混合式教学下Python课程模块化任务教学实践

刘菊香，郭海如*，胡绪龙，万兴

湖北工程学院计算机与信息科学学院，孝感 432000

摘　要　针对高职院校Python程序设计课程学生逻辑迁移能力薄弱、团队协作流于形式的教学痛点，本研究构建融合模块化任务解构的混合式教学模式。该模式将复杂编程项目拆解为低耦合独立逻辑单元，课前依托线上资源完成语法预习与学情摸底；课中开展分层分组教学，专项小组攻克编程重难点后，项目小组依次完成模块接口对接与系统联调。以社交媒体舆情分析系统实训为载体，落实需求拆解、接口拟定、难点攻关、模块联调及Git代码冲突处理全教学流程。依托Git提交记录、代码圈复杂度、PEP8编码规范，搭建多维度过程性评价体系。教学实践表明，该模式可有效提升学生编程逻辑能力、代码工程质量与团队协作效率，操作简便，适合在高职各类程序设计课程中推广应用。

关键字：Python 程序设计；线上线下混合教学；模块化任务解构；接口契约联调；多元量化评价

Modular Task Teaching Practice of Python Course Under Blended Teaching Mode

Juxiang Liu, Hairu Guo, Xulong Hu, Xing Wang

School of Computer and Information Science

Hubei Engineering University

Xiaogan 432000, China

Corresponding author email: 58288239@qq.com; ljx@hbeu.edu.cn

Abstract—Aiming at the problems of students' weak logical transfer ability and superficial team collaboration in higher vocational colleges Python Programming courses, this work constructs a blended teaching mode integrated with modular task decomposition. Complex programming projects are divided into independent low-coupling logical units. Online resources are used for pre-class grammar preview and learning situation investigation. Hierarchical group teaching is adopted in class: special teams solve key and difficult programming points first, and project teams complete module interface docking and system joint debugging. Taking social media public opinion analysis system training as the carrier, this paper implements the complete teaching process including demand decomposition, interface formulation, difficulty tackling and Git code conflict resolution. A multi-dimensional process evaluation system is established based on Git submission records, code cyclomatic complexity and PEP8 specifications. Practice results show that the mode improves students' programming competence, code quality and team collaboration efficiency, which is convenient to operate and worthy of promotion in higher vocational programming courses.

Keywords—Python programming education, blended online-offline teaching, modular task decomposition, interface integration and debugging, multi-dimensional quantitative assessment

1 引　言

在人工智能技术快速发展背景下，国内众多高职院校将《Python程序设计》设置为计算机类专业入门编程核心课程[1]。Python凭借语法简洁、入门难度低、行业应用场景丰富等优势，获得师生广泛认可。但现阶段传统混合式教学模式应用于该课程时，仍存在两类典型教学弊端。

在个体学习层面，Python的低门槛易使学生形成被动学习习惯，线上学习多限于观看视频与模仿示例，缺乏自主设计代码和梳理逻辑的意识。面对复杂编程场景，学生难以将零散代码整合为结构化模块，导致程序冗余且逻辑不完整。在小组协作层面，多数学生虽能掌握基础模块化语法，但模块拆分缺乏标准化接口约定，分组实训中常出现任务分配不均，能力强者承担核心编码，弱者仅做辅助工作，进而拉大编程能力差距。

针对上述问题，本文提出模块化任务解构教学思

*基金资助：湖北省教育科学规划课题（2025GB231），基于产教融合的 AI 课赛融通模式：新工科双向赋能研究；湖北工程学院教改项目—AI 驱动的新工科教学范式改革实践（JY2025055）。湖北工程学院 2026 年课程思政优秀项目（KCSZ202626）—Python 程序设计。

路[2]。依托Python原生模块与包机制，将大型项目拆解为高内聚、低耦合的独立单元。课前通过线上资源进行学情诊断，课中实行分层分组教学，先由专项小组攻克难点，再由项目小组完成模块对接与系统联调。该模式契合协作编程的信息依存与个体责任理念，符合工程能力培养要求，有助于提升学生的逻辑思维与工程实践能力。

2 Python程序设计教学现状分析

伴随混合式教学改革全面落地，Python程序设计课程对学习者逻辑思维与工程实践能力的要求持续提升，传统混合式教学模式在编程课程中的适配短板逐步暴露。基于课程实训课堂的长期教学观察，当前高职Python程序设计教学主要存在四类共性问题[1]。

2.1教学形态同质化，逻辑构建动力不足

现阶段多数高职院校Python混合式教学流程趋于固化，普遍采用线上视频导学加线下验证性实验与机械代码练习的固定套路，课堂多沿袭教师演示、学生复刻代码的老路，学生沦为单纯的代码复刻执行者。Python语法简洁，入门门槛低，这本是优势，却也弱化了学生对算法逻辑与程序架构设计的重视。常规课堂缺少有针对性的编程逻辑引导，学生很难独立完成从业务需求到标准化工程代码的转化，计算思维培育效果并不理想[3]。

2.2协作机制浅层化，知识传递存在黑盒

程序设计课程本应依托团队协作实现编程思路的交互与难点共研，但当前课堂缺少规范的代码评审与逻辑研讨环节。线上导学阶段师生、生生之间的双向互动不足，知识传递以单向灌输为主。递归、装饰器、多线程这类重难点内容，学生在线上攒下的困惑到了线下课堂也得不到针对性解决。课堂表面看着热闹，学生核心的编程疑点却在持续堆积，隐性学习问题难以暴露[4, 5]。

2.3任务分配极化，“技术潜水”现象突出

高职生源编程基础差异显著，传统强弱混搭的分组模式没有设置硬性的模块逻辑隔离边界，课程项目开发任务常常分配失衡。班级里大约20%编程基础较好的学生扛下了80%的核心代码开发与逻辑设计，其余学生只负责文档撰写、界面调试、页面美化这类边缘事务。长期被动参与团队项目，不少学生养成了“技术潜水”的习惯—能躲就躲，能混就混[6]。这种分工方式不仅拉大了编程能力的两极分化，也背离了协作教学的初衷。

2.4评价维度单一，个体贡献难以量化

现有课程考核多半只看程序运行结果对不对，对编程全过程的行为观测基本缺失。学生代码调试、模块重构、问题攻坚这些过程中的投入，教师很难精准掌握。一个完整的项目成果交上来，小组内部谁干得多谁干得少根本看不出来[7]。边缘学生缺少过程性考核的约束，也得不到正向激励，学习的内生动力自然提不上去。

3“模块化任务解构”教学方案的构建与实施路径

针对当前Python教学存在的代码逻辑碎片化、团队协作形式化两大核心痛点，研究借鉴软件工程组件化开发思想，对传统拼图教学法加以改造，构建了逻辑单元演进教学模式。该模式依托模块物理逻辑隔离机制，明确每位学习者独立模块开发的权责，落实团队协作中的个体学习责任。前文已依托拼图教学法核心理论完成基础阐述，本章不再重复标注原有引文序号[2]。

该教学方案以Aronson拼图课堂教学理论为框架，通过专家组与原生项目组轮转协作的机制，实现团队内部信息相互依存与个体责任绑定，对协作学习中的社会懈怠问题起到了一定的遏制效果。编程课程团队学习中社会懈怠本就高发，尤其是在任务可拆分、个人贡献难以直观量化的实训场景中，这种现象更为突出。结合Python原生模块化开发特性，本研究增设了逻辑单元隔离与接口契约双重约束，要求学习者独立完成单一.py代码模块或程序类的开发，同步规范代码类型提示与异常捕获机制，让个人开发贡献可以直观量化，从工程开发层面遏制团队协作中的“搭便车”行为。

相较于传统翻转课堂教学模式，本方案优化了课前学情诊断环节，依据学生测评结果推送个性化微课补丁资源，替代过去那种“一刀切”的通用教学视频；课中引入Git代码管理钩子与PEP8代码规范自动化检测工具[8]，完善教学过程监控与闭环反馈，弥补了常规混合式教学在学生软件工程素养培育方面的短板。

3.1方案架构：基于工程能力的“双动力”协同模型

本次构建的教学方案以模块化任务解构为核心主轴，依托专家研修组、项目开发组双向轮转机制提供教学实施动力，整体分为两大核心模块。

任务解构为核心定轴：结合Python模块与包管理的原生特性，将综合性编程项目拆解为高内聚、低耦合的独立功能单元，统一固化模块接口契约。相比常规小组随机分工，这种做法的分工标准更贴近软件工程开发规范。

双组轮转为实施动力：同质专家小组聚焦课程共性编程难点开展集中攻坚，异质项目小组负责多模块集成与系统联调，兼顾学生编程知识深度拓展与项目

工程集成能力的培养。

3.2“八位一体”动态实施链路

研究设计了八位一体动态教学实施链路，整体流程如图1所示，八大核心实施环节如下：

（1）线上学情画像+补丁式微课：依托线上平台完成编程基础测评，依据学情数据推送个性化微课资源，实现前置知识的查漏补缺；

（2）逻辑隔离+接口协议签署：划分独立程序开发模块，明确单人独立开发权责，统一规范代码类型提示与异常处理规则，固化模块开发边界；

（3）专家组横向攻关+成果反哺：同质专家小组集中攻克项目共性技术难点，将标准化解决方案同步回流至原生项目组；

（4）契约联调+Git合并+代码走查：学生自主完成模块接口联调、代码冲突解决与规范化代码审查[8]，形成教学反馈闭环；

（5）工程指纹多维评价：依托Git提交日志、代码圈复杂度、PEP8规范合规率等多类指标开展量化考核，将考核重心从结果评价转向全过程评价[9]。

该链路采用线上赋能、线下实操、项目内化双循环结构，打破了传统线性教学的局限，将软件工程思维融入Python编程实训的全过程（图1）。

图1“八位一体”动态专长培养模式实施链路图

3.3“模块化”协作的规范约束与保障机制

为保障模块化任务解构教学模式能够真正落地，避免模块拆分流于形式，研究搭建了接口契约逻辑解耦、Git代码工程质量监控双重保障机制。

项目编码开始前，所有团队必须编制标准化《模块接口定义单》，统一明确模块输入输出参数、异常处理规则，以契约先行的方式锁定模块开发边界，避免模块逻辑重叠、开发责任推诿等问题。具体分工模型见表1和图2所示。

（1）基于“接口契约”的逻辑解耦规范

表1 基于“工程架构”的项目组与专家组异质化分工表

角色类型	分组形态	核心职能描述	逻辑边界约束
项目统筹员	项目组（异质）	维护API契约、主枝合并（Merge）、冲突调停	严禁接入任何模块代码，仅负责逻辑集成
数据专家	专家组（同质）	攻坚反爬策略、实现矢量化数据清洗算法	仅交互标准化JSON/CSV结构数据流
分析专家	专家组（同质）	构建统计建模、封装核心运算逻辑类	交付解耦后的运算模块接口（Interface）
交互专家	专家组（同质）	执行Matplotlib渲染、响应GUI事件驱动	仅交付UI模块对运算结果的视图呈现

异步采集模块（async_collector.py）由爬虫专家负责开发。其输入参数包括：urls: List[str]（待爬取的URL

列表）、proxy: Optional[Dict[str, str]]（代理配置，可选）、headers: Dict[str, str]（请求头伪装，必填User-Agent等）。输出采用生成器形式，yield Dict[str, Any]，每条数据为标准字典，包含raw_text、url、fetch_time等字段。

异常处理约定为：ConnectionError和TimeoutError日志记录后重试3次，失败则抛出自定义异常CrawledFailed；ValueError（URL格式错误）直接抛出，不进行重试。

该模块依赖Scrapy >= 2.5.0和redis >= 4.0，统一使用专家组共享的requirements.txt。前置条件为Redis服务已启动且队列可用、代理池至少保有5个有效代理且网络可访问目标站点。后置条件要求输出数据必须为标准字典格式，可直接转为pandas.DataFrame，严禁修改或删除原始采集内容。

接口稳定性测试需满足pytest覆盖率≥80%，必测场景包括：空URL列表返回空生成器、代理失效触发超时异常、验证码拦截返回空数据、超时重试3次失败等。

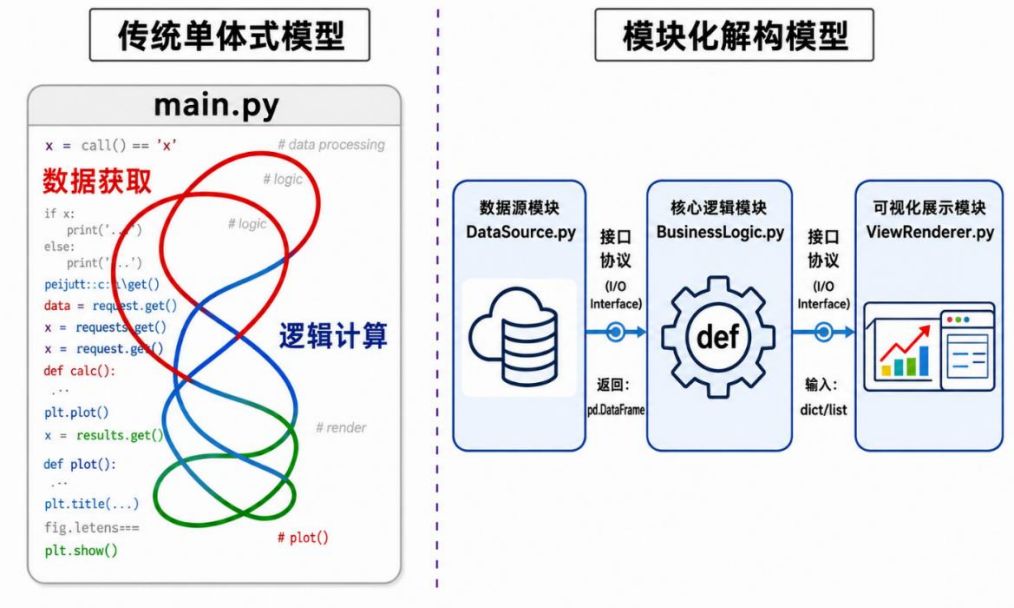

图2《Python程序设计》项目模块化解构与逻辑领地划分模型

接口定义单模板，模板统一规范模块基础信息、参数格式、异常规则与依赖环境，所有内容均采用Python标准类型提示语法编写。接口文件经专家组编制、小组交叉审核、教师终审签字后正式生效，作为后续Git代码合并与模块联调的官方依据，从流程上杜绝上游代码变更引发下游程序报错。标准化的接口单据可同步支撑pytest自动化契约测试，实现接口规范性的自动核验。

（2）引入Git钩子的工程质量监控

系统配置了Git前置提交钩子，学生推送本地代码至私有代码仓库之前，系统自动完成PEP8代码规范检测与圈复杂度扫描[8, 9]，不合规的代码无法提交，实现了代码质量的全过程刚性管控。这一机制推动课程考核从结果导向转向全过程行为画像，有助于夯实学生的基础代码工程规范。完整评价指标体系如表3所示。

表3 基于“工程指纹”的多维评价指标体系

评价维度	观测指标（Metrics）	采集工具/方法	权重占比
技术指纹	PEP8遵循度、Try-Except覆盖深度、代码复杂度	Pylint/Flake8静态扫描	40%
协作指纹	Git提交频率、有效变更行数、冲突消解贡献度	Git Inspector历史溯源	30%
工程交付	API文档规范性、模块联调通过率、接口稳定性	单元测试回归分析/同行评审	20%
职业素养	要求响应度、代码注释率、团队决策参与度	统筹员评价+教师行为观察	10%

关代码圈复杂度、PEP8规范遵循率这两项考核阈值，均结合行业标准、本刊同类教改研究成果与高职学情综合设定。代码圈复杂度分级标准参考软件工程通用风险分级规则，结合高职学生编程能力，划定单模块复杂度≤10为基础合格标准，11~20为工程达标标准，超过20则强制要求代码重构[9]。PEP8代码规范遵循率阈值设定为95%，兼顾企业代码审查标准与高职学生的学习承受能力，在规范养成与学习积极性保护

之间寻求平衡[8]。整套量化评价指标经过职教专家、企业后端开发工程师多轮论证，能够适配Python程序设计课程实训考核要求，评价结果客观可信。

4 教学实证案例分析—以“社交媒体舆情分析系统”为例

表4“舆情分析系统”模块契约与逻辑隔离规范

模块名称	负责人定位	核心技术栈	交付契约（I/O约束）	职责红线（隔离点）
异步采集	爬虫专家	Scrapy/Redis	yield: dict（详见表6模板示例）	严禁介入清洗，仅负责反爬对抗
特征工程	数据专家	Pandas	clean_df: DataFrame	不触碰源头，仅处理数据降维
情感极性	算法专家	Snownlp/Jieba	score: float	独立算法域，封装为静态类
流式渲染	交互专家	Flask/Echarts	render: html	仅消费结果，不干涉底层逻辑

为精准剖析编程课程团队协作中的伪协作、搭便车等共性问题[4]，选取社交媒体舆情分析系统作为实训实证案例。该项目涵盖异步爬虫采集、矢量数据清洗、文本情感分析、前端可视化渲染等完整开发链路，业务逻辑链条完整、模块耦合关系清晰，能够真实还原工程项目团队协作全流程，适配编程实训复杂协作场景的教学检验需求。

4.1任务解构：基于职责红线的逻辑领地划分

实训开展初期，依托接口契约机制完成四大核心模块物理逻辑隔离，明确各岗位开发权责与不可逾越的职责红线，模块划分规范如表4所示。

本次任务解构核心在于代码逻辑解耦与开发权限硬性约束。以异步采集模块为例，爬虫模块仅能输出标准化字典生成器，下游数据模块只能通过既定接口读取数据，无法篡改上游底层代码。模块联调阶段会主动抛出参数类型错误、键值缺失错误等异常，依靠程序运行报错倒逼学生坚守自身开发边界，从技术层面遏制小组内无偿代做、责任推诿等行为，规范团队协作秩序。

4.2空间移位：专家组专项攻坚消解逻辑分歧

在拼图教学专家组轮转环节，打破原有项目小组边界，将全班同岗位学生集中组建专项专家小组，集中攻克同类技术难点。全体爬虫专家统一研讨分布式代理池搭建、验证码自动识别等共性爬虫难题，对齐全班级代码开发标准与接口规范[10]。

教学观测数据表明，集中研修模式可推动学生学习思维从独立闭门编码转向团队标准协同对齐。例如在请求头伪装开发环节，部分学生因缺失动态Referer校验字段导致程序请求失败，专家组全员共同复盘HTTP底层通信逻辑，统一请求报文编写规范。专家组形成的标准化依赖配置文件、通用避坑方案同步回流至原生项目组，可快速解决多版本依赖冲突、库函数调用异常等联调常见问题，帮助学生完整建立工程化依赖管理思维。

4.3代码指纹溯源：数字化识别个体贡献差异

依托Git版本管理日志与静态代码检测工具，构建代码工程指纹评价体系[5][11]，剥离小组整体成果掩盖的个体贡献差异，实现团队成员贡献度客观量化，以实训第5组为例的量化审计结果如表5所示。

表5“工程指纹”审计与个体贡献量化分位表（以第5组为例）

学生姓名	模块定位	Commit频次	逻辑圈复杂度	PEP8报错率	贡献判定与成绩分位
学生A	爬虫专家	45（高频）	14（工程级）	2（极低）	核心(0.96)：逻辑闭环
学生B	算法专家	32（活跃）	22（重逻辑）	12（一般）	骨干(0.92)：算法迭代深度
学生C	项目统筹	60（极高）	5（架构级）	0（标准）	灵魂(0.94)：主导分支合并
学生D	交互专家	3（异常）	2（极低）	35（高/乱）	预警(0.60)：典型搭便车

审计结果直观体现小组内个体参与度差异：学生D代码提交频次极低、代码无业务逻辑复杂度且格式错误频发，属于典型协作懈怠群体；项目统筹人员高频参与代码合并与冲突处理，团队整体贡献度突出。该量化评价模式将考核重心从最终项目成品下沉至代码开发全过程，实现个体贡献可追溯、可量化，有效提升团队考核公平性，抑制协作搭便车现象。

4.4实证效应：从“脚本思维”向“工程思维”的转变

为期4周实操结果显示，学生在以下方面取得积极变化：

容错意识增强：自己处理循环导入、版本冲突，

对包管理有了实感；

逻辑参与度提高：契约一签，代劳现象少了很多，90%以上的学生主动开发自己模块；

代码规范性有所改善：在PEP8反馈下，大家开始主动改命名、调格式，代码看着舒服多了。

这些变化说明，该模式确实能帮助学生从零散脚本思维转向系统工程思维[12]。

5 教学效果评价与成效分析

本方案实施后，教学成效主要体现在学生过程参与度、代码规范性和工程意识都有明显进度。

研究采用准实验设计：选取湖北工程学院计算机相关专业的两个平行自然班作为样本：实验班（N=48）采用“模块化任务解构”模式，对照班（N=45）采用传统分组+线上资源模式。数据采集包括改革前（第1周诊断测评）和改革后（第4周末项目交付）的Git提交记录、PEP8扫描结果（使用Pylint工具）以及学生自评贡献度问卷，确保前后测可比性[11]。

5.1逻辑参与提升：缓解“技术潜水”问题

通过“模块化契约”机制，学生在任务中的参与度得到提高。在“舆情分析系统”实训中，接口契约将每位学生限定在特定.py模块责任域内。

关键指标“个体模块贡献率”（基于Git有效变更行数占比计算）前后测配对t检验结果显示：$t(47)=8.42$, $p<0.001$，均值从45.2%±18.3%提升至91.7%±7.5%，效应量Cohen's $d=2.81$（大效应）。与对照班独立样本t检验比较，实验班后测贡献率显著高于对照班（$t(91)=6.78$, $p<0.001$）。结果表明，该架构设计有助于提升边缘学生的逻辑参与度，实现“人人有专长、节点无盲区”的初步效果[11]。

5.2评价模型转型：过程审计精度提高

借助“代码指纹”溯源技术（Git提交记录、圈复杂度审计），实现了对个体产出的量化分析。表6汇总了实验班前后测及与对照班的后测比较，结果显示差异均高度显著（$p<0.001$），表明该模式对过程参与和规范遵守具有积极的影响。

表6 关键指标前后测与组间统计比较

指标	实验班前测(M±SD)	实验班后测(M±SD)	t值(配对)	p值	对照班后测(M±SD)	t值(独立样本)	p值
贡献率	45.2±18.3	91.7±7.5	8.42	<0.001	52.6±16.4	6.78	<0.001
PEP8 遵循率	60.4±22.1	94.8±5.2	7.15	<0.001	68.3±19.7	5.97	<0.001

在工程标准的持续反馈下，学生PEP8规范遵循率前后测t检验：$t(47)=7.15$，$p<0.001$，从60.4%±22.1%提升至94.8%±5.2%。该变化主要源于联调过程中的格式报错自我修正，体现了过程评价对规范养成的促进作用[12]。

5.3产出与岗位接轨

试点班级近两年在省级高职职业技能竞赛（如大数据与人工智能、软件测试赛项）中累计获奖12次，学生的Debug能力和多分支协作意识得到了评委的认可。

学生利用实训开发的舆情监控模块参与校企真实项目联调，部分代码进入企业预生产环境。毕业生在相关企业（如春晖、安恒信息）就业后，其模块化协同意识与开发流程适配良好，有效缩短岗前培训周期[4]。

6 结语

本次研究将线上线下混合式教学模式与拼图协作学习理论相结合，依托代码逻辑物理隔离机制，将被动团队协作压力转化为学生自主逻辑攻坚的学习动力。研究证实，Python程序设计课程小组分工不能停留在表层任务划分，必须依托软件工程标准实现模块硬隔离。双组轮转模式保障项目整体开发质量，接口契约机制守住团队协作公平底线。

依托真实工程项目开展分层协作实训，可全方位提升学生工程实践素养、代码规范意识与编程学习自信心，完成从零散脚本编写思维到系统化工程开发思维的转型升级。

后续研究可将智能代码辅助工具嵌入现有教学闭环，将GitHub Copilot等大模型插件接入Git前置提交钩子，在代码提交阶段自动完成长代码片段检测、冗余代码扫描与代码重构建议推送[13]。智能化前置检测可降低教师人工代码评审工作量，同时让学生在编码全过程获取实时优化建议，主动完成代码结构迭代。

该优化方案可贴合当下企业AI协同开发主流工作模式，提前培养学生人机协同编程能力。同时模块化契约教学模式具备良好可移植性，可推广至全部程序设计类课程及跨专业综合实训项目，助力学生实现从

基础代码编写，到高质量规范化代码开发与项目全流程代码管理的能力进阶。

参考文献

[1] 王瑞平,诸强,周雪忠,熊轲.基于OBE理念的Python语言程序设计课程教学改革与实践[J].计算机技术与教育学报,2026,14(1):55-59.

[2] 孟文龙,张策,张小东. 新工科背景下“计算机程序设计基础”课程教学改革探索[J]. 计算机技术与教育学报, 2024, 12(01): 85-90.

[3] 王雅娣, 谢玉琳, 郭小丁, 等. 数字化转型背景下行业大数据应用课程教学改革[J]. 计算机技术与教育学报, 2025, 13(2): 35-39.

[4] 陈磊, 鲍蓉. 应用型高校计算机网络类课程产教深度融合教学探索[J]. 计算机技术与教育学报, 2024, 12(4): 69-73.

[5] 潘娅, 范勇, 杨春明, 等. 以学生为中心建设可持续发展的大学生创新实践团队[J]. 计算机技术与教育学报, 2024, 12(6): 8-12.

[6] 李志刚, 杨吉斌, 张睿, 王彩玲, 吴永芬. 基于大模型试错与概念图的“程序设计”翻转课堂教学方法探索[J]. 计算机技术与教育学报, 2025, 13(4): 112-116.

[7] 何施茗, 曹嵘晖, 徐梓桑, 黄敏, 龙际珍. 长沙理工大学对面向系统能力培养实践课程体系的改革和成效综述[J]. 计算机技术与教育学报, 2025, 13(1): 71-75.

[8] 张晓明, 卓政, 田小平, 张莉, 崔宁. 基于三驱动四层次 AI 赋能计算机实践教学模式探索[J]. 计算机技术与教育学报, 2025, 13(4): 86-90.

[9] 叶鹏, 冯亚梦, 叶璐瑶, 等. 课程教学改革中计算机类本科生研究能力培养探索与实践[J]. 计算机技术与教育学报, 2025, 13(4): 62-70.

[10]石蕴玉,刘翔,汤显,单亮. 面向对象课程设计提升学生参与度的教学探索与实践[J]. 计算机技术与教育学报, 2025, 13(3): 93-100.

[11]陈磊, 鲍蓉. 应用型高校计算机网络类课程产教深度融合教学探索[J]. 计算机技术与教育学报, 2024, 12(4): 69-73.

[12]刘珂伶,魏冬梅. 头歌智慧教学平台赋能计算机编程教学改革探索[J]. 计算机技术与教育学报, 2025, 13(4): 250-254.