Advances in Education 教育进展, 2026, 16(2), 563-570
Published Online February 2026 in Hans. https://www.hanspub.org/journal/ae
https://doi.org/10.12677/ae.2026.162331

Hans汉斯

大模型环境下Python教学的范式迁移与能力重构

刘柏霆，黄勇萍*，刘小满，赵俊涛

广西民族师范学院数学与计算机科学学院，广西 崇左

收稿日期：2026年1月1日；录用日期：2026年1月28日；发布日期：2026年2月6日

摘　要

随着生成式人工智能的爆发，传统的编程教育面临巨大挑战。本文针对《Python程序设计》课程中存在的语法教学碎片化、学生解决复杂问题能力弱等痛点，提出了一种“AI协同编程”的教学新模式。通过引入提示词工程、AI辅助纠错及大模型项目驱动教学，旨在培养学生在AI时代下的算法思维与系统集成能力。

关键词

生成式人工智能，Python程序设计，教学范式重构，提示词工程，AI协同编程

Paradigm Shift and Competence Reconstruction in Python Programming Education under the LLM Environment

Boting Liu, Yongping Huang*, Xiaoman Liu, Juntao Zhao

School of Mathematics and Computer Science, Guangxi Minzu Normal University, Chongzuo Guangxi

Received: January 1, 2026; accepted: January 28, 2026; published: February 6, 2026

Abstract

With the rapid emergence of generative artificial intelligence, traditional programming education is facing significant challenges. Focusing on the issues in the *Python Programming* course—such as fragmented syntax instruction and students' limited ability to solve complex problems—this paper

*通讯作者。

文章引用: 作者 1, 作者 2, 作者 3. 大模型环境下 Python 教学的范式迁移与能力重构[J]. 教育进展, 2026, 16(2): 563-570. DOI: 10.12677/ae.2026.162331

proposes a new teaching model of "AI-collaborative programming." By incorporating prompt engineering, AI-assisted debugging, and large language model–driven project-based learning, the approach aims to cultivate students' algorithmic thinking and system integration capabilities in the AI era.

Keywords

Generative Artificial Intelligence, Python Programming, Teaching Paradigm Reconstruction, Prompt Engineering, AI-Collaborative Programming

1. 引言

自 2022 年末大语言模型(Large Language Model, LLM)爆发以来，软件开发领域经历了一场前所未有的范式转移[1]。以 OpenAI 的 GPT 系列、DeepSeek 为代表的生成式人工智能，展现出了卓越的代码生成、调试与优化能力。在工业界，AI 辅助编程(AI-Augmented Programming)已成为开发者提升效率的标准配置。然而，反观高等院校的《Python 程序设计》课程，教学模式仍普遍滞后于技术变革。传统的教学大纲大多围绕变量、循环、数据结构等语法碎片展开，这种"手工业时代"的教学思维，在面对"只需自然语言描述即可生成高质量代码"的 AI 时代时，正面临着严峻的生存危机与合法性挑战。

在长期的教学实践中，传统的 Python 教学模式暴露出了三大核心痛点："见木不见林"的语法陷阱，学生花费大量精力记忆繁琐的语法规则(如复杂的切片操作、闭包等)，却在面对实际工程问题时，缺乏将业务逻辑转化为程序架构的系统性思维；调试过程的挫败感，初学者往往因为一个缩进错误或库版本冲突而陷入长时间的停滞，这种低效的报错处理消耗了学习热情，阻碍了高阶思维的培养；考核机制的滞后性，传统的闭卷考试或标准答案式作业，在 AI 触手可及的今天，既无法真实反映学生的工程能力，也难以避免学术诚信的风险。

现有的 Python 课程往往过度强调语法细节的严苛性，而忽略了在 AI 工具辅助下，学生应当具备的高阶思维[2]。当学生进入职场时，他们面对的是已经高度自动化、智能化的生产环境，如果他们在学校仅学会了如何闭卷编写简单的算法，而不会利用 AI 进行辅助推理、系统集成和复杂 Debug，将难以适应现代企业的岗位需求。Python 语言因其简洁的语法和强大的库生态，已不仅仅是计算机专业的编程语言，更成为了全学科通用的"数字工具"。在 AI 环境下，教学重心应当从手工编码转向 AI 辅助推理。

本研究旨在探索一种全新的《Python 程序设计》教学范式。该范式不再将 AI 视为禁忌或作弊工具，而是将其作为"数字领航员"引入课堂。通过重塑教学目标、解构课程内容、创新评价体系，培养学生掌握"提示词工程(Prompt Engineering)"与"代码评审(Code Review)"的核心能力。这不仅是为了应对 AI 的冲击，更是为了赋予学生在智能时代驾驭复杂系统的能力，实现从"被动接受语法"向"主动驱动智能"的跨越式转变。

2. 大模型驱动下的教学范式重构设计

在企业生产环境转向"AI + 程序员"协作模式的背景下，本课程教学改革的核心逻辑在于：将 Python 语言视为"逻辑表达的载体"，将 AI 视为"执行与调试的引擎"。本章提出"1 + N"融合教学框架，旨在实现从语法驱动向逻辑推理驱动的转型。

2.1. “1 + N”融合教学框架构建

“1 + N”融合教学框架强调以稳定的人类逻辑能力为中心(1)，辅以可变的、多类型 AI 工具(N)，形成一种具备可迁移性和可持续演化能力的教学结构。

2.1.1. “1”：以 Python 逻辑思维为主线的稳定核心

“1”并非指具体语法体系，而是指通过 Python 这一通用语言所承载的计算逻辑、问题建模与抽象表达能力，如图 1 所示。在教学实施中，教师不再围绕“函数如何书写”展开，而是围绕以下逻辑能力进行系统引导：问题的结构化拆解(输入 - 处理 - 输出)、算法与数据流的因果关系理解、条件判断与异常分支的逻辑完备性分析。

在具体教学情境中，教师从“如何拆解问题”入手，而非“如何记忆语法”。例如，在数据分析主题中，教学重点放在理解数据处理流程(数据获取→清洗→转换→分析→可视化)，而非要求学生机械记忆 Pandas 或 Matplotlib 的具体 API 调用方式。

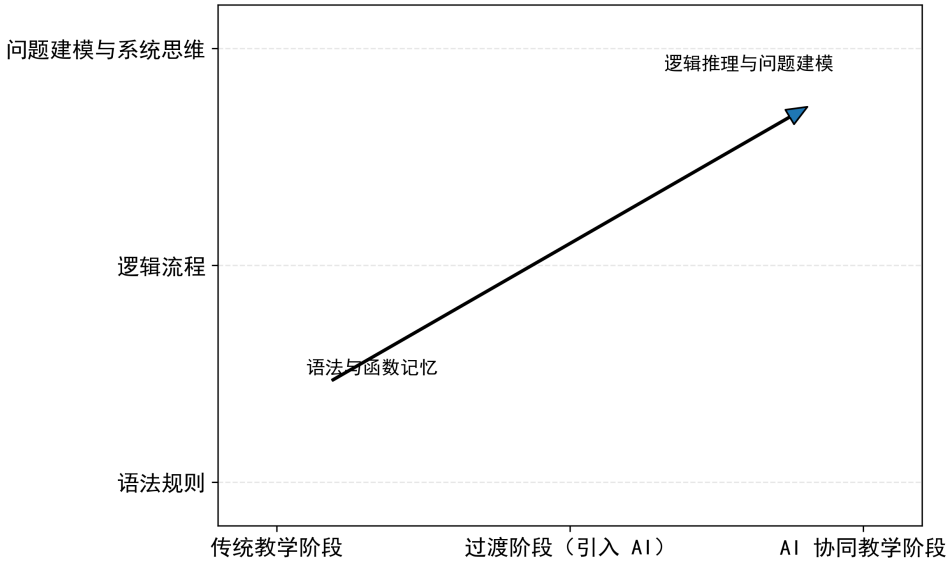

Figure 1. Illustration of the shift in teaching focus
图 1. 教学关注点迁移示意图

2.1.2. “N”：多维度大模型工具的动态嵌入

“N”代表以大语言模型为核心的一系列智能工具集合[3]，其角色并非替代学习者，而是嵌入到不同教学环节，作为认知放大器与反馈引擎存在。AI 赋能点主要体现在以下三个关键阶段：

代码生成初稿阶段，利用 AI 将自然语言逻辑快速转化为可运行代码，使学生将注意力集中于“逻辑是否正确”，而非“代码是否能写出来”。

运行报错与异常排查阶段，AI 作为“第二视角”，辅助学生分析报错原因，训练其从错误信息反推逻辑缺陷的能力。

代码优化与重构阶段，借助 AI 分析代码复杂度与可维护性，引导学生理解不同实现方案之间的权衡关系。

2.2. 课程内容的解构与重组

为适应 AI 辅助推理型学习方式，课程内容不再按“知识点难度递进”线性展开，而是依据学生认

Table 1. The functional positioning of AI in various stages of teaching
表 1. AI 在教学各阶段中的功能定位

教学阶段	学生主要任务	AI 角色	能力培养重点
需求理解	逻辑拆解	需求翻译器	抽象建模
编码实现	方案评估	代码生成器	逻辑判断
调试优化	问题定位	推理助手	反向推理

知角色的变化，重构为三个递进层级，并重新分配教学时间与实践比重，如表 1 所示。然而在使用 AI 工具前，需引入基础能力测评。要求学生在受控的离线编译器中，完成逻辑密度高、但代码量小的核心算法。只有通过该测评的学生，方可获得后续实验中调用 AI 工具的权限。

2.2.1. 基础模块：从“手动编码”到“人机共读”

在基础阶段，课程并未削弱 Python 基础语法的重要性，但教学目标由“会写”转向“会读、会解释”。

(1) 语法内化：从书写者到解释者

如图 2 所示，学生重点训练阅读 AI 生成代码的能力，能够说明每一段代码的功能、关键变量与逻辑分支的作用和代码与原始需求之间的映射关系。

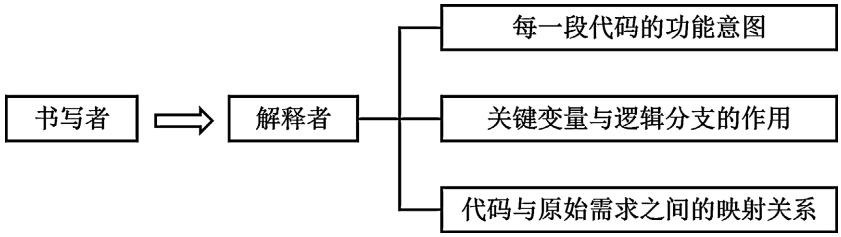

Figure 2. Illustration of the shift in teaching focus
图 2. 教学关注点迁移示意图

(2) Prompt 基础训练：计算思维的语言化表达。

引入“计算思维 + 语言工程”的复合训练方式，教学如何使用结构化自然语言描述编程需求，明确输入输出边界、指定判断条件与异常处理规则和通过变量命名强化逻辑表达。

2.2.2. 进阶模块：从“功能实现”到“逻辑校验”

在 AI 可快速生成可用代码的前提下，学生的核心角色转变为“代码质量守门员”，如表 2 所示。

Table 2. Comparison table of student role transitions
表 2. 学生角色转变对比表

阶段	学生主要行为	认知层级
传统	写代码	技能执行
AI 协同	审代码	逻辑判断与评价

(1) 逆向调试实训

教师借助 AI 生成包含隐性逻辑缺陷的代码案例(如边界条件缺失、性能隐患)，要求学生通过推理定位问题来源，而非直接运行试错。

(2) 代码评审(Code Review)机制引入

模拟企业开发流程，学生在 AI 辅助下分析代码的时间复杂度与空间复杂度，并撰写评审报告，重点解释：AI 给出的优化建议是否合理、不同方案在可读性与性能之间的权衡。

2.2.3. 综合模块：从“闭环练习”到“开放式系统集成”

在综合阶段，课程模拟真实生产环境，取消标准答案导向，强调系统集成与架构思维。

(1) API 驱动开发实践

引导学生调用大模型 API 或第三方服务，开发具备真实应用价值的 Python 程序，如智能问答助手、自动文本摘要系统等[4]。

(2) 复杂系统拆解训练

教学重点转向如何将大型需求拆解为 AI 可处理的子任务，训练学生以“系统设计者”的视角统筹功能模块与数据流。

2.3. 教学模式对比分析

表 3 总结了本研究提出的改革方案与传统方案的本质区别。

Table 3. Comparison table of teaching models
表 3. 教学模式对比分析表

维度	传统教学模式(工业 2.0 遗存)	改革后教学模式(AI 协同范式)
核心目标	熟练掌握 Python 语法规则	培养基于 AI 的问题解决与推理能力
程序员角色	低阶重复性编码工作	监制/架构师(逻辑设计与代码审计)
报错处理	查阅文档、反复盲目修改	利用 AI 推理报错原因并进行逻辑修复
对 AI 的态度	排斥性防御	深度集成(视为结对编程伙伴)
生产力体现	编码速度与准确率	需求拆解深度与 AI 工具驾驭水平
维度	传统教学模式(工业 2.0 遗存)	改革后教学模式(AI 协同范式)

2.3.1. 认知负荷的有效转移：从“语法存贮”到“逻辑评估”

在传统模式下，学生的认知资源大量被复杂的语法规则(如 Python 的装饰器、深浅拷贝、闭包等)所占用。心理学中的认知负荷理论认为，当短期记忆被琐碎规则填满时，高阶的逻辑推理能力会被抑制。改革后的模式通过 AI 承担了“语法存储”与“初级代码实现”的重任，将学生的认知负荷从内部认知负荷(记忆语法)转向了相关认知负荷(理解算法逻辑与系统架构)。这种转移使得非计算机专业的学生也能快速跨越“语法红利期”，直接进入利用 Python 解决专业问题的实质阶段。

2.3.2. 调试范式的重构：从“盲目试错”到“逻辑溯源”

传统的报错处理通常是“查阅文档–盲目修改–运行报错”的低效迭代陷阱，学生往往在挫败感中丧失兴趣。引入 AI 辅助推理后，调试过程转变为一种“对话式诊断”。学生需要将 AI 给出的报错分析与自己的业务逻辑进行对标，通过推理判断 AI 的建议是否符合当前的业务上下文。这种“逻辑溯源”过程本质上是高强度的思维训练，它要求学生不仅要知其然(代码能跑通)，更要知其所以然(代码为何这样改)，从而在真实生产环境中具备更强的故障排查韧性。

2.3.3. 角色定位的职业化对接：从“代码搬运工”到“技术审计师”

如引言所述，企业真实环境中纯手工编码的岗位正在萎缩。传统教学培养的是“执行者”，而新模式培养的是“决策者”。通过在课堂中引入“代码评审(Code Review)”环节，学生扮演的是技术审计师

的角色。他们需要评估 AI 生成代码的可维护性、安全性和执行效率。这种角色的提前转变，使得教学目标与工业界的“架构师思维”无缝对接。在这一过程中，学生学会了如何管理 AI 的输出，而非被 AI 的错误所误导，这正是 AI 时代程序员的核心护城河。

2.3.4. 评价维度的多元化：从“结果唯一”到“过程协同”

传统的考核往往通过 OJ (Online Judge)平台进行自动化评测，只看输出结果是否匹配。在新模式下，单纯的“结果正确”已不再是衡量水平的唯一指标。分析认为，教学评价应转向“人机协同的效能比”。例如，评价学生如何通过三轮 Prompt 的迭代，将一个冗余的算法优化为高效的生产级代码。这种评价维度的变化，能够真实反映出学生对复杂问题的拆解能力和对智能工具的驾驭深度。

3. 关键技术与 AI 协同教学实践路径

在明确了范式重构的设计思路后，本章重点探讨如何将“AI 辅助推理”具象化为可操作的教学实践。通过引入提示词工程(Prompt Engineering)、逆向工程思维训练以及人机协同实验设计，构建一套闭环的教学路径。

3.1. 提示词工程(Prompt Engineering)的融入与能力培养

在真实生产环境中，向 AI 描述需求的能力即是生产力。我们将提示词工程从简单的“提问”提升为一种“结构化编程思维”。

需求拆解训练：教学中不再直接给学生代码题目，而是给出一个复杂的业务场景。要求学生先不写代码，而是利用“思维链”技术，将大目标拆解为：数据抓取、格式清洗、逻辑运算、图表渲染四个子任务。

结构化提示词设计：引入角色(Role)、上下文(Context)、任务(Task)、约束(Constraint)的四要素框架。学生需学习如何编写“高质量提示词”来驱动 AI 给出生产级的 Python 代码。

3.2. 逆向工程思维：基于 AI 错误生成的逻辑纠偏

为了防止学生产生“技术依赖”而丧失思考能力，本课程引入了“主动干扰”教学法。

AI 幻觉识别：教师演示 AI 在处理复杂库(如深度学习框架或特定金融接口)时可能产生的“代码幻觉”或过时语法。

逻辑逻辑缺陷识别：利用 AI 生成一段在特定边缘条件下会崩溃的代码。例如，一段未处理 ZeroDivisionError 的数据平均值计算函数，要求学生通过推理找出逻辑漏洞并进行加固。

性能对比实验：让学生用 AI 生成三种不同时间复杂度的排序或搜索方案，并使用 timeit 模块进行实测，分析 AI 方案在处理大数据量时的优劣，从而培养学生对代码效率的底层推理能力。

3.3. 实验环节：从“复现代码”向“系统集成”转型

传统的 Python 实验多为复现经典算法[5]，改革后的实验设计更强调人机协同的系统构建。

API 驱动的模块化开发：实验项目不再局限于本地运行。学生需学习如何通过 Python 调用大模型 API (如 DeepSeek、OpenAI)、天气 API 或地图 API。通过 AI 生成接口调用模板，学生重点负责各模块之间的逻辑联调与异常处理。

自动化测试驱动：在实验中强制引入单元测试。要求学生利用 AI 编写测试用例，通过“测试失败、AI 修复、测试通过”的闭环，模拟工业级软件开发的质量控制流程。

3.4. “1 对 1”AI 助教的常态化部署

为了解决大班教学中教师无法兼顾每位学生 Debug 需求的问题，本研究建议在课程中部署专门的 AI

助教平台。

实时代码解析：学生在遇到看不懂的代码段时，可一键发送给 AI 助教进行逐行解释，但系统设置限制 AI 直接给出作业答案，而是给出逻辑指引。

个性化错题集：AI 助教记录学生频繁出错的逻辑点(如：递归深度理解、全局变量误用)，定期生成个性化的逻辑强化练习题。

4. 评价体系重塑：从“结果导向”转向“人机协同过程导向”

在引入 AI 驱动的教学模式后，传统的“标准答案式考核”已完全失效[6]。如果评价标准不随之改革，学生极易滑向盲目复制 AI 生成结果的深渊。因此，本章提出一套全新的多元化评价体系，旨在精准衡量学生在 AI 辅助下的真实推理能力与工程思维。

4.1. 考核维度的解构与权重重组

新评价体系不再单一考查代码的正确性，而是将其拆解为“人机协同”的多个环节，如表 4。

Table 4. Deconstruction and reweighting of assessment dimensions
表 4. 考核维度的解构与权重重组

评价指标	考核重点	权重分配
需求建模与拆解	能否将模糊业务需求转化为清晰的逻辑步骤	25%
逻辑校验与评审	对 AI 生成代码的 Debug 能力、性能优化建议及安全性评估	30%
系统集成能力	多个 AI 生成模块的联调、第三方 API 调用及整体项目稳定性	25%
编程伦理与规范	代码注释、规范性、以及对开源协议和数据隐私的遵循	10%
传统手写能力	在无 AI 环境下完成核心算法的编写	10%

4.2. 过程化动态评价：引入“Prompt 演进记录”

为了遏制学术不端并观察学生的思维演进，评价过程从“静态终点”转向“动态全景”。

Prompt 追溯机制：学生提交作业时，必须附带其与 AI 交互的对话记录或 Prompt 迭代历史。教师通过审查学生如何根据 AI 的反馈调整指令，判断其对问题的拆解是否深入[7]。

版本控制审计：引入 Git 教学，要求学生在项目开发的不同阶段进行 commit。通过分析代码从“AI 初稿”到“人工微调”再到“最终定稿”的演进路径，评估学生的参与度与贡献值。

4.3. 考核形式的创新：从“闭卷考试”到“开放式逻辑辩论”

反向面试(Reverse Interview)：教师或 AI 考官针对学生提交的项目代码，随机抽取片段提问：“为什么这里选择这种数据结构？”“如果 AI 给出的方案存在并发冲突，你会如何修改？”。通过口头答辩验证代码是否被学生真正“内化”。

盲测 Debug 挑战赛：提供一段由 AI 生成且包含隐蔽逻辑缺陷的代码，要求学生在规定时间内利用推理能力定位错误并修复。这种形式比传统的代码填空更能反映学生在真实生产环境中的应变能力。

4.4. 针对 AI 伦理与诚信的评价指标

在评价体系中首次引入“AI 伦理系数”。

归因声明：学生必须明确标注哪些代码由 AI 生成，哪些由自己原创。

安全评估：考查学生是否会对AI生成的代码进行安全性校验(如防止SQL注入、硬编码敏感信息等)，这不仅是技术评价，更是对职业操守的预演[8]。

5. 结论与展望

本文针对人工智能浪潮下编程教育的滞后性，系统性地提出了《Python 程序设计》课程的教学改革方案。通过对企业真实生产环境的调研发现，“AI辅助推理”已取代“纯手工编码”成为行业主流范式。基于此，本研究重构了教学大纲，引入了提示词工程、逆向逻辑校验及人机协同实训，并配套构建了过程化、多维度的评价体系。实践证明，这种模式不仅有效缓解了初学者在语法细节上的认知负荷，更显著提升了学生解决复杂工程问题的系统性思维，实现了从“语法背诵者”向“智能系统架构师”的职业素养跨越。

尽管引入 AI 极大提升了教学效率，但在实践中仍面临一定的挑战。一方面，过度依赖 AI 可能导致部分学生忽略对底层原理(如内存管理、基本算法复杂度)的理解。未来需进一步平衡“工具调用”与“底层内功”的关系。另一方面，改革对授课教师提出了极高要求，教师不仅要精通 Python，还需具备深厚的提示词设计经验及敏锐的代码审计能力。

随着大模型技术的迭代，未来的 Python 教学将向“全自然语言驱动”进一步演进。教学的终极目标将不再是消灭代码，而是消灭“低效代码”。未来的研究将探索如何利用国产大模型构建更加垂直的编程教育智能体，实现真正意义上的“千人千面”个性化编程教学。

基金项目

广西高等教育本科教学改革工程项目(2025JGA373)；广西民族师范学院科研基金项目(2024YB124)。

参考文献

[1] 王聪, 万聪. 大模型时代计算机程序设计类课程教学模式探索[J]. 计算机教育, 2025(4): 137-141.

[2] 牟玉亭, 龙寰, 蒋浩. 融入 AI 大模型的计算机程序设计教学实践[J]. 电气电子教学学报, 2024, 46(4): 128-131.

[3] 王昊, 童峥嵘. 大模型驱动下 Python 程序设计教学模式创新研究[C]//北京高校电子信息类专业群暨教育部电子信息类专业虚拟教研室全国院校教育教学研究成果论文集. 2025: 390-395.

[4] 苏瑶, 李磊. AIGC 赋能程序设计类课程教学模式改革研究[J]. 福建电脑, 2025, 41(12): 115-118.

[5] 林国宇, 温海标, 李明彤. 大语言模型在“Python 程序设计”教学改革中的应用探索[C]//北京高校电子信息类专业群暨教育部电子信息类专业虚拟教研室全国院校教育教学研究成果论文集. 2025: 241-245.

[6] 刘凌, 吴永芬, 陈卫卫, 等. 大语言模型对高校程序设计类课程的挑战与机遇[J]. 计算机教育, 2025(2): 118-122.

[7] 张彦芳, 高璐, 李艳. AI 技术项目驱动下的程序设计类课程教学模式创新[J]. 计算机教育, 2025(2): 159-163.

[8] 李朋朋, 刘涛, 张黎明, 等. 大语言模型赋能测绘类专业 Python 程序设计课程教学改革探索[J]. 测绘通报, 2025(S2): 339-342.