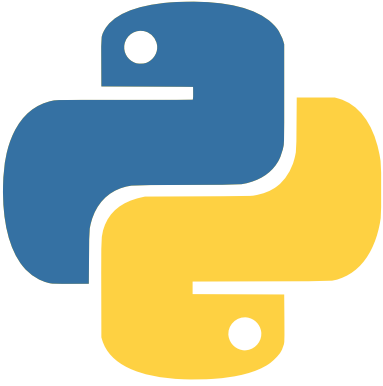
Universität Stuttgart
Projekt digit@L – BOOST. SKILLS. SUPPORT.



Programmierkurs Python

Vererbung

Dominik Göddeke

IANS SimTech

Stiftung Innovation in der Hochschullehre

Vererbung

Vererbung

- **Vererbung (Inheritance)**: Klassen von anderen Klassen abgeleitet

Vererbung

- **Vererbung (Inheritance)**: Klassen von anderen Klassen abgeleitet
- Erlaubt **hierarchische Modellierung**

Vererbung

- **Vererbung (Inheritance)**: Klassen von anderen Klassen abgeleitet
- Erlaubt **hierarchische Modellierung**
- Im Fahrzeugbeispiel

Vererbung

- **Vererbung (Inheritance)**: Klassen von anderen Klassen abgeleitet
- Erlaubt **hierarchische Modellierung**
- Im Fahrzeugbeispiel
  - Nie Instanz einer Klasse `Vehicle`

Vererbung

- **Vererbung (Inheritance)**: Klassen von anderen Klassen abgeleitet
- Erlaubt **hierarchische Modellierung**
- Im Fahrzeugbeispiel
  - Nie Instanz einer Klasse `Vehicle`
  - Dort Attribute und Methoden, die allen konkreteren Klassen gemeinsam sind

Vererbung

- **Vererbung (Inheritance)**: Klassen von anderen Klassen abgeleitet
- Erlaubt **hierarchische Modellierung**
- Im Fahrzeugbeispiel
  - Nie Instanz einer Klasse `Vehicle`
  - Dort Attribute und Methoden, die allen konkreteren Klassen gemeinsam sind
  - `num_seats`, `color`, `has_navi`

Vererbung

- **Vererbung (Inheritance)**: Klassen von anderen Klassen abgeleitet
- Erlaubt **hierarchische Modellierung**
- Im Fahrzeugbeispiel
  - Nie Instanz einer Klasse `Vehicle`
  - Dort Attribute und Methoden, die allen konkreteren Klassen gemeinsam sind
  - `num_seats`, `color`, `has_navi`
  - `get_color()`, `calc_average_consumption()`

Vererbung

- **Vererbung (Inheritance)**: Klassen von anderen Klassen abgeleitet
- Erlaubt **hierarchische Modellierung**
- Im Fahrzeugbeispiel
  - Nie Instanz einer Klasse `Vehicle`
  - Dort Attribute und Methoden, die allen konkreteren Klassen gemeinsam sind
  - `num_seats`, `color`, `has_navi`
  - `get_color()`, `calc_average_consumption()`
- In allen abgeleiteten Klassen: Attribute und Methoden der (abstrakten) Basisklasse verfügbar

Vererbung

- **Vererbung (Inheritance)**: Klassen von anderen Klassen abgeleitet
- Erlaubt **hierarchische Modellierung**
- Im Fahrzeugbeispiel
  - Nie Instanz einer Klasse `Vehicle`
  - Dort Attribute und Methoden, die allen konkreteren Klassen gemeinsam sind
  - `num_seats`, `color`, `has_navi`
  - `get_color()`, `calc_average_consumption()`
- In allen abgeleiteten Klassen: Attribute und Methoden der (abstrakten) Basisklasse verfügbar
- Wichtig bei privaten Attributen in der Basisklasse: **einfacher Unterstrich**

Vererbung

- Hierarchische Modellierung: direkt übertragbar auf Programmierung

Vererbung

- Hierarchische Modellierung: direkt übertragbar auf Programmierung
- Erhöhung der Lesbarkeit, Wartbarkeit und Erweiterbarkeit

Vererbung

- Hierarchische Modellierung: direkt übertragbar auf Programmierung
- Erhöhung der Lesbarkeit, Wartbarkeit und Erweiterbarkeit
- Durch Vermeidung von Redundanz und gute Strukturierung

Vererbung

- Hierarchische Modellierung: direkt übertragbar auf Programmierung
- Erhöhung der Lesbarkeit, Wartbarkeit und Erweiterbarkeit
- Durch Vermeidung von Redundanz und gute Strukturierung
- Großer Vorteil in **Kombination mit Überladung**

Vererbung

- Hierarchische Modellierung: direkt übertragbar auf Programmierung
- Erhöhung der Lesbarkeit, Wartbarkeit und Erweiterbarkeit
- Durch Vermeidung von Redundanz und gute Strukturierung
- Großer Vorteil in **Kombination mit Überladung**
  - Methoden behalten Signatur, verhalten sich aber spezialisiert

Vererbung

- Hierarchische Modellierung: direkt übertragbar auf Programmierung
- Erhöhung der Lesbarkeit, Wartbarkeit und Erweiterbarkeit
- Durch Vermeidung von Redundanz und gute Strukturierung
- Großer Vorteil in **Kombination mit Überladung**
  - Methoden behalten Signatur, verhalten sich aber spezialisiert
  - Gemeinsamer Code verbleibt in Basisklasse

Vererbung

- Hierarchische Modellierung: direkt übertragbar auf Programmierung
- Erhöhung der Lesbarkeit, Wartbarkeit und Erweiterbarkeit
- Durch Vermeidung von Redundanz und gute Strukturierung
- Großer Vorteil in **Kombination mit Überladung**
  - Methoden behalten Signatur, verhalten sich aber spezialisiert
  - Gemeinsamer Code verbleibt in Basisklasse
  - Beispiel Fahrzeug: Sportwagen berücksichtigt „Testosteronmodus an/aus“, Familienkutsche eher nicht

Beispiel: Polygone

Beispiel: Polygone

- Beispiel aus der schulmathematischen Geometrie

Beispiel: Polygone

- Beispiel aus der schulmathematischen Geometrie
- **Basisklasse** für beliebiges Polygon in 2D

Beispiel: Polygone

- Beispiel aus der schulmathematischen Geometrie
- **Basisklasse** für beliebiges Polygon in 2D
- Beschrieben durch geordnete Liste der Knoten beschreiben, in Form eines privaten Attributs

Beispiel: Polygone

- Beispiel aus der schulmathematischen Geometrie
- **Basisklasse** für beliebiges Polygon in 2D
- Beschrieben durch geordnete Liste der Knoten beschreiben, in Form eines privaten Attributs
- Fokus auf Vererbung, also Polygon nach der Erzeugung statisch, und Linienzug zur „implizit“ durch Start- und Endpunkte

Beispiel: Polygone

- Beispiel aus der schulmathematischen Geometrie
- **Basisklasse** für beliebiges Polygon in 2D
- Beschrieben durch geordnete Liste der Knoten beschreiben, in Form eines privaten Attributs
- Fokus auf Vererbung, also Polygon nach der Erzeugung statisch, und Linienzug zur „implizit“ durch Start- und Endpunkte
- Zwei Methoden: Stringifizierung und Flächenberechnung

So geht das im Code

Beispiel: Polygone

Beispiel: Polygone

- **Spezialisierung** der Klasse `Polygon` zu einem Dreieck

Beispiel: Polygone

- **Spezialisierung** der Klasse `Polygon` zu einem Dreieck
- **Syntax**: Angabe der Basisklasse in runden Klammern bei der Klassendefinition

Beispiel: Polygone

- **Spezialisierung** der Klasse `Polygon` zu einem Dreieck
- **Syntax**: Angabe der Basisklasse in runden Klammern bei der Klassendefinition
- Wichtig: nur geänderte Methoden müssen überladen werden

Beispiel: Polygone

- **Spezialisierung** der Klasse `Polygon` zu einem Dreieck
- **Syntax**: Angabe der Basisklasse in runden Klammern bei der Klassendefinition
- Wichtig: nur geänderte Methoden müssen überladen werden
- Befehl `super()`: übernimmt im Konstruktor sämtliche Attribute und Methoden aus der Basisklasse

So geht das im Code

Fortgeschrittenes Beispiel im Notebook
Exceptions als Klassenhierarchie

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