# 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：能够区分继承与多态的概念并正确应用，掌握类之间关系的建立方式。   
- 技能目标：能够编写继承的类并实现多态，解决继承中的方法冲突，完成简单继承与多态的编程任务。   
- 素养目标：培养良好的编程规范和团队协作能力，通过实际项目练习提升代码质量与团队协作效率。

## 教学重点

• 继承的语法结构：掌握类的定义语法（class）和继承语法（class A(B)），理解父类与子类的关系，明确子类如何继承父类的属性和方法。   
• 多态的实现方式：理解方法重写（overriding）的概念，掌握如何通过子类覆盖父类方法，并通过实例调用实现多态行为。   
• 类的初始化与方法调用：掌握如何通过\_\_init\_\_方法初始化对象，理解实例方法的调用逻辑，以及如何通过继承实现方法的动态绑定。

## 教学难点

• 继承的机制与super()的使用：学生可能难以理解父类与子类之间的关系（如方法覆盖、属性继承），以及如何正确使用`super()`函数调用父类方法，尤其是当存在多重继承时的调用顺序问题。   
• 多态的实现与动态绑定：学生可能混淆方法重写（overriding）与方法覆盖（overriding）的概念，或难以理解如何通过实例化不同类的对象调用不同的实现方法（如`int`类型与`str`类型的`\_\_add\_\_()`方法）。

## 教学活动

新课导入【5分钟】：   
教师用故事导入，讲述“动物分类”案例，提出问题：“如何让不同动物共享相同的行为？例如，狗和猫都能跑，但动作不同。”通过提问引导学生思考，激发兴趣。结合实物展示（如狗、猫图片），运用\*\*故事导入\*\*和\*\*提问法\*\*，学生分组讨论后回答，教师总结“继承”概念。   
  
讲授新课【15分钟】：   
1. \*\*讲解继承\*\*（5分钟）：   
 - 用“动物类”示例，讲解类与对象关系，强调继承的定义（“子类继承父类属性和方法”）。   
 - 通过代码示例（如`class Dog(Pet):`）演示，教师板书关键点（如`super()`用法）。   
 - \*\*讲授法\*\*结合\*\*案例分析法\*\*，学生分析`Dog`和`Pet`的代码差异。   
  
2. \*\*多态讲解\*\*（5分钟）：   
 - 用“动作”案例，展示`run()`方法在`Dog`和`Cat`中的不同实现。   
 - 教师演示`Animal`类定义`run()`方法，学生编写`Dog`和`Cat`子类并调用。   
 - \*\*讨论法\*\*：学生分组讨论“多态为何重要”，教师总结“同一方法不同实现”体现抽象与具体。   
  
3. \*\*互动练习\*\*（5分钟）：   
 - 教师提出问题：“如果`Dog`和`Cat`都继承`Animal`，如何调用`run()`？”   
 - 学生分组编写代码，教师巡视指导，强调\*\*代码调试\*\*和\*\*问题解决\*\*。   
 - \*\*练习法\*\*结合\*\*游戏法\*\*（如“代码拼图”游戏，学生用碎片拼出完整类结构）。   
  
巩固练习【10分钟】：   
1. \*\*基础练习\*\*（5分钟）：   
 - 学生独立完成“动物类”代码，要求继承`Animal`并实现`run()`方法。   
 - 教师抽查学生代码，纠正错误，强调\*\*语法规范\*\*。   
  
2. \*\*拓展练习\*\*（5分钟）：   
 - 学生添加`eat()`方法到`Dog`类，要求`Cat`类重写该方法。   
 - 分组竞赛：用不同颜色标记代码，完成时间最快组获胜，激发\*\*竞争意识\*\*。   
  
归纳总结【5分钟】：   
- 教师提问：“继承的核心是什么？多态如何体现？”   
- 学生举手回答，教师汇总要点（如“继承：类间关系；多态：同一方法不同实现”）。   
- \*\*头脑风暴法\*\*：学生自由发言，教师记录关键词（如“抽象”“具体”“动态”）。   
  
教学方法融合：\*\*故事导入+讲授+案例分析+讨论+练习+游戏\*\*，确保学生参与度与知识掌握。

## 教学资源

• 教学设备和工具   
 • 投影仪/白板用于展示示例代码和图表   
 • 计算机（配备Python开发环境，如PyCharm、Jupyter Notebook或VS Code）   
 • 交互式白板（可选，用于实时演示代码运行）   
 • Python解释器（如Python 3.x）   
 • 示例代码编写工具（如Codecademy、Replit或在线IDE）   
  
• 推荐的参考资料或网站   
 • 书籍：   
 • 《Python编程：从入门到实践》（推荐基础教程）   
 • 《Python核心编程》（深入语法与实践）   
 • 在线资源：   
 • Python官方文档（https://docs.python.org/3/）   
 • Codecademy Python课程（https://www.codecademy.com/）   
 • Coursera《Python for Everybody》（https://www.coursera.org/）   
 • 练习平台：   
 • LeetCode（练习编程题，涉及继承与多态）   
 • HackerRank（实践面向对象编程）   
 • GitHub（开源项目示例，如Animal类继承示例）   
 • 实战项目：   
 • 创建“动物”类（继承自基类Animal，实现多态方法如`make\_sound()`）   
 • 使用Python装饰器或抽象类（如`abc`模块）实现抽象方法   
  
• 其他补充资源   
 • 面向对象编程视频教程（如YouTube频道“Traversy Media”）   
 • 交互式编程练习网站（如Exercism）   
 • 课堂练习题库（如“Python for Beginners”练习题）

## 教学反思

• 教学效果方面：需关注学生是否能准确理解继承的层级关系与方法调用顺序，以及多态中不同类的同一方法如何实现差异。部分学生可能对super()的使用存在混淆，或在多继承场景下出现方法冲突问题。   
• 学生反馈方面：部分学生反映继承概念抽象，难以建立类与类之间的层次关系；多态应用案例中，学生可能对实际场景（如不同形状的面积计算）的理解存在困难，或对动态绑定机制的原理不清晰。   
• 改进建议：   
• 优化教学方法：通过可视化图示展示继承层级结构，结合代码示例演示super()的使用场景，分步骤讲解多继承中的方法冲突问题。   
• 增强实践环节：设计阶梯式练习题，从单继承到多继承逐步推进，结合实际案例（如图形绘制）强化多态应用。   
• 深化反馈机制：在课堂上预留时间解答学生疑问，通过即时反馈工具（如在线测验）检测学生对继承与多态的理解程度，针对性调整教学内容。   
• 强化概念关联：将继承与类的构造函数、方法覆盖等知识结合讲解，帮助学生建立完整知识体系，提升多态应用的实践能力。

## 教学评价

• 过程性评价   
• 课堂参与：观察学生在讨论、提问、代码编写中的主动性和表达能力，记录课堂表现（优秀/良好/需改进）   
• 作业完成情况：检查学生是否按时提交作业，是否正确应用继承和多态概念（优秀/良好/需改进）   
• 小组合作：评估团队成员分工是否明确、沟通是否有效，是否存在协作障碍（优秀/良好/需改进）   
  
• 结果性评价   
• 小测验：通过选择题和简答题检验学生对继承和多态的核心概念理解（满分100分，优秀≥90分，良好≥75分）   
• 项目作业：学生需编写一个包含继承和多态的类（如动物类与哺乳动物类），提交代码并说明设计思路（优秀：代码规范、逻辑清晰，良好：基本正确，需改进：存在逻辑错误或未完全应用概念）   
  
• 评价标准与方式   
• 过程性评价：采用观察记录表、课堂反馈问卷、作业批改评分（满分100分）   
• 结果性评价：小测验（40分）+ 项目作业（60分）总分100分，按等级划分（优秀/良好/需改进）   
  
• 评分细则   
• 课堂参与：50分（优秀/良好/需改进）   
• 作业完成：40分（优秀/良好/需改进）   
• 小组合作：10分（优秀/良好/需改进）   
• 小测验：30分（优秀/良好/需改进）   
• 项目作业：40分（优秀/良好/需改进）   
  
• 附加说明   
• 评价周期：课时结束后24小时内完成，结果反馈教师并用于教学调整   
• 评价工具：课堂观察表、作业批改记录、项目代码提交文档