Unterrichtsentwurf für die Lehrübung

Vor- und Nachname Lorenz Bung		
Schulanschrift (mit Telefonnummer) Walther-Rathenau-Gewerbeschule, Friedrichstr. 51, 79098 Freiburg. 0761/201-7942 Schulleiter/-in Renate Storm		
Mentor/-in Leonie Feldbusch	Ausbilder/-in Jochen Pogrzeba	
Datum 08.05.2025	Uhrzeit 13:45–14:30	
Klasse und Schulart E2FI3 – Berufsschule: Fachinformatiker, 2. Lehrjahr		Raum 025
Fach Software- und Anwendungsentwicklung (SAE)		

Thema des Unterrichts

Objektorientierte Programmierung: Vererbung (Python)

1. Überblick und zentrales Anliegen

Thema	Objektorientierte Vererbung: Umsetzung des Konzepts der Vererbung in einer objektorientierten Programmiersprache (Python)
Lehrplanbezug	Lernfeld 8: **Daten systemübergreifend bereitstellen** "Die Schülerinnen und Schüler ermitteln für einen Kundenauftrag Datenquellen und **analysieren** diese hinsichtlich ihrer Struktur [...]." "Sie **entwickeln** Konzepte zur Bereitstellung der gewählten Datenquellen für die weitere Verarbeitung […]." "Die Schülerinnen und Schüler **implementieren** [...] ihr Konzept mit vorhandenen sowie dazu passenden Entwicklungswerkzeugen und Produkten."
Zentrales Anliegen	Die SuS können das Konzept der Vererbung mithilfe der objektorientierten Programmiersprache Python umsetzen und konkrete Beispiele implementieren. Weiterhin können die SuS mit UML-Diagrammen modellierte Sachverhalte, in welchen Vererbung zum Einsatz kommt, in Python-Code überführen.
Lehr-Lernarrangement	Die Vorstellung der Hausaufgabe durch die SuS dient als wiederholender Einstieg, mit dem in der letzten Stunde fehlende SuS auf einen arbeitsfähigen Stand gebracht werden sollen. Die anschließende Erarbeitungsphase erfolgt lehrerzentriert; es wird die Umsetzung der Vererbung in Python anhand eines Merkblatts erklärt, welches anschließend per Moodle zur Verfügung gestellt wird. In der Übungsphase soll zunächst in Einzelarbeit das in der vergangenen Stunde gemeinsam entwickelte UML-Diagramm mit Python umgesetzt werden. Anschließend soll in Partnerarbeit die Richtung einer bestehenden Vererbungsrelation umgekehrt und über deren Sinnhaftigkeit diskutiert. Die Ergebnisse der Arbeitsphase werden zum Schluss in der Sicherungsphase im Plenum vorgestellt und diskutiert. Als Puffer steht eine Zusatzaufgabe bereit, in welcher das bereits angesprochene Problem der Mehrfachvererbung in Python umgesetzt werden soll.

2. Unterrichtsverlaufsplan

Phase	Unterrichtsstruktur (mit Zeitplanung)	Lehrerhandeln	Schülerhandeln	Lernziele (fachliche und überfachliche)	Medien
Unterrichts-einstieg (13:45–13:53)	Wiederholung anhand Hausaufgabe 8 min.	Begrüßung der SuS Moderation von Meldungen Ergänzung der Beiträge	Präsentation der HA Fragen stellen Lösung ergänzen	1. TZ: Die SuS *nennen* zentrale Begriffe der Vererbung und *erläutern* UML-Diagramme unter Verwendung dieser Begriffe. (AFB I)	Digitale Tafel Schülerlösung Hausaufgabe
Erarbeitung I (13:53–14:00)	Lehrervortrag, darbietend Erklärung Vererbung in Python 7 min.	"Vorprogrammieren" Erklären des Codes Beantwortung von Fragen	Zuhören Fragen stellen	Syntax kennenlernen	Lehrer-PC
Übung I (14:00–14:15)	Einzelarbeit: Aufgabe 1 (Gartencenter) 15 min.	Freischalten des ABs in Moodle Beantwortung von Fragen Unterstützung von SuS, welche gefehlt haben technische Hilfestellung	Bearbeitung von Aufgabe 1+2 Verständnisfragen stellen nach Beendigung der Aufgabe: Upload der eigenen Lösung in Moodle	2. TZ: Die SuS *implementieren* das Konzept der Vererbung in Python. (AFB II)	Arbeitsblatt "OOP (Python)", Aufgabe 1
Übung II (14:15–14:20)	Partnerarbeit: Aufgabe 2 (Richtung der Vererbung) 5 min.			3. TZ: Die SuS *bewerten* die vorgegebene Vererbungsrelation hinsichtlich logischer Sinnhaftigkeit. (AFB III)	Arbeitsblatt, Aufgabe 2
Sicherung (14:20–14:30)	Besprechung Aufgabe 1+2 im Plenum 10 min.	Moderation von Meldungen Ergänzung der vorgestellten Lösung Beantwortung von Fragen	Präsentation der eigenen Lösung Ergänzung der vorgestellten Lösung Verständnisfragen stellen Diskussion im Plenum	Konsolidierung Stärkung der Präsentations- und Sozialkompetenz	Digitale Tafel Schülerlösung Arbeitsblatt "OOP"
Maximal-planung: Übung	Bonusaufgabe (Mehrfach-vererbung) 10 min.	Beantwortung von Fragen technische Hilfestellung	Bearbeitung der Bonus-aufgabe Verständnisfragen stellen	Die SuS *implementieren* eine Situation, in der Mehrfachvererbung zum Einsatz kommt und *analysieren* diese hinsichtlich möglicher Probleme. (AFB II-III)	Arbeitsblatt, Bonus-aufgabe

(Hinweise zur Ergebnissicherung werden in den Spalten Lehrer- bzw. Schülerhandeln eingetragen)

Anhang

3. Quellenverzeichnis

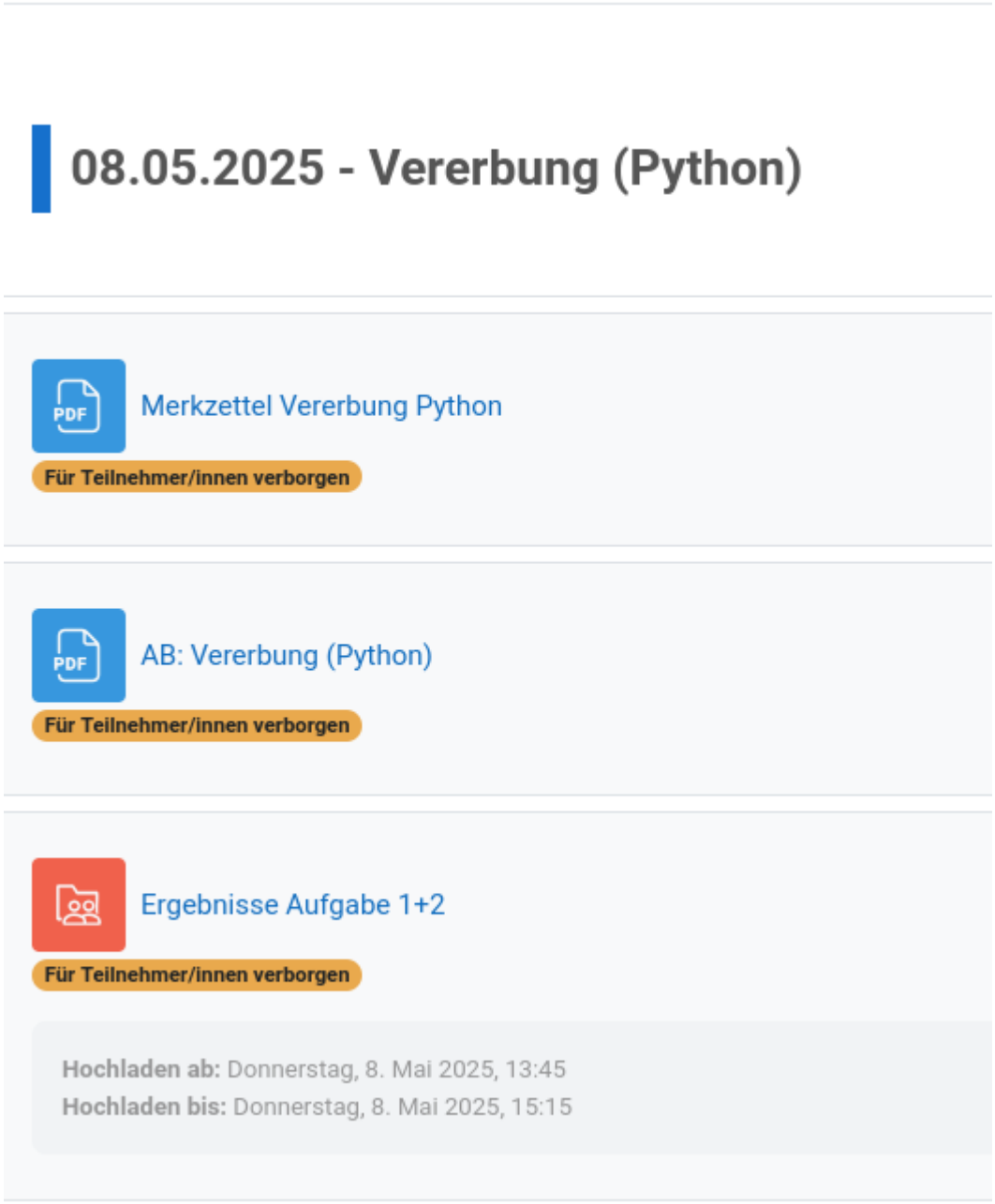
Ministerium für Kultus, Jugend und Sport Baden-Württemberg (2019). *Bildungsplan für die Berufsschule: Fachinformatiker und Fachinformatikerin*. Zentrum für Schulqualität und Lehrerbildung (ZSL). https://bildungsplaene-bw.de/site/bildungsplan/get/documents_E201170933/lsbw/Bildungsplaene-BERS/MediaCenter/bs/bs_berufsbez/BS_Technische%20IT%20Berufe_2019-12-13.pdf (abgerufen am 04.05.2025)

Klaus Becker (2024). *Fachkonzept Vererbung*. Inf-Schule. https://inf-schule.de/oop/python/bank/vererbung/konzept_vererbung (abgerufen am 04.05.2025)

4. Weitere Materialien

- Screenshot der Struktur des Moodle-Kurses (1 Seite)
- Arbeitsblatt "OOP: Vererbung (Python)" (2 Seiten)
- Musterlösung Arbeitsblatt "OOP: Vererbung (Python)" (2 Seiten)
- Merkzettel "Vererbung in Python" (1 Seite)

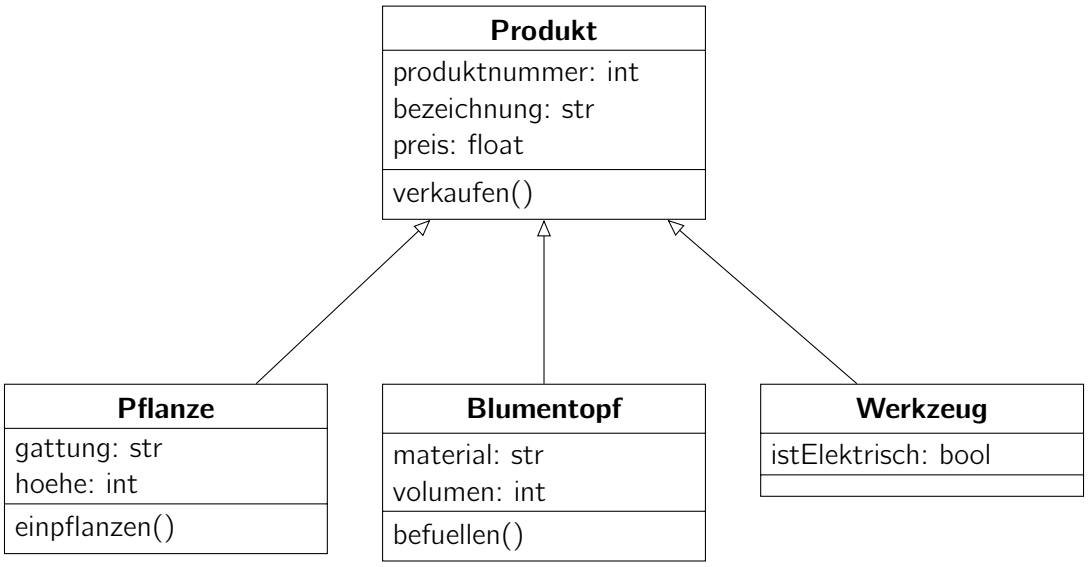
Screenshot der Struktur des Moodle-Kurses



SAE L. Bung	**OOP: Vererbung (Python)**	Klasse: Datum:

Aufgabe 1: Gartencenter-Produktverwaltung in Python

Nachdem wir in der letzten Stunde gemeinsam ein Softwarekonzept für die Produktverwaltung des Gartencenters entworfen haben, soll dieses nun implementiert werden.



Setzen Sie das obenstehende UML-Diagramm in Python um. Bilden Sie insbesondere die Vererbungsstrukturen korrekt ab.

Aufgabe 2: Richtung der Vererbung

Betrachten Sie den folgenden Code.

```
1 class Laptop:
2   def leistung(self):
3     return "Alltagstauglich"
4 class GamingLaptop(Laptop):
5   def leistung(self):
6     return "Sehr leistungsstark"
```

a) Verändern Sie den Code, sodass die Richtung der Vererbung umgedreht wird.

b) Diskutieren Sie, welche Variante mehr Sinn ergibt.

⊕ Bonusaufgabe 3: Mehrfachvererbung

Am Ende der letzten Stunde haben wir kurz über die Probleme gesprochen, die bei mehrfacher Vererbung entstehen können (Beispiel: `Wasserflugzeug` als Subklasse von `Flugzeug` und `Boot`).

a) Recherchieren Sie, wie in Python mit Mehrfachvererbung umgegangen wird.

b) Setzen Sie das oben beschriebene Beispiel in Python um und halten Sie Ihre Beobachtungen fest.

SAE	**OOP: Vererbung (Python)**	Klasse:
L. Bung	**Musterlösung**	Datum:

Aufgabe 1: Gartencenter-Produktverwaltung in Python

```python
class Produkt():
    def __init__(self, produktnummer, bezeichnung, preis):
        self.produktnummer = produktnummer
        self.bezeichnung = bezeichnung
        self.preis = preis
    def verkaufen(self):
        print(f"Das Produkt mit der Nummer {self.produktnummer} wurde für {self.preis} verkauft.")

class Pflanze(Produkt):
    def __init__(self, produktnummer, bezeichnung, preis, gattung, hoehe):
        super().__init__(produktnummer, bezeichnung, preis)
        self.gattung = gattung
        self.hoehe = hoehe
    def einpflanzen(self):
        print(f"Die Pflanze mit der Produktnummer {self.produktnummer} wurde eingepflanzt.")

class Blumentopf(Produkt):
    def __init__(self, produktnummer, bezeichnung, preis, material, volumen):
        super().__init__(produktnummer, bezeichnung, preis)
        self.material = material
        self.volumen = volumen
    def befuellen(self):
        print(f"Der Topf mit der Nummer {self.produktnummer} wurde mit {self.volumen} Liter befüllt.")

class Werkzeug(Produkt):
    def __init__(self, produktnummer, bezeichnung, preis, istElektrisch):
        super().__init__(produktnummer, bezeichnung, preis)
        self.istElektrisch = istElektrisch
```

Aufgabe 2: Richtung der Vererbung

a) Verändern Sie den Code, sodass die Richtung der Vererbung umgedreht wird.

```
1 class GamingLaptop:
2   def leistung(self):
3     return "Sehr leistungsstark"
4 class Laptop(GamingLaptop):
5   def leistung(self):
6     return "Alltagstauglich"
```

b) Diskutieren Sie, welche Variante mehr Sinn ergibt.

Die Variante, in der `GamingLaptop` von `Laptop` erbt, ergibt mehr Sinn: Ein Gaming-Laptop ist eine spezialisierte Variante eines "normalen" Laptops und nicht umgekehrt. Es könnte beispielsweise auch noch eine zweite Subklasse `ArbeitsLaptop` geben.

Aus logischer Sicht ergibt das Codebeispiel aus der Aufgabe so wie es dasteht daher keinen Sinn.

SAE	**Merkzettel**	Klasse:
L. Bung	**Vererbung in Python**	Datum:

```
1  class Fahrzeug():
2    def __init__(self, geschwindigkeit):
3      self.geschwindigkeit = geschwindigkeit
4    def fahren(self):
5      print("Das Fahrzeug fährt.")
6  class Auto(Fahrzeug):
7    def __init__(self, geschwindigkeit, farbe):
8      super().__init__(geschwindigkeit)
9      self.farbe = farbe
10   def hupen(self):
11     print("Das Auto hupt.")
12 auto1 = Auto(100, "rot")
13 auto1.hupen()   #Ausgabe: Das Auto hupt.
14 auto1.fahren()  #Ausgabe: Das Fahrzeug fährt.
```

Vererbung in Python

Um eine Vererbungsrelation in Python anzugeben, wird der Name der Superklasse bei der Definition der Subklasse in Klammern geschrieben. Im obigen Beispiel erbt also die Klasse `Auto` von der Klasse `Fahrzeug`.
Objekte der Subklasse können auf Attribute und Methoden zugreifen, die in der Superklasse definiert wurden. Beispielsweise kann `auto1.fahren()` aufgerufen werden, obwohl die Methode `fahren()` in der Klasse `Fahrzeug` definiert wurde.

Aufruf von Methoden der Superklasse

Mit `super()` können Methoden der Superklasse aufgerufen werden. Dies ist insbesondere bei den Konstruktoren (`__init__()`-Methoden) hilfreich, um doppelten Code zu sparen.
In Zeile 8 müsste sonst das Attribut `geschwindigkeit` manuell gesetzt werden. Bei vielen Parametern wäre das sehr aufwändig.