Inhaltsverzeichnis

Programmierung in Python

Univ.-Prof. Dr. Martin Hepp, Universität der Bundeswe

Einheit 3: Objektorientierte Programmierung in Python

Version: 2019-12-10

http://www.ebusiness-unibw.org/wiki/Teaching/PIP

, martin.hepp@unibw.de

Inhaltsverzeichnis

1 Funktionen und Modularis

1.1 Motivation

- Programmteile, die oft wiederholt werden, sollten ni vorhanden sein.
- Stattdessen sollte man sie über einen Namen aufru
- Einen Programmteil, den man über seinen Namen a nennt man in der Programmierung **Funktion** oder **M**
- Programme werden dadurch kürzer und übersichtli

, martin.hepp@unibw.de

Beispiel: Im Folgenden wird zwei Mal 'Hallo UniBwM' a


```
In [1]: print('Hallo UniBwM')
        print('Hallo UniBwM')
```

```
Hallo UniBwM
Hallo UniBwM
```

Besser wäre es, wenn man diese Funktion ein Mal defin
dann bei Bedarf aufrufen könnte.

, martin.hepp@unibw.de

1.2 Definition von Funktionen

Funktionen werden in Python mit dem Schlüsselwort `de` [siehe auch Python 3 Reference].

In [2]:
```
def say_hello():
    print('Hallo UniBwM')
```

Anschließend kann man sie jederzeit über ihren Namen

In [3]:
```
say_hello()
say_hello()
```

```
Hallo UniBwM
Hallo UniBwM
```

Hier spart man zwar nicht wirklich viel an Programmläng

, martin.hepp@unibw.de

Wenn man den Begrüßungstext ändern möchte, muss m

Inhaltsverzeichnis

1.3 Übergabeparameter

Eine Funktion kann so definiert werden, dass man ihr W
Parameter übergibt, die dann das Verhalten der Funktio
(vgl. Python 3 Reference).

Dazu definiert man in runden Klammern eine Liste von N
die der jeweils übergebene Wert **innerhalb der Funktio**
ist.

Beispiel:

In [4]:
```
def say_text(text):
    print(text)
```

In [5]:
```
say_text('Hallo UniBwM')
```

```
Hallo UniBwM
```

, martin.hepp@unibw.de

Natürlich können auch mehrere Parameter übergeben w

Inhaltsverzeichnis

In [6]:
```
def multipliziere(wert_1, wert_2):
    """Multipliziert <wert_1> mit <v
    ergebnis = wert_1 * wert_2
    print(ergebnis)
```

In [7]:
```
# Aufruf mit Wer
multipliziere(3,
```

```
15
```

In [8]:
```
# Aufruf mit Var
a = 3
b = 5
multipliziere(a,
```

```
15
```

, martin.hepp@unibw.de

1.3.1 Arten von Parametern

Beim Aufruf müssen die übergebenen Parameter zu der
in der Definition passen. Die Parameter einer Funktion o
nennt man auch Signatur.

Hier gibt es mehrere Möglichkeiten.

, martin.hepp@unibw.de

1.3.1.1 Positional Arguments

Parameter können einfach über ihre Reihenfolge bestim
im obigen Beispiel). In den folgenden Beispielen ist der
beim Aufruf der Funktion `funktion` an erster Stelle üb
innerhalb der Funktion über den Namen `parameter_1`
und der Wert, der an zweiter Stelle übergeben wird, übe
`parameter_2`:

In [9]:
```
def funktion(parameter_1, parameter_
    print(parameter_1)
    print(parameter_2)

funktion(1, 5)
```

```
1
5
```

In [10]:
```
funktion(5, 1)
```

```
5
1
```

, martin.hepp@unibw.de

Die Reihenfolge bestimmt also die Zuordnung:

Inhaltsverzeichnis

In [11]:
```
def teile(dividend, divisor):
    print(dividend / divisor)

teile(15, 3)
```

```
5.0
```

In [12]:
```
teile(3, 5)
```

```
0.6
```

, martin.hepp@unibw.de

1.3.1.2 Keyword Arguments

Alternativ kann man auch Namen für Parameter vorgebe
verwendet werden müssen. Dies erlaubt auch Default-W

In [13]:
```
def funktion(parameter_name='UniBwM'):
    print(parameter_name)
```

In [14]:
```
# Aufruf mit Parameter
funktion(parameter_name='hallo')
```

```
hallo
```

In [15]:
```
# Variable text
text = 'TUM'
funktion(paramet
```

```
TUM
```

In [16]:
```
# Wenn der Parameter fehlt, wird der Default-Wert verwendet:
funktion()
```

```
UniBwM
```

, martin.hepp@unibw.de

Keyword Arguments können auch über ihre Position angesprochen werden:


```
In [17]: funktion('Guten Tag')

         Guten Tag
```

, martin.hepp@unibw.de

Inhaltsverzeichnis

1.3.1.3 Variable Anzahl an Positional Arguments (nic klausurrelevant)

Es ist auch möglich, eine flexible Anzahl an Positional A zuzulassen. Dazu fügt man einen Parameternamen mit v Sternchen ein, wie z.B. `*weitere_parameter`. Dann weiteren übergebenen Parameter im Inneren der Funktio Tupel mit dem Namen `weitere_parameter` ansprech

Die folgende Funktion erfordert mindestens ein Paramet Nachnamen und 0..n weitere Parameter als Vornamen.

In [18]:
```
def schreibe_namen(nachname, *vornamen):
    for vorname in vornamen:
        print(vorname, end=' ')   # end=' ' unterdrückt den Zeile
    print(nachname)
```

In [19]:
```
schreibe_namen('Müller')
```

Inhaltsverzeichnis

1.3.1.4 Variable Anzahl an Keyword Arguments (nich klausurrelevant)

Es ist ebenso möglich, eine flexible Anzahl an Keyword zuzulassen. Dazu fügt man einen Parameternamen mit Doppelstern ein, wie z.B. `**weitere_werte`. Dann sin weiteren übergebenen Parameter im Inneren der Funktio Dictionary mit dem Namen `weitere_werte` ansprechl

Die folgende Funktion erfordert mindestens ein Paramet Kundennummer und 0..n weitere Paare aus Keyword un zusätzliche Parameter.

In [23]:
```
def drucken_kundendaten(kundennummer, **eigenschaften):
    print('Kundennummer:', kundennummer)
    for key in eigenschaften:
        print(key, '=', eigenschaften[key])
```

, martin.hepp@unibw.de

Inhaltsverzeichnis

1.3.1.5 Weitere Informationen zu Parametern in Funl (nicht klausurrelevant)

Siehe https://docs.python.org/3.7/tutorial/controlflow.ht defining-functions.

, martin.hepp@unibw.de

1.4 Rückgabewerte

Eine Funktion kann einen Wert als Ergebnis zurückliefern, das Schlüsselwort `return`.

Dann muss man den Aufruf der Funktion einer Variable z ausgeben oder auf andere Weise in einem Ausdruck ver

In [25]:
```
def verdopple(parameter_1):
    ergebnis = parameter_1 * 2
    return ergebnis

print(verdopple(10))
mein_wert = verdopple(10)
print(mein_wert)
```

```
20
20
```

, martin.hepp@unibw.de

Der Aufruf einer Funktion mit Rückgabewert kann wie je Wert oder Ausdruck verwendet werden.

Inhaltsverzeichnis


```
In [26]: print(verdopple(verdopple(4)))

16
```

```
In [27]: print(verdopple(

23.799999999999
```

, martin.hepp@unibw.de

Wenn man mehrere Werte zurückliefern möchte, muss n
Datentyp verwenden, der Unterelemente enthält. Beispi

- Tupel
- Dictionary
- Benutzerdefinierte Objekte

In [28]:
```
def zwei_werte(parameter_1):
    wert_1 = parameter_1 * 2
    wert_2 = parameter_1 * 3
    return (wert_1, wert_2)

ergebnis = zwei_werte(4)
print(ergebnis)
a, b = ergebnis
print(a, b)
```

(8, 12)

, martin.hepp@unibw.de

Übungsaufgabe: Schreiben Sie eine Funktion, die die Q
einer als Parameter übergebenen Ganzzahl zurückliefert

Inhaltsverzeichnis

In [29]:
```
def quersumme(zahl):
    zahl = str(zahl)
    ergebnis = 0
    for ziffer in zahl:
        ergebnis = ergebnis + int(ziffer)
    return ergebnis

print(quersumme(426))
```

```
12
```

, martin.hepp@unibw.de

1.5 Dokumentation von Funktion (nicht klausurrelevant)

Jede Funktion sollte eine kurze Beschreibung und Anga Aufruf- und Rückgabeparameter enthalten.

Eine genaue Beschreibung dieser 'docstrings' finden Sie folgenden Links:

- https://docs.python.org/3/tutorial/controlflow.html#strings
- https://www.python.org/dev/peps/pep-0257/
- http://google.github.io/styleguide/pyguide.html#383-and-methods

, martin.hepp@unibw.de

Beispiel:

Inhaltsverzeichnis

In [30]:

```python
def quersumme(zahl):
    """Berechnet die Quersumme einer Ganzzahl.

    Diese Funktion addiert die Ziffernwerte der übergebenen Ganz

    Args:
        zahl: Die zu verwendende Ganzzahl.

    Returns:
        Die Quersumme als Ganzzahl.
    """
    zahl = str(zahl)
    ergebnis = 0
    for ziffer in zahl:
        ergebnis = ergebnis + int(ziffer)
    return ergebnis
```

, martin.hepp@unibw.de

Inhaltsverzeichnis

2 Bibliotheken und Import v Modulen (nicht klausurreleva

In Python können Module mit vordefinierten Funktionen (s.u.) bei Bedarf zum eigenen Programm hinzugefügt we 3 Reference].

2.1 Import von Modulen mit `impo`

Durch die Anweisung `import <modulname>` wird das `<modulname>` geladen. Danach können alle Namen au Modul über `<modulname>.<lokaler_name>` angesp werden.

, martin.hepp@unibw.de

Beispiel:

In [31]:
```
import math

# Im Modul math gibt es einen Wert pi
print(math.pi)
```

```
3.141592653589793
```

In [32]:
```
# Und eine Funktion ceil(x), die die kleinste Ganzzahl zurücklie
print(math.ceil(3.14))
```

```
4
```

Für weitere Informationen, siehe hier. Die vollständigen t
Details sind etwas kompliziert und hier beschrieben.

, martin.hepp@unibw.de

2.2 Standard-Bibliothek von Pyth

Python enthält eine sehr reichhaltige Bibliothek an vorde
Funktionen und Klassen. Für eine vollständige Liste, siel
Python Standard Library.

, martin.hepp@unibw.de

Inhaltsverzeichnis

2.3 Vorinstallierte Bibliotheken in Anaconda

Die Anaconda-Distribution enthält viele weitere, vorinsta
Module; diese variieren zum Teile je nach Betriebssyster
Eine Liste ist hier verfügbar.

, martin.hepp@unibw.de

2.4 Installation zusätzlicher Bibli

Dieses Thema wird im Rahmen der Vorlesung nicht beha
hier nur der Vollständigkeit halber erwähnt. Für eine List
verfügbarer Module, siehe https://pypi.org/.

, martin.hepp@unibw.de

Inhaltsverzeichnis

3 Objektorientierung als Programmierparadigma

3.1 Motivation

3.2 Klassen

3.3 Instanzen

3.4 Methoden

3.5 Beispiel

, martin.hepp@unibw.de

4 Definition einer Klasse (ni klausurrelevant)

https://docs.python.org/3/tutorial/classes.html

Inhaltsverzeichnis

, martin.hepp@unibw.de

4.1 Grundgerüst

- Klasse
- Parameter
- DOCSTRING
- Konstruktor-Methode
- Klassenmethoden
- Klassenvariablen
- Instanzenvariablen

, martin.hepp@unibw.de

4.2 Variablen auf Instanzebene

Dieses Thema wird im HT 2019 nicht behandelt.

, martin.hepp@unibw.de

4.3 Konstruktor-Methode

Dieses Thema wird im HT 2019 nicht behandelt.

, martin.hepp@unibw.de

4.4 Methoden auf Klassenebene

Dieses Thema wird im HT 2019 nicht behandelt.

, martin.hepp@unibw.de

Inhaltsverzeichnis

5 Quellenangaben und weiterführende Literatur

[Pyt2019] Python Software Foundation. Python 3.8.0 Documentation. https://docs.python.org/3/.

, martin.hepp@unibw.de

Vielen Dank!

http://www.ebusiness-unibw.org/wiki/Teaching/PIP

, martin.hepp@unibw.de