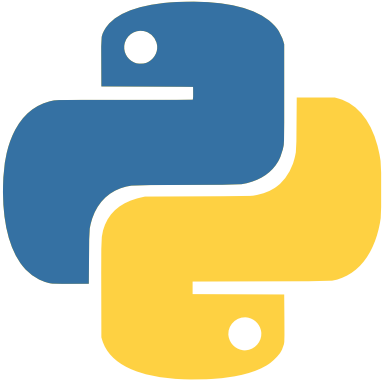
Universität Stuttgart
Projekt digit@L – BOOST. SKILLS. SUPPORT.



Programmierkurs Python

Klassen und Objekte

Dominik Göddeke

IANS SimTech

Stiftung Innovation in der Hochschullehre

Klassen und Objekte

Klassen und Objekte

- Bisher: Wort `class` ignoriert

Klassen und Objekte

- Bisher: Wort `class` ignoriert
- Moral: „alles“ in Python verbunden mit Klassen

Klassen und Objekte

- Bisher: Wort `class` ignoriert
- Moral: „alles“ in Python verbunden mit Klassen
- Von profanen Zahlen über Zeichenketten bis hin zu Funktionen

Klassen und Objekte

- Bisher: Wort `class` ignoriert
- Moral: „alles“ in Python verbunden mit Klassen
- Von profanen Zahlen über Zeichenketten bis hin zu Funktionen
- Ziel dieser Einheit: grundlegendes Verständnis dafür

Klassen und Objekte

- Bisher: Wort `class` ignoriert
- Moral: „alles“ in Python verbunden mit Klassen
- Von profanen Zahlen über Zeichenketten bis hin zu Funktionen
- Ziel dieser Einheit: grundlegendes Verständnis dafür
- Nebeneffekt (aber nicht primäres Ziel): Fähigkeit zur **objektorientierten Programmierung**

Klassen und Objekte

- Im Beispiel: Zahlen `1.2` und `2.3`, Zeichenketten `„a“`, `“Hallo Welt“` heißen **Objekte** bzw. **Instanzen** der zugrundeliegenden **Klassen**

Klassen und Objekte

- Im Beispiel: Zahlen `1.2` und `2.3`, Zeichenketten `„a“`, `“Hallo Welt“` heißen **Objekte** bzw. **Instanzen** der zugrundeliegenden **Klassen**
- Ziel nun: recht abstraktes Vorgehen, zunächst ohne Python

Klassen und Objekte

- Im Beispiel: Zahlen `1.2` und `2.3`, Zeichenketten `„a“`, `“Hallo Welt“` heißen **Objekte** bzw. **Instanzen** der zugrundeliegenden **Klassen**
- Ziel nun: recht abstraktes Vorgehen, zunächst ohne Python
- Grund: objektoriertierte Programmierung (OOP) ist **Programmierparadigma**

Klassen und Objekte

- Im Beispiel: Zahlen `1.2` und `2.3`, Zeichenketten `„a“`, `“Hallo Welt“` heißen **Objekte** bzw. **Instanzen** der zugrundeliegenden **Klassen**
- Ziel nun: recht abstraktes Vorgehen, zunächst ohne Python
- Grund: objektoriertierte Programmierung (OOP) ist **Programmierparadigma**
- Beinhaltet Aspekte der **Modellierung**

Klassen und Objekte

- Im Beispiel: Zahlen `1.2` und `2.3`, Zeichenketten `„a“`, `“Hallo Welt“` heißen **Objekte** bzw. **Instanzen** der zugrundeliegenden **Klassen**
- Ziel nun: recht abstraktes Vorgehen, zunächst ohne Python
- Grund: objektoriertierte Programmierung (OOP) ist **Programmierparadigma**
- Beinhaltet Aspekte der **Modellierung**
  - Nicht nur: schreiben eines Programms das funktioniert

Klassen und Objekte

- Im Beispiel: Zahlen `1.2` und `2.3`, Zeichenketten `„a“`, `“Hallo Welt“` heißen **Objekte** bzw. **Instanzen** der zugrundeliegenden **Klassen**
- Ziel nun: recht abstraktes Vorgehen, zunächst ohne Python
- Grund: objektoriertierte Programmierung (OOP) ist **Programmierparadigma**
- Beinhaltet Aspekte der **Modellierung**
  - Nicht nur: schreiben eines Programms das funktioniert
  - Sondern auch: Modellierung eines Systems in der realen Welt, so dass es im Computer abbildbar ist als geschlossene Entität

Klassen und Objekte

- Im Beispiel: Zahlen `1.2` und `2.3`, Zeichenketten `„a“`, `“Hallo Welt“` heißen **Objekte** bzw. **Instanzen** der zugrundeliegenden **Klassen**
- Ziel nun: recht abstraktes Vorgehen, zunächst ohne Python
- Grund: objektoriertierte Programmierung (OOP) ist **Programmierparadigma**
- Beinhaltet Aspekte der **Modellierung**
  - Nicht nur: schreiben eines Programms das funktioniert
  - Sondern auch: Modellierung eines Systems in der realen Welt, so dass es im Computer abbildbar ist als geschlossene Entität
  - Und: **Wiederverwendbarkeit** der Entität in anderen Kontexten

Abstrakte prosaische Definition

Abstrakte prosaische Definition

- Kernidee von OOP: klare **Trennung** zwischen **(konkreten) Daten** und **(allgemeinen) Methoden**, die unterschiedliche, aber prinzipiell gleichartige Daten verarbeiten

Abstrakte prosaische Definition

- Kernidee von OOP: klare **Trennung** zwischen **(konkreten) Daten** und **(allgemeinen) Methoden**, die unterschiedliche, aber prinzipiell gleichartige Daten verarbeiten
- Klassen: gewissermaßen „Konstruktionspläne“ (Vorlagen, abstrakte Definitionen, Rahmenbeschreibungen)

Abstrakte prosaische Definition

- Kernidee von OOP: klare **Trennung** zwischen **(konkreten) Daten** und **(allgemeinen) Methoden**, die unterschiedliche, aber prinzipiell gleichartige Daten verarbeiten
- Klassen: gewissermaßen „Konstruktionspläne“ (Vorlagen, abstrakte Definitionen, Rahmenbeschreibungen)
- Klassen spezifizieren zwei Dinge

Abstrakte prosaische Definition

- Kernidee von OOP: klare **Trennung** zwischen **(konkreten) Daten** und **(allgemeinen) Methoden**, die unterschiedliche, aber prinzipiell gleichartige Daten verarbeiten
- Klassen: gewissermaßen „Konstruktionspläne“ (Vorlagen, abstrakte Definitionen, Rahmenbeschreibungen)
- Klassen spezifizieren zwei Dinge
  - „Platzhalter“ für konkrete Daten (konzeptionell ähnlich), Sprechweise: **Eigenschaften** oder **Attribute**

Abstrakte prosaische Definition

- Kernidee von OOP: klare **Trennung** zwischen **(konkreten) Daten** und **(allgemeinen) Methoden**, die unterschiedliche, aber prinzipiell gleichartige Daten verarbeiten
- Klassen: gewissermaßen „Konstruktionspläne“ (Vorlagen, abstrakte Definitionen, Rahmenbeschreibungen)
- Klassen spezifizieren zwei Dinge
  - „Platzhalter“ für konkrete Daten (konzeptionell ähnlich), Sprechweise: **Eigenschaften** oder **Attribute**
  - „Operationen“ die die Daten manipulieren, unabhängig von den konkreten Werten der Daten, Sprechweise: **Funktionen** oder **Methoden**

Verdeutlichung

Verdeutlichung

- Beispiel: Modellierung verschiedener Fahrzeuge

Verdeutlichung

- Beispiel: Modellierung verschiedener Fahrzeuge
- Abstrakte Eigenschaften, d.h. Attribute der Klasse `Fahrzeug`

Verdeutlichung

- Beispiel: Modellierung verschiedener Fahrzeuge
- Abstrakte Eigenschaften, d.h. Attribute der Klasse `Fahrzeug`
  - Farbe, Sitzplätze, Batteriestand, Höchstgeschwindigkeit, ...

Verdeutlichung

- Beispiel: Modellierung verschiedener Fahrzeuge
- Abstrakte Eigenschaften, d.h. Attribute der Klasse `Fahrzeug`
  - Farbe, Sitzplätze, Batteriestand, Höchstgeschwindigkeit, ...
- Operationen für Fahrzeuge, d.h. Methoden der Klasse `Fahrzeug`

Verdeutlichung

- Beispiel: Modellierung verschiedener Fahrzeuge
- Abstrakte Eigenschaften, d.h. Attribute der Klasse `Fahrzeug`
  - Farbe, Sitzplätze, Batteriestand, Höchstgeschwindigkeit, ...
- Operationen für Fahrzeuge, d.h. Methoden der Klasse `Fahrzeug`
  - `aufladen()`, `entladen()`, `reichweite()` oder `umlackieren()`

Abstrakte prosaische Definition

Abstrakte prosaische Definition

- Beobachtung: formulierte Ziele nicht mit Klassen alleine erreichbar

Abstrakte prosaische Definition

- Beobachtung: formulierte Ziele nicht mit Klassen alleine erreichbar
- Deshalb: **Objekte** als **Instanzen** (konkrete Realisierungen) von Klassen

Abstrakte prosaische Definition

- Beobachtung: formulierte Ziele nicht mit Klassen alleine erreichbar
- Deshalb: **Objekte** als **Instanzen** (konkrete Realisierungen) von Klassen
- Füllen die Platzhalter für die Daten

Abstrakte prosaische Definition

- Beobachtung: formulierte Ziele nicht mit Klassen alleine erreichbar
- Deshalb: **Objekte** als **Instanzen** (konkrete Realisierungen) von Klassen
- Füllen die Platzhalter für die Daten
- Genügen der Vorlage (dem Konstruktionsplan)

Abstrakte prosaische Definition

- Beobachtung: formulierte Ziele nicht mit Klassen alleine erreichbar
- Deshalb: **Objekte** als **Instanzen** (konkrete Realisierungen) von Klassen
- Füllen die Platzhalter für die Daten
- Genügen der Vorlage (dem Konstruktionsplan)
- Bereits Klassen bieten Methoden zur Datenmanipulation an

Abstrakte prosaische Definition

- Beobachtung: formulierte Ziele nicht mit Klassen alleine erreichbar
- Deshalb: **Objekte** als **Instanzen** (konkrete Realisierungen) von Klassen
- Füllen die Platzhalter für die Daten
- Genügen der Vorlage (dem Konstruktionsplan)
- Bereits Klassen bieten Methoden zur Datenmanipulation an
- **Kapselung**: Daten eines Objekts nur über dafür vorgesehene Methoden manipulierbar

Verdeutlichung

Verdeutlichung

- Zwei konkrete Instanzen der Klasse `Fahrzeug`

Verdeutlichung

- Zwei konkrete Instanzen der Klasse `Fahrzeug`
  - Familienkutsche mit Farbe rot, 5 Sitzplätzen, einem aktuellen Batteriestand in Ampere-Stunden und irgendeiner Maximalgeschwindigkeit

Verdeutlichung

- Zwei konkrete Instanzen der Klasse `Fahrzeug`
  - Familienkutsche mit Farbe rot, 5 Sitzplätzen, einem aktuellen Batteriestand in Ampere-Stunden und irgendeiner Maximalgeschwindigkeit
  - SuperTurbo mit Farbe giftgrün, 2 Sitzplätzen, usw.

Verdeutlichung

- Zwei konkrete Instanzen der Klasse `Fahrzeug`
  - Familienkutsche mit Farbe rot, 5 Sitzplätzen, einem aktuellen Batteriestand in Ampere-Stunden und irgendeiner Maximalgeschwindigkeit
  - SuperTurbo mit Farbe giftgrün, 2 Sitzplätzen, usw.
- Objekte spezifizieren definierte Attribute der Klasse mit konkreten Werten

Verdeutlichung

- Zwei konkrete Instanzen der Klasse `Fahrzeug`
  - Familienkutsche mit Farbe rot, 5 Sitzplätzen, einem aktuellen Batteriestand in Ampere-Stunden und irgendeiner Maximalgeschwindigkeit
  - SuperTurbo mit Farbe giftgrün, 2 Sitzplätzen, usw.
- Objekte spezifizieren definierte Attribute der Klasse mit konkreten Werten
- `umlackieren()` kapselt für jedes Fahrzeug-Objekt Änderung des Attributs `Farbe`

Verdeutlichung

- Zwei konkrete Instanzen der Klasse `Fahrzeug`
  - Familienkutsche mit Farbe rot, 5 Sitzplätzen, einem aktuellen Batteriestand in Ampere-Stunden und irgendeiner Maximalgeschwindigkeit
  - SuperTurbo mit Farbe giftgrün, 2 Sitzplätzen, usw.
- Objekte spezifizieren definierte Attribute der Klasse mit konkreten Werten
- `umlackieren()` kapselt für jedes Fahrzeug-Objekt Änderung des Attributs `Farbe`
- Analog `aufladen()` und `entladen()` für Attribut `Batteriestand`

Verdeutlichung

- Zwei konkrete Instanzen der Klasse `Fahrzeug`
  - Familienkutsche mit Farbe rot, 5 Sitzplätzen, einem aktuellen Batteriestand in Ampere-Stunden und irgendeiner Maximalgeschwindigkeit
  - SuperTurbo mit Farbe giftgrün, 2 Sitzplätzen, usw.
- Objekte spezifizieren definierte Attribute der Klasse mit konkreten Werten
- `umlackieren()` kapselt für jedes Fahrzeug-Objekt Änderung des Attributs `Farbe`
- Analog `aufladen()` und `entladen()` für Attribut `Batteriestand`
- `reichweite()`: reine Leseoperation für Attribut `Batteriestand`, inkl. Umrechnung Amperestunden zu Kilometer

Abstrakte prosaische Definition

Abstrakte prosaische Definition

- **Interaktion von Objekten** über weitere Methoden

Abstrakte prosaische Definition

- **Interaktion von Objekten** über weitere Methoden
- Unterstützte Interaktion bereits auf Klassenebene vorgegeben

Abstrakte prosaische Definition

- **Interaktion von Objekten** über weitere Methoden
- Unterstützte Interaktion bereits auf Klassenebene vorgegeben
- Kein Datenzugriff aufgrund der Kapselung

Verdeutlichung

Verdeutlichung

- Einbau eines Pannenservice in Klasse `Fahrzeug`: Überbrückungskabel

Verdeutlichung

- Einbau eines Pannenservice in Klasse `Fahrzeug`: Überbrückungskabel
- Mögliche Methode: `querladen(f)`

Verdeutlichung

- Einbau eines Pannenservice in Klasse `Fahrzeug`: Überbrückungskabel
- Mögliche Methode: `querladen(f)`
- Argument `f`: weitere Instanz der Klasse `Fahrzeug`

```
superturbo.querladen(familienkutsche)
```

Verdeutlichung

- Einbau eines Pannenservice in Klasse `Fahrzeug`: Überbrückungskabel
- Mögliche Methode: `querladen(f)`
- Argument `f`: weitere Instanz der Klasse `Fahrzeug`

  `superturbo.querladen(familienkutsche)`
- Unter der Haube schön gekapselt: `a = entladen(1 Ah)` und `aufladen(a)`

Beispiele möglicher Klassen aus der Mathematik

Beispiele möglicher Klassen aus der Mathematik

- **Vektoren** $x = (x_1, \ldots, x_n)^\top$

Beispiele möglicher Klassen aus der Mathematik

- **Vektoren** $x = (x_1, \ldots, x_n)^\top$
- **Quiz: sinnvolle Attribute und Methoden?**

Beispiele möglicher Klassen aus der Mathematik

- **Vektoren** $x = (x_1, \ldots, x_n)^\top$
- **Quiz: sinnvolle Attribute und Methoden?**
  - Koordinaten x_i, Dimension n, zugrundeliegender Raum

Beispiele möglicher Klassen aus der Mathematik

- **Vektoren** $x = (x_1, \ldots, x_n)^\top$
- **Quiz: sinnvolle Attribute und Methoden?**
  - Koordinaten x_i, Dimension n, zugrundeliegender Raum
  - Lesemethoden für Attribute, Norm, Addition mit anderem Vektor, Skalierung, ...

Beispiele möglicher Klassen aus der Mathematik

- **Vektoren** $x = (x_1, \ldots, x_n)^\top$
- **Quiz: sinnvolle Attribute und Methoden?**
  - Koordinaten x_i, Dimension n, zugrundeliegender Raum
  - Lesemethoden für Attribute, Norm, Addition mit anderem Vektor, Skalierung, ...
  - In Python: realisiert bspw. in NumPy

Beispiele möglicher Klassen aus der Mathematik

- **Vektoren** $x = (x_1, \ldots, x_n)^\top$
- **Quiz: sinnvolle Attribute und Methoden?**
  - Koordinaten x_i, Dimension n, zugrundeliegender Raum
  - Lesemethoden für Attribute, Norm, Addition mit anderem Vektor, Skalierung, ...
  - In Python: realisiert bspw. in NumPy
- **Polynome** $p(x) = \sum_{i=0}^{n} a_i x^i$

Beispiele möglicher Klassen aus der Mathematik

- **Vektoren** $x = (x_1, \ldots, x_n)^\top$
- **Quiz: sinnvolle Attribute und Methoden?**
  - Koordinaten x_i, Dimension n, zugrundeliegender Raum
  - Lesemethoden für Attribute, Norm, Addition mit anderem Vektor, Skalierung, ...
  - In Python: realisiert bspw. in NumPy
- **Polynome** $p(x) = \sum_{i=0}^{n} a_i x^i$
- **Quiz: sinnvolle Attribute und Methoden?**

Beispiele möglicher Klassen aus der Mathematik

- **Vektoren** $x = (x_1, \ldots, x_n)^\top$
- **Quiz: sinnvolle Attribute und Methoden?**
  - Koordinaten x_i, Dimension n, zugrundeliegender Raum
  - Lesemethoden für Attribute, Norm, Addition mit anderem Vektor, Skalierung, ...
  - In Python: realisiert bspw. in NumPy
- **Polynome** $p(x) = \sum_{i=0}^{n} a_i x^i$
- **Quiz: sinnvolle Attribute und Methoden?**
  - Dimension n, Koeffizienten a_i, zugrundeliegender Raum

Beispiele möglicher Klassen aus der Mathematik

- **Vektoren** $x = (x_1, \ldots, x_n)^\top$
- **Quiz: sinnvolle Attribute und Methoden?**
  - Koordinaten x_i, Dimension n, zugrundeliegender Raum
  - Lesemethoden für Attribute, Norm, Addition mit anderem Vektor, Skalierung, ...
  - In Python: realisiert bspw. in NumPy
- **Polynome** $p(x) = \sum_{i=0}^{n} a_i x^i$
- **Quiz: sinnvolle Attribute und Methoden?**
  - Dimension n, Koeffizienten a_i, zugrundeliegender Raum
  - Auswertung $p(x)$, Addition, ...

Beispiele möglicher Klassen aus der Mathematik

- **Vektoren** $x = (x_1, \ldots, x_n)^\top$
- **Quiz: sinnvolle Attribute und Methoden?**
  - Koordinaten x_i, Dimension n, zugrundeliegender Raum
  - Lesemethoden für Attribute, Norm, Addition mit anderem Vektor, Skalierung, ...
  - In Python: realisiert bspw. in NumPy
- **Polynome** $p(x) = \sum_{i=0}^{n} a_i x^i$
- **Quiz: sinnvolle Attribute und Methoden?**
  - Dimension n, Koeffizienten a_i, zugrundeliegender Raum
  - Auswertung $p(x)$, Addition, ...
  - In Python: realisiert bspw. in SymPy

Beispiele möglicher Klassen aus der Mathematik

- **Vektoren** $x = (x_1, \ldots, x_n)^\top$
- **Quiz: sinnvolle Attribute und Methoden?**
  - Koordinaten x_i, Dimension n, zugrundeliegender Raum
  - Lesemethoden für Attribute, Norm, Addition mit anderem Vektor, Skalierung, ...
  - In Python: realisiert bspw. in NumPy
- **Polynome** $p(x) = \sum_{i=0}^{n} a_i x^i$
- **Quiz: sinnvolle Attribute und Methoden?**
  - Dimension n, Koeffizienten a_i, zugrundeliegender Raum
  - Auswertung $p(x)$, Addition, ...
  - In Python: realisiert bspw. in SymPy
- Weitere Beispiele: **Brüche, Plots und Grafiken, ...**

Abschließende Überlegungen

Abschließende Überlegungen

- Blöde Frage: warum ist in Python bereits etwas Profanes wie eine Zahl eine Instanz einer Klasse?

Abschließende Überlegungen

- Blöde Frage: warum ist in Python bereits etwas Profanes wie eine Zahl eine Instanz einer Klasse?
- Antwort fast schon klar: alles sind Klassen, deshalb existiert Schwung an Methoden, die Objekte mit Instanzen anderer Klassen „kombinieren“

Abschließende Überlegungen

- Blöde Frage: warum ist in Python bereits etwas Profanes wie eine Zahl eine Instanz einer Klasse?
- Antwort fast schon klar: alles sind Klassen, deshalb existiert Schwung an Methoden, die Objekte mit Instanzen anderer Klassen „kombinieren“
- Nur deshalb möglich: „irgendetwas in `print()`“ stopfen

Abschließende Überlegungen

- Blöde Frage: warum ist in Python bereits etwas Profanes wie eine Zahl eine Instanz einer Klasse?
- Antwort fast schon klar: alles sind Klassen, deshalb existiert Schwung an Methoden, die Objekte mit Instanzen anderer Klassen „kombinieren“
- Nur deshalb möglich: „irgendetwas in `print()`“ stopfen
- Unter der Haube: jede Klasse stellt Methode bereit, um eine ihrer Instanzen in eine (sinnvolle) Zeichenkette transformiert

Abschließende Überlegungen

- Blöde Frage: warum ist in Python bereits etwas Profanes wie eine Zahl eine Instanz einer Klasse?
- Antwort fast schon klar: alles sind Klassen, deshalb existiert Schwung an Methoden, die Objekte mit Instanzen anderer Klassen „kombinieren“
- Nur deshalb möglich: „irgendetwas in `print()`“ stopfen
- Unter der Haube: jede Klasse stellt Methode bereit, um eine ihrer Instanzen in eine (sinnvolle) Zeichenkette transformiert
  - Für Listen: eckige Klammern, Kommata als Trennzeichen

Impressum, Danksagung und Quellen

Stiftung Innovation in der Hochschullehre

Gefördert durch die Stiftung Innovation in der Hochschullehre im Rahmen des Projekts digit@L, https://stiftung-hochschullehre.de

SimTech

Gefördert mit Mitteln der Deutschen Forschungsgemeinschaft (EXC 2075 - 390740016) im Rahmen der Exzellenzstrategie

---

Autor: Dominik Göddeke, IANS, Universität Stuttgart

ZOERR

Veröffentlicht auf dem Zentralen OER Repositorium Baden-Württemberg, https://www.zoerr.de