

Built-in Functions

Wissen, wo Funktionen bereits vorhanden
sind

Built-in Functions

Als Entwickler können Sie eine Reihe von eingebauten Funktionen ohne Einbindung eines Moduls verwenden.

Built-in Functions

<code>abs()</code>	Betrag einer Zahl
<code>bin()</code>	binäre (duale) Zahl
<code>bytes()</code>	Objekt des Datentyps bytes
<code>chr()</code>	Zeichen zu Unicode-Zahl
<code>eval()</code>	ausgeführten Python-Ausdruck
<code>exec()</code>	(Ausführung einer Anweisung)
<code>filter()</code>	Iterable, für das eine Funktion True ergibt
<code>float()</code>	Zahl mit Nachkommastellen
<code>frozenset()</code>	unveränderliches Set
<code>hex()</code>	hexadezimale Zahl

Built-in Functions

<code>input()</code>	Eingabe des Benutzers
<code>int()</code>	ganze Zahl
<code>len()</code>	Anzahl der Elemente
<code>map()</code>	Iterable mit Ergebnissen mehrerer Aufrufe
<code>max()</code>	größtes Element
<code>min()</code>	kleinstes Element
<code>oct()</code>	oktale Zahl
<code>open()</code>	Verweis auf geöffnete Datei
<code>ord()</code>	Unicode-Zahl zu Zeichen
<code>print()</code>	Ausgabe

Built-in Functions

<code>range()</code>	Iterable über Bereich
<code>reversed()</code>	Liste in umgekehrter Reihenfolge
<code>round()</code>	gerundete Zahl
<code>set()</code>	Set
<code>sorted()</code>	sortierte Liste
<code>str()</code>	Zeichenkette
<code>sum()</code>	Summe der Elemente
<code>type()</code>	Typ eines Objekts
<code>zip()</code>	Verbindung von Iterables

Max, min, sum

Die Funktionen `max()` und `min()` liefern den größten bzw. kleinsten Wert eines Iterables. Die Funktion `sum()` liefert die Summe der Elemente eines Iterables und wird nur mit einem Parameter aufgerufen.



```
1 # 18-min-max-sum.py
2
3 print("Max. Value of a Tuple:", max(3, 2, -7))
4 print("Min. Value of a Set:", min(set([3, 2, 3])))
5 print("Sum of a Tuple:", sum((3, 2, -7)))
```

chr() und ord()

Die Funktion chr() liefert das zugehörige Zeichen zu einer Unicode-Zahl.

Umgekehrt erhalten Sie mithilfe der Funktion ord() die Unicode-Zahl zu einem Zeichen.

```
1 # 18-chr-ord.py
2
3 for i in range(48,58):
4     print(chr(i), end=" ")
5 print()
6 for i in range(65,91):
7     print(chr(i), end=" ")
8 print()
9 for i in range(97,123):
10    print(chr(i), end=" ")
11 print()
12 for z in "abcde":
13    print(ord(z), end=" ")
14 print()
15 for z in "abcde":
16    print(chr(ord(z)+1), end=" ")
```

reversed() und sorted()

Die Funktion `reversed()` liefert die Elemente einer Sequenz in umgekehrter Reihenfolge.

Mithilfe der Funktion `sorted()` wird eine sortierte Liste der Elemente einer Sequenz erstellt und geliefert.

```
1 # 18-reversed-sorted.py
2
3
4 t = 4, 12, 6, -2
5 print(t)
6 print(sorted(t))
7 for i in reversed(t):
8     print(i, end=" ")
9 print()
10 dc = {"Peter":31, "Julia":28, "Werner":35}
11 print(dc)
12 va = dc.values()
13 print(va)
14
15 print(sorted(va))
16 for i in reversed(va):
17     print(i, end=" ")
```


Übung: filter()

Besuchen Sie bitte folgende Webseite:

<https://docs.python.org/3/library/functions.html> und finden Sie die Dokumentation zur filter() built-in Funktion.

Sobald Sie dieses verstanden haben, erstellen Sie eine Liste mit 10 Nummern und schreiben Sie eine Filter-Funktion welche ihnen nur gerade Zahlen zurückgibt.

Lösung: Funktionen



```
1 # 18-filter-func.py
2
3 def check_even(number):
4     if number % 2 == 0:
5         return True
6
7     return False
8
9 numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
10
11 # if an element passed to check_even() returns True, select it
12 even_numbers_iterator = filter(check_even, numbers)
13
14 # converting to list
15 even_numbers = list(even_numbers_iterator)
16
17 print(even_numbers)
```

Ende

Das war alles für dieses Kapitel
