Funktionen und Modularisierung

Claas de Boer, Tilman Hinnerichs

17. Dezember 2020

Python-Grundlagen

Rückblick

- Datentypen: `int, str, list, set, dict, ...`
- Komperatoren: `==, !=, ...`
- Schleifen: `while, for`
- Ein-/Ausgabe: `input, print`

Heute: Funktionen

Funktionen

- Eine **Funktion** ist ein in sich geschlossener Codeblock, der eine bestimmte Aktion ausführt.
- **Funktionen** können mit **Parametern** aufgerufen werden und können **Rückgabewerte** liefern.
- **Funktionen** erlauben es Code zu modularisieren, zu organisieren und wiederverwendbar zu machen.

Built-In Funktionen

```python
# Die Anzahl der Elemente eines Containers bestimmen
numbers = [1, 2, 3, 4, 5]
len(numbers)

# Den Typ einer Variable herausfinden
n_float = 42.8
type(n_float)

# Informationen über ein Objekt bekommen
set_of_even_numbers = set(n for n in range(1, 10) if n % 2 == 0)
help(set_of_even_numbers)

# Weitere: sorted(), enumerate(), ...
```

Hier findet ihr eine Liste der Python 3.9.1 Built-in Funktionen.

Eigene Funktionen I

```python
# Funktionsdefiniton startet mit dem Keyword 'def'

def simple_function():

    # Hier beginnt der Körper (body) der Funktion

    print("The mitochondria is the powerhouse of the cell.")

    # Hier endet der Körper der Funktion
```

- Kein **Parameter**, kein **Rückgabewert**

Eigene Funktionen II

```python
# Der Wert eines übergebenen Parameters wird an eine
# Variable im Körper der aufgerufenen Funktion gebunden

def format_greeting(name):
    return f"Hello {name}. Nice to meet you again!"

# Der Rückgabewert einer Funktion kann
# in einer Variable gespeichert werden

result = format_greeting("Sun")
print(result)
```

- Ein **Parameter**, ein **Rückgabewert**

Eigene Funktionen III

```
1 def calculate_price(item, price, amount=5):
2     costs = amount * price
3     return f"{amount} {items}'s will cost you {costs} $."
```

- Funktionen lassen sich auf verschiedene Arten aufrufen.

```
1 >>> calculate_price("Apple", 0.3, 10)
2 # 10 Apple's will cost you 3.0 $
3
4 >>> calculate_price(price=0.3, amount=10, item="Apple")
5 # 10 Apple's will cost you 3.0 $
6
7 >>> calculate_price("Apple", 0.3)
8 # 10 Apple's will cost you 1.5 $
```

Parametrisieren will gekonnt sein!

Bei den Parametern einer Funktion unterscheidet man zwischen verschiedenen Typen.

- **Positionelle** Parameter (Erforderliche Parameter)
- **Keyword** Parameter
- **Optionale** Parameter (Parameter mit Default Wert)

Mini-Exkurs: Documentation

```
    def complicated_funktion(gamma, offset, inertia):
        """This function computes the result
        of calculation foobar for some given
        parameters.

        A function can get really complicated!
        But we can describe what a function does in
        plain text in the function docstring.
        """
        ...

```

- Code wird häufiger gelesen als geschrieben.
- **Dokumentiert euren Code** und ihr müssst weniger lesen!
- Gebt euren Funktionen und ihren Parametern **deskriptive Namen**!

Einige Aufgaben™6

1. Schreibe eine Funktion, welche zwei Zahlen addiert.
2. Schreibe eine Funktion, welche die Anzahl der Vokale einer Zeichenkette berechnet.
3. Schreibe eine Funktion, welche die Länge eines Videos im Format `mm:ss` (bspw. `02:54`) entgegen nimmt und die Länge des Videos in Sekunden zurückgibt.
4. Schreibe eine Funktion, welche die `Fakultät` einer Ganzahl berechnet. Die Fakultät einer Ganzzahl ist das Produkt der Ganzzahl mit allen kleineren positiven Ganzzahlen.
5. Schreibe eine Funktion, welche die `n-te` Zahl der `Fibonacci-Folge` berechnet.

Ressourcen zum Üben/Informieren

- Real Python Artikel: Defining your own python function
- Advent of Code