Grundlagen der Objektorientierten Programmierung

Proseminar Python

Sefkan Demir

Fachbereich Informatik
Fakultät für Mathematik, Informatik und Naturwissenschaften
Universität Hamburg

2022-06-28

Gliederung (Agenda)

1 Einleitung

2 Klassen

3 Objekte

4 Setter und getter

5 Scooping

6 Public-, Protected- und Private Instanzvariablen

7 Zusammenfassung

Einleitung

- Was ist Objekt Orientierte Programmierung
- Warum ist sie Sinnvoll?
- Was werden wir heute lernen?

Einleitung

- Aus welchen Instanzen setzt sich das Programm zusammen?
- Instanz Modellieren
- Instanzen in beziehung setzen
- Instanz = Objekt
- Attribut = Feld

Klassen

- Klassen in Python
  - beispiel folgt
  - class ist das Schlüsselwort
  - sind nur Objekttyp, aber selber kein Objekt

Klassen

- Klassen in Python
  - Auto als beispiel:

```
1  class auto()
2      def __init__(self):
3          self.auto_hersteller = None
4          self.auto_PS = None
5          self.auto_farbe = None
6          self.x_position = 5
7          self.y_position = 5
8
9      def fahren(self, x, y):
10         self.x_position += x
11         self.y_position += y
```

Objekte

- Objekte in Python
  - Wie werden sie Definiert?
  - Syntax: VariableName = klassenName()
  - Nur Referenz, also Speicheradresse ist gespeichert

- Objekte in Python

```
1 auto1 = auto()
2
3 print(auto1.auto_hersteller)
4
5 auto1.fahren()
```

Objekte erzeugen

- Rezept und Kuchen
- in Referenzvariable gespeichert
- Objekt = Instanz

Self-Parameter

- in Java: this
- gibt Referenz des ausführenden obj. zurück
- muss nicht übergeben werden als Parameter

Konstruktor

- init

```
1  # Neuer Bauplan fuer eine Person:
2  # Beim Erstellen eines Objekt der Klasse Person
3  # werden die Instanzvariablen direkt definiert.
4
5  class Person:
6      def __init__(self, name, vorname, geb_datum,
           ↪ gewicht):
7          self.name = name
8          self.vorname = vorname
9          self.geb_datum = geb_datum
10         self.gewicht = gewicht
11
12     //ruft Konstruktor auf
13     //Self muss nicht uebergeben werden
```

Listing 1: https://pythonbuch.com/objekte.html

Klassen Methoden, Klassen Attribute

- class Method
  - bezieht sich nicht auf instanz
  - classmethod Schlüsselwort
  - erster Parameter eine Refer. auf die Klasse,für die sie aufgerufen werden
  - Klassen Methoden sind ein Spezielles Konzept
  - hauptsaechlich in zusammenhang mit Metaklassen verwendet wird

- Klassen Attribute
  - Klassen Attribute kann direkt in body class erzeugt werden
  - direkt ueber Klasse als auch ueber Instanzen der Klasse auf K.Attribute zugreifen

```
1
2  class D:
3      X = 10
4
5      print(D.X)
6
7      d = D()
8      print(d.X)
9
10     //Ausgabe
11     //10
12     //10
```

Listing 2: Peter Kaiser Python 3 Das umfassende Handbuch

Statische Methoden

- ein alternativer Konstruktor
- können auch ohne Instanz aufgerufen werden
- staticmethod built-in funtion
- bezieht sich nicht auf eine Instanz
- kann aufgerufen werden, ohne dass zuvor eine instanz erzeugt wurde

Statische Methoden

```
1  class Date(object):
2      def __init__(self,year, month, day):
3          self.year = year
4          self.month = month
5          self.day = day
6      @staticmethod
7      def now():
8          t = time.localtime()
9          return Date(t.tm_year, t.tm_mon, t.tm_day)
10     @staticmethod
11     def tomorrow():
12         t = time.localtime(time.time() +86400)
13         return Date(t.tm_year, t.tm_mon, t.tm_day)
14   //beispiele
15   a = Date(2000, 1, 1)
16   b = Date.now()
17   c = Date.tomorrow()
```

Setter methode

- Hubraum private
- wir können Wert kontrollieren
- kein Negativer Wert

```
1  class Motorrad():
2      def __init__(self, marke, hubraum):
3          self.marke = marke
4          self.__hubraum = hubraum
5
6      def set_hubraum(self, kubik):
7          if (kubik <= 0):
8              print("Error: Negativer Wert fuer den
                   ↪ Hubraum! \
9              Der Wert wurde nicht geaendert")
10         else:
11             self.__hubraum = kubik
12             print("Hubraum wurde geaendert.")
```

Listing 4: https://pythonbuch.com/objekte.html

getter methode

- um den Wert abzufragen

```
1  class Motorrad():
2      def __init__(self, marke, hubraum):
3          self.marke = marke
4          self.__hubraum = hubraum
5
6      def set_hubraum(self, kubik):
7          if (kubik <= 0):
8              print("Error: Negativer Wert fuer den
                   ↪ Hubraum! \
9              Der Wert wurde nicht geaendert")
10         else:
11             self.__hubraum = kubik
12             print("Hubraum wurde geaendert.")
13
14     def get_hubraum(self):
15         return self.__hubraum
```

Scooping - Reichweite

- mangel an Reichweite unterschied zu java oder c++

```
1  class Foo(object)
2      def bar(self):
3          print("bar):
4
5      def spam("bar!):
6
7          bar(self):    //nameError funktioniert
               ↪ nicht
8          self.bar()    //funktioniert
9          foo.self.bar()    //funktioniert
```

Listing 6: David M. Beazley : Python Essential Reference

Public-, Protected- und Private Instanzvariablen

```
1 class A():
2     def __init__(self):
3         self.__priv = "Ich bin privat"
4         self._prot = "Ich bin protected"
5         self.pub = "Ich bin oeffentlich"
```

Listing 7: https://pythonbuch.com/objekte.html

Zusammenfassung

- Klassen
  - Klassen werden mit Class deklariert
  - Klassen sind keine objekte
- Objekte
  - syntax
  - Objekte erzeugen
- Welche Methoden und unterschiede zu anderen Programmier Sprachen
  - Self-Parameter
  - Die Reichweite

Literatur

- David M. Beazley : Python Essential Reference
- Peter Kaiser, Johannes Ernesti : Python 3 Das umfassende Handbuch
- Bernhard Lahres co : Objektorientierte Programmierung Das umfassende Handbuch
- https://pythonbuch.com/objekte.html
- https://www.programiz.com/python-programming/methods/built-in/staticmethod: :text=What