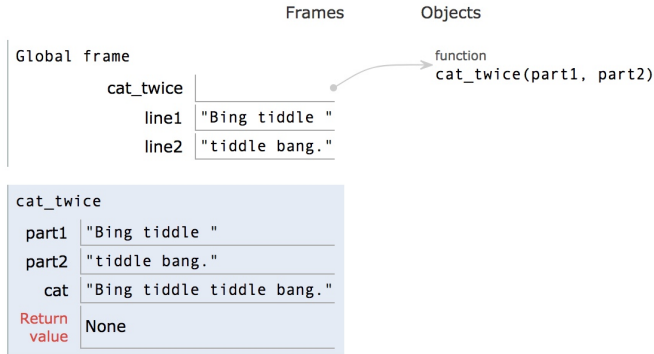
Globale Variablen

Rückgabe-
werte



- Die Variablenbelegungen innerhalb von Funktionsaufrufen können durch eine **Kellertabelle** visualisiert werden (hier hilft pythontutor.com).

Ende von cat_twice



Funktions-
Aufrufe

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Lokale Variablen
und Parameter

Kellertabelle

Traceback

Globale Variablen

Rückgabe-
werte



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

Python-Interpreter

```
>>> def cat_twice(part1, part2):  
...     cat = part1 + part2  
...     print_twice(cat)  
...  
>>> cat_twice('foo ', 'bar!')
```

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Lokale Variablen
und Parameter

Kellertabelle

Traceback

Globale Variablen

Rückgabe-
werte



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

Python-Interpreter

```
>>> def cat_twice(part1, part2):
...     cat = part1 + part2
...     print_twice(cat)
...
>>> cat_twice('foo ', 'bar!')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 3, in cat_twice
NameError: name 'print_twice' is not defined
```

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Lokale Variablen
und Parameter
Kellertabelle

Traceback
Globale Variablen

Rückgabe-
werte



- Funktionen sollen vorrangig **lokale Variable** und **Parameter** nutzen.
- Eine **globale Variable** ist ausserhalb der Funktion definiert.
- Funktionen können globale Variablen **lesen**, falls es keine lokale Variable gleichen Namens gibt.

```
master = 666
def slave():
    return master
print("slave_returns", slave())
def independent(master):
    return master
print("independent_returns", independent(42))
def ignorant():
    master = 333
    return master
print("ignorant_returns", ignorant(), "master=", master)
```

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Lokale Variablen
und Parameter

Kellertabelle

Traceback

Globale Variablen

Rückgabe-
werte



Rückgabewerte

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- Alle Funktionen geben einen Wert zurück.
- Funktionen wie `print` geben einen speziellen Wert `None` zurück, der **nicht angezeigt** wird.

Python-Interpreter

```
>>> result = print('Bruce')
Bruce
>>> result
>>> print(result)
None [das ist nicht der String 'None!']
```

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

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- Das Schlüsselwort `return` erlaubt die Definition des Rückgabewerts.

```
def sum3(a, b, c):  
    return a + b + c  
  
s123 = sum3(1, 2, 3)
```

- Keine Ausgabe, aber `s123` enthält den Wert 6
- Funktionen ohne `return` (wie `cat_twice`) geben `None` zurück.

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte

return \neq print



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

```
def printsum3(a, b, c):  
    print(a + b + c)
```

Python-Interpreter

```
>>> sum3(1, 2, 3)  
6
```

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte

return \neq print



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

```
def printsum3(a, b, c):  
    print(a + b + c)
```

Python-Interpreter

```
>>> sum3(1, 2, 3)  
6  
>>> sum3(1, 2, 3) + 4
```

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte

return \neq print



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

```
def printsum3(a, b, c):  
    print(a + b + c)
```

Python-Interpreter

```
>>> sum3(1, 2, 3)  
6  
>>> sum3(1, 2, 3) + 4  
10
```

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte

return \neq print



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

```
def printsum3(a, b, c):  
    print(a + b + c)
```

Python-Interpreter

```
>>> sum3(1, 2, 3)  
6  
>>> sum3(1, 2, 3) + 4  
10  
>>> printsum3(1, 2, 3) + 4
```

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte

return \neq print



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

```
def printsum3(a, b, c):  
    print(a + b + c)
```

Python-Interpreter

```
>>> 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'
```

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- **Funktionen** sind benannte Programmstücke

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- **Funktionen** sind benannte Programmstücke
- Sie können vorgegeben sein (Standardfunktionen) oder selbst definiert.

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- **Funktionen** sind benannte Programmstücke
- Sie können vorgegeben sein (Standardfunktionen) oder selbst definiert.
- Beim Aufruf einer Funktion müssen **Argumente** angegeben werden, die die formalen **Parameter** mit Werten belegen.

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- **Funktionen** sind benannte Programmstücke
- Sie können vorgegeben sein (Standardfunktionen) oder selbst definiert.
- Beim Aufruf einer Funktion müssen **Argumente** angegeben werden, die die formalen **Parameter** mit Werten belegen.
- Funktionen geben immer einen **Funktionswert** zurück, der mit `return` festgelegt wird (sonst `None`).

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- **Funktionen** sind benannte Programmstücke
- Sie können vorgegeben sein (Standardfunktionen) oder selbst definiert.
- Beim Aufruf einer Funktion müssen **Argumente** angegeben werden, die die formalen **Parameter** mit Werten belegen.
- Funktionen geben immer einen **Funktionswert** zurück, der mit `return` festgelegt wird (sonst `None`).
- Der Rumpf einer Funktionsdefinition führt einen neuen **Namensraum** für Parameter und **lokale Variablen** (durch Zuweisung eingeführt) ein.

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- **Funktionen** sind benannte Programmstücke
- Sie können vorgegeben sein (Standardfunktionen) oder selbst definiert.
- Beim Aufruf einer Funktion müssen **Argumente** angegeben werden, die die formalen **Parameter** mit Werten belegen.
- Funktionen geben immer einen **Funktionswert** zurück, der mit `return` festgelegt wird (sonst `None`).
- Der Rumpf einer Funktionsdefinition führt einen neuen **Namensraum** für Parameter und **lokale Variablen** (durch Zuweisung eingeführt) ein.
- **Globale Variablen** können gelesen werden, falls sie nicht durch einen Parameter oder eine lokale Variable überdeckt werden.

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte



- **Funktionen** sind benannte Programmstücke
- Sie können vorgegeben sein (Standardfunktionen) oder selbst definiert.
- Beim Aufruf einer Funktion müssen **Argumente** angegeben werden, die die formalen **Parameter** mit Werten belegen.
- Funktionen geben immer einen **Funktionswert** zurück, der mit `return` festgelegt wird (sonst `None`).
- Der Rumpf einer Funktionsdefinition führt einen neuen **Namensraum** für Parameter und **lokale Variablen** (durch Zuweisung eingeführt) ein.
- **Globale Variablen** können gelesen werden, falls sie nicht durch einen Parameter oder eine lokale Variable überdeckt werden.
- pythontutor.com visualisiert die Programmausführung mit Hilfe von Kellertabellen.

Funktions-
Aufrufe

Mathemati-
sche
Funktionen

Funktions-
Definition

Namens-
raum

Rückgabe-
werte