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

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

Built-in Functions

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

Built-in Functions

range()	Iterable über Bereich
reversed()	Liste in umgekehrter Reihenfolge
round()	gerundete Zahl
set()	Set
sorted()	sortierte Liste
str()	Zeichenkette
sum()	Summe der Elemente
type()	Typ eines Objekts
zip()	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.

```
# 18-chr-ord.py

for i in range(48,58):
    print(chr(i), end="")
print()
for i in range(65,91):
    print(chr(i), end="")
print()
for i in range(97,123):
    print(chr(i), end="")
print()
for z in "abcde":
    print(ord(z), end=" ")
print()
for z in "abcde":
    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.

```
# 18-reversed-sorted.py


t = 4, 12, 6, -2
print(t)
print(sorted(t))
for i in reversed(t):
    print(i, end=" ")
print()
dc = {"Peter":31, "Julia":28, "Werner":35}
print(dc)
va = dc.values()
print(va)

print(sorted(va))
for i in reversed(va):
    print(i, end=" ")
```

zip()

Die Funktion zip() liefert ein **`Iterable`**, in dem die Elemente anderer Iterables miteinander verbunden werden.

```python
# 18-zip.py

plz = [49808, 78224, 55411]
city = ["Dietikon", "Wohlen", "Zollikofen"]
canton = ["NS", "BW", "RP"]
combo = zip(plz, city, canton)


for element in combo:
    print(element)
```

map()

Die Funktion map() gibt Ihnen die Möglichkeit, eine Funktion in einer Anweisung **mehrfach** aufzurufen, jeweils **mit anderen Parametern**.

Sie liefert ein Iterable, das aus den Ergebnissen besteht.

```python
# 18-map.py

def quad(x):
    return x * x

def sum(a,b,c):
    return a + b + c


z = map(quad, [4, 2.5, -1.5])
print("Quadratic:")
for element in z:
    print(element)
print()

z = map(sum, [3, 1.2, 2], [4.8, 2], [5, 0.1,
print("Sum:")
for element in z:
    print(element)
```

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

```python
# 18-filter-func.py

def check_even(number):
    if number % 2 == 0:
        return True

    return False

numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# if an element passed to check_even() returns True, select it
even_numbers_iterator = filter(check_even, numbers)

# converting to list
even_numbers = list(even_numbers_iterator)

print(even_numbers)
```

Ende

Das war alles für dieses Kapitel