# Python网络编程 教案

周次：6

课次：2

章节内容：HTTP请求处理

课时：2

## 单元教学目标

- 知识目标：能够描述HTTP请求的各个组成部分，包括请求方法、请求行、请求头和请求体   
- 技能目标：能够使用Python发送HTTP请求并获取响应数据，包括发送GET/POST请求及解析响应内容   
- 素养目标：能够遵守网络伦理规范，确保请求符合法律法规，避免恶意请求行为

## 教学重点

• 使用requests库发送HTTP请求并解析响应数据   
• 区分GET和POST请求的差异及对应数据格式   
• 掌握HTTP响应状态码的含义及常见错误处理机制

## 教学难点

• 学生可能难以掌握HTTP请求的构造方法，尤其是GET和POST请求的参数传递方式差异。GET请求通过URL参数传递数据（如`?key=value`），而POST请求需通过请求体（body）传递数据，需正确使用`requests`库的`params`和`data`参数，并注意编码格式（如URL编码与JSON编码的区别）。   
  
• 如何正确设置请求头（如`User-Agent`）并理解其作用是难点。学生可能误以为请求头与请求体无关，或不清楚如何通过`requests`库的`headers`参数灵活配置，导致请求被服务器拒绝或返回错误响应。

## 教学活动

新课导入【5分钟】   
教师用故事导入：讲述外卖平台用户反馈系统案例，提出问题：“如何用Python模拟用户下单的HTTP请求？”引导学生思考网络编程在实际生活中的应用场景，激发学习兴趣。通过提问引发学生讨论，激活原有认知。   
  
讲授新课【15分钟】   
1. \*\*讲授法\*\*（5分钟）：讲解HTTP请求的基本结构（GET/POST）、请求头（Host、Content-Type）、响应码（200/404）及响应体（JSON格式）。   
2. \*\*案例分析法\*\*（5分钟）：用Python requests库示例演示发送GET请求，分析代码逻辑，强调参数传递与响应解析。   
3. \*\*讨论法\*\*（3分钟）：分组讨论“如何处理跨域请求”问题，教师引导学生思考CORS机制与Python代码的关联。   
4. \*\*角色扮演法\*\*（2分钟）：学生分角色扮演开发者（需求分析）与测试员（响应验证），模拟实际开发场景。   
  
巩固练习【10分钟】   
1. \*\*基础练习\*\*（3分钟）：编写Python脚本发送GET请求获取网页数据，要求输出响应内容。   
2. \*\*进阶练习\*\*（4分钟）：用POST请求提交表单数据，解析JSON响应并验证成功与否。   
3. \*\*小组合作\*\*（3分钟）：学生分组完成任务，教师巡视指导，确保不同水平学生参与。   
  
归纳总结【5分钟】   
教师总结HTTP请求的核心要点：请求方法、参数传递、响应结构，强调代码调试与异常处理的重要性。学生用5分钟时间复述课堂重点，教师补充关键术语（如HTTP状态码、CORS），布置课后作业：编写一个简易的HTTP请求程序并记录运行结果。

## 教学资源

</think>  
  
• \*\*教学设备\*\*：   
 - 计算机（用于运行Python环境）   
 - 展示设备（如投影仪或屏幕）   
 - Python开发环境（如PyCharm、VS Code、Jupyter Notebook）   
 - 网络连接设备（用于测试HTTP请求）   
 - 互联网接入（用于访问相关网站和文档）   
  
• \*\*推荐参考资料或网站\*\*：   
 - 《Python网络编程》（书籍）   
 - Python官方文档（https://docs.python.org/3/library/requests.html）   
 - W3Schools（https://www.w3schools.com/python/）   
 - MDN Web Docs（https://developer.mozilla.org/zh-CN/docs/Web）   
 - GitHub（https://github.com/PythonWeb）   
 - 网络请求工具：`requests` 库的官方文档和示例   
 - 互动式学习平台：Codecademy、Coursera、Udemy（Python网络编程课程）

## 教学反思

• 教学效果方面：学生是否能准确区分GET和POST请求的区别？是否掌握请求头、响应头的基本结构？实践环节是否达到预期目标？   
• 学生反馈方面：是否对HTTP协议的底层机制（如TCP/IP、SSL/TLS）产生兴趣？是否因理论抽象或实践难度导致理解困难？   
• 实践环节设计：是否提供足够清晰的代码示例？学生是否能独立完成HTTP请求的发送与解析？是否需要增加更多案例（如API调用、网页爬虫）？   
• 理论与实践结合：是否在讲解理论后立即安排实践任务？是否通过代码调试帮助学生理解HTTP响应的结构（如status\_code、headers、body）？   
• 分层次教学：是否针对不同水平的学生设计差异化练习（如基础练习与进阶项目）？是否对理解能力较弱的学生提供额外支持？   
• 技术细节准确性：是否正确讲解HTTP版本（如HTTP/1.1）的特性？是否强调安全机制（如HTTPS）的重要性？   
• 时间安排合理性：是否在2小时内完成所有教学内容？是否需要调整节奏（如增加案例讲解或预留时间答疑）？   
• 实际应用场景：是否结合真实案例（如调用公开API、分析网页数据）提升学生兴趣？是否引导学生思考HTTP在实际开发中的作用？

## 教学评价

• 过程性评价   
• 课堂参与度：通过观察学生在实验课中的发言频率、提问积极性及小组讨论表现，记录参与度评分（1-5分）   
• 作业与实验完成情况：评估学生是否按时提交实验报告，代码是否符合规范，是否存在错误或遗漏（1-5分）   
• 小组合作表现：观察小组分工是否合理，沟通是否顺畅，任务完成度是否达标（1-5分）   
• 学习反思：通过学生提交的反思日志或课堂总结，评估其对HTTP请求流程的理解程度及问题解决能力（1-5分）   
  
• 结果性评价   
• 项目成果：根据学生完成的HTTP请求处理项目（如爬虫、API调用等）的完整性、正确性及创新性进行评分（1-5分）   
• 测试成绩：通过单元测试或编程作业得分，评估学生对HTTP请求基础概念（如GET/POST、URL解析、响应码等）的掌握程度（1-5分）   
• 项目展示：观察学生在课堂展示中的表现，包括内容逻辑性、技术细节准确性及表达清晰度（1-5分）   
• 学习成果：通过问卷调查或小测验，评估学生是否能独立完成HTTP请求的实现，并理解其实际应用场景（1-5分）   
  
• 评价方式   
• 过程性评价：结合课堂观察、作业提交、实验记录等实时数据动态反馈   
• 结果性评价：采用项目评分表、测试成绩、展示评分及学习成果问卷综合判定