Python面向对象 教案

|  |  |  |  |
| --- | --- | --- | --- |
| **周次** | **课次** | **章节内容** | **课时** |
| 3 | 2 | 继承与多态 | 2 |
| **单元教学目标** | | | |
| - 知识目标：能够描述继承的层次结构并举例说明，掌握super()函数的使用场景，理解多态的定义及实现方式。  - 技能目标：能够编写继承的类，正确使用super()函数，并实现方法重写，能够通过多态实现接口方法的差异行为。  - 素养目标：能够小组合作完成项目，遵守编码规范，展示良好的沟通能力与问题解决能力。 | | | |
| **教学重点** | | **教学难点** | |
| • 继承的基本概念与语法：掌握类的继承关系（如class Child(Parent)），理解父类属性和方法的继承机制，以及子类如何扩展父类功能。  • 多态的实现方式：理解方法重写（override）和动态绑定（dynamic binding）的概念，掌握子类如何通过继承调用父类方法并实现差异化行为。  • 子类与父类方法的调用关系：明确子类如何继承父类方法，理解方法调用的优先级（如覆盖父类方法时的动态绑定规则）。 | | • 学生可能难以理解如何在子类中正确调用父类的方法，尤其是在重写方法时，容易混淆是否需要使用`super()`函数或直接调用父类方法。  • 多态的实现方式（如通过方法重写）可能让学生困惑，尤其是如何在不同类中实现同一方法的差异化行为，以及如何通过继承传递多态特性。 | |
| **教学活动** | | | |
| 新课导入【10分钟】  1. \*\*故事导入\*\*：教师讲述“动物分类”故事，引入父类Animal（有方法say()）和子类Dog（重写say()）、Cat（重写say()），引导学生思考继承与多态在现实中的应用。  2. \*\*提问互动\*\*：提问“如果Dog和Cat都叫‘汪汪’和‘喵喵’，如何让程序识别不同动物的叫声？”激发学生兴趣。   讲授新课【30分钟】  1. \*\*讲授法\*\*：   - 解释继承概念：父类定义共性，子类扩展功能（例：Animal类定义say()方法，Dog类重写为“Woof”）。   - 重点讲解多态：同一方法在不同子类中表现不同（例：列表中添加Dog和Cat对象，调用say()输出不同结果）。  2. \*\*案例分析法\*\*：   - 分析“交通工具”案例：父类Vehicle（有方法move()），子类Car（重写为“Driving”）、Bicycle（重写为“Biking”）。   - 学生分组讨论：如何用继承实现交通工具的统一接口。  3. \*\*讨论法\*\*：   - 讨论“继承的局限性”：如类属性无法共享，需显式传递参数。   - 课堂问答：如何避免父类方法被子类覆盖？（答案：重写或使用抽象类）。  4. \*\*练习法\*\*：   - 学生独立编写继承类，完成Dog和Cat的say()方法，教师巡视指导。   巩固练习【15分钟】  1. \*\*分层练习\*\*：   - \*\*基础题\*\*：编写Animal类，子类Dog和Cat重写say()方法。   - \*\*进阶题\*\*：创建列表混合Dog和Cat对象，调用say()方法并输出结果。  2. \*\*小组合作\*\*：   - 学生分组完成练习，教师巡视并点评，重点检查是否理解多态原理。  3. \*\*角色扮演\*\*：   - 学生扮演开发者，设计一个“食物分类”系统，使用继承实现不同食物的共性（如eat()）和特性（如cooked()）。   归纳总结【10分钟】  1. \*\*头脑风暴\*\*：   - 学生总结继承的规则（如方法重写、属性继承）、多态的实现方式（如动态绑定）。   - 教师补充：抽象类、接口的概念，强调继承的层级性。  2. \*\*课堂问答\*\*：   - 问题1：继承与多态的关系？   - 问题2：如何避免父类方法被子类覆盖？（答案：重写或使用抽象类）。  3. \*\*学生自述\*\*：   - 学生用一句话概括本节课重点，教师总结并强调多态在实际编程中的价值。   教学方法：讲授法、案例分析、讨论法、练习法、角色扮演。 | | | |
| **教学资源** | | | |
| • 教学设备：投影仪、白板、电脑（用于演示代码）、Python编程环境（如PyCharm/VS Code）、打印设备（用于打印练习题或示例代码）  • 推荐参考资料：   - 书籍：《Python编程：从入门到实践》（经典入门教材）   - 在线资源：Python官方文档（https://docs.python.org/zh-cn/3/）   - 网站：GeeksforGeeks（Python教程与实例）   - 练习平台：Exercism（实践编程） | LeetCode（巩固语法）  • 教学方法：   - 分步骤讲解继承（类、方法、属性继承）与多态（方法重写、动态绑定）   - 通过实际案例（如动物类继承、图形绘制）演示抽象概念   - 引导学生编写代码并调试，结合小组讨论加深理解  • 其他资源：   - 项目实践：用继承实现“动物-鸟类-鸟类子类”结构，展示多态应用   - 代码示例：   ```python   class Animal:   def speak(self):   pass    class Dog(Animal):   def speak(self):   print("Woof!")    class Cat(Animal):   def speak(self):   print("Meow!")    dog = Dog()   cat = Cat()   dog.speak() # 输出: Woof!   cat.speak() # 输出: Meow!   ```   - 互动工具：使用Replit或Jupyter Notebook进行实时代码运行与可视化演示 | | | |
| **教学反思** | | | |
| • 教学效果：学生是否能准确区分继承与多态的概念，能否正确使用super()函数，是否掌握类方法与实例方法的区别  • 学生反馈：课堂互动是否有效（如提问、小组讨论），是否有学生因抽象概念（如多态）产生困惑，是否对实际应用场景（如图形绘制、数据处理）理解不足  • 教学方法：是否通过案例演示（如Animal类继承Dog类）帮助学生理解继承关系，是否用多态示例（如不同动物的run()方法）说明多态特性  • 技术难点：是否因抽象概念（如多态的实现机制）导致学生理解困难，是否需要补充图示或代码注释辅助解释  • 课堂参与度：学生是否主动提问或参与讨论，是否在练习环节（如编写Animal类）中表现出学习积极性  • 评估方式：是否通过小测验或编程任务检验学生对继承与多态的理解，是否发现普遍性错误（如混淆父类与子类方法）  • 教学资源：是否提供代码示例（如Dog类、Animal类）帮助学生理解，是否因资源不足导致部分学生理解偏差  • 实践应用：是否引导学生将继承与多态应用于实际项目（如模拟动物行为），是否因缺乏项目引导导致学习兴趣下降  • 分层教学：是否针对不同水平学生（如基础薄弱者 vs 程序员）调整教学难度，是否需要补充额外辅导材料  • 课堂时间：是否因内容过快导致学生消化困难，是否需要调整讲解节奏或增加复习环节 | | | |
| **教学评价** | | | |
| • 过程性评价  • 课堂参与度（30%）：观察学生在课堂讨论、代码编写、小组协作中的主动性和表达能力，记录参与频率与质量。  • 作业完成情况（40%）：评估学生是否独立完成继承与多态的代码任务，检查代码逻辑、注释及是否理解抽象类/接口概念。  • 课堂笔记质量（20%）：通过学生整理的笔记内容判断其对继承规则（如继承顺序、方法覆盖）、多态实现（如方法重写）的理解深度。   • 结果性评价  • 课堂测试（30%）：设计简答题（如“解释继承与多态的区别”）和编程题（如“实现Animal类的继承与多态示例”），检验知识掌握程度。  • 项目展示（40%）：要求学生完成一个包含继承（如Dog类继承Animal类）和多态（如Animal接口定义方法）的程序，并展示代码逻辑、运行结果及设计思路。  • 作业评分（30%）：根据代码规范性、功能完整性、注释清晰度等维度进行评分，重点考察是否正确应用继承与多态原理。   • 评价标准  • 过程性评价：参与度（30%）、作业质量（40%）、笔记深度（30%）  • 结果性评价：测试成绩（30%）、项目展示（40%）  • 评分细则：   - 课堂参与度：1-3分（完全参与）/ 2-1分（部分参与）/ 1-0分（无参与）   - 作业质量：1-5分（完全正确）/ 4-2分（部分正确）/ 1-0分（错误）   - 项目展示：1-5分（完全正确）/ 4-2分（部分正确）/ 1-0分（错误）   • 评价方式  • 过程性：课堂观察记录、作业提交、小组互评  • 结果性：纸质测试卷、代码提交文件、项目演示视频 | | | |