# 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：能够区分HTTP请求方法（GET/POST）并理解其应用场景，掌握请求头（Headers）和请求体（Body）的基本结构。   
- 技能目标：能够使用Python发送GET和POST类型的HTTP请求，并正确解析响应内容（如状态码、响应头、响应体）。   
- 素养目标：具备网络安全意识，能够识别并避免在HTTP请求中泄露敏感信息，遵守网络道德规范。

## 教学重点

• 掌握HTTP请求的基本方法（GET/POST）及应用场景   
• 理解参数传递方式（查询参数/表单数据/JSON数据）及编码规范   
• 熟悉请求头（headers）与响应数据（status\_code、json/text）的解析与处理

## 教学难点

• 学生可能难以掌握HTTP请求的构造方式，尤其是如何正确区分请求行、请求头和请求体的结构，例如：   
 - GET请求需在请求行中指定URL，而POST请求需在请求体中发送数据，学生可能混淆二者区别。   
  
• 处理HTTP响应的解析逻辑是难点，例如：   
 - 如何区分响应行（如`200 OK`）和响应体（如HTML内容），以及如何解析响应头（如`Content-Type`）以判断数据类型。   
 - 如何处理可能的错误状态码（如`404 Not Found`）并根据响应内容调整后续操作。

## 教学活动

新课导入【10分钟】   
1. 教师用故事导入：讲述“小明通过浏览器访问新闻网站”的场景，引出HTTP协议的必要性。   
2. 角色扮演活动：学生分组扮演“网络工程师”和“用户”，模拟HTTP请求流程，讨论如何通过GET/POST方式传递数据。   
3. 提问引导：提出“为什么网页能动态显示？背后是什么技术？”激发学生兴趣。   
  
讲授新课【25分钟】   
1. \*\*讲授法\*\*：讲解HTTP协议的基本结构（请求行、请求头、响应码），用示例代码演示GET请求（如`GET /index.html?name=Alice`）。   
2. \*\*案例分析法\*\*：分析“表单提交”案例，拆解请求行（`POST /submit HTTP/1.1`）和请求头（`Content-Type: application/x-www-form-urlencoded`）。   
3. \*\*讨论法\*\*：分组讨论“POST请求与GET请求的区别”，教师点评重点（如数据嵌入 vs. 隐藏在URL中）。   
4. \*\*演示法\*\*：用Python代码演示POST请求（`requests.post()`），并解释响应码200的含义。   
5. \*\*游戏法\*\*：设计“HTTP请求接龙”游戏，学生依次说出请求类型（如GET、POST、PUT），教师快速判断正确性。   
  
巩固练习【20分钟】   
1. \*\*基础练习\*\*：学生用Python代码编写GET请求，模拟访问示例网页（如`requests.get('https://example.com')`）。   
2. \*\*进阶练习\*\*：设计“文件上传”任务，学生用POST请求发送文件（示例：`requests.post('https://upload.example.com', files={'file': open('test.txt', 'rb')})`）。   
3. \*\*分层练习\*\*：   
 - \*\*基础层\*\*：完成简单GET请求代码编写。   
 - \*\*进阶层\*\*：解析HTTP响应码（如200、404）并解释其含义。   
 - \*\*挑战层\*\*：模拟跨域请求（如`requests.get('https://other-site.com')`），讨论CORS机制。   
4. \*\*小组竞赛\*\*：分组完成“网页登录”任务，用POST请求发送用户名和密码，教师点评优劣。   
  
归纳总结【10分钟】   
1. \*\*头脑风暴\*\*：学生列出“HTTP请求的关键要素”（如请求方法、URL、头字段、数据体）。   
2. \*\*角色扮演\*\*：学生扮演“网络工程师”，复述HTTP请求流程并解释其应用场景。   
3. \*\*教师总结\*\*：强调重点（如GET/POST的区别、请求头的用途、响应码的意义），用简明语言回顾核心知识点。   
4. \*\*即时反馈\*\*：教师提问“为什么需要HTTP协议？”学生回答后，教师总结HTTP作为“互联网的呼吸系统”概念。

## 教学资源

• 教学设备：   
 - 计算机实验室（需配备Python开发环境，如Python 3.x及以上版本）   
 - 服务器或虚拟机（如Ubuntu/Windows Server，用于演示HTTP请求处理）   
 - 本地网络测试工具（如curl、nmap、Wireshark）   
 - 桌面/笔记本电脑（用于课堂演示和实践操作）   
  
• 推荐参考资料或网站：   
 - 书籍：   
 - 《Python网络编程》（Simon Real）   
 - 《Pro Python网络编程》（Michael Dawson）   
 - 官方文档：   
 - Python官方文档（https://docs.python.org/3/library/）   
 - requests库文档（https://requests.readthedocs.io/）   
 - 在线资源：   
 - MDN Web Docs（https://developer.mozilla.org/zh-CN/docs/Web/HTTP）   
 - GitHub开源项目（如https://github.com/requests/requests）   
 - 课程平台：   
 - Coursera《Python for Everybody》   
 - Udemy《Python Network Programming》   
 - 实验工具：   
 - Flask/Django（用于搭建HTTP服务器）   
 - Postman（用于测试HTTP请求）   
 - 基础知识：   
 - HTTP协议基础（https://en.wikipedia.org/wiki/Hyper-text\_Transfer\_Protocol）   
 - TCP/IP协议详解（https://en.wikipedia.org/wiki/Transmission\_Control\_Protocol）

## 教学反思

• 教学效果方面：学生是否能准确区分GET和POST请求的差异？是否掌握HTTP响应码（如200、404、500）的含义？实践环节中是否能独立完成简单的HTTP请求发送与解析？   
• 学生反馈方面：部分学生对HTTP协议的底层原理（如TCP/IP、请求头解析）理解不足，导致操作失误；是否因时间限制导致实践任务完成度不足？   
• 教学设计优化：是否需要增加HTTP请求的调试工具（如curl、Postman）对比演示？是否需分层教学（基础语法+实战案例+进阶应用）？   
• 课堂互动效果：是否通过小组协作或实时问答提升学生对HTTP方法（如PUT、DELETE）的掌握？是否因内容抽象导致学生注意力分散？   
• 技术工具适配：是否使用可视化工具（如PyQt）辅助教学，帮助学生直观理解HTTP请求的结构？是否需补充网络编程基础（如IP地址、端口）的预习内容？   
• 实践任务难度：是否因任务过于简单（如简单发送请求）导致学生缺乏挑战性？是否需设计复杂场景（如动态网页解析、文件上传）提升学习深度？   
• 学习资源覆盖：是否提供代码示例、调试工具链接或在线练习平台？是否需补充HTTP协议的进阶知识（如HTTPS、缓存机制）？   
• 教学节奏把控：是否因讲解过快导致学生消化困难？是否需分段讲解（如先讲协议再讲工具）或增加复习环节？   
• 学生能力差异：是否针对不同水平学生设计差异化任务（如基础任务+进阶挑战）？是否需补充基础概念（如TCP/IP、HTTP/1.1）的预习内容？

## 教学评价

• 过程性评价   
• 课堂参与：观察学生在课堂讨论、代码编写、小组合作中的表现，记录积极发言、主动提问、协作态度等，评分标准为：积极参与（4分）/ 一般参与（2分）/ 无参与（0分）   
• 作业完成情况：检查学生是否按时提交代码作业，代码是否符合规范（如语法正确、注释完整），评分标准为：优秀（5分）/ 良好（3分）/ 一般（2分）/ 不合格（0分）   
• 课堂小测验：通过选择题或简答题检验学生对HTTP请求基础概念（如GET/POST、请求头、响应码）的理解，评分标准为：优秀（5分）/ 良好（3分）/ 一般（2分）/ 不合格（0分）   
• 代码质量：通过代码提交的格式、注释、逻辑清晰度等评估学生编程能力，评分标准为：优秀（5分）/ 良好（3分）/ 一般（2分）/ 不合格（0分）   
  
• 结果性评价   
• 项目作业：学生需独立完成一个HTTP请求程序（如发送GET/POST请求并解析响应），评分标准为：优秀（5分）/ 良好（3分）/ 一般（2分）/ 不合格（0分）   
• 课堂小测验：通过考试或测试题检验学生对HTTP请求核心知识点（如请求方法、响应码、异常处理）的掌握程度，评分标准为：优秀（5分）/ 良好（3分）/ 一般（2分）/ 不合格（0分）   
• 课堂表现综合评分：结合课堂参与、作业、项目表现等综合评定，满分10分，评分标准为：优秀（10分）/ 良好（8分）/ 一般（6分）/ 不合格（0分）   
  
• 评价方式   
• 形成性评价：通过课堂观察、作业批改、项目评审等方式持续记录学生进展   
• 总结性评价：通过项目作业和考试最终评估学生对HTTP请求处理的掌握程度   
• 评分权重：过程性评价占60%，结果性评价占40%