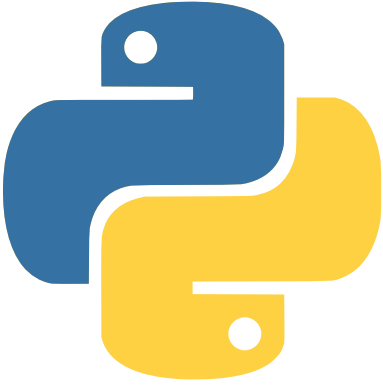
Universität Stuttgart
Projekt digit@L – BOOST. SKILLS. SUPPORT.



Programmierkurs Python

Klassen und Objekte in Python

Dominik Göddeke

IANS SimTech

Stiftung Innovation in der Hochschullehre

Klassen und Objekte in Python

Klassen und Objekte in Python

- **Ziel dieses Abschnitts**: Konzepte von OOP als Python-Sprachkonstrukte

Klassen und Objekte in Python

- **Ziel dieses Abschnitts**: Konzepte von OOP als Python-Sprachkonstrukte
- Wenig verblüffend: Definition von Klassen mit `class name_of_class: block`

Klassen und Objekte in Python

- **Ziel dieses Abschnitts**: Konzepte von OOP als Python-Sprachkonstrukte
- Wenig verblüffend: Definition von Klassen mit `class name_of_class: block`
- Erinnerung: übliche Einrückungsregel für Blöcke

Klassen und Objekte in Python

- **Ziel dieses Abschnitts**: Konzepte von OOP als Python-Sprachkonstrukte
- Wenig verblüffend: Definition von Klassen mit `class name_of_class: block`
- Erinnerung: übliche Einrückungsregel für Blöcke
- Trick, nicht nur bei OOP: `pass` macht leeren Block ausführbar

Klassen und Objekte in Python

- **Ziel dieses Abschnitts**: Konzepte von OOP als Python-Sprachkonstrukte
- Wenig verblüffend: Definition von Klassen mit `class name_of_class: block`
- Erinnerung: übliche Einrückungsregel für Blöcke
- Trick, nicht nur bei OOP: `pass` macht leeren Block ausführbar
- Instanzen der Klasse, also Objekte: Name der Klasse, gefolgt von leeren Klammern

Klassen und Objekte in Python

- **Ziel dieses Abschnitts**: Konzepte von OOP als Python-Sprachkonstrukte
- Wenig verblüffend: Definition von Klassen mit `class name_of_class: block`
- Erinnerung: übliche Einrückungsregel für Blöcke
- Trick, nicht nur bei OOP: `pass` macht leeren Block ausführbar
- Instanzen der Klasse, also Objekte: Name der Klasse, gefolgt von leeren Klammern
- Zuweisung in Variable nicht vergessen

Klassen und Objekte in Python

- **Ziel dieses Abschnitts**: Konzepte von OOP als Python-Sprachkonstrukte
- Wenig verblüffend: Definition von Klassen mit `class name_of_class: block`
- Erinnerung: übliche Einrückungsregel für Blöcke
- Trick, nicht nur bei OOP: `pass` macht leeren Block ausführbar
- Instanzen der Klasse, also Objekte: Name der Klasse, gefolgt von leeren Klammern
- Zuweisung in Variable nicht vergessen
- Kennen wir schon: `my_list = list()`

Klassen und Objekte in Python

- **Ziel dieses Abschnitts**: Konzepte von OOP als Python-Sprachkonstrukte
- Wenig verblüffend: Definition von Klassen mit `class name_of_class: block`
- Erinnerung: übliche Einrückungsregel für Blöcke
- Trick, nicht nur bei OOP: `pass` macht leeren Block ausführbar
- Instanzen der Klasse, also Objekte: Name der Klasse, gefolgt von leeren Klammern
- Zuweisung in Variable nicht vergessen
- Kennen wir schon: `my_list = list()`
- **Konvention**: CamelCase-Namen, `Car`, `AnotherClass`, `MySuperCoolClass`

Codebeispiele

Attribute einer Klasse

Attribute einer Klasse

- Leere Klasse nicht nützlich

Attribute einer Klasse

- Leere Klasse nicht nützlich
- Deshalb: **Hinzufügen von Attributen**

Attribute einer Klasse

- Leere Klasse nicht nützlich
- Deshalb: **Hinzufügen von Attributen**
- Einfachste Möglichkeit: Einführung bei erster Verwendung

Attribute einer Klasse

- Leere Klasse nicht nützlich
- Deshalb: **Hinzufügen von Attributen**
- Einfachste Möglichkeit: Einführung bei erster Verwendung
- Interne Speicherung als `dict`

Attribute einer Klasse

- Leere Klasse nicht nützlich
- Deshalb: **Hinzufügen von Attributen**
- Einfachste Möglichkeit: Einführung bei erster Verwendung
- Interne Speicherung als `dict`
- `__dict__` erlaubt direkten Zugriff, nicht sinnvoll für Kapselung

Attribute einer Klasse

- Leere Klasse nicht nützlich
- Deshalb: **Hinzufügen von Attributen**
- Einfachste Möglichkeit: Einführung bei erster Verwendung
- Interne Speicherung als `dict`
- `__dict__` erlaubt direkten Zugriff, nicht sinnvoll für Kapselung
- Klasse so nur „Hülle“ um Dictionary

Attribute einer Klasse

- Leere Klasse nicht nützlich
- Deshalb: **Hinzufügen von Attributen**
- Einfachste Möglichkeit: Einführung bei erster Verwendung
- Interne Speicherung als `dict`
- `__dict__` erlaubt direkten Zugriff, nicht sinnvoll für Kapselung
- Klasse so nur „Hülle“ um Dictionary
- Aber: inkrementelle Konstruktion von Klassen, deshalb temporäre Einschränkung

Codebeispiele

Methoden einer Klasse

Methoden einer Klasse

- Klassen (besser Objekte) dienen nicht nur der Speicherung von Daten

Methoden einer Klasse

- Klassen (besser Objekte) dienen nicht nur der Speicherung von Daten
- Bieten (fast immer) **Methoden zur Manipulation** von Daten (Attributen)

Methoden einer Klasse

- Klassen (besser Objekte) dienen nicht nur der Speicherung von Daten
- Bieten (fast immer) **Methoden zur Manipulation** von Daten (Attributen)
- Kernmechanismus für das Ziel der **Kapselung**

Methoden einer Klasse

- Klassen (besser Objekte) dienen nicht nur der Speicherung von Daten
- Bieten (fast immer) **Methoden zur Manipulation** von Daten (Attributen)
- Kernmechanismus für das Ziel der **Kapselung**
- Definition im **Block der Klasse**

Methoden einer Klasse

- Klassen (besser Objekte) dienen nicht nur der Speicherung von Daten
- Bieten (fast immer) **Methoden zur Manipulation** von Daten (Attributen)
- Kernmechanismus für das Ziel der **Kapselung**
- Definition im **Block der Klasse**
- Aufruf einer Methode der Instanz `obj` durch `obj.method()`

Methoden einer Klasse

- Klassen (besser Objekte) dienen nicht nur der Speicherung von Daten
- Bieten (fast immer) **Methoden zur Manipulation** von Daten (Attributen)
- Kernmechanismus für das Ziel der **Kapselung**
- Definition im **Block der Klasse**
- Aufruf einer Methode der Instanz `obj` durch `obj.method()`
  - Kennen wir schon, bspw. `my_list.append(x)`

Methoden einer Klasse

- Klassen (besser Objekte) dienen nicht nur der Speicherung von Daten
- Bieten (fast immer) **Methoden zur Manipulation** von Daten (Attributen)
- Kernmechanismus für das Ziel der **Kapselung**
- Definition im **Block der Klasse**
- Aufruf einer Methode der Instanz `obj` durch `obj.method()`
  - Kennen wir schon, bspw. `my_list.append(x)`
- Definition von Methoden direkt in der Definition der Klasse

Methoden einer Klasse

- Klassen (besser Objekte) dienen nicht nur der Speicherung von Daten
- Bieten (fast immer) **Methoden zur Manipulation** von Daten (Attributen)
- Kernmechanismus für das Ziel der **Kapselung**
- Definition im **Block der Klasse**
- Aufruf einer Methode der Instanz `obj` durch `obj.method()`
  - Kennen wir schon, bspw. `my_list.append(x)`
- Definition von Methoden direkt in der Definition der Klasse
  - Erstes Argument **immer** das eigene Objekt, bezeichnet als `self`

Methoden einer Klasse

- Klassen (besser Objekte) dienen nicht nur der Speicherung von Daten
- Bieten (fast immer) **Methoden zur Manipulation** von Daten (Attributen)
- Kernmechanismus für das Ziel der **Kapselung**
- Definition im **Block der Klasse**
- Aufruf einer Methode der Instanz `obj` durch `obj.method()`
  - Kennen wir schon, bspw. `my_list.append(x)`
- Definition von Methoden direkt in der Definition der Klasse
  - Erstes Argument **immer** das eigene Objekt, bezeichnet als `self`
- Aufruf von Klassenmethoden wie bekannt

Methoden einer Klasse

- Klassen (besser Objekte) dienen nicht nur der Speicherung von Daten
- Bieten (fast immer) **Methoden zur Manipulation** von Daten (Attributen)
- Kernmechanismus für das Ziel der **Kapselung**
- Definition im **Block der Klasse**
- Aufruf einer Methode der Instanz `obj` durch `obj.method()`
  - Kennen wir schon, bspw. `my_list.append(x)`
- Definition von Methoden direkt in der Definition der Klasse
  - Erstes Argument **immer** das eigene Objekt, bezeichnet als `self`
- Aufruf von Klassenmethoden wie bekannt
- `OBJEKT.METHODE(ARGUMENTE)`: Abkürzung für `KLASSE.METHODE(OBJEKT,ARGUMENTE)`, klar woher `self` stammt

Methoden einer Klasse

- Bekannte Konzepte der Definition von Funktionen übertragbar

Methoden einer Klasse

- Bekannte Konzepte der Definition von Funktionen übertragbar
  - Beliebige Anzahl an Argumente, benannte Argumente, Keyword-Argumente, Rekursion usw.

Methoden einer Klasse

- Bekannte Konzepte der Definition von Funktionen übertragbar
  - Beliebige Anzahl an Argumente, benannte Argumente, Keyword-Argumente, Rekursion usw.
- Aufruf einer Methode einer Klasse durch Methode derselben Klasse: explizit `self.my_other_method()`

Codebeispiele

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