

Informatik I: Einführung in die Programmierung

8. Objekte und Datenklassen

Albert-Ludwigs-Universität Freiburg



UNI
FREIBURG

Prof. Dr. Peter Thiemann

18. November 2025

1 Objekte und Datenklassen

- Objekte
- Datenklassen für Records
- Klassendefinition
- Erzeugung von Instanzen
- Funktionen auf Records
- Geschachtelte Records

Objekte und
Datenklas-
sen

Objekte
Datenklassen für
Records
Klassendefinition
Erzeugung von
Instanzen
Funktionen auf
Records
Geschachtelte
Records

Zusammen-
fassung

1 Objekte und Datenklassen

- Objekte
- Datenklassen für Records
- Klassendefinition
- Erzeugung von Instanzen
- Funktionen auf Records
- Geschachtelte Records

Objekte und
Datenklas-
sen

Objekte

Datenklassen für
Records

Klassendefinition

Erzeugung von
Instanzen

Funktionen auf
Records

Geschachtelte
Records

Zusammen-
fassung

Objekte und Attribute



- Alle Werte in Python sind *Objekte*.
- Damit ist gemeint, dass sie assoziierte *Attribute* und *Methoden* haben, auf die mit der *Punktnotation*

expr.attribut bzw. *expr.method(...)*

zugegriffen werden kann:

```
>>> " ".join(["I","think","therefore"])
'I think therefore'
>>> "spam".index("a")
2
>>> (10 + 10).__neg__()
-20
```

Objekte und
Datenklassen

Objekte

Datenklassen für
Records

Klassendefinition
Erzeugung von
Instanzen

Funktionen auf
Records

Geschachtelte
Records

Zusammen-
fassung

1 Objekte und Datenklassen

- Objekte
- Datenklassen für Records
- Klassendefinition
- Erzeugung von Instanzen
- Funktionen auf Records
- Geschachtelte Records

Objekte und
Datenklas-
sen

Objekte

Datenklassen für
Records

Klassendefinition

Erzeugung von
Instanzen

Funktionen auf
Records

Geschachtelte
Records

Zusammen-
fassung

Klassen beschreiben Objekte

- Bisher haben wir vorgefertigte Objekte verwendet.
- Jetzt beginnen wir selbst welche zu bauen!
- Dafür benötigen wir einen Bauplan, eine **Klasse**.

Definition

Ein **Record** ist ein Objekt, das mehrere untergeordnete Objekte, die **Attribute**, enthält. Eine **Klasse** definiert (zunächst nur), welche Attribute vorhanden sind.

Alternative Bezeichnungen

- Record: auch **Struct**; deutsch: Reihung, Struktur
- Objekte: auch **Instanzen**.
- Attribute: auch **Felder** oder **Properties**.

Objekte und
Datenklassen

Objekte

Datenklassen für
Records

Klassendefinition

Erzeugung von
Instanzen

Funktionen auf
Records

Geschachtelte
Records

Zusammen-
fassung

Beispiel für Datenmodellierung mit Records: Ware



Beschreibung für Ware

Ein Händler beschreibt eine Ware durch den **Namen** und den **Angebotspreis**.

Schritt 1: Bezeichner und Datentypen

Ein Händler beschreibt eine Ware (**Article**) durch die **Attribute**

- **name** : **str**, den Namen und
- **price** : **int**, den Angebotspreis (**in cent**), immer ≥ 0 .

Objekte und
Datenklassen

Objekte

Datenklassen für
Records

Klassendefinition

Erzeugung von
Instanzen

Funktionen auf
Records

Geschachtelte
Records

Zusammen-
fassung

1 Objekte und Datenklassen



- Objekte
- Datenklassen für Records
- Klassendefinition
- Erzeugung von Instanzen
- Funktionen auf Records
- Geschachtelte Records

Objekte und
Datenklas-
sen

Objekte

Datenklassen für
Records

Klassendefinition

Erzeugung von
Instanzen

Funktionen auf
Records

Geschachtelte
Records

Zusammen-
fassung

Definieren von Records und Klassen



```
from dataclasses import dataclass
@dataclass
class Article:
    name : str
    price : int
```

- Neue Records und Klassen werden mit der `class`-Anweisung eingeführt (Konvention: **CamelCase**-Namen).
- Die `class`-Anweisung muss **ausgeführt werden**. Sie darf nicht in einer bedingten Anweisung verborgen sein!
- Sie definiert einen **neuen Typ** mit Namen `Article`.
- Im Rumpf werden Attribute der Klasse definiert (Funktionalität von `dataclass`).

Objekte und
Datenklassen

Objekte
Datenklassen für
Records

Klassendefinition
Erzeugung von
Instanzen

Funktionen auf
Records
Geschachtelte
Records

Zusammen-
fassung

1 Objekte und Datenklassen



- Objekte
- Datenklassen für Records
- Klassendefinition
- Erzeugung von Instanzen
- Funktionen auf Records
- Geschachtelte Records

Objekte und
Datenklas-
sen

Objekte

Datenklassen für
Records

Klassendefinition

Erzeugung von
Instanzen

Funktionen auf
Records

Geschachtelte
Records

Zusammen-
fassung

Einfache Klasse: Article



Schritt 2: Klassengerüst

```
from dataclasses import dataclass
@dataclass
class Article:
    name : str
    price : int
```

- Jeder Aufruf der Klasse Article als Funktion mit zwei Parametern (name, price) erzeugt eine neue Instanz der Klasse.
- Alle Instanzen haben garantiert die Attribute name und price, die beim Aufruf initialisiert werden.
- Instanzen von Datenklassen sind gleich (==), falls alle Attribute gleich sind.

Objekte und
Datenklassen

Objekte

Datenklassen für
Records

Klassendefinition

Erzeugung von
Instanzen

Funktionen auf
Records

Geschachtelte
Records

Zusammen-
fassung

Beispiel



```
>>> from dataclasses import dataclass
>>> @dataclass
... class Article:
...     name : str
...     price : int
...
>>> phone = Article("Smartphone", 49500)
>>> phone
Article(name='Smartphone', price=49500)
>>> phone.price * 0.19 / 1.19
7903.361344537815
>>> myphone = Article("Smartphone", 49500)
>>> myphone == phone
True
```

Objekte und
Datenklassen

Objekte

Datenklassen für
Records

Klassendefinition

Erzeugung von
Instanzen

Funktionen auf
Records

Geschachtelte
Records

Zusammen-
fassung

1 Objekte und Datenklassen



- Objekte
- Datenklassen für Records
- Klassendefinition
- Erzeugung von Instanzen
- Funktionen auf Records
- Geschachtelte Records

Objekte und
Datenklas-
sen

Objekte

Datenklassen für
Records

Klassendefinition

Erzeugung von
Instanzen

Funktionen auf
Records

Geschachtelte
Records

Zusammen-
fassung

Beispiel

Angebotspreis

Der Händler will seine Preise am Black Friday um 25% herabsetzen. Der Angebotspreis soll dynamisch nur an der Kasse berechnet werden.

Schritt 1: Bezeichner und Datentypen

Der Händler braucht für die Kasse eine Funktion `sale_price`, die als Parameter

- `article` : `Article`, die Ware, und
- `discount` : `int`, den Rabattsatz (in Prozent zwischen 0 und 100)

erwartet und den Verkaufspreis : `int` (in Cent) berechnet.

Objekte und
Datenklassen

Objekte

Datenklassen für
Records

Klassendefinition

Erzeugung von
Instanzen

Funktionen auf
Records

Geschachtelte
Records

Zusammen-
fassung

Beispiel Angebotspreis

Schritt 2: Funktionsgerüst

```
def sale_price (
    article : Article,
    discount : int) -> int:
    # fill in
    return 0
```

- Neu: der Rumpf kann die Attribute von `article` über die Punktnotation verwenden.

Objekte und Datenklassen

Objekte
Datenklassen für Records

Klassendefinition
Erzeugung von Instanzen

Funktionen auf Records
Geschachtelte Records

Zusammenfassung

Beispiel Angebotspreis

Schritt 3: Beispiele

```
a1 = Article ("Mausefalle", 2000)
a2 = Article ("Promo Lutscher", 0)
a3 = Article ("Nougat", 2000)
assert sale_price (a1, 25) == 1500
assert sale_price (a1, 10) == 1800
assert sale_price (a3, 10) == 1800
assert sale_price (a2, 25) == 0
```

Objekte und
Datenklassen

Objekte

Datenklassen für
Records

Klassendefinition
Erzeugung von
Instanzen

Funktionen auf
Records

Geschachtelte
Records

Zusammen-
fassung

Beispiel Angebotspreis

Schritt 4: Funktionsdefinition

```
def sale_price (
    article : Article,
    discount : int) -> int:
    return article.price * (100 - discount) // 100
```

Objekte und
Datenklassen

Objekte

Datenklassen für
Records

Klassendefinition
Erzeugung von
Instanzen

Funktionen auf
Records

Geschachtelte
Records

Zusammen-
fassung

1 Objekte und Datenklassen



- Objekte
- Datenklassen für Records
- Klassendefinition
- Erzeugung von Instanzen
- Funktionen auf Records
- Geschachtelte Records

Objekte und
Datenklas-
sen

Objekte

Datenklassen für
Records

Klassendefinition

Erzeugung von
Instanzen

Funktionen auf
Records

Geschachtelte
Records

Zusammen-
fassung

Beispiel

Terminplanung

Ein Termin hat einen **Titel**, **Teilnehmer**, eine **Anfangszeit** und eine **Endzeit**. Eine Zeit wird durch **Stunde** und **Minute** repräsentiert.

- 1 Wie lange dauert ein Termin?
- 2 Stehen zwei Termine in Konflikt?

Bemerkungen

- Zwei Datenklassen beteiligt: Termin und Zeit
- Frage 2 muss noch präzisiert werden

Objekte und
Datenklassen

Objekte

Datenklassen für
Records

Klassendefinition
Erzeugung von
Instanzen

Funktionen auf
Records

Geschachtelte
Records

Zusammen-
fassung

Schritt 1: Bezeichner und Datentypen

Eine Zeit Time besteht aus

- einer Stundenzahl `hour` : `int` zwischen 0 und 23 inklusive.
- einer Minutenzahl `minute` : `int` zwischen 0 und 59 inklusive.

Schritt 2: Klassengerüst

```
@dataclass
```

```
class Time:
```

```
    hour    : int # 0 <= hour < 24
    minute : int # 0 <= minute < 60
```

Objekte und
Datenklassen

Objekte
Datenklassen für
Records

Klassendefinition
Erzeugung von
Instanzen

Funktionen auf
Records

Geschachtelte
Records

Zusammen-
fassung

Terminplanung/Termin

Schritt 1: Bezeichner und Datentypen

Ein Termin Appointment hat

- einen Titel title : str
- (mehrere) Teilnehmer participants : list[str]
- eine Anfangszeit start : Time
- eine Endzeit end : Time nicht vor start

Bemerkung

- Ein Appointment-Objekt enthält zwei Time-Objekte

Objekte und
Datenklassen

Objekte
Datenklassen für
Records

Klassendefinition
Erzeugung von
Instanzen

Funktionen auf
Records

Geschachtelte
Records

Zusammen-
fassung

Schritt 2: Klassengerüst

```
@dataclass
class Appointment:
    title: str
    participants: list[str]
    start: Time
    end: Time      # not less than start
```

Objekte und
Datenklassen

Objekte

Datenklassen für
Records

Klassendefinition
Erzeugung von
Instanzen

Funktionen auf
Records

Geschachtelte
Records

Zusammen-
fassung

Schritt 1: Bezeichner und Datentypen

Wie lange dauert ein Termin?

Die Funktion `duration` nimmt einen Termin `app : Appointment` und bestimmt seine Dauer in Minuten (`int`).

Schritt 2: Funktionsgerüst

```
def duration (app : Appointment) -> int:  
    # fill in  
    return 0
```

Objekte und
Datenklassen

Objekte

Datenklassen für
Records

Klassendefinition

Erzeugung von
Instanzen

Funktionen auf
Records

Geschachtelte
Records

Zusammen-
fassung

Funktionen der Terminplanung

Schritt 3: Beispiele

```
t1 = Time (12, 50)
t2 = Time (13, 10)
t3 = Time (10, 05)
t4 = Time (12, 45)
m1 = Appointment ("lunch", [], t1, t2)
m2 = Appointment ("lecture", [], t3, t4)
m3 = Appointment ("alarm", [], t4, t4)
assert duration(m1) == 20
assert duration(m2) == 160
assert duration(m3) == 0
```

Objekte und
Datenklassen

Objekte
Datenklassen für
Records

Klassendefinition
Erzeugung von
Instanzen

Funktionen auf
Records

Geschachtelte
Records

Zusammen-
fassung

Schritt 4: Funktionsdefinition

```
def duration (app : Appointment) -> int:  
    return time_difference (app.end, app.start)
```

Prinzip Wunschdenken

- Zur Erledigung der Aufgabe in Appointment benötigen wir eine Operation, die nur mit Time zu tun hat.
- Daher lagern wir sie in eine Hilfsfunktion aus!
- **Wunschdenken** heißt, wir geben der gewünschten Funktion einen Namen und erstellen einen Vertrag für sie.
- Dann verwenden wir sie, bevor sie entworfen und implementiert ist.

Objekte und
Datenklassen

Objekte

Datenklassen für
Records

Klassendefinition
Erzeugung von
Instanzen

Funktionen auf
Records

Geschachtelte
Records

Zusammen-
fassung

Schritt 1: Bezeichner und Datentypen

Bestimme die Differenz zweier Zeitangaben.

Die Funktion `time_difference` nimmt zwei Zeitangaben `t1, t2 : Time` und bestimmt die Differenz `t1 - t2` in Minuten (`int`). Dabei nehmen wir an, dass `t1 >= t2` ist.

Schritt 2: Funktionsgerüst

```
def time_difference (t1 : Time, t2 : Time) -> int:  
    # fill in  
    return 0
```

Objekte und
Datenklassen

Objekte

Datenklassen für
Records

Klassendefinition
Erzeugung von
Instanzen

Funktionen auf
Records

Geschachtelte
Records

Zusammen-
fassung

Schritt 3: Beispiele

```
t1 = Time (12, 50)
t2 = Time (13, 10)
t3 = Time (10, 05)
t4 = Time (12, 45)
assert time_difference(t2, t1) == 20
assert time_difference(t4, t3) == 160
assert time_difference(t1, t1) == 0
```

Objekte und
Datenklassen

Objekte

Datenklassen für
Records

Klassendefinition
Erzeugung von
Instanzen

Funktionen auf
Records

Geschachtelte
Records

Zusammen-
fassung

Schritt 4: Funktionsdefinition

```
def time_difference (t1 : Time, t2 : Time) -> int:  
    return ((t1.hour - t2.hour) * 60  
            + t1.minute - t2.minute)
```

Konvention

- In Funktionen die Punktnotation nur zum Zugriff auf direkte Attribute verwenden.
- Also nicht tiefer als eine Ebene zugreifen.

Objekte und
Datenklassen

Objekte

Datenklassen für
Records

Klassendefinition
Erzeugung von
Instanzen

Funktionen auf
Records

Geschachtelte
Records

Zusammen-
fassung

Präzisierung der Fragestellung

Stehen zwei Termine in Konflikt?

- Überschneiden sich zwei Termine zeitlich?
- Haben zwei Termine gemeinsame Teilnehmer?
- Konflikt nur, falls beides zutrifft!

Schritt 1: Bezeichner und Datentypen

Stehen zwei Termine in Konflikt?

Die Funktion `conflict` nimmt zwei Termine `a1, a2 : Appointment` und stellt fest, ob sie in Konflikt stehen (`bool`).

Objekte und
Datenklassen

Objekte
Datenklassen für
Records

Klassendefinition
Erzeugung von
Instanzen

Funktionen auf
Records

Geschachtelte
Records

Zusammen-
fassung

Schritt 2: Funktionsgerüst

```
def conflict (a1 : Appointment,  
              a2 : Appointment) -> bool:  
    # fill in  
    return False
```

Objekte und
Datenklassen

Objekte

Datenklassen für
Records

Klassendefinition
Erzeugung von
Instanzen

Funktionen auf
Records

Geschachtelte
Records

Zusammen-
fassung

Schritt 3: Beispiele

```
t1 = Time (12, 00)
t2 = Time (12, 30)
t3 = Time (10, 05)
t4 = Time (12, 45)
a1 = Appointment ("lunch", ["jim", "jack"], t1, t2)
a2 = Appointment ("lecture", ["jeff", "jim"], t3, t4)
a3 = Appointment ("coffee", ["jack", "jill"], t2, t4)
#
assert conflict(a1, a2) and conflict (a2, a1)
assert not conflict(a1, a3)
assert not conflict(a2, a3)
```

Objekte und
Datenklassen

Objekte

Datenklassen für
Records

Klassendefinition
Erzeugung von
Instanzen

Funktionen auf
Records

Geschachtelte
Records

Zusammen-
fassung

Schritt 4: Funktionsdefinition

```
def conflict (a1 : Appointment,  
              a2 : Appointment) -> bool:  
    time_ok = (before(a1.end, a2.start)  
               or before(a2.end, a1.start))  
    participants_ok = not (  
        intersection (a1.participants, a2.participants))  
    return not (time_ok and participants_ok)
```

Objekte und
Datenklassen

Objekte

Datenklassen für
Records

Klassendefinition
Erzeugung von
Instanzen

Funktionen auf
Records

Geschachtelte
Records

Zusammen-
fassung

Terminkonflikt

Wunschdenken

```
def before (t1 : Time, t2 : Time) -> bool:  
    ''' is t1 earlier than t2? '''  
    return False  
  
def intersection (lst1 : list, lst2 : list) -> list:  
    ''' returns the list of elements both in lst1 and lst2 '''  
    return []
```

Weitere Ausführung selbst

- `before`: Bedingung auf den Attributen von `Time`-Objekten
- `intersection`: `for`-Schleife auf einer der Listen, Akkumulator fürs Ergebnis

Objekte und
Datenklassen

Objekte

Datenklassen für
Records

Klassendefinition

Erzeugung von
Instanzen

Funktionen auf
Records

Geschachtelte
Records

Zusammen-
fassung

2 Zusammenfassung



UNI
FREIBURG

Objekte und
Datenclas-
sen

Zusammen-
fassung

Zusammenfassung



UNI
FREIBURG

Objekte und
Datenklassen

Zusammen-
fassung

- Alle Werte in Python sind Objekte.
- Veränderliche Objekte besitzen eine **Identität**.
- Eine **Klasse** beschreibt Objekte/Instanzen.
- Eine Instanz enthält **Attribute**, d.h. untergeordnete Objekte.
- Eine **Datenklasse** (Record) enthält nichts anderes als Attribute.
- Funktionsentwurf mit **einfachen Records**.
- Funktionsentwurf mit **geschachtelten Records**.