# Python面向对象 教案

周次：3

课次：2

章节内容：继承与多态

课时：2

## 单元教学目标

- 知识目标：能够解释继承的概念并写出基本语法，如class Child(Parent)   
- 技能目标：能够编写继承的类并调用父类和子类的方法，通过实例验证继承关系   
- 素养目标：能够团队协作完成项目，并遵守编程规范，体现职业素养与协作能力

## 教学重点

• 继承的语法结构：掌握类的继承语法（class A(B)）及子类如何继承父类的方法和属性，理解\_\_init\_\_方法的调用顺序   
• 方法的覆盖与重写：掌握子类如何通过重写父类方法实现差异化行为，理解\_\_override\_\_方法的使用场景   
• 多态的概念与应用：理解动态绑定机制（如子类对象调用父类方法时的多态表现），掌握通过子类实例调用父类方法的典型示例

## 教学难点

• 学生可能难以理解继承的层次结构，尤其是父类与子类方法的调用关系。例如，如何通过`super()`函数调用父类方法，或者在子类中重写父类方法时如何保持原有行为。部分学生可能混淆继承的顺序（如父类→子类）或忽略继承链的层级关系，导致代码逻辑错误。   
  
• 多态概念的抽象性较强，学生可能难以区分“方法重写”与“方法覆盖”的区别，以及如何通过实例类型动态调用不同实现的函数。例如，当使用`int`类型变量调用`str`类方法时，学生可能无法理解这种动态绑定的底层机制，或误以为所有类的同一方法必须重写。

## 教学活动

新课导入【5分钟】   
教师通过一个生活案例（如动物分类）引出继承与多态的概念，用动画演示动物类的继承关系（猫、狗、鸟），激发学生兴趣。结合提问：“如果我要创建一个会飞的动物类，如何实现？”引导学生思考，激活已有知识。  
  
讲授新课【20分钟】   
1. \*\*讲授概念\*\*（5分钟）   
 - 教师用PPT讲解继承的定义（子类继承父类属性方法）和多态的含义（同一方法不同实现）。   
 - 通过代码示例展示：`class Animal: def speak(self): print("动物发声")`，并解释继承语法 `class Dog(Animal)`。   
  
2. \*\*案例分析\*\*（7分钟）   
 - 提供“水果类”示例（水果、苹果、香蕉），让学生分组讨论如何用继承实现属性共享。   
 - 教师引导学生编写代码：`class Fruit: def info(self): print("水果")`，`class Apple(Fruit)：def info(self): print("苹果")`。   
  
3. \*\*互动练习\*\*（6分钟）   
 - 学生尝试编写“交通工具”类（汽车、自行车），使用继承实现属性共享。   
 - 教师巡视指导，重点讲解如何通过`super()`调用父类方法。   
  
4. \*\*多态实践\*\*（2分钟）   
 - 用`class Animal: def speak(self): print("动物发声")`，`class Dog(Animal): def speak(self): print("汪!")`，演示多态应用。   
 - 学生分组用不同动物类实现`speak()`，验证多态性。   
  
巩固练习【10分钟】   
1. \*\*基础练习\*\*（4分钟）   
 - 学生编写“水果类”代码，实现属性共享（如`fruit\_color`），教师讲解继承语法。   
  
2. \*\*进阶练习\*\*（6分钟）   
 - 设计“交通工具”类，要求学生用继承实现不同交通工具的`move()`方法（如`car.move()`, `bike.move()`）。   
 - 学生分组讨论如何用`super()`调用父类方法，教师点评优化方案。   
  
归纳总结【5分钟】   
教师通过提问引导总结：   
- 继承如何实现属性共享？   
- 多态在Python中如何体现？   
- 学生用头脑风暴法分享学到的关键词（如“父类”“子类”“多态”）。   
- 教师板书核心要点，强调继承的“is-a”关系和多态的“can-do”特性。

## 教学资源

• 教学设备：投影仪、白板、编程环境（如Python 3解释器）、计算机终端（用于学生实践操作）   
• 工具：Python代码编辑器（如VS Code、PyCharm）、在线代码运行平台（如Replit）、示例代码库（如GitHub上的Python教程项目）   
  
• 推荐参考资料：   
1. Python官方文档（https://docs.python.org/3/）   
2. 《Python编程：从入门到实践》（书籍）   
3. Real Python（https://realpython.com）   
4. LeetCode（https://leetcode.com）：练习继承与多态的编程题   
5. 网站：Codecademy Python课程（https://www.codecademy.com/learn/Python）   
  
• 实践资源：   
- 示例代码：   
 ```python   
 class Animal:   
 def speak(self):   
 pass   
  
 class Dog(Animal):   
 def speak(self):   
 print("Woof!")   
  
 class Cat(Animal):   
 def speak(self):   
 print("Meow!")   
 ```   
- 练习题：   
 - 实现一个`Shape`类，继承自`Object`，包含`area()`方法，子类`Circle`和`Rectangle`覆盖该方法   
 - 用多态演示不同动物的叫声（如`Dog`和`Cat`的`speak()`方法）   
  
• 教学辅助材料：   
- 交互式教学平台：Python Tutor（https://pythontutor.com）   
- 课件模板：包含流程图（继承关系）和代码示例的PPT或PDF   
- 课堂活动：分组编写动物类，实现继承与多态的实践操作

## 教学反思

• 教学效果方面：学生对继承的层次结构理解存在偏差，部分学生混淆父类与子类的方法覆盖逻辑；多态应用案例中，部分学生未能正确区分运行时绑定与编译时绑定的差异   
• 学生反馈方面：课堂练习中出现"继承后父类方法被覆盖但子类未重写"的错误操作，反映对继承机制的掌握存在薄弱环节；部分学生对多态的"运行时多态性"概念理解模糊，误认为多态仅适用于方法重载   
• 教学设计优化：需增加抽象类与接口的对比说明，补充多态应用场景的典型案例（如图形绘制系统）；建议通过可视化工具展示继承层级结构，强化学生对类层次关系的认知   
• 教学节奏调整：对基础薄弱学生进行分层教学，通过代码调试任务巩固继承的"方法覆盖"概念；针对高阶需求学生设计多态应用场景的编程任务，提升实践能力   
• 技术细节补充：需明确区分"方法重写"与"方法覆盖"的异同，补充多态实现的底层原理（如虚函数表机制）；建议结合实际项目案例（如游戏中的角色系统）深化理解   
• 互动环节改进：增加小组协作任务，要求学生设计继承树结构并实现多态应用，通过协作学习强化概念记忆；使用代码审查工具检测学生作业中的继承/多态错误   
• 师生互动优化：设置即时反馈环节，通过代码示例展示继承错误的调试过程，帮助学生理解运行时绑定的动态特性；建立学习日志记录学生对多态概念的掌握程度

## 教学评价

• 过程性评价   
• 课堂参与度（10分）：观察学生在课堂讨论、小组协作中的发言频率、主动性，记录参与行为。   
• 作业完成情况（10分）：检查学生是否按时提交继承与多态相关练习题，是否准确理解概念并完成代码。   
• 小组合作表现（10分）：评估小组成员分工是否明确、协作是否有效，能否解决代码中的问题或完成项目任务。   
• 课堂提问回答（10分）：记录学生在教师提问中的逻辑清晰度、准确性及对核心概念（如多态、继承规则）的掌握程度。   
  
• 结果性评价   
• 课堂测试（20分）：设计选择题、填空题和简答题，考查学生对继承语法（如`class A(B)`）、多态定义（如`def method(self): ...`）、方法重写（如`def method(self): ...`）的理解。   
• 项目展示（20分）：要求学生用继承与多态完成一个小型项目（如动物类继承、图形绘制），展示代码逻辑并解释实现思路。   
• 代码提交质量（20分）：评估学生提交的代码是否符合规范（如缩进、注释），能否正确应用继承与多态原理，是否存在错误需修正。   
• 课堂表现评分（10分）：综合学生在课堂讨论、作业、项目中的综合表现，给予最终评分。   
  
• 评价标准与方式   
• 评分维度：过程性评价（40分） + 结果性评价（60分）   
• 评分细则：   
 - 过程性评价：按条目满分10分累加，总分30分   
 - 结果性评价：按项目/测试/课堂表现满分20分累加，总分60分   
• 评价方式：课堂观察+作业批改+项目展示+测试题+教师评分   
• 附加说明：鼓励学生通过代码注释、小组讨论等方式主动巩固知识，教师可提供反馈建议。