Was sind Funktionen?

- Nützliches Programmierwerkzeug, um *wiederkehrende* Prozesse nicht immer wieder aufzuschreiben
- Stattdessen: 1x definieren, beliebig oft nutzen!
- Beispiele für wiederkehrende Prozesse:
  - Mathematische Formeln
  - Erstellen eines neuen Nutzers in einer Webapp
  - Auslesen von Dateien
  - usw.

Woraus besteht eine Funktion?

```
def funktionsname(a, b):
        c = a * b
        return c
print(funktionsname(5, 10))
```

- Schlüsselwort, welches dem Python-Interpreter mitteilt, dass eine Funktionsdefinition folgt

Woraus besteht eine Funktion?

```
def funktionsname(a, b):
        c = a * b
        return c
print(funktionsname(5, 10))
```

- Der Name der Funktion
- Wird relevant, wenn Funktion aufgerufen wird
- Achtung: In Python gibt es gewisse Regeln für Namen
  - Siehe Syntax von Python.pdf

Woraus besteht eine Funktion?

```
def funktionsname(a, b):
        c = a * b
        return c
print(funktionsname(5, 10))
```

- Parameter
- Platzhalter für Werte, mit denen die Funktion aufgerufen wird
- Existieren nur im Rahmen der Funktionsdefinition, nicht außerhalb
- Keine Angabe des Datentyps nötig (Eigenheit von Python)

Woraus besteht eine Funktion?

```
def funktionsname(a, b):
        c = a * b
        return c
print(funktionsname(5, 10))
```

- Anweisungen, Befehle, Berechnungen etc.
- Hängt von der Logik der Funktion ab
- „Was macht die Funktion?“

Woraus besteht eine Funktion?

```
def funktionsname(a, b):
        c = a * b
        return c
print(funktionsname(5, 10))
```

- Schlüsselwort, welches dem Python-Interpreter mitteilt, dass ein Rückgabewert folgt

Woraus besteht eine Funktion?

```
def funktionsname(a, b):
        c = a * b
        return c
print(funktionsname(5, 10))
```

- Der Wert, den die Funktion zurückgibt
- Achtung: Der Datentyp dieses Wertes ist auch der Datentyp, den die Funktion zurückgibt!
- Achtung: Dieser Wert wird nicht in die Konsole geschrieben!
  - Außer es gibt um den Funktionsaufruf ein print()

Woraus besteht eine Funktion?

```
def funktionsname(a, b):
        c = a * b
        return c
print(funktionsname(5, 10))
```

- Aufruf der definierten Funktion mit Parameter 5 und 10
- Reihenfolge der Parameter entscheidend:
  - Dem Parameter a wird der Wert 5 zugewiesen
  - Dem Parameter b wird der Wert 10 zugewiesen
- Aus funktionsname(5, 10) wird der Wert 50
  - Somit steht in der letzten Zeile print(50)

Woraus besteht eine Funktion?

```
def funktionsname(a, b):
        c = a * b
        return c
x = funktionsname(5, 10)
```

- Alternativ: Den Rückgabewert (return-Wert) einer Variable zuweisen

FAQ

F: *„Muss meine Funktion etwas zurückgeben?“*

- Explizit müssen Funktionen in Python nichts zurück
- Implizit steht in diesem Fall: return None
- None ist ein Datentyp, der kennzeichnet, dass eine Variable keinen Wert hat

F: *„Müssen bei der Definition immer Parameter angegeben werden?“*

- Nein, z.B. Lesen von Konfigurationswerten
- Aber: Parameter machen die Funktion flexibler

Weiteres

- Funktionsparameter können Default-Werte haben
  - Dadurch ist es optional beim Funktionsaufruf dem Parameter einen Wert zuzuweisen!
  - Beispiel:

```
def func(x, y=5):
        return x+y
print(func(3))     # Ausgabe: 8
```

- Die Anzahl der Parameter lässt sich sogar komplett flexibel gestalten!
  - Fortgeschrittenes Thema!
  - Bei Interesse: https://www.programiz.com/python-programming/args-and-kwargs

Weiteres

- Es lassen sich *Hinweise* auf den Datentyp von Parameter und Ausgabe einbauen:

```
def my_func(a: int, b: str, c: bool) -> float:
```

- Kein Type-Check!
  - a muss nicht int sein,
  - b muss nicht str sein,
  - c muss nicht bool sein,
  - return muss nicht float sein!