Python网络编程 教案

|  |  |  |  |
| --- | --- | --- | --- |
| **周次** | **课次** | **章节内容** | **课时** |
| 6 | 2 | HTTP请求处理 | 2 |
| **单元教学目标** | | | |
| - 知识目标：能够区分HTTP请求方法（GET/POST）的用途及区别，掌握HTTP状态码的常见值（如200、404、500）  - 技能目标：能够使用Python requests库发送HTTP GET和POST请求，并解析返回的JSON数据，完成简单的网页爬取或表单提交任务  - 素养目标：养成规范的代码书写习惯，能通过调试工具排查HTTP请求中的错误，具备基础的网络调试与问题分析能力 | | | |
| **教学重点** | | **教学难点** | |
| • 使用requests库发送HTTP请求（GET/POST方法）  • 解析HTTP响应数据（状态码、响应头、内容）  • 处理网络异常（如超时、连接错误） | | • 学生可能难以掌握如何正确构造HTTP请求参数（如url、data、json等）并确保其与服务器预期的格式匹配，例如未正确设置headers或参数顺序导致请求失败。  • 如何解析HTTP响应数据（如提取状态码、响应内容或JSON数据）时，需处理可能的异常（如网络错误、解析失败）并理解响应对象的属性（如response.status\_code、response.text）。 | |
| **教学活动** | | | |
| 新课导入【5分钟】  教师通过播放一段网络爬虫抓取网页数据的短视频，引出HTTP请求的现实应用场景。提问：“若要从网页中提取数据，需向服务器发送什么类型的请求？”学生回答后，教师总结HTTP请求是网络通信的基础，激发学习兴趣。   讲授新课【20分钟】  1. \*\*讲授法\*\*（5分钟）：讲解HTTP请求的组成（请求行、请求头、请求体），结合GET/POST示例说明请求方法。  2. \*\*案例分析法\*\*（7分钟）：分组分析“爬取知乎主页”的代码片段，讨论如何通过GET请求获取数据，并指出参数编码规则。  3. \*\*讨论法\*\*（5分钟）：学生分组讨论“为何需区分GET和POST请求”，教师引导归纳区别（数据暴露 vs 数据封装）。  4. \*\*角色扮演法\*\*（2分钟）：模拟发送HTTP请求，学生扮演客户端和服务器，体验请求过程。   巩固练习【10分钟】  1. \*\*基础练习\*\*（4分钟）：学生独立编写代码，使用requests库发送GET请求获取网页数据，并记录响应内容。  2. \*\*进阶练习\*\*（6分钟）：分组完成“发送POST请求提交表单数据”的任务，要求处理JSON参数并解析响应，小组展示成果。   归纳总结【5分钟】  教师总结HTTP请求的关键点：请求方法、请求头、请求体的结构，强调安全实践（如使用HTTPS）。学生通过头脑风暴列举常见HTTP方法（如DELETE、HEAD），教师补充注意事项（如跨域问题）。 | | | |
| **教学资源** | | | |
| • 教学设备：   - 投影仪/白板（用于演示）   - 多台电脑（需安装Python环境）   - 网络环境（确保可访问HTTP服务示例）   - 本地服务器（如Python Flask或Django）   - 调试工具：Postman、curl、Wireshark（可选）   • 推荐参考资料或网站：   1. 书籍：《Python网络编程》（作者：李云峰）   2. 官方文档：[Python官方文档](https://docs.python.org/3/library/requests.html)   3. 在线资源：   - MDN Web Docs：[HTTP请求](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Requests)   - GitHub开源项目：[requests](https://github.com/requests/requests)   4. 实践平台：   - LeetCode（Python网络编程练习题）   - HackerRank（HTTP请求相关挑战）   5. 工具推荐：   - `requests`库（基础HTTP请求）   - `BeautifulSoup`（解析HTML响应）   - `Scrapy`（高级网络爬虫示例）   - `WebSocket`库（支持实时通信）   • 其他建议：   - 提供HTTP请求示例代码（如GET/POST请求）   - 引入实际案例（如爬取网页数据、发送表单）   - 强调安全实践（如使用HTTPS、验证证书） | | | |
| **教学反思** | | | |
| • 教学效果方面：学生是否能够准确区分GET和POST请求的差异？是否掌握请求头（headers）和请求体（body）的使用场景？实践环节是否有效巩固了HTTP协议的各组成部分（如方法、状态码、响应码）？  • 学生反馈方面：部分学生反映对请求头的语法结构（如`headers={...}`）理解困难，或对实际应用（如发送表单数据）缺乏直观体验；是否因时间限制导致实践任务深度不足？  • 教学设计优化：是否需要增加更多可视化示例（如用工具模拟HTTP请求）或分组项目（如开发简易Web爬虫）以提升参与感？  • 知识点覆盖深度：是否遗漏了HTTPS协议的基础知识（如TLS加密）或跨域请求（CORS）的简要介绍？  • 差异化教学需求：是否对不同水平学生（如初学者与进阶者）的实践任务设计不够兼顾？  • 工具辅助效果：是否通过PyTest或Jupyter Notebook等工具增强代码调试体验，提升学习效率？  • 课堂节奏把控：是否因讲解过快导致学生消化困难，或因实践时间不足影响学习效果？  • 延伸学习资源：是否提供可运行的示例代码或配套文档，方便学生课后进一步探索？  • 互动性提升：是否通过问答、代码调试比赛等形式增强学生对HTTP请求的主动理解？  • 技术前沿关联：是否在讲解中提及HTTP/2或REST API设计原则，拓展学生技术视野？ | | | |
| **教学评价** | | | |
| • 过程性评价  • 课堂参与度：观察学生是否积极回答问题、主动提问或讨论，记录参与情况（如发言次数、互动频率）  • 实践操作能力：评估学生是否能正确使用Python的`requests`库发送HTTP请求，处理响应数据（如状态码、内容解析）  • 问题解决能力：记录学生在调试错误（如网络异常、语法错误）时的思考过程及解决策略  • 学习态度：通过学生完成任务的主动性、按时性及合作意愿进行评估（如是否主动完成课后练习）   • 结果性评价  • 作业完成情况：检查学生是否独立完成HTTP请求处理的代码任务（如发送GET/POST请求并输出结果）  • 代码质量：评价代码的结构清晰度、语法正确性、注释完整性及是否符合最佳实践（如使用`try-except`捕获异常）  • 知识应用能力：通过学生能否将HTTP请求知识应用于实际场景（如模拟网页爬虫、发送表单数据）进行验证  • 学习成果：通过学生提交的代码或项目成果（如HTTP请求示例程序）评估其对HTTP协议的理解深度（如区分GET/POST、处理JSON响应）   • 评价方式  • 过程性：课堂观察记录、学生自评/同伴互评、实践操作日志  • 结果性：作业评分、代码评审、项目展示/演示、课堂测试（如HTTP请求功能验证）   • 评价标准  • 过程性：参与度（30%）、操作能力（30%）、问题解决（20%）、态度（20%）  • 结果性：作业质量（30%）、代码规范（20%）、知识应用（20%）、学习成果（10%） | | | |