Grundlagen der Programmierung 1

Prof. Dr. Detlef Krömker

Dipl. Inf. Alexander Wolodkin

Kaddour Alnaasan, 0016285

2018-12-21

Aufgabe 8.1: Prozeduren und Funktionen

Die Arbeit mit OOP und Modulen erfordert Grundkenntnisse in Arbeit mit Prozeduren und Funktionen.

a) *Was verstehen wir in der Informatik unter dem Begriff der Signatur einer Funktion? Aus welchen Bestandteilen setzt sich diese zusammen?*
Losung:
In der Informatik bedeutet die Signatur einer Funktion seine Inputs und Outputs.
Sie setzt sich von ihrer Parameter, der Typen der Parameter und die Rückkehr von Werten.
< https://en.wikipedia.org/wiki/Type_signature>

b) *Was ist der Unterschied zwischen einer Prozedur und einer Funktion?*
Lösung:
Prozedur kann keinen Wert zurückgeben aber die Funktion soll einen Wert zurückgeben.
< https://de.wikipedia.org/wiki/Prozedur_(Programmierung)>

c) *Was ist der Unterschied zwischen call-by-value und call-by-object? Welche Vor- und Nachteile können damit verbunden sein?*
Lösung:

Aufgabe 8.2: OOP-Grundbegriffe

Erklären Sie in eigenen Worten folgende Begriffe und ihre Realisierung in Python:

a) Klasse, Objekt
Die Klasse ist wie ein Container, der eigene Attribute und Methoden hat.
Das Objekt ist die Abbildung von Gegenstand mit seinen Eigenschaften und Verhaltensweisen (Methoden) in ein Programm.

```
class Car():
    def wheels(self):
        print("Car has 4 Wheels.")

car = Car()
```

< https://www.python-kurs.eu/klassen.php>

b) Vererbung:
Die Vererbung ist, dass eine Klasse (B) von der Klasse (A) erbt. Das heißt, Die Klasse (B) erbt die Eigenschaften und Methoden der Klasse (A).
< https://pythonspot.com/vererbung/>

```
class Company():
    def __init__(self):
        self.name = "BMV"

class Car(Company):
```

Kaddour Alnaasan, 0016285

2018-12-21

```
    def __init__(self):
        self.id = 1
        self.name_car = "310i"

    def wheels(self):
        print("Car has 4 Wheels.")
```

c) Interface:

Das Interface zeigt uns, was eine Funktion tun kann. Bei die objektorientierten Programmierung ist es eine Schnittstelle Bei der objektorientierten Programmierung ist das Interface ein Satz öffentlich Methoden an einem Objekt, die von anderen Teilen des Programms zur Interaktion mit diesem Objekt verwendet werden können
< http://masnun.rocks/2017/04/15/interfaces-in-python-protocols-and-abcs/>

```
import abc

class MyInterface(abc.ABC):
    @abc.abstractclassmethod
    def func1(self):
        pass

class MyClass(MyInterface):
    def func1(self):
        print("Hallo")
my_interface = MyInterface()
print(my_interface.func1)

>> Hallo
```

d) Überschreiben von Methoden

Es bedeutet, dass eine Methode durch eine andere Methode mit dem selben Namen und mit anderen Inhalt übergeschrieben wird.
< https://www.python-kurs.eu/python3_vererbung.php>

```
class Company():
    def __init__(self):
        self.name = "BMV"

    def get_name(self):
        return self.name

class Car(Company):
    def __init__(self):
```

Kaddour Alnaasan, 0016285

2018-12-21

```
        self.id = 1
        self.name = "WV"
        self.name_car = "310i"

    def get_name(self):
        return self.name

c = car()
print(c.get_name)

>> WV
```

e) Überladen von Methoden:

Python unterschätzt nicht das Überladen von Methoden, aber man kann mit der Hilfe vom Default-Attribut eine überladende Methoden definieren.
< https://www.quora.com/What-is-the-difference-between-function-overloading-and-overriding-in-Python>

```
def sum_numbers(a, b, c=None):
    if c == None:
        return a + b
    else:
        return a + b + c
```

f) Polymorphie:

Polymorphie ist eine Funktion, die dieselbe Funktion von verschiedenen Klassen aufrufen kann.
< https://pythonspot.com/polymorphie/>

```
class Car():
    def wheels(self):
        print("Car has 4 Wheels.")


class Bicycle():
    def wheels(self):
        print("Bicycle has 2 Wheels.")


def print_wheels(thingsType):
    thingsType.wheels()

c = Car()
```

Kaddour Alnaasan, 0016285

2018-12-21

```
b = Bicycle()
print_wheels(c)
print_wheels(b)
```

g) Konstruktor:

Der Konstruktor ist eine Methode, um Instanzen der Klasse zu erzeugen.
< https://www.python-kurs.eu/klassen.php>

```
class Car():
    def __init__(self):
        self.id = 1
        self.name = "310i"

    def wheels(self):
        print("Car has 4 Wheels.")
```

h) Destruktor:

Destruktor ist eine Methode, die innerhalb einer Klasse definieren soll, um alle Referenz zwischen ihre Klasse und Instanz zu löschen.
< https://www.python-kurs.eu/klassen.php>

```
class Car():
    def __init__(self):
        self.id = 1
        self.name = "WV"
        self.name_car = "310i"

    def __del__(self):
        print("Destruktor gestartet")
car = Car()
car.__del__()
```

i) Kapselung und Information Hiding:

- Kapselung bedeutet, dass der Zugriff auf Data nur über Zugriffsmethoden möglich ist.
- Information Hiding besteht darin, interne Informationen und Implementierungsdetails nach außen zu verbergen.

< https://www.python-kurs.eu/python3_klassen.php>

Grundlagen der Programmierung 1

Prof. Dr. Detlef Krömker

Dipl. Inf. Alexander Wolodkin

Kaddour Alnaasan, 0016285

2018-12-21

```python
class Car():
    def __init__(self):
        self.id = 1
        self.name = "WV"
        self.name_car = "310i"

    def get_name(self):
        return self.name

car = Car()
print(car.get_name())
```