# 这不是我需要的反馈！ - 学生与 GenAI 反馈在导师 Kai 中的互动

Sven Jacobs
University of Siegen
Siegen, Germany
sven.jacobs@uni-siegen.de

Maurice Kempf
University of Siegen
Siegen, Germany
maurice.kempf@uni-siegen.de

Natalie Kiesler
Nuremberg Tech
Nuremberg, Germany
natalie.kiesler@th-nuernberg.de



## 摘要

生成式人工智能（GenAI）在计算教育中生成反馈的潜力一直是许多研究的主题。然而，关于计算机科学学生如何参与这种反馈以及它在多大程度上支持他们解决问题的研究仍然有限。出于这个原因，我们建立了一个定制的网络应用程序，为学生提供 Python 编程任务、代码编辑器、GenAI 反馈和编译器反馈。通过包括眼动追踪和对 11 名本科生进行的访谈协议，我们调查了（1）生成的反馈在多大程度上吸引了学习者的注意力以及（2）生成的反馈有多大帮助（或没有帮助）。此外，还将学生对 GenAI 反馈的注意力与他们对编译器反馈的关注进行了比较。我们进一步研究了有编程经验的学生和没有编程经验的学生之间的差异。研究结果表明，GenAI 反馈通常会吸引大量的视觉关注，没有经验的学生花费的眼动时间是两倍。更有经验的学生较少请求使用 GenAI，并且能更好地利用它来解决问题。对于没有经验的学生来说，理解 GenAI 反馈更具挑战性，因为他们并不总能理解这些反馈。他们经常只依赖于 GenAI 反馈，而忽视编译器反馈。了解学生对 GenAI 反馈的注意力和感知对于开发支持学习者学习的教育工具至关重要。

*UKICER ’ 25, September 04–05, 2025, Edinburgh, UK*



## CCS Concepts

• **Social and professional topics → Computer science education**.

## Keywords

编程教育，反馈，大型语言模型，生成式人工智能，GenAI

**ACM Reference Format:**

Sven Jacobs, Maurice Kempf, and Natalie Kiesler. 2025. 这不是我需要的反馈！ - 学生与 GenAI 反馈在导师 Kai 中的互动. In *Proceedings of 2025 Conference on United Kingdom & Ireland Computing Education Research (UKICER ’ 25)*. ACM, New York, NY, USA, 10 pages. https://doi.org/XXXXXXX.XXXXXXX

## 1 介绍

学习编程在认知上是复杂的，学生在整个学习过程中都需要支持，例如通过个性化和及时的反馈 [14]. 然而，这项任务对于教育者来说变得越来越具有挑战性，尤其是在大型大学课程中拥有多样化的学生成分时。尽管自 1960 年代以来就存在用于编程练习的自动化工具，但这些工具主要集中在识别错误上，而不是建议下一步行动或指导解决问题 [16]. 此外，有效的反馈必须考虑干预的时间、背景和学习者本身。不同能力水平的个体需要不同类型的支持，范围从基本的错误识别到战略问题解决指导 [31, 43].

生成式 AI（GenAI）工具基于大型语言模型（LLMs），为编程教育中的个性化反馈自动生成提供了有前景的

可能性，特别是在 [2, 3] 的背景下。然而，早期的研究表明，GenAI 工具经常提供任务和学生问题 [19, 20] 的即时代码解决方案，而不是提供学生们更倾向于独立学习的支持方法 [8]。最近对 GenAI 反馈的评估集中在诸如完整性和准确性这样的质量指标上，通常通过专家评估 [1, 9, 11, 19] 或使用额外模型进行比较分析 [22] 来衡量。Scholl et al. [41] 开始研究入门课程 [39, 41] 中的学生使用模式。然而，在初级编程教育中学生如何具体参与 GenAI 反馈的经验研究方面存在空白 [45]。

为了弥补这一差距，我们采用一种混合方法，结合眼动追踪和一种包括与**目标**调查学生对 GenAI 与编译器反馈的关注度，以及这些反馈对他们的帮助程度的半结构化访谈。我们还在研究不同的学生（缺乏经验与有经验的学习者）。因此，本研究**贡献**旨在增进对学生在入门编程背景下对 GenAI 反馈的感知和使用的理解。

## 2 相关工作

及时和个性化的反馈对编程教育至关重要 [14]。然而，在大规模实施这种反馈仍然是具有挑战性的。传统的自动化反馈系统通常关注简单的测试用例结果或编译器错误，而生成式人工智能已经作为生成自然语言反馈的有前景的方法出现。

在过去几年中，开发了各种定制的生成式 AI 工具。这些包括用于反馈和提示生成的工具如 CodeAid [15]、Codehelp [25]、SCRIPT [40]、LLM Hint Factory [47]、StAP-tutor [37] 或（上下文感知）聊天机器人如 CS50 Duck [26, 27] 和 CodeTutor [29]。例如，Kiesler et al. [19] 分析了 ChatGPT 对入门编程任务的反馈，展示了其提供详细解释和建议的能力，同时识别出输出之间的显著差异，并且还存在误导性信息。对 GPT-4 Turbo 的全面评估进一步提高了我们对 GenAI 反馈能力的理解 [1]。对学生提交的 55 次反馈分析显示，在结构和一致性方面有了显著改进 [1]，与早期模型相比。然而，只有 52% 的反馈是完全正确且完整的，强调了生成可靠反馈，特别是对初学者而言，仍然面临的持续挑战 [1]。

研究人员开发了更复杂的方法，基于这些初步发现来提高反馈质量。Phung et al. [33] 引入了一种双模型方法，结合 GPT-4 作为导师与 GPT-3.5 作为学生验证器。Jacobs and Jaschke [12] 使用检索增强生成将讲座材料纳入编程任务的 GenAI 反馈中。在他们的研究中，学生们根据问题的复杂性战略性地选择快速通用反馈或更详细的情境感知回应。为了满足对生成反馈进行更多控制的需求，Lohr et al. [28] 研究了基于既定反馈分类法 [16] 生成特定类型反馈的问题。通过迭代提示工程，他们展示了 GPT-4 能够在 66 个案例中的 63 个中生成所需类型的反馈。

尽管这些研究显著提升了我们对生成式人工智能（GenAI）反馈生成的理解，但在学生如何在问题解决过程中与 GenAI 互动方面仍存在一个关键的缺口。Prather et al. [34] 进行了一项混合方法研究，包括访谈、眼动追踪和对学生观察，涉及 21 名初学者程序员，以探讨使用 GenAI 工具在入门级编程中的益处和挑战。他们发现，一些学生能够有效地利用这些工具加速他们的工作，但另一些学生则产生了能力错觉 [35] 并面临新的困难。Prather et al. [34] 的研究确定了与 GenAI 相关的三种元认知困难：（1）频繁的 AI 建议导致中断，（2）误导性的代码建议，以及（3）虚假的进步意识。他们的发现表明，GenAI 工具可能会扩大准备充分和准备不足的学生之间的差距。其他研究还报告了 GenAI 的问题使用情况，尤其是对于没有先前知识的学习者 [21]。同样，Nam 等人 [30] 结论是，他们所开发的 GenAI 系统 “可能因参与者背景或技能的不同而有所差异” 的益处。这项研究进一步解决了 [45] 研究缺口，即理解学生如何在入门级编程教育中与 GenAI 反馈互动。

## 3 反馈生成与导师凯伊

为了系统地评估在编程教育环境中使用和感知到的生成式 AI 反馈的帮助性，我们开发了一个定制的网络应用程序——导师凯 [11–13]。该应用作为学生参与编程任务并在此研究过程中接收生成式 AI（以及编译器）反馈的界面。

## 3.1 网页界面

该网络应用包含四个兴趣区域（AoIs），如图 1 所示：

(1) 任务描述（左上）：为学生提供要解决的编程问题。
(2) 代码编辑器（右上角）：允许学生编写和编辑他们的代码。
(3) 生成 *AI* 反馈（左下）：显示生成的反馈。
(4) 编译器反馈（右下角）：显示代码编译的结果（编译器输出）。

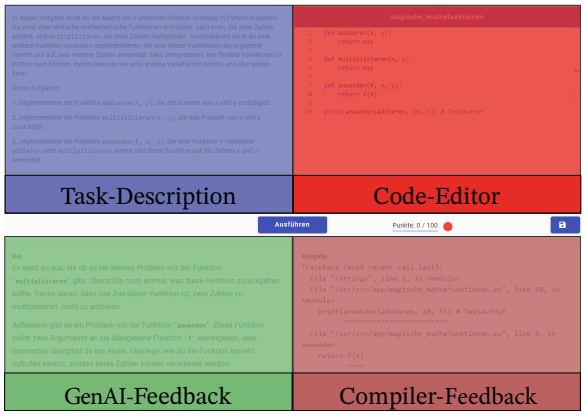


**图 1: 带有突出显示的 AoI 的网络应用**

在图 1 的中心，有一个执行代码的按钮。执行后，解决方案会自动针对编程任务的预定义单元测试进行测试。单元测试结果以点的形式显示。代码执行后，会出现一个额外的按钮。它允许学生请求反馈。每当代码更改时，反馈按钮将保持禁用状态，直到再次执行代码。这种设计有两个目的：(1) 鼓励学生在请求生成式 AI 反馈之前根据编译器反馈和单元测试结果进行改进；(2) 从技术上讲，执行结果是生成反馈所必需的。学生可以无限次地执行代码并请求生成式 AI 反馈。

## 3.2 生成 AI 反馈的提示工程

为了生成反馈，我们使用了 OpenAI 的"gpt-4-turbo-2024-04-09" 模型，并将温度参数设置为 0 以最大化输出一致性。提示工程过程对于确保相关且教育意义充足的反馈至关重要。我们基于先前的工作 [11] 迭代开发了提示。最终的提示 [10] 采用 Markdown 语法编写，包含全面的上下文信息，包括任务描述、学生的当前程序代码、编译器输出、单元测试结果以及预期长度（最多 6 句话）。我们采用了零样本提示方法结合角色和风格提示技术 [42]。

# 4 方法论

本研究由以下研究问题（RQs）指导：

RQ1 编程学习者对生成式人工智能反馈与编译器反馈的关注程度有何不同？
RQ2 *GenAI* 反馈在多大程度上对编程学习者有帮助？

对于这两个研究问题，我们特别调查有无编程经验的学生之间的差异。

结合眼动追踪与大声思考协议和半结构化访谈被选用来捕捉可观测的行为模式和自我报告的感知。眼动追踪技术提供了网络应用程序中注意力分配的客观衡量标准，而大声思考协议则为决策和问题解决过程提供了个人见解。访谈允许进行回顾性反思和澄清。这些方法的三角互证支持对学生对所提供反馈的全面理解。

访谈指南、编程任务、思维 aloud 指示、用于生成 AI 反馈的提示、生成 AI 和编译器反馈示例可以在在线仓库 [10] 中获得。

## 4.1 数据收集

眼动追踪技术 [5, 6] 被用于测量不同兴趣区域（AoIs）上视觉注意力的分布。为了研究 GenAI 和编译器反馈对经验不足和有经验的学习者的帮助，我们使用了思考 aloud 方法 [44]。该方法常用于了解认知过程中的情况，例如调试 [46]、代码重构 [7] 或反馈处理 [18]。

*4.1.1* **参与者**. 研究于 2024 年夏季学期初在一所大学进行。参与者是选修"面向对象和函数式编程"（OOFP）课程的学生，他们通常是计算机科学专业的一、二年级学生。所有课程参与者（约 200 人）均通过电子邮件及辅导课上的现场公告收到邀请。参与研究的报酬为 20€。

*4.1.2* **任务设计**. 六个 Python 编程任务是专门为本研究开发的。这些任务经过精心设计，以符合课程大纲

并逐步增加复杂性，涵盖了变量操作、控制结构、循环和递归等基本编程概念（所有任务均可在我们补充数据存储库 [10] 中获得）。为确保适当的难度水平和时间长度，所有的任务都先由两名已经完成该课程的学生导师进行了试做。他们大约用了 40 分钟完成了所有任务。

*4.1.3* **设置**. 研究在一个配备 27 英寸显示器和标准输入设备（鼠标和键盘）的受控实验室环境中进行。主要的眼动追踪装置包括 Tobii Pro Glasses 3，辅以 Tobii Pro Lab 分析软件。该设置基于其在编程教育研究中的可靠性而被选中，Tobii 技术在这类研究中有 55%的应用。[32] 为了确保数据收集的连续性，我们使用 Tobii Eye Tracker 5 实施了一个备份系统。所有参与者在接受有关数据保护、记录程序和数据处理的简报后均表示同意。

*4.1.4* **程序**. 每位参与者被安排进行为期两小时的实验，实验结构如下：

(1) 引言（15-30 分钟）：
- 思声协议指令和练习
- 眼动追踪设备熟悉和校准
- 告知同意过程

(2) 任务阶段/问题解决（60 分钟）：
- 编程任务的完成
- 连续思考出声陈述
- 目标追踪和屏幕录制
- 正面和侧面角度的视频和音频录制

(3) 面试阶段（15-30 分钟）：
- 观测行为的澄清
- 经验的反映

在整个环节中，有一名指导教师在场以解决技术问题并在需要时提供组织指导，同时确保在解决问题过程中最少的干预。

## 4.2 数据分析

RQ1 通过分析每个学生的眼动追踪数据并将其映射到兴趣区域来解决。为了回答 RQ2，我们将眼动追踪数据（以及相应的 AoIs）与学生的解决问题步骤进行三角测量，以查看 GenAI 反馈或编译器消息是否确实有助于他们完成任务。我们进一步分析了 GenAI 反馈没有帮助的情况。半结构化访谈增加了学生对 GenAI 反馈的自我报告的帮助程度。

*4.2.1* **眼动分析**. 为了分析学生的注意力模式，我们基于 Tutor Kai 的关键界面元素定义了四个兴趣区域 (AoI)：任务描述、代码编辑器、GenAI 反馈和编译器反馈（参见图 1）。我们使用 Tobii Pro Lab 分析软件的辅助映射功能，将来自眼动追踪眼镜视频录制的凝视数据映射到这些 AoI。对于参与者 S01、S02 和 S10，设备兼容性问题需要使用 Tobii Eye Tracker 5 作为 Tobii Pro Glasses 3 的替代品。对于他们，我们使用逐帧分析手动映射他们的凝视数据。我们分析了所有参与者在每个 AoI 上的注视时间 [32]，从而提供注视分布的定量度量。为了进一步分析，我们导出了凝视数据录制，并将其与屏幕截图和外部摄像头录像同步，为每个参与者创建同步视频，以便进行详细分析（即凝视可视化）。

请注意，”aloud” 未进行翻译是因为它可能是特定术语或有特殊含义，在没有更多上下文的情况下保持原样。如果需要进一步调整，请告知具体要求。] 思维 aloud 协议分析与眼动追踪的三角互证法分析

请注意，”aloud” 未进行翻译是因为它可能是特定术语或有特殊含义，在没有更多上下文的情况下保持原样。如果需要进一步调整，请告知具体要求。首先，对音频进行转录，以便对思维导向研究进行详细分析。然后，将转录文本与同步视频链接起来，以便在 MAXQDA [23] 中进行评估。所有在任务阶段期间执行的代码实例或反馈请求均已标记。然后，我们分析了在生成反馈之前学生注视的 AoI，以确定他们正在查看的内容（即，编译器反馈、任务描述等）。

作为下一步，学生的进步被编码以反映他们的改进（或缺乏改进），以确定反馈的客观有效性并回答 RQ2。我们使用了以下类别（数据来源用括号表示）：

**已帮助：**编译器或生成式 AI 反馈中的信息被读取（注视可视化）。随后，要么明确表示该信息有帮助的思想被口头表达出来（大声思考），或者代码以直接与

提供的信息相关的方式得到改进。例如，参与者阅读了生成式 AI 的反馈 “[...] 你的函数还没有实现返回语句 [...]” 并口头表达了 “当然 [...] 返回 [...]” 同时纠正了代码。即使没有口头表达，由于注视可视化确认参与者在添加返回语句之前确实读取了相关的生成式 AI 反馈部分，此实例也会被编码为**有帮助的**。

**未提供帮助：**编译器或生成式 AI 反馈中的信息被阅读（凝视可视化）。参与者没有说出任何表明该信息有所帮助的内容（思考声化）。与提供的信息相关的代码改进并未进行，例如，一些学生读了编译器的反馈但立即要求生成式 AI 反馈而不是据此行动。为了未来改进反馈生成，我们也对为什么生成式 AI 反馈无帮助进行了分类。

**尚未阅读：**编译器或生成式 AI 反馈中的信息未被阅读（注视可视化）。例如，在读取编译器反馈之前就请求了生成式 AI 的反馈，或者在生成式 AI 反馈生成过程中请求了反馈。或者学生独立进行了改进，并立即请求了新的生成式 AI 反馈。

*4.2.2* **面试分析**. 我们进行了半结构化访谈，以获得更多关于参与者对编译器和生成式 AI 反馈的有用性的看法。完整的访谈指南详见 [10]。特别是，我们希望让学生评定（1）任务难度（采用 10 点李克特量表，从非常容易到非常难）和（2）生成式 AI 反馈的帮助程度（同样采用 10 点李克特量表，从完全无帮助到非常有帮助）。响应进行了定量分析并报告。此外，我们评估了参与者在回应特定问题时明确表达的思维声化协议和眼动追踪设备的干扰情况。最后，我们分析了学生是否具有任何先前编程经验，从而得出二元区分。

# 5 结果

## 5.1 学生样本

11 名学生自愿参与并完成了这项研究（S01-S11）。为本研究招募的学生都熟悉该工具，因为它在 OOFP 课程中经常被使用。7 人自认为是男性，4 人为女性。他们的平均年龄为 21 岁（标准差=2.79）。对学生先前编程经验的分析迅速揭示了招募的学生中有新手和有经验者（如预期的那样）。6 名学生是缺乏经验的，报告几乎没有或完全没有先前的编程经验。其余 5 名学生被归类为经验丰富的，因为他们报告有先前的编程经验。这包括通过高中计算机科学课程或在之前的学期中完成与编程相关的课程而接触过编程。

应注意到，OOFP 课程的注册学生在先前知识方面通常具有多样性。课程内的初步调查（N=97）显示，有 43 名学生（大约一半）没有先前的知识。因此，我们假设我们的样本具有代表性，即代表了这种多样性。

我们的分析显示了这两组经验在任务完成率和感知难度水平方面存在明显差异。有经验的学生平均完成了 4.2 个任务，而没有经验的学生则只完成了 1.8 个任务。通过使用 10 分李克特量表（1=非常简单，10=非常困难）进行半结构化访谈收集到的学生对任务难度的主观评估与这些绩效差异一致：有经验的学生将任务评为中等挑战性（M = 4.2），而没有经验的参与者则持续报告更高的难度水平（M = 7.0）。因此，这些数据证实了学生自我报告的先前编程经验（或缺乏经验）。

## 5.2 RQ1: 学生的注意力分布

代码编辑器（53.91%）获得了最长的注视时间，这意味着在所有 11 名学生中都是如此。与其他兴趣区相比，生成式 AI 反馈占总注视时间的 23.79%，这比接收到的编译器反馈时间长了三倍多（7.00%）。与任务描述（15.29%）相比，对生成式 AI 反馈的注视时间也更高。比较分析显示，有经验的学生花费了大约 15.49%的时间在 GenAI 反馈上，这大约是缺乏经验学生所花时间（30.71%）的一半。对于编译器反馈，则呈现相反的模式，有经验的学生注视时间几乎是缺乏经验学生的两倍（9.35%对 5.05%）。关于任务描述和代码编辑区域，有经验的学生比缺乏经验的学生给予了更多的关注（见图 2）。

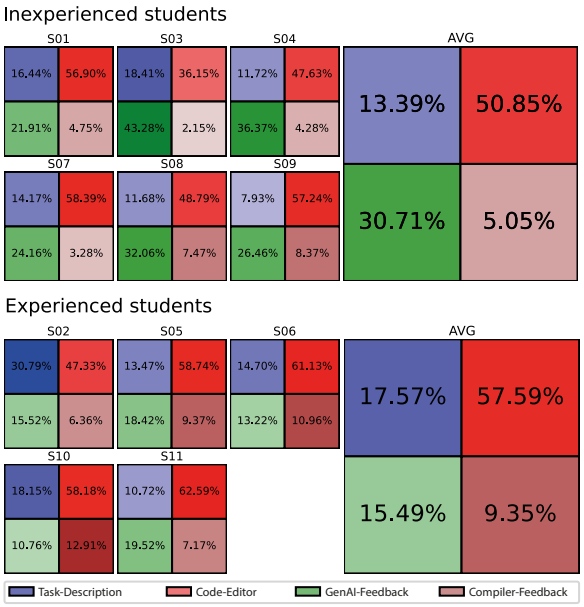


**图 2: 兴趣区域的固定时间**

**表 1: 固定 AoI 在 GenAI 反馈请求之前**

| | Task-Description | Code-Editor | Compiler-Feedback |
|---|---|---|---|
| Inexperienced students | 0 (0%) | 68 (63.55%) | 39 (36.45%) |
| Experienced students | 1 (1.56%) | 19 (29.69%) | 44 (68.75%) |
| All students | 1 (0.58%) | 87 (50.88%) | 83 (48.54%) |

## 5.3 RQ2: 生成 AI 反馈的 helpfulness（有用性）

首先，我们观察了生成反馈之前的注视区域 AoI，以确定学生正在看什么（见表 1）。代码编辑器和编译器反馈分别在大约 50%的 171 次 GenAI 反馈请求前被注视到。任务描述仅被注视了一次。不熟练的学生比熟练的学生更少地注视编译器反馈（36.45% 对比 68.75%）。

有 171 个生成 AI 反馈输出，其中 107 个针对无经验学生，64 个针对有经验学生。其中 49.7%的反馈有助于推进解决问题的过程并正确完成任务（参见图 3）。在 287 条编译器反馈消息中，信息的帮助性显著较低（总体仅为 17.4%）。

我们的分析显示这两组学生之间存在显著差异。如图 3 所示，对于有经验的学生（60.9%），GenAI 反馈更有帮助，而相对于没有经验的学生（43.0%）来说则不那么明显。有经验的学生平均请求的 GenAI 反馈少 28%（总共只有 64 次），相比之下，没有经验的学生提出了 107 次 GenAI 反馈请求。同步思维 aloud 协议的分析显示，对于有经验的学生，编译器反馈几乎是四倍更有帮助（25.9%）比没有经验的学生（7%）。眼球追踪数据证实了，在 129 条编译器反馈消息中，39.5%未被没有经验的学生阅读（图 3 的左下象限）。AoIs 上的固定时间进一步证明，没有经验的学生在编译器反馈上的停留时间只有有经验学生的一半（见图 2）。与其先阅读编译器反馈，没有经验的学生经常立即请求 GenAI 反馈，这解释了为什么 GenAI 反馈请求的数量更高。

**表 2: 半结构化访谈的结果**

| | 缺乏经验的学生 | | | | | | 有经验的学生 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | S01 | S03 | S04 | S07 | S08 | S09 | S02 | S05 | S06 | S10 | S11 |
| 已完成任务数（共 6 项） | 1 | 2 | 2 | 2 | 2 | 2 | 5 | 4 | 5 | 4 | 3 |
| 李克特量表项目（1-10）： | | | | | | | | | | | |
| 感知任务难度 | 5 | 8 | 7 | 7 | 8 | 7 | 1 | 6 | 5 | 4 | 4 |
| 反馈评分 | 8 | 7.5 | 10 | 10 | 10 | 8 | 9 | 8 | 7 | 7.5 | 10 |
| 先前的编程经验 | ○ | ○ | ○ | ○ | ○ | ○ | ● | ● | ● | ● | ● |
| 反馈： | | | | | | | | | | | |
| ...was easy to understand | ● | ● | ● | ● | ● | ○ | ● | ● | ● | ● | ● |
| ...was sometimes too vague | ● | ● | ● | ● | ● | ● | ○ | ● | ● | ● | ○ |
| ...had too complex vocabulary | ○ | ○ | ● | ● | ● | ○ | ○ | ● | ○ | ○ | ● |
| ...increased motivation | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● |

我们对 77 个被归类为“尚未帮助”的 GenAI 反馈实例进行了分析，发现了四个问题类别：(1) 25 例包含各种不同的原因，这些原因阻碍了进一步明确的分类，例如学生对反馈的误解。(2) 在 23 个实例中（其中 22 个是不熟练的学生，1 个是有经验的学生），学生无法理解该反馈。这种情况发生在 GenAI 反馈引用了对学生而言不够熟悉的某些基础概念知识（如“循环”）时，而没有提供足够的解释或示例来说明。(3) 在 17 个实例中（其中 12 个是不熟练的学生，5 个是有经验的学生），学生理解了解决方案的含义，但对所需的 Python 语法不熟悉，而这正是 GenAI 反馈未能提供的内容（例如通过代码示例）。他们表示这并不是他们需

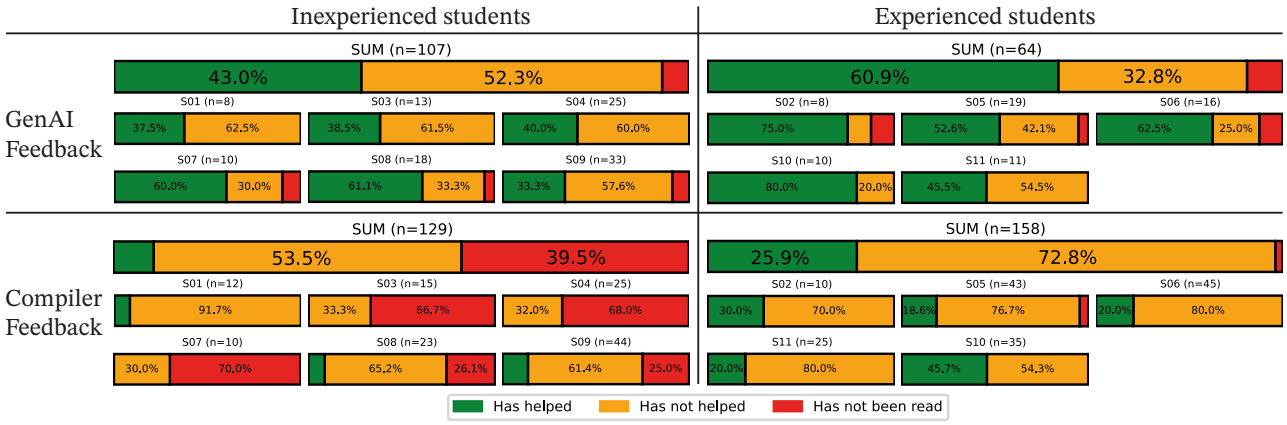


**图 3: 经验丰富的学生和缺乏经验的学生从 GenAI 和编译器反馈中获得的帮助效果（RQ2）**

要的反馈。(4) 在 13 个实例中，GenAI 反馈中的错误定位缺乏足够的精确度（例如，指出了缩进错误，但对于学生来说仍然难以识别这些错误）。

学生对生成式 AI 反馈的整体感知是积极的。在访谈中，学生们在其帮助性方面给出了 8.59 分（满分为 10 分的李克特量表，1=完全没有帮助，10=非常有帮助，见表 2）。这个高评分在两个经验组中都是一致的，没有任何参与者给反馈打分低于 7（见表 2）。然而，对大声思考协议的深入分析以及与眼动追踪数据的三角验证表明，感知的帮助性与实际帮助性的关系更为复杂。关于学生反思的其余访谈结果有些不确定（见表 2）。十一位中有十位学生描述生成式 AI 反馈通常是可以理解的，但有时也不够精确。五名学生提到，生成式 AI 反馈有时使用了过于复杂的专业词汇。所有学生表示，生成式 AI 反馈提高了他们解决编程任务的动力。

## 6 讨论

我们关于生成式 AI 反馈的发现反映出计算教育领域由于生成式 AI 工具的出现而产生的更广泛模式。在利用生成式 AI 反馈方面，有经验与无经验的学生之间存在的差距具有重要意义。最近的研究已经报告了引入生成式 AI 工具与入门编程课程中失败率急剧增加之间的巧合 [21]。这种结果上的分歧与观察相符：一些学生能够有效地使用生成式 AI 来“加速”他们的编程工作，而另一些学生则可能形成一种“能力错觉”。AI 工具似乎引入了额外的元认知困难，而学生们通常对此并不了解 [34]。

此外，结果表明，缺乏经验的学生通常不会阅读编译器反馈。相反，他们阅读了 GenAI 的反馈。这并不令人惊讶，因为编程错误消息对学生来说通常是问题重重的——尽管它们作为反馈代理发挥着重要作用 [3]。如果这些缺乏经验的学生继续使用这种策略，他们可能无法学会正确解释编译器输出。同时，有人可能会认为，由于 GenAI 支持无处不在，解释编译器输出在未来可能不再必要。毕竟，编译器消息可以直接由 LLM[24, 38] 解释。虽然 Reeves et al. [36] 建议通过 GenAI 进行自然语言编程作为编程抽象的下一步，但我们的结果显示学生仍然需要发展基本的编程知识和技能来理解 GenAI 反馈。

将 GenAI 成功整合到编程教育中可能需要一种平衡的方法，结合传统基础学习与逐步接触由 GenAI 辅助的编程技术。在这种情况下，Keuning et al. [17] 建议采用掌握式学习方法，使学生在发展基础技能的同时逐渐融入 GenAI 工具。最终，我们需要考虑学生应该培养哪些能力。

## 7 有效性威胁

本研究的方法论局限性应予注意。例如，思维 aloud 协议和眼动追踪技术可能会干扰问题解决过程。在访谈

中，三名学生报告说 think-aloud 协议具有干扰性，指出难以表达他们的想法或选择要表达的想法感到负担。观察设置本身影响了一些参与者的舒适度水平。四名学生表示，在被观察时对他们自己的代码和思维感到尴尬。关于眼动追踪设备，两名学生经历了物理干扰：一名报告说眼镜在使用过程中变暖，另一名报告说由眼镜传感器引起的干扰。技术局限性是由于用于 GenAI 反馈生成的非确定性大语言模型；LLMs 的输出生成仍然不透明 [42]。研究性质是定性和探索性的。通过样本量为 11 名学生和 171 条 GenAI 反馈，我们已经达到了足够大的样本量 [4]。

## 8 结论

在这项研究中，我们评估了 11 名学生在使用 Tutor Kai 解决编程任务时如何利用由生成式 AI 产生的反馈。我们的混合方法结合了眼动追踪、大声思考协议和访谈，揭示了以下结果：

(1) 学生将他们视觉注意力的相当大一部分（23.79%）集中在了生成式 AI 反馈上。然而，不同经验水平之间存在显著差异。没有经验的学生在生成式 AI 反馈上的注视时间几乎是更有经验的同学的两倍。
(2) 经验丰富的学生展示了更有效地利用了生成式 AI 的反馈。他们较少频繁地请求反馈，并从中获得了更多的益处，有 60.9%的生成式 AI 反馈帮助推进了他们的问题解决能力，相比之下，经验不足的学生仅有 43.0%的比例。其中一个原因是经验不足的学生无法理解他们收到的 20.6%（107 个案例中的 22 个）生成式 AI 反馈，而经验丰富的学生只有 1.6%（64 个案例中的 1 个）。
(3) 不经验丰富的学生经常完全忽略编译器的反馈，仅仅依赖于 GenAI 的反馈。这引发了对传统调试技能发展可能产生影响的担忧。

我们的研究结果强调了 GenAI 反馈在编程教育中的潜力以及基于学生经验制定个性化方法的必要性。先前的研究 [8] 表明这对学生也很重要。未来的研究应该集中在开发能够根据学生的先验知识和经验提供适应性反馈的系统上。此外，它们还应能适应学生个人的信息需求。如本研究所示，有经验和无经验的学生似乎以不同的方式与编译器和 GenAI 反馈互动。因此，这项研究应在更多的参与者中重复进行。更重要的是，我们需要为具有不同编程知识背景的不同学生群体开发教学方法，使他们能够学会如何利用编译器和 GenAI 反馈。

## References


[1] Imen Azaiz, Natalie Kiesler, and Sven Strickroth. 2024. Feedback-Generation for Programming Exercises With GPT-4. In *Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1* (Milan, Italy) *(ITiCSE 2024)*. ACM, New York, USA, 31 – 37. https://doi.org/10.1145/3649217.3653594

[2] Brett A Becker, Michelle Craig, Paul Denny, Hieke Keuning, Natalie Kiesler, Juho Leinonen, Andrew Luxton-Reilly, James Prather, and Keith Quille. 2024. Generative AI in Introductory Programming. In *Computer Science Curricula 2023*. ACM, New York, USA, 438–439.

[3] Brett A. Becker, Paul Denny, James Finnie-Ansley, Andrew Luxton-Reilly, James Prather, and Eddie Antonio Santos. 2023. Programming Is Hard - Or at Least It Used to Be: Educational Opportunities and Challenges of AI Code Generation. In *Proc. of the 54th ACM Techn. Symp. V. 1*. ACM, New York, 500–506. https://doi.org/10.1145/3545945.3569759

[4] Clive Roland Boddy. 2016. Sample size for qualitative research. *Qualitative market research: An international journal* 19, 4 (2016), 426–432.

[5] Teresa Busjahn, Carsten Schulte, and Andreas Busjahn. 2011. Analysis of Code Reading to Gain More Insight in Program Comprehension. In *Proceedings of the 11th Koli Calling International Conference on Computing Education Research*. ACM, Koli Finland, 1–9. https://doi.org/10.1145/2094131.2094133

[6] Teresa Busjahn, Carsten Schulte, Bonita Sharif, Simon, Andrew Begel, Michael Hansen, Roman Bednarik, Paul Orlov, Petri Ihantola, Galina Shchekotova, and Maria Antropova. 2014. Eye Tracking in Computing Education. In *Proceedings of the Tenth Annual Conference on International Computing Education Research*. ACM, 3–10. https://doi.org/10.1145/2632320.2632344

[7] Eduardo Carneiro Oliveira, Hieke Keuning, and Johan Jeuring. 2024. Investigating Student Reasoning in Method-Level Code Refactoring: A Think-Aloud Study *(Koli Calling '24)*. ACM, New York. https://doi.org/10.1145/3699538.3699550

[8] Paul Denny, Stephen MacNeil, Jaromir Savelka, Leo Porter, and Andrew Luxton-Reilly. 2024. Desirable Characteristics for AI Teaching Assistants in Programming Education. In *Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1*. ACM, 408–414. https://doi.org/10.1145/3649217.3653574

[9] Arto Hellas, Juho Leinonen, Sami Sarsa, Charles Koutcheme, Lilja Kujanpää, and Juha Sorva. 2023. Exploring the Responses of Large Language Models to Beginner Programmers' Help Requests. In *Proceedings of the 2023 ACM Conference on International Computing Education Research - Volume 1*. ACM, New York, USA, 93–105. https://doi.org/10.1145/3568813.3600139

[10] Sven Jacobs. 2025. Data: That' s Not the Feedback I Need! - Student Engagement with GenAI Feedback in the Tutor Kai. https://doi.org/10.

17605/OSF.IO/US9HJ

[11] Sven Jacobs and Steffen Jaschke. 2024. Evaluating the Application of Large Language Models to Generate Feedback in Programming Education. In *2024 IEEE Global Engineering Education Conference (EDUCON)*. IEEE. https://doi.org/10.1109/EDUCON60312.2024.10578838

[12] Sven Jacobs and Steffen Jaschke. 2024. Leveraging Lecture Content for Improved Feedback: Explorations with GPT-4 and Retrieval Augmented Generation. In *2024 36th International Conference on Software Engineering Education and Training*. IEEE. https://doi.org/10.1109/CSEET62301.2024.10663001

[13] Sven Jacobs, Henning Peters, Steffen Jaschke, and Natalie Kiesler. 2025. Unlimited Practice Opportunities: Automated Generation of Comprehensive, Personalized Programming Tasks. In *Proceedings of the 30th ACM Conference on Innovation and Technology in Computer Science Education V. 1* (Nijmegen, Netherlands) *(ITiCSE 2025)*. ACM, New York, 319 – 325. https://doi.org/10.1145/3724363.3729089

[14] Johan Jeuring, Hieke Keuning, Samiha Marwan, Dennis Bouvier, Cruz Izu, Natalie Kiesler, Teemu Lehtinen, Dominic Lohr, Andrew Peterson, and Sami Sarsa. 2022. Towards Giving Timely Formative Feedback and Hints to Novice Programmers. In *Proceedings of the 2022 Working Group Reports on Innovation and Technology in Computer Science Education*. ACM, New York, USA, 95–115. https://doi.org/10.1145/3571785.3574124

[15] Majeed Kazemitabaar, Runlong Ye, Xiaoning Wang, Austin Zachary Henley, Paul Denny, Michelle Craig, and Tovi Grossman. 2024. CodeAid: Evaluating a Classroom Deployment of an LLM-based Programming Assistant that Balances Student and Educator Needs. In *Proc. of the CHI Conference on Human Factors in Computing Systems*. ACM, New York. https://doi.org/10.1145/3613904.3642773

[16] Hieke Keuning, Johan Jeuring, and Bastiaan Heeren. 2019. A Systematic Literature Review of Automated Feedback Generation for Programming Exercises. *ACM ToCE* 19 (2019), 1–43. https://doi.org/10.1145/3231711

[17] Hieke Keuning, Andrew Luxton-Reilly, Claudia Ott, Andrew Petersen, and Natalie Kiesler. 2024. Goodbye Hello World - Research Questions for a Future CS1 Curriculum. In *Proceedings of the 24th Koli Calling International Conference on Computing Education Research*. ACM, Koli Finland, 1–2. https://doi.org/10.1145/3699538.3699591

[18] Natalie Kiesler. 2023. Investigating the Use and Effects of Feedback in CodingBat Exercises: An Exploratory Thinking Aloud Study. *2023 Future of Educational Innovation-Workshop Series Data in Action: Digital Ecosystem and Emerging Tools for Education* (2023), 1–12. https://doi.org/10.1109/IEEECONF56852.2023.10104622

[19] Natalie Kiesler, Dominic Lohr, and Hieke Keuning. 2023. Exploring the Potential of Large Language Models to Generate Formative Programming Feedback. In *2023 IEEE Frontiers in Education Conference (FIE)*. IEEE, College Station, USA, 1–5. https://doi.org/10.1109/FIE58773.2023.10343457

[20] Natalie Kiesler and Daniel Schiffner. 2023. Large Language Models in Introductory Programming Education: ChatGPT's Performance and Implications for Assessments. https://doi.org/10.48550/arXiv.2308.08572 arXiv:2308.08572 [cs.SE]

[21] Natalie Kiesler, Ingo Scholz, Jens Albrecht, Friedhelm Stappert, and Uwe Wienkop. 2024. Novice Learners of Programming and Generative AI - Prior Knowledge Matters. In *Proceedings of the 24th Koli Calling International Conference on Computing Education Research*. ACM, Koli Finland, 1–2. https://doi.org/10.1145/3699538.3699580

[22] Charles Koutcheme, Nicola Dainese, Arto Hellas, Sami Sarsa, Juho Leinonen, Syed Ashraf, and Paul Denny. 2024. Evaluating Language Models for Generating and Judging Programming Feedback. https://doi.org/10.48550/arXiv.2407.04873

[23] Udo Kuckartz and Stefan Rädiker. 2019. *Analyzing qualitative data with MAXQDA*. Springer.

[24] Juho Leinonen, Arto Hellas, Sami Sarsa, Brent Reeves, Paul Denny, James Prather, and Brett A. Becker. 2023. Using Large Language Models to Enhance Programming Error Messages. In *Proc. of the 54th ACM Technical Symposium on Computer Science Education V. 1*. ACM, 563–569. https://doi.org/10.1145/3545945.3569770

[25] Mark Liffiton, Brad E Sheese, Jaromir Savelka, and Paul Denny. 2024. CodeHelp: Using Large Language Models with Guardrails for Scalable Support in Programming Classes. In *Proceedings of the 23rd Koli Calling International Conference on Computing Education Research*. ACM, New York, USA. https://doi.org/10.1145/3631802.3631830

[26] Rongxin Liu, Carter Zenke, Charlie Liu, Andrew Holmes, Patrick Thornton, and David J. Malan. 2024. Teaching CS50 with AI: Leveraging Generative Artificial Intelligence in Computer Science Education. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*. 750 – 756. https://doi.org/10.1145/3626252.3630938

[27] Rongxin Liu, Julianna Zhao, Benjamin Xu, Christopher Perez, Yuliia Zhukovets, and David J. Malan. 2025. Improving AI in CS50: Leveraging Human Feedback for Better Learning. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1* (Pittsburgh, PA, USA) *(SIGCSETS 2025)*. ACM, New York, NY, USA, 715 – 721. https://doi.org/10.1145/3641554.3701945

[28] Dominic Lohr, Hieke Keuning, and Natalie Kiesler. 2025. You're (Not) My Type- Can LLMs Generate Feedback of Specific Types for Introductory Programming Tasks? *Journal of Computer Assisted Learning* 41, 1 (2025), e13107. https://doi.org/10.1111/jcal.13107 e13107 JCAL-24-434.R1.

[29] Wenhan Lyu, Yimeng Wang, Tingting Rachel Chung, Yifan Sun, and Yixuan Zhang. 2024. Evaluating the Effectiveness of LLMs in Introductory Computer Science Education: A Semester-Long Field Study. *arXiv:2404.13414* (2024).

[30] Daye Nam, Andrew Macvean, Vincent Hellendoorn, Bogdan Vasilescu, and Brad Myers. 2024. Using an LLM to Help With Code Understanding. In *Proc. of the IEEE/ACM 46th ICSE*. ACM. https://doi.org/10.1145/3597503.3639187

[31] Susanne Narciss. 2008. Feedback strategies for interactive learning tasks. *Handbook of research on educational communications and technology* 3 (2008), 125–144.

[32] Unaizah Obaidellah, Mohammed Al Haek, and Peter C.-H. Cheng. 2019. A Survey on the Usage of Eye-Tracking in Computer Programming. *Comput. Surveys* 51, 1 (Jan. 2019), 1–58. https://doi.org/10.1145/3145904

[33] Tung Phung, Victor-Alexandru Pădurean, Anjali Singh, Christopher Brooks, José Cambronero, Sumit Gulwani, Adish Singla, and Gustavo Soares. 2024. Automating Human Tutor-Style Programming Feedback: Leveraging GPT-4 Tutor Model for Hint Generation and GPT-3.5 Student Model for Hint Validation. In *Proceedings of the 14th Learning Analytics and Knowledge Conference*. ACM, Kyoto, Japan, 12–23. https://doi.org/10.1145/3636555.3636846

[34] James Prather, Brent N Reeves, Juho Leinonen, Stephen MacNeil, Arisoa S Randrianasolo, Brett A. Becker, Bailey Kimmel, Jared Wright, and Ben Briggs. 2024. The Widening Gap: The Benefits and Harms of Generative AI for Novice Programmers. In *Proc. of ICER 2024*, Vol. 1. ACM, New York, 469–486. https://doi.org/10.1145/3632620.3671116

[35] Rajendra Raj, Mihaela Sabin, John Impagliazzo, David Bowers, Mats Daniels, Felienne Hermans, Natalie Kiesler, Amruth N. Kumar, Bonnie MacKellar, Renée McCauley, Syed Waqar Nabi, and Michael Oudshoorn. 2021. Professional Competencies in Computing Education: Pedagogies and Assessment. In *Proceedings of the 2021 Working Group Reports on Innovation and Technology in Computer Science Education*. 133 – 161. https://doi.org/10.1145/3502870.3506570

[36] Brent N. Reeves, James Prather, Paul Denny, Juho Leinonen, Stephen MacNeil, Brett A. Becker, and Andrew Luxton-Reilly. 2024. Prompts First, Finally. https://doi.org/10.48550/arXiv.2407.09231

[37] Lianne Roest, Hieke Keuning, and Johan Jeuring. 2023. Next-Step Hint Generation for Introductory Programming Using Large Language Models. In *Proceedings of the 26th Australasian Computing Education Conference*. ACM, Sydney, Australia, 144–153. https://doi.org/10.1145/3636243.3636259

[38] Eddie Antonio Santos and Brett A. Becker. 2024. Not the Silver Bullet: LLM-enhanced Programming Error Messages are Ineffective in Practice. In *Proceedings of the 2024 UKICER*. ACM, Manchester United Kingdom, 1–7. https://doi.org/10.1145/3689535.3689554

[39] Andreas Scholl and Natalie Kiesler. 2024. How Novice Programmers Use and Experience ChatGPT when Solving Programming Exercises in an Introductory Course. In *2024 IEEE Frontiers in Education Conference (FIE)*. 1–9. https://doi.org/10.1109/FIE61694.2024.10893442

[40] Andreas Scholl and Natalie Kiesler. 2025. SCRIPT - Supportive Chatbot for Resolving Introductory Programming Tasks. In *Proceedings of the 30th ACM Conference on Innovation and Technology in Computer Science Education V. 2*. ACM, New York, 759. https://doi.org/10.1145/3724389.3730786

[41] Andreas Scholl, Daniel Schiffner, and Natalie Kiesler. 2024. Analyzing Chat Protocols of Novice Programmers Solving Introductory Programming Tasks with ChatGPT. In *Proceedings of DELFI 2024*, Sandra Schulz and Natalie Kiesler (Eds.). 63–79. https://doi.org/10.18420/delfi2024_05

[42] Sander Schulhoff, Michael Ilie, Nishant Balepur, Konstantine Kahadze, Amanda Liu, and Chenglei et al. Si. 2024. The Prompt Report: A Systematic Survey of Prompting Techniques. https://doi.org/10.48550/arXiv.2406.06608

[43] Valerie J. Shute. 2008. Focus on formative feedback. *Review of Educational Research* 78, 1 (2008).

[44] Paul Solomon. 1994. *The Think Aloud Method: A Practical Guide to Modelling Cognitive Processes*. Academic Press, London.

[45] Irene Stone. 2024. Exploring Human-Centered Approaches in Generative AI and Introductory Programming Research: A Scoping Review. In *Proceedings of the 2024 Conference on United Kingdom & Ireland Computing Education Research*. ACM, Manchester United Kingdom, 1–7. https://doi.org/10.1145/3689535.3689553

[46] Jacqueline Whalley, Amber Settle, and Andrew Luxton-Reilly. 2023. A Think-Aloud Study of Novice Debugging. *ACM Trans. Comput. Educ.* 23, 2 (June 2023), 28:1–28:38. https://doi.org/10.1145/3589004

[47] Ruiwei Xiao, Xinying Hou, and John Stamper. 2024. Exploring How Multiple Levels of GPT-Generated Programming Hints Support or Disappoint Novices. In *Extended Abstracts of the 2024 CHI Conference*. ACM, New York. https://doi.org/10.1145/3613905.3650937