基于大模型与前端技术的闯关答题游戏，对 Python 课堂教改的实践探索

吴新刚 吴明泽 黄雅婕 李昱 李晓薇

烟台科技学院，山东烟台，265699；

摘要：本文设计并实现了一个基于大型语言模型和游戏化机制的 Python 编程闯关答题系统。本文从产业实际情况出发，以爬取招聘网站中 Python 岗位信息的形式：逆向梳理当下最热门的爬虫岗位、数据分析岗位、Web 开发岗位和人工智能岗位这四大主要职位要求的相关 python 技术路线的技能要点，并基于这些知识点利用大模型动态生成闯关答题系统的题目及试题分析和评分说明。游戏化学习策略的实施：可以让枯燥的 Python 学习变得更有意思也更有针对性。期望本研究对其他智能化教育工具提供可复制的样板及工程经验参考。

关键词：大语言模型；Python 教学；游戏化设计；系统实现；智能答题系统

DOI：10.64216/3080-1516.25.10.002

引言

Python 已经成为了当今最受欢迎的编程语言之一，广泛用于数据分析、人工智能、Web 开发等众多领域。然而，戴茂胤（2025）指出传统的 Python 教学模式仍存在：教学方法单一、实践环节薄弱、课程内容与实际应用脱节等问题，难以满足差异化学习和产业对接的需求。近些年，伴随着信息技术的快速发展，游戏化学习：应运而生，并在教学领域的探索日益深入。潘靓等（2025）的研究：强调了游戏化激励机制在提升学生课堂参与度与创新能力方面的潜力。此外，俞新龙（2025）提出：通过设计具有挑战性的考题，可以有效提高学生的解题能力和数学运算能力。向俊杰（2013）通过方法实践、展示了如何通过游戏化教学提高学生的题兴趣和能力。

本文旨在设计并实现利用大模型和前端开发技术完成的基于 Python 的闯关答题游戏，通过将产业的需求、和题目的动态生成、与游戏化激励机制深度融合，构建形成辅助学习 Python 编程的软件系统，旨在使学生更愿意、更深入、更容易地掌握有关编程的核心知识。

1 系统需求分析与总体设计

1.1 需求分析

1.1.1 功能需求

•Python 技术路线与岗位需求管理：爬虫每天定时：爬取并分析招聘网的岗位需求、倒推 Python 各技术路线、以及所需的知识点。

•学情问卷调研：提前了解学生的个性化学习情况与倾向。

•用户选择技术路线：根据自己的兴趣或者未来的规划，学生可选择自己感兴趣的技术线路，例如 Python 爬虫、Python 数据分析等；

•动态导入题库：针对用户选择的不同技术路线，系统动态加载相应试题；

•提交并进行判题：学生相应的答案，提交到数据库；所有的答案提交以后，系统会自动给出相应得分和是否正确；

•系统分析答题的结果，最终生成个性化反馈和激励内容。

1.1.2 技术需求

•大模型集成：需要集成有大模型生成接口，实现大模型自动生成智能答题等模块；

•数据库设计：完成游戏关卡和用户答题的数据库结构设计；

•前后端交互和游戏化展示：展示游戏化界面，实现前后端交互，提供良好的用户体验。

1.2 系统架构

系统采用前后端分离架构：

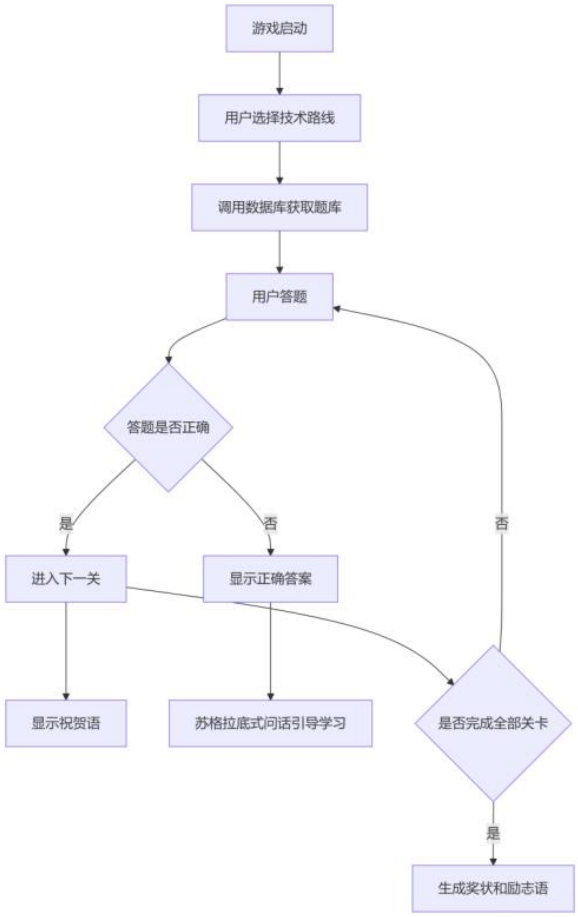
•前端：使用 Vue.js 实现响应式游戏界面，提供关卡的可视化、答题界面和实时反馈提示，增强游戏沉浸感。

•后端：采用 Node.js 与 Express 框架提供 RESTfulAPI，集成大模型服务（本系统采用百度千帆大模型平台），处理业务逻辑和数据交互。

•数据库：选用 MySQL，存储用户、试题及游戏进度数据。

2 系统详细设计与实现

2.1 核心功能模块详细设计与实现



2.1.1 技术路线与知识点动态获取模块

本模块是系统实现“产教融合”的关键基础，属于“事前”模块，核心是：流水线自动化的数据处理。

实现方案：采用来也科技的 uibot 低代码工具编写爬虫，定时爬取国内主流招聘平台中与 Python 相关的职位描述（JD）。

数据处理：爬取的原始 JD 文本经由大模型进行关键信息抽取。通过 Prompt（例如：“请从以下招聘文本中提取出要求的 Python 技术栈和核心技能点，并以 JSON 格式输出：{JD_TEXT}”），模型能够从非结构化的文本中识别并结构化输出技术路线（如“Python 数据分析”）和具体知识点（如“掌握 Pandas 进行数据清洗”）。

流程：爬取原始 JD->大模型信息抽取与清洗->人工审核->更新技术路线表（route）和知识点表（knowledge_point）。

2.1.2 试题动态生成与存储模块

该模块负责根据技术路线、知识点：自动生成高质量的关卡试题。

实现方案：系统将知识点名称、难度等级、题型要求作为输入，构造 Prompt 调用千帆大模型的文本生成 API。编程题的示例 Prompt：“你是资深的 Python 考官。请生成一道考查『{知识点名称}』的{难度等级}编程题。要求如下： 1.题目描述清晰，包含一个明确待解决的问题； 2.给出预期的输入输出示例； 3.提供完整的标准答案代码。输出格式为 JSON，包含 question_description, io_sample,standard_answer 三个字段。”

2.1.3 答题与判题模块

本模块是：系统最核心的交互模块，判题逻辑的准确性至关重要。具体策略如下：

选择题/填空题：实现简单的字符串匹配或选项比对，效率高。

编程题：采用 Docker 沙箱安全隔离技术实现动态判题。

实现流程：

用户提交代码后，系统将其与题目的测试用例（在生成试题时一并由 AI 生成并存储）打包。

启动 Docker 容器（内含预配置的 Python 运行环境）。

在容器内执行用户代码，注入测试用例。

捕获程序输出、运行错误或超时异常。

将输出结果与标准答案进行比对，并返回判题结果。

2.1.4 智能反馈与激励生成模块

苏格拉底式反馈：当用户答题错误时，系统将用户错误答案、题目信息及上下文作为输入，调用大模型生成引导性反馈。示例 Prompt：“用户在做一道关于『{知识点}』的题目时，提交了错误答案：{user_answer}。标准答案是：{standard_answer}。请不要直接指出错误，而是模仿苏格拉底的风格，通过提出 2-3 个启发式的问题，引导用户自己意识到问题所在。回复直接以问题开始。”

激励内容生成：通关后，系统根据用户的学习数据（如学习路径、通关时长、常错知识点）生成个性化奖

状和励志语。示例 Prompt：“用户已成功通关『{技术路线名称}』所有关卡，总用时{total_time}，在『{知识点 A}』等知识点上曾遇到挑战但最终克服。请生成一段热情洋溢的祝贺语，并颁发一个创意性电子奖状，突出其坚持和解决问题的能力。”

2.2 数据库设计

系统关键核心表的设计以下：

2.2.1 技术路线表(route)

id(BIGINT,PK):主键，自增。

name(VARCHAR(64)):技术路线名称。

description(TEXT):路线的详细描述。

source_from(VARCHAR(255)):数据爬取来源。

2.2.2 知识点表(knowledge_point)

id(BIGINT,PK):主键，自增。

route_id(BIGINT):所属技术路线 ID。

name(VARCHAR(128)):知识点名称（如“Django ORM”）。

difficulty(ENUM('easy','medium','hard')):难度等级。

2.2.3 试题表(question)

id(BIGINT,PK):主键，自增。

knowledge_point_id(BIGINT):关联的知识点 ID。

question_type(ENUM('single_choice','multiple_choice','fill_blank','programming')):题型。

content(TEXT):题干(题目描述和内容)。

io_sample(TEXT,NULL):仅编程题有效，输入输出示例。

answer(TEXT):答案。

test_cases(JSON,NULL):仅编程题有效，存储判题用的测试用例集。

difficulty(ENUM('easy','medium','hard')):试题自身难度。

2.2.4 用户答题记录表(user_question_log)

id(BIGINT,PK):主键，自增。

user_id(BIGINT):用户 ID。

question_id(BIGINT):试题 ID。

user_answer(TEXT):用户提交的答案。

is_correct(BOOLEAN):是否正确。

ai_feedback(TEXT):答题后 AI 生成的反馈内容。

time_spent(INT):本题耗时（秒）。

created_at(DATETIME):答题时间。

3 系统测试与分析

为确保系统的稳定性、可靠性及有效性，本研究进行了全面的系统测试与分析。

3.1 功能测试

功能测试采用黑盒测试方法，重点验证系统核心功能点的正确性和完整性。

表 3-1：核心功能测试结果统计

测试模块	测试用例数	通过率	主要问题	解决情况
爬虫与数据处理	45	100%	无	无
试题自动生成	68	97.1%	5%的生成题目格式不规范	已优化提示词工程
答题与判题	120	99.2%	复杂编程题判题偶发超时	调整 Docker 超时参数
智能反馈	85	95.3%	15%的苏格拉底式引导不够精准	持续优化提示词

3.2 性能测试

性能测试主要评估系统在高并发条件下的表现，采用负载测试、压力测试等方法。使用自研开发的多线程工具模拟不同并发用户数，对系统关键接口进行性能测试。

表 3-2：系统性能测试结果

性能指标	预期值	测试结果	是否达标
试题生成接口响应时间	≤3s	2.8s	是
答题判题接口响应时间	≤5s	4.5s	是
系统支持最大并发用户数	≥200	238	是
CPU 利用率（100 并发）	≤80%	75.3%	是
内存利用率（100 并发）	≤70%	68.2%	是

3.3 测试总结

系统核心功能模块的平均通过率达到 98.3%，少量问题聚焦在：大模型生成内容的质量和判题模块的稳定性上。通过优化提示词工程和调整 Docker 容器参数，问题得以有效解决。

系统各项性能指标均达到或超过预期值。通过峰值测试发现，当短时间内负载大幅度超出正常负载时，系统通过扩容策略仍能保持服务可用性。

4 总结

4.1 成果总结

本文提出一个基于大模型技术的 Python 闯关答题游戏，主要成果和创新如下：

基于产教融合的逆向设计方法，从招聘网站 Python 岗位场景出发、逆向反推技术路线与知识点体系，使得学习内容与产业需求紧密结合。

引入大模型能力产生试题并提供实时反馈与评价信息，使游戏系统与学习内容深度融合，激发学习兴趣、保持学习热情；结合学习内容与学习者的兴趣爱好，创造个性化的沉浸式游戏体验环境。

4.2 研究局限

尽管本研究取得了预期成果，但仍存在局限性：

依赖外部数据和服务：技术路线和知识点体系依赖于招聘网站的数据；系统功能依赖于大模型 API 服务。要考虑这些第三方资源的稳定性和开销问题。

应用范围还不广：目前系统仅在 Python 编程基础课程中进行了一个班小规模试点。

评价体系有待进一步完善：现有评价主要由闯关成功率、题目正确率决定，而对于编程思维、能力的评价和创新能力的评估有待探讨。

4.3 未来研究方向

未来可以用更新颖的“多模态”的大模型为支持的图像、视频等多种试题方式和回答方式，并采用算法对试题的难度进行自适配调整；并且未来还可扩展到数据科学或者其他编程语言与能力培养领域；再有，未来可开展游戏化学习的长期效果研究：持续跟踪调查与分析，作为系统进一步发展的基础。

4.4 结语

总之，本研究是一次信息技术和教育深度融合的成功尝试，集产教融合逆向设计、大模型技术赋能以及游戏化机制于一体，大大提高了编程学习的趣味性、个性化和实用性，同时也探索了技术教育的一种新模式，期望为此后课堂教学改革带来些许新思维启发和新方向启发。

参考文献

[1]孙莉霞.建立生动有趣的课堂评价体系[J].教育,2016,(25):65.

[2]潘靓,宋冰.针对小学三年级认知特点的信息科技双层PBL融合评价模式研究[J].中国信息技术教育,2025,(11):33-36.

[3]俞新龙.一道 T8 联考题的解题反思[J].广东教育(高中版),2025,(06):33-36.

[4]向俊杰.从游戏中得到的解题方法[J].语数外学习(初中版上旬),2013,(Z1):64-65.

[5]戴茂胤.基于 PBL 理念的 Python 程序设计课程教学改革与实践[J].电脑知识与技术,2025,21(08):171-173.DOI:10.14004/j.cnki.ckt.2025.0399.

[6]中华人民共和国国家市场监督管理总局,中国国家标准化管理委员会.GB/T39788-2021 系统与软件工程性能测试方法[S].北京:中国标准出版社,2021.