

WIRTSCHAFTSINFORMATIK 2

DATENVERARBEITUNG MIT PYTHON

PROF. DR. CHRISTIAN BOCKERMANN, PROF. DR. VOLKER KLINGSPOR

HOCHSCHULE BOCHUM

SOMMERSEMESTER 2026

Funktionen und Module in Python

Häufig benötigt man in einem Programm Code-Stücke, die immer wieder verwendet werden. Dafür bieten sich *Funktionen* an. Funktionen sind Teilprogramme, die aufgerufen werden können, um bestimmte Berechnungen durchzuführen. Python bietet eine Vielzahl fertiger Funktionen an, die benutzt werden können. Ein Beispiel ist die Funktion **max**, die das Maximum von mehreren Zahlen berechnet:

```
zahl1 = 5  
zahl2 = 8  
  
maximum = max( zahl1, zahl2 )
```

Funktionen kennen Sie bereits aus der Schule – dort wurden mathematische Funktionen eingeführt, die eine Berechnung darstellen, z.B.

$$f(x) = x^2 + 3x + 2$$

Funktionen haben allerdings nicht immer ein Ergebnis. Wir haben schon häufig die Funktion **print**

Funktionen in Python werden mit **def** definiert

- Funktionen bestehen aus einem Namen, den Parametern (optional) und einem Code Block als Rumpf
- Dem Rumpf kann eine Beschreibung (**docstring**) vorangestellt werden (optional!)

Beispiel: $f(x) = x^3 + 5x^2 + 27$

```
def f(x):  
    """  
    Die ist ein Beschreibung der Funktion  
    """  
    return x ** 3 + 5 * x ** 2 + 27
```

```
def f(x):  
    """  
    Die ist ein Beschreibung der Funktion  
    """  
    return x ** 3 + 5 * x ** 2 + 27
```

Wenn die Funktion definiert wurde, kann Sie aufgerufen werden:

```
ergebnis = f(5)
```



◀ Probieren Sie es im Notebook aus!

Ein weiteres Beispiel - Betragsfunktion

```
def betrag(x):  
    """  
    Liefert den Betrag von x zurueck  
    """  
    if x >= 0:  
        return x  
    else:  
        return -x
```

Wofür ist der Hilfetext?

Der Hilfetext in der Funktionsdefinition ermöglicht es, die Wirkung der Funktion und ggf. die Bedeutung der Parameter zu beschreiben. Python hat eine eingebaute Funktion `help`, mit der man den Hilfetext abrufen kann.

```
def betrag(x):  
    """  
    Liefert den Betrag von x zurueck  
    """  
    if x >= 0:  
        return x  
    else:  
        return -x
```

Wenn die Funktion `betrag` wie oben definiert wurde, zeigt der folgende Befehl den Hilfetext an:

```
help(betrag)
```

Funktionsparameter

Funktionen können mit Parametern definiert werden. Diese Parameter werden bei der Ausführung der Funktion mit den Werten gefüllt, mit denen die Funktion aufgerufen wurde:

```
def g(x, y):  
    return 2*x + y
```

Welchen Wert liefert die Funktion bei den folgenden beiden Aufrufen zurück?

```
g(3, 2)  
g(2, 3)
```



Probieren Sie es im Notebook aus!

Funktionsparameter

Parameter von Funktionen sind lokale Variablen, die nur innerhalb der Funktion bekannt sind. Dies kann man mit dem folgenden Code gut testen:

```
x = 5  
  
def mal7(x):  
    return 7 * x  
  
y = mal7(3)
```

Welchen Wert hat **y**?



Probieren Sie es im Notebook aus!

Funktionsparameter können Standardwerte haben

```
def greet(name = 'Welt'):  
    """  
    Sagt Hallo zum angegebenen Namen bzw. zur Welt,  
    wenn kein Name angegeben wurde.  
    """  
    print( "Hallo, " + name + "!")  
  
# Aufruf der Funktion ohne und mit Parameter  
greet()  
greet("Data Science")
```



◀ Probieren Sie es im Notebook aus!

Funktionen und Module in Python

PYTHON MODULE

Module

Ein **Modul** stellen Klassen und Funktionen bereit, die meistens zusammenhängen. Python enthält große **Standardbibliothek** mit vielen Modulen.

Um Funktionen aus Modulen zu benutzen, muss vorher festgelegt werden, dass das jeweilige Modul in das Programm importiert werden soll. Dies geschieht mit dem Schlüsselwort **import**:

Beispiel: `time` enthält Funktionen für Umgang mit Zeit

```
import time

jetzt = time.time()
print(jetzt)
```

**Vieles braucht man nicht selbst zu programmieren!
Es geht mehr um die Verbindung der richtigen Teile ; -)**

Module und die `help(..)` Funktion

Gerade bei Verwendung von Modulen ist z.B. die Funktion `help` sehr hilfreich, weil Sie eine genaue Dokumentation der Funktionen eines Moduls bereitstellt:

```
import time  
  
help(time.time)
```



◀ Probieren Sie es im Notebook aus!

Module und Namensräume

Ein Modul wird mit **import** eingebunden und ermöglicht dann den Zugriff auf alle Funktionen, die dieses Modul bereitstellt. Dazu muss aber der Modulname mit angegeben werden:

```
import datascience

# Benutze die Funktion betrag aus dem Modul datascience:
x = -3
datascience.betrag(x)
```

Mit **import .. as** lässt sich der Namesraum eines Moduls ändern:

```
import datascience as ds

m = ds.MyModel()
```

^oDas Modul **datascience** aus dem Beispiel ist ein fiktives Modul.

Import in den Namensraum

Manchmal ist es hilfreich, Funktionen/Klassen, die man häufig verwendet, in den globalen Namensraum eines Programms zu importieren. Damit erspart man sich für diese Funktionen, dass immer der Modulname mit angegeben werden muss:

```
from datascience import betrag
```

```
x = -3  
betrag(x)
```

statt

```
import datascience
```

```
x = -3  
datascience.betrag(x)
```