LU09b - Funktionen erweitert

Parameter mit Namen aufrufen

In Python können wir auch beim Aufruf von Funktionen den Variablennamen mitgeben. Damit können wir unseren Code besser lesbar machen.

Wir haben diese Funktion und möchten sie verwenden.

```
def value_beween(value, lower_bound,
upper_bound):
    '''
    Returns True if Value is between
lower_bound and upper_bound
    '''
    return lower_bound <= value <=
upper_bound
```

Wir können diese Funktion nun so aufrufen:

```
value_beween(25,0,30) # True
```

Oder wir rufen die Funktion mit benennten Argumenten auf:

```
value_beween(value = 25, lower_bound = 0,
upper_bound = 30) # True
```

Defaultwerte für Parameter

Wenn Sie `argument_name=default_value` in der Funktionsdefinition verwenden, wird der Standardwert verwendet, wenn das entsprechende Argument weggelassen wird.

```
def func_default(arg1, arg2='default_x',
arg3='default_y'):
    print(arg1)
    print(arg2)
    print(arg3)

func_default('a')

a
```

```
default_x
default_y

func_default('a', 'b')

a
b
default_y

func_default('a', arg3='c')

a
default_x
c
```

Beachte: Das der Parameter `arg1` keinen default Wert hat, muss der Parameter `arg1` beim Funktionsaufruf immer angegeben werden.

Mehrere Return-Werte

Eine Funktion kann mehr als nur einen Return-Wert zurückgeben, indem sie einfach durch Kommas getrennt werden.

Definieren wir zum Beispiel eine Funktion, die eine Zeichenkette und eine ganze Zahl zurückgibt, wie folgt:

```
def test():
    return 'abc', 100
```

In Python werden kommagetrennte Werte als `Tupel` (nicht veränderbare Liste) betrachtet. Aus diesem Grund gibt die Funktion im obigen Beispiel ein Tupel mit jedem Wert als Element zurück.

```
result = test()

print(result)
print(type(result))

('abc', 100)
```

```
<class 'tuple'>
```

Jedes Element hat jedoch auch einen eigenen definierten Typ.

```
print(result[0])
print(type(result[0]))

abc
<class 'str'>

print(result[1])
print(type(result[1]))

100
<class 'int'>
```

Die Angabe eines Index, der die Anzahl der definierten Rückgabewerte überschreitet, führt natürlich zu einem Fehler.

```
print(result[2])

IndexError: tuple index out of range
```

Sie können mehrere Rückgabewerte auch direkt an verschiedene Variablen vergeben.

```
a, b = test()
print(a)
print(b)

abc
100
```

Dasselbe gilt für drei oder mehr Rückgabewerte.

```
def test2():
    return 'abc', 100, [0, 1, 2]

a, b, c = test2()
print(a)
```

```
print(b)
print(c)

abc
100
[0, 1, 2]
```

M319-F2G, M319-F2F, M319-F2E