Advances in Social Sciences 社会科学前沿, 2026, 15(7), 226-232
Published Online July 2026 in Hans. https://www.hanspub.org/journal/ass
https://doi.org/10.12677/ass.2026.157567

Hans汉斯

基于支架式教学的Python编程实验教学改革与实践

陈　豪*，吴　航，高宇航，朱庆香

淮北理工学院电子与信息工程学院，安徽 淮北

收稿日期：2026年5月29日；录用日期：2026年7月13日；发布日期：2026年7月22日

摘　要

针对人工智能专业Python程序设计课程实验教学中存在的理论脱离实践、学生编程畏难情绪严重、实验完成率低等问题，借鉴支架式教学理论，构建“范例支架→代码补全支架→改错支架→独立编程”四阶递进式实验教学模式。通过逐级降低支架支撑力度，引导学生在最近发展区内实现从代码阅读理解到独立编程的能力跨越。实践表明，该模式有效降低了学生编程认知负荷，显著提升了实验参与度和完成率，为编程类课程实验教学改革提供了可操作的参考路径。

关键词

支架式教学，Python编程，实验教学，代码补全，人工智能专业

Reform and Practice of Python Programming Laboratory Teaching Based on Scaffolding

Hao Chen*, Hang Wu, Yuhang Gao, Qingxiang Zhu

College of Electronic and Information Engineering, Huaibei Institute of Technology, Huaibei Anhui

Received: May 29, 2026; accepted: July 13, 2026; published: July 22, 2026

Abstract

In response to issues in the practical teaching of Python programming courses for artificial intelligence majors—such as a disconnect between theory and practice, students' significant reluctance to engage with programming, and low completion rates—we have drawn upon scaffolding theory to construct a

*第一作者。

文章引用: 陈豪, 吴航, 高宇航, 朱庆香. 基于支架式教学的 Python 编程实验教学改革与实践[J]. 社会科学前沿, 2026, 15(7): 226-232. DOI: 10.12677/ass.2026.157567

four-stage progressive practical teaching model: "example scaffolding → code completion scaffolding → error correction scaffolding → independent programming". By gradually reducing the level of scaffolding support, this approach guides students to make the leap from reading and understanding code to independent programming within their zone of proximal development. Practice has demonstrated that this model effectively reduces students' cognitive load in programming, significantly enhances experimental participation and completion rates, and provides a practical reference framework for the reform of experimental teaching in programming courses.

Keywords

Structured Teaching, Python Programming, Practical Teaching, Code Completion, Artificial Intelligence Degree Programme

1. 引言

随着人工智能技术的迅猛发展，Python 凭借其简洁易读的语法和丰富的第三方库生态，已成为人工智能领域最主流的编程语言，也是各高校人工智能专业的核心基础课程。2017 年教育部发布的《新工科研究与实践项目指南》[1]明确提出，要深化产教融合，强化学生工程实践能力培养。在此背景下，Python 程序设计课程的实验教学环节承担着将语法知识转化为编程能力的关键使命，其教学质量直接影响学生后续机器学习、深度学习等专业课程的学习效果。

然而，在长期的教学实践中，Python 编程实验环节的效果并不理想。尤其在面向大一新生开展实验教学时，"课堂听得懂、实验写不出"的困境尤为突出，如何有效降低编程入门的认知门槛、激发学生动手编程的内驱力，成为亟待解决的教学问题。本文结合人工智能专业 Python 程序设计课程的教学实践，引入支架式教学理论，探索渐进式实验教学模式，旨在为编程类课程实验教学改革提供参考。

2. 编程实验教学痛点

2.1. 认知负荷

Python 程序设计课程一般在大一学年开设，授课对象多为编程零基础学生。传统实验采用"从零编程"模式，要求学生面对空白编辑器，从无到有地编写完整程序。这一过程需要学生同时兼顾语法规则、算法逻辑和调试策略等多重认知任务，认知负荷远超初学者的承受范围[1]。以笔者所授课程为例，在"列表推导式与数据处理"这节实验中，对 2023 级 68 名人工智能专业学生的实验课堂数据统计显示，在要求独立编写完整程序的实验任务中，课内完成率仅为 32.5%，且超过 60%的学生在空白编辑器前停留时间超过 15 分钟无从下手，部分学生甚至盲目依赖 AI 或其他同学给出答案，逐步形成了"编程很难、我不适合"的自我否定心理。

2.2. 理论实践脱节

从学习迁移理论来看，课堂上的代码片段演示属于"近迁移"任务，而综合编程实验属于"远迁移"任务[2]。由于缺乏必要的中间桥梁，学生在面对要素组合复杂的远迁移任务时极易产生迁移障碍。理论

[1]http://www.moe.gov.cn/srcsite/A08/s7056/201707/t20170703_308464.html

课堂上，大多以代码片段的讲解与演示为主，学生能够接收单个知识点的含义，但在实验任务中，需要综合运用多个知识点。例如，有些学生能准确说出 for 循环语句和 if 条件句的语法结构，却无法将二者有机组合实现简单的成绩筛选功能。这种懂而不会的现象根源就在于学生缺少从“理解语法”到“应用编程”的中间支撑环节[3]，理论与实践之间缺少有效的桥梁，传统实验课堂直接要求学生从空白编辑器开始从零编写代码，跳过了模仿与半独立练习的阶段，导致认知跨越过大。

2.3. 评价体系单一

传统实验教学中的考核多以“程序能否正确运行”作为评分依据，在上一学年的课程学生评教中“缺乏成就感”的描述占比达 30%，这种结果导向的评价方式虽能迅速直观地展示学习成果，但也存在两方面问题，其一是忽视了学生调试和纠错中的努力过程，导致学生遇到频繁报错时失去耐心、直接求助或复制他人代码，而非独立排查问题；其二是这种评价方式无法区分“理解但未完成”与“完全不理解”的差异，统一归类为评价较差的一档，挫伤了部分学生的积极性。此外，部分实验题目难度偏高且缺少梯度设计，学生难以在完成过程中获得阶梯式成就感，自行完成实验的动力也随之下降。

3. 基于支架式教学的 Python 编程实验教学改革举措

3.1. 理论基础

维果茨基的“最近发展区”理论中提出了支架式教学这一概念[4]。该理论认为，学生在学习过程中存在两种发展水平：一种是已经达到的现有水平，另一种是在他人指导或同伴帮助下可能达到的潜在水平，两者之间的差距即为“最近发展区”。在此区域内提供适当的学习支架，能够最大程度地帮助学生跨越能力差距；同时，随着学生能力的增长逐步将支架移去，提高学生学习的独立自主能力。

近年来，国际编程教育领域也涌现了多种旨在降低初学者认知门槛的渐进式教学法，如 PRIMM 模型[5]和 Use-Modify-Create 模型[6]。这些经典研究均强调从代码阅读理解到独立编写的过渡，与支架式教学有共通之处。然而，本文提出的四阶递进式实验支架模型与它们相比具有显著的差异化特征：在支架颗粒度上，PRIMM 和 Use-Modify-Create 模型将“修改”作为单一阶段，而本文模型将其进一步细化为“代码补全支架”和“改错支架”两个独立阶梯。这种细化专门针对零基础初学者面对空白编辑器时的高认知负荷与语法焦虑，通过填空提供语法托底，通过纠错专项培养调试能力。因此，本研究在借鉴国际经典编程教学法的基础上，进行了更具专业针对性和颗粒度更细的本土化创新。

编程实验的技能习得过程天然符合“模仿→半独立→独立”的渐进规律，与支架式教学“搭建支架→逐步撤除”的核心思想高度契合。习得编程技能需要大量阅读代码，范例支架可满足这一需求；从读懂代码到自行写出代码之间存在明显的能力要求差异，代码补全和改错支架可构成完整的桥梁。因此，将支架式教学应用于 Python 编程实验教学就非常契合。

3.2. 支架体系建设

根据以上论述，我们提出了“范例支架→代码补全支架→改错支架→独立编程”的四阶递进式实验支架框架，见图 1，四个层次由简到难，支架的支持程度逐渐减弱，在学生的最近发展区促进其不断提高自身水平。

第一阶段：范例支架。这一步骤的实验提供完整可执行代码以及详细注释。学生任务是运行代码、阅读代码、理解每一行代码作用之后，回答教师事先准备的代码理解问题。范例支架目的是让学生形成“正确代码应该如何编写”概念，减少他们对于编程陌生与畏惧心理。如在“函数定义与调用”实验里，老师会给出一个完整文本处理函数(包含参数传递、返回结果、异常处理等信息)，学生运行之后需要回答“这个

函数有几个参数？它返回什么样的结果？第 12 行代码的作用是什么？”等问题，而不是蜻蜓点水式浏览。

第二阶：代码补全支架。此阶段给出部分代码，删除主要部分(例如循环终止条件、函数返回值、列表下标等)，学生只需要在空缺处填入少量代码即可让程序正常工作，这正是对传统的“从零开始编程”的改进——把“编写整个程序”这个高认知负担的任务变为“理解上下文并填写关键代码”，让学生可以把精力集中在本次实验需要掌握的知识点上。比如，在“文件读写”实验中，老师给出完整的文件打开、异常处理以及关闭代码，挖空文件逐行读取以及字符串分割这部分内容，学生只需要完成 3~5 行的核心代码即可。这样的“减量不减质”，让学生在较短时间内由“看懂”到“写得出”。

第三阶：改错支架。此阶段给出包含典型错误的代码，让学生找到问题所在并改正。不同于代码补全，改错支架旨在培养学生调试能力——这是一项在编码过程中非常重要的但往往被忽略的能力[7]。改错题目中所涉及的错误包括初学者易犯的一些缩进错误、变量名拼写错误、类型不匹配以及逻辑条件错误等，这些都是老师多年总结的学生常出现的问题。如在“字典与数据统计”一节的教学活动中，给出一段有缩进层次问题、字典键名书写错误以及数值类型的代码让学生找出其中的问题，用 print 调试法找到问题所在并进行改正，同时记录下“找到了几个错误、都是什么错误”。这样有利于锻炼学生的调试能力，也有助于学生今后遇到问题能够自己解决。

第四阶：独立编程。此阶段除去所有支架，给定需求说明，让学生从头到尾自己动手写程序。独立编程一般放在课程末尾以及综合实训部分进行，这时的学生已经经过前三阶的学习掌握了大量代码阅读、片段编写以及调试的经验，有独立编程的能力。在学期末的综合实训里，让同学们自主开发一个“学生成绩统计分析程序”，包括输入成绩、求平均值、给成绩排名和输出等操作。可以全面考察学生的编程水平，也是支架式教学“去掉支架，放手让学生自己做”的目的之一。

需要特别指出的是，四阶递进式支架模型并非僵化的线性流程，而是具备动态调整的灵活性。在具体教学实践中，教师依托过程导向的评价体系(如代码补全的尝试次数、改错阶段的报错频率等过程性数据)，可以根据学生的实时表现动态增减支架的强度。例如，对于领悟较快、在“代码补全”阶段表现优异的学生，教师可允许其跳过“改错”阶段，直接进入“独立编程”挑战；而对于在“改错”阶段反复受挫、认知负荷依然过高的学生，教师则可动态插入“提示性子支架”(如高亮错误行、提供同类正确代码参考)或引入同伴互助，使其在撤除主干支架前获得额外支撑。这种基于学情的动态微调，确保了教学支持始终停留在学生的“最近发展区”内，避免了“一刀切”带来的拔苗助长或支撑不足，体现了该理论模型在应用中的灵活性与适应性。

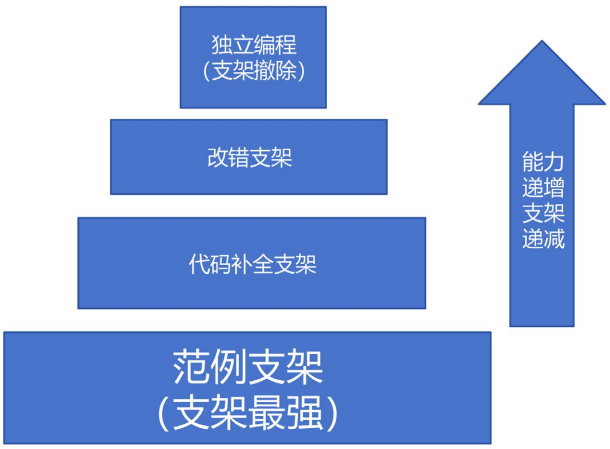

Figure 1. Schematic diagram of a four-stage progressive experimental scaffold model
图 1. 四阶递进式实验支架模型示意图

3.3. 过程性考核体系转变

为了适应四阶递进式实验教学需要，在实验考试方面也进行相应改革，由结果导向转变为过程导向[8]，即从以下三个方面进行：首先是对代码补全准确性和完整性(占实验成绩 30%)，考查学生对代码的理解程度以及能够写出合理代码的能力；其次是对改错实验中所找到错误及其解决过程(占 20%)，要求学生不仅能够指出问题所在而且要记录下自己是如何发现这个问题以及是如何解决这个问题的过程，以培养学生一种“发现问题–分析问题–解决问题”的思维方式；最后是独立编程中的代码质量和完成情况(占 50%)，全面反映学生编程水平。此外，在改错阶段加入同伴互评，学生互相检查对方程序，通过指出他人的问题而思考自己的程序，这有利于提高学生对程序进行检查的认识。

3.4. 题目场景 AI 化

为了提高学生对专业的认同感以及学习的积极性，我们对原有的实验题进行 AI 情景化改造，在一般的编程训练上加入与人工智能相关的工作内容。如表 1 所示，在保证基本知识点的基础上，将题目背景由普通的场景转变为 AI 相关的情境，让学生在编写代码的过程中体验到“所学即所用”，减少“不知道学这个能用来做什么”的迷茫。

Table 1. Comparison of traditional experiment questions and their AI-adapted versions
表 1. 传统实验题目与 AI 场景化改编对照

传统题目	AI 场景化改编
字符串处理练习	清洗文本数据(去停用词等)
列表排序练习	对数据集特征值进行排序分析
字典综合练习	构建词频统计与关键词提取工具
函数综合练习	实现简单的 KNN 分类器
文件读写练习	读取 CSV 格式的图像标注文件

AI 场景化设计的主要思想是“以实际问题引导学生进行编程”。当学生明白所编写字符串操作程序可用于进行文本数据清洗，所编写文件读写程序可用于载入真实数据集等，便会更有动力去学习，也更愿意参与到实验中。

4. 改革实践成效与反思

4.1. 实践成效

本文所提改进措施已经在作者所在学院的人工智能专业 2024 级大一学生们的 Python 程序设计课程上试行一个学期，该门课程有 80 个课时(理论课 48 个课时、实验课 32 个课时)，每星期安排一次实验课，在 PyCharm 本地 IDE 上进行。因为改革的时间较短，还没有得到足够的统计数据来比较，但是从没有进行改革之前的班级或者以前的 Python 程序设计课的教学感受以及同学们的意见来看，可以得出如下结论：

第一，学生实验参与度大大提高。以前，在实验课中后期有部分学生开始分神甚至放弃做题，但是现在特别是在代码补全以及纠错这部分内容上，大多数学生都可以在课堂上完成所有题目。一个明显的变化就是：以前实验课下课时学生都已走光，而现在有很多同学愿意留下来把剩下的题目做完，积极性较高，也更有成就感。

第二，编程畏难情绪显著降低。代码补全支架把“从零开始编写程序”分解成“理解框架、填写主要代码”，使初学者更容易接受。很多同学表示：“原来编程并没有想象中的那么困难”，由“不敢写”到

“愿意尝试”，这有利于今后的自主编程学习。

第三，调试意识开始建立。改错支架让学生把调试变成了“主动找问题”而不是“被动出错后解决”。学生在记录 Bug 修正过程时，逐渐养成了“先看报错信息、定位行号、分析原因”的调试习惯，而不再是以前那种“出现错误就换个写法尝试”的盲目试错方式。

4.2. 不足与反思

在取得一定成果的同时，改革过程中也出现了一些问题，在今后的教学中需要不断加以完善。

第一，分层次支架需要进一步完善。目前四阶递进式支架对所有同学均适用，但是每个班中学生的基础水平不同——有的有一定编程经验的同学认为代码补全过于简单，而基础较差的同学仍然觉得单独编程有难度。后期可以尝试分层次支架的设计，根据不同水平的学生给予不同的支架力度，做到因材施教。

第二，改错题库需要不断完善。目前改错实验中的 Bug 题目主要是老师的经验总结，数量较少且范围较窄。未来可设计一套学生编程错误收集的方法，在使用 PyCharm 进行代码检查以及程序运行过程中产生的异常信息的基础上不断丰富初学者常见的编程错误，形成按知识点分类以及按错误类型分类的改错题库。

第三，信息化教学手段不足。目前实验环节仍主要依靠人工批阅以及课上巡查进行指导，效率低下，在今后可以尝试使用一些在线编程测评系统(如 PTA、洛谷等)来进行代码补全或纠错类型的题目自动生成及即时反馈；另外也要注意研究大语言模型在教育领域的应用，寻求一种基于 AI 的个性化支架推送方法，让支架式教学由“统一支架”转变为“智能适配支架”。

5. 结语

编程学习不是一朝一夕的事情，特别是对于大一刚接触编程的学生来说，理解语法与自己动手写程序之间存在着较大的差距。本文以支架理论为基础，提出一种由“范例支架－代码补全支架－改错支架－独立编程”的四阶段梯度式实验教学方法，在不断减少支架支持下促使学生处在最近发展区中进行逐步提高编程技能。初步的教学效果显示，这种做法可以有效减轻学生的畏难心理，增加他们对实验的积极性以及完成实验的动力。支架式教学对于解决编程相关课程出现的“知而不做”的现象提供了切实可行的办法，而且这种方法也适用于 C 语言、Java 等其他编程课程的实验教学之中。

基金项目

等级：安徽省高等教育质量工程智慧；名称：数据库原理及应用；编号：2024aijy544。

参考文献

[1] 刘建明, 咸琳涛, 刘晓兰, 等. 程序设计类课程“个性协同化”智慧实验教学改革的探索[J]. 实验室研究与探索, 2023, 42(12): 179-183.

[2] 陈国权, 吴凡. 学习迁移的系统理论: PPEE 理论模型的建构和意义[J]. 中国管理科学, 2018, 26(9): 183-196.

[3] 卢慧, 巩政, 赵俊峰, 等. 人工智能与课程思政双向浸润的 Python 程序设计实验教学改革研究[J]. 软件导刊, 2025, 24(11): 54-59.

[4] 王艳芝, 张春莉. 学习支架何以促学——基于皮亚杰与维果茨基思想的综合视角[J]. 教育科学研究, 2024(11): 76-82.

[5] Sentance, S., Waite, J. and Kallia, M. (2019) Teaching Computer Programming with PRIMM: A Sociocultural Perspective. *Computer Science Education*, **29**, 136-176. https://doi.org/10.1080/08993408.2019.1608781

[6] Martin, F., Lee, I., Lytle, N., Sentance, S. and Lao, N. (2020) Extending and Evaluating the Use-Modify-Create Progression

for Engaging Youth in Computational Thinking. *Proceedings of the* 51*st ACM Technical Symposium on Computer Science Education*, Portland, 11-14 March 2020, 807-808. https://doi.org/10.1145/3328778.3366971

[7] 陈娟, 徐新海, 王学慧. 如何提高学生的实际编程能力[J]. 计算机工程与科学, 2014, 36(S2): 213-219.

[8] 谢全亮, 谢双全, 李鑫, 等. 细胞工程课程过程性考核教学改革实施与探索[J]. 社会科学前沿, 2023, 12(8): 4422-4428.