AI 如何影响技能形成

Judy Hanwen Shen* Alex Tamkin†

February 4, 2026

摘要

AI 辅助在各专业领域产生了显著的生产力提升，尤其是对新手工作者。然而，这种辅助如何影响有效监督 AI 所需的技能发展尚不明确。严重依赖 AI 来完成不熟悉任务的新手工作者可能会在这个过程中损害自己的技能习得。我们进行了随机实验，研究开发人员在有和没有 AI 辅助的情况下掌握一个新的异步编程库的情况。我们发现，使用 AI 会损害概念理解、代码阅读和调试能力，而不会带来显著的效率提升。完全委托编码任务的参与者显示出了一些生产力改善，但代价是未能学习该库。我们确定了六种不同的 AI 交互模式，其中三种涉及认知参与，即使参与者接受 AI 辅助也能保持学习成果。我们的研究结果表明，AI 增强的生产力并非通往能力的捷径，应谨慎地将 AI 辅助纳入工作流程以保持技能形成——尤其是在安全关键领域。

1 引言

自工业革命以来，劳动力市场的技能随着新技术的引入不断变化；工人的角色通常从执行任务转变为监督任务 [Autor et al., 2001]。例如，工厂机器人的自动化使人类能够从体力劳动转向监督工作，会计软件使专业人士能够从进行原始计算转向识别更好的簿记和税务策略。在这两种情况下，人类都要对最终产品的质量负责，并对任何错误承担责任 [Bleher and Braun, 2022]。即使自动化改变了完成任务的过程，识别和修复错误的技术知识仍然极其重要。

随着 AI 承诺成为从软件工程到创业等广泛应用的自动化和生产力催化剂 [Dell'Acqua et al., 2023, Peng et al., 2023, Cui et al., 2024, Otis et al., 2024, Brynjolfsson et al., 2025]，AI 对劳动力的影响尚未完全理解。尽管越来越多的工人依靠 AI 来提高生产力，但在工作场所使用 AI 辅助是否会阻碍对概念的核心理解或阻止开发监督自动化任务所需的技能尚不清楚。虽然大多数研究集中在 AI 辅助的最终*产品*上（例如，编写的代码行数、提出的想法质量），但同样重要，甚至更关键的问题是接受 AI 辅助的*过程*如何影响工人。当人类依赖 AI 进行头脑风暴、写作和一般批判性思维等技能时，这些技能的发展可能会根据 AI 辅助的使用方式而发生显著变化。

软件工程尤其被认为是 AI 工具可以 readily 应用的职业，AI 辅助显著提高了日常任务的生产力 [Peng et al., 2023, Cui et al., 2024]。初级或新手工作者在编写代码时从 AI 辅助中受益最多。在高风险应用中，AI 编写的代码可能需要在软件准备好部署之前由人类进行调试和测试。这种增强安全性的额外验证只有在人类工程师自己拥有理解代码和识别错误的技能时才可能实现。随着 AI 的发展，如果人类理解代码的能力较弱，监督越来越强大的 AI 系统的问题就会变得更加困难 [Bowman et al., 2022]。当复杂的软件任务需要人机协作时，即使他们的软件技能与 AI 的优势互补，人类仍然需要了解代码开发的基本概念 [Wang et al., 2020]。高风险环境中的持续能力要求与 AI 辅助带来的已证明的生产力提升的结合，使软件工程成为研究 AI 如何影响技能形成的理想试验台。

我们调查使用和依赖 AI 是否会影响软件工程技能的发展 [Handa et al., 2025]。基于 AI 在软件工程中的快速采用，我们的动机是工程师在工作岗位上学习新技能的场景。虽然使用 AI 工具可能会提高这些工程师的生产力，但它们是否也会抑制技能形成？更具体地说，AI 辅助的任务完成工作流程是否会阻止工程师深入了解用于完成这些任务的工具？我们进行了随机实验，通过要求参与者使用他们以前从未使用过的新库完成编码任务来衡量技能形成。这代表了工程师学习和获得新技能的一种方式，因为 Python 等语言经常引入新库。然后我们评估他们对该新库的熟练程度。我们的主要研究问题是：(1) AI 是否提高了需要新概念和技能的编码任务的生产力，(2) 使用 AI 是否会降低对这些新概念和技能的理解水平。

*工作作为 Anthropic Fellows Program 的一部分完成，judy@anthropic.com

†Anthropic，atamkin@anthropic.com

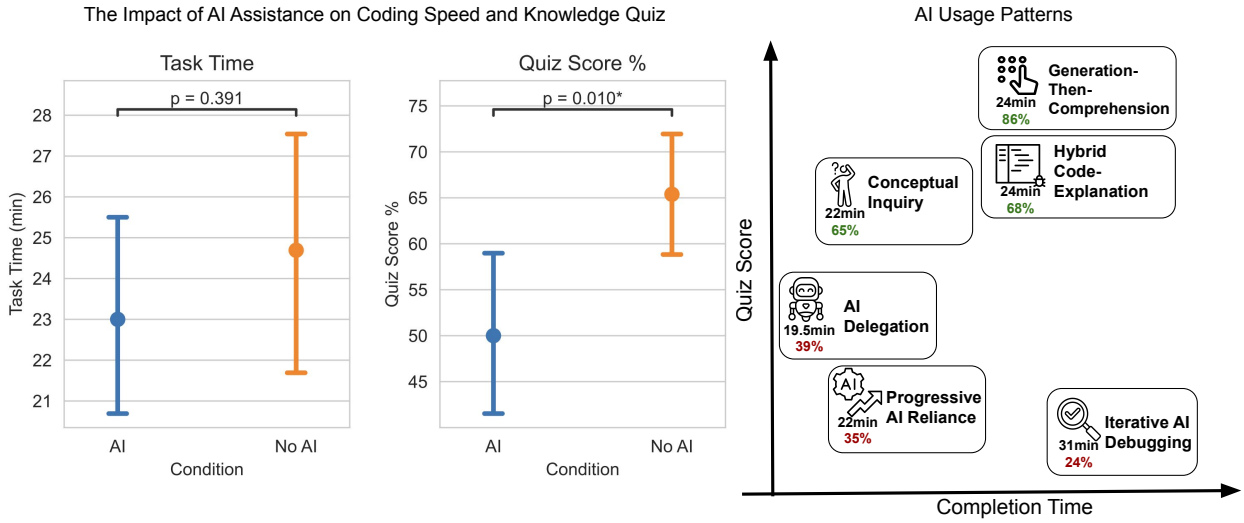

图 1: 结果概述：（左）我们发现，使用 AI 辅助完成新 Python 库任务的工人在库特定技能（概念理解、代码阅读和调试）方面显著下降。（右）我们归类了 AI 使用模式，发现了三种高技能发展模式，参与者在这些模式下在使用 AI 辅助时保持认知参与。

1.1 我们的结果

受 AI 和软件技能这一突出场景的启发，我们围绕相对较新的异步 Python 库 Trio 设计了编码任务和评估，并进行了随机实验，以了解 AI 辅助对任务完成时间和技能发展的影响。我们发现，使用 AI 辅助完成涉及这个新库的任务导致评估分数降低 17% 或两个等级分（Cohen's $d = 0.738$，$p = 0.010$）。同时，我们没有发现 AI 辅助在完成时间上有统计学显著的加速（图 6）。

通过深入的定性分析，我们观看了主要研究中每位参与者的屏幕录制，我们通过一些参与者在与 AI 助手交互上投入的额外时间来解释缺乏 AI 生产力提升的原因。一些参与者提出了多达 15 个问题，或者花费了超过总可用任务时间的 30% 来撰写查询（图 12）。我们将对照组技能发展的提升归因于独立遇到并随后解决错误的过程。我们将 AI 交互行为归类为六种常见模式，并发现三种最能保持技能发展的 AI 交互模式（图 11）。这三种与 AI 的交互模式在我们的技能评估中获得了更高的分数，涉及更多的认知努力和独立思考（例如，仅要求解释或仅提出概念性问题）。

2 背景

2.1 AI 使用的影响

自 2022 年底 ChatGPT、Copilot、Claude 和其他高级对话助手广泛可用以来，AI 工具已在许多不同领域得到广泛使用。研究基于提示的使用的研究有助于详细检查 AI 的实际应用 [Tamkin et al., 2024, Shen and Guestrin, 2025]。例如，AI 工具正在软件开发、教育、设计和科学等专业领域使用 [Handa et al., 2025]。

生产力提升 许多研究发现使用这些 AI 助手可以提高生产力。例如， Brynjolfsson et al. 发现，基于 AI 的对话助手使呼叫中心工人平均能够解决的问题数量增加了 15%。 Dell'Acqua et al. 在咨询公司发现了类似的结果，在 AI 的帮助下，咨询公司平均完成的任务多出 12.2%。虽然基于技能的效果在不同研究中有所不同，但在呼叫中心工作、咨询、法律问答和写作中出现了一致的模式：经验较少和技能较低的工作者往往受益最多 [Brynjolfsson et al., 2025, Dell'Acqua et al., 2023, Choi and Schwarcz, 2023, Noy and Zhang, 2023]。一个例外是当向肯尼亚小企业主提供 GPT-4 时，AI 商业建议帮助高绩效者（按收入）改善了业务结果，而低绩效者的结果恶化 [Otis et al., 2024]。

特别是在软件工程方面，Peng et al. 发现，使用 copilot 的众包软件开发人员完成任务的速度比对照组快 55.5%，新手程序员从 AI 编码辅助中受益更多。对主要软件公司开发人员的后续研究发现，AI 生成的代码

补全提供了 26.8% 的生产力提升，这是通过拉取请求、提交和软件产品构建来衡量的 [Cui et al., 2024]。这项研究还发现，经验较少的编码员获得了更大的生产力提升。虽然研究发现初级或经验较少的开发人员从使用 AI 中获得更大的生产力提升，但这些同样的工人应该在工作场所快速学习新技能。然而，这些工具对这一子群体的技能形成的影响仍然未知。这种生产力是免费的还是有代价的?

认知卸载 最近的研究强调了 AI 辅助和技能 depletion 的影响。例如，接受过 AI 辅助培训的医疗专业人员可能无法发展出识别某些条件的敏锐视觉技能 [Macnamara et al., 2024]。在对知识工作者的调查中，频繁使用 AI 与较差的批判性思维能力和增加的认知卸载有关 [Gerlich, 2025]。此外，知识工作者报告说，在使用生成 AI 工具时，认知努力和信心较低 [Lee et al., 2025]。然而，这些调查是观察性的，可能无法捕捉 AI 使用的因果效应。

技能保留 与我们研究相关的一个相邻问题是人类在 AI 辅助后如何很好地保留知识和技能。 Wu et al. 发现，即使生成 AI 提高了内容创建任务（例如，写 Facebook 帖子、写绩效评估、起草欢迎电子邮件）的即时性能，性能提升也不会在人类随后独立执行的任务中持续存在。对于数据科学任务， Wiles et al. 将 AI 对非技术顾问的影响描述为 “外骨骼”，AI 启用的增强技术能力在工人不再可以使用 AI 时不会持续存在。我们的工作提出了一个自然的后续问题，即使用 AI 工具是否会导致技术专业人员本身在工作场所获得技能的学习结果更差。

过度依赖 尽管经济学中关于 AI 增强生产力的许多文献隐含假设 AI 的一代是值得信赖的，但现实是生成 AI 可能产生不正确 [Longwell et al., 2024] 或幻觉内容 [Maleki et al., 2024]。当模型容易出错，但仍部署用于协助人类时，跟随错误模型决定的人类决策被称为 “过度依赖” [Buçinca et al., 2021, Vasconcelos et al., 2023, Klingbeil et al., 2024]。虽然已经提出了减少过度依赖的方法，但这些主要集中在决策时的信息上，例如解释 [Vasconcelos et al., 2023, Reingold et al., 2024] 或辩论 [Kenton et al., 2024]。

2.2 CS 教育和 AI 辅助

衡量技能的习得高度依赖于领域。特别是对于计算机科学，大多数入门课程通过多项选择题、代码编写和代码阅读/解释来衡量学习 [Cheng et al., 2022]。最近的工作发现代码面试和学生代码的积极讨论可以产生积极的学习成果 [Kannam et al., 2025]。

几项观察性研究描述了学生在计算机科学课程的背景下如何使用 AI 工具。Poitras et al. 发现，在一个学期内，学生使用 AI 工具编写代码、修复错误和解释算法概念；编码能力较差的学生更有可能寻求 AI 辅助。其他工作使用调查发现，学生可能由于 “依赖担忧”（即过度依赖编码工具）而犹豫是否使用 AI 编码辅助工具 [Pan et al., 2024]。对于形式方法，Prasad et al. 编码了学生在课程工作中使用 LLM 的不同方式，并发现参加课程的高年级学生不依赖 LLM 辅助，只在开始时提出了几个问题。

也在专业发展环境中进行了用户研究。Wang et al. 研究了在完成编码谜题和开发任务时，有和没有聊天访问 AI 模型的用户之间的不同使用模式。他们发现了丰富的交互模式，包括交互式调试、代码讨论和提出具体问题。参与者的范围从要求 ChatGPT 然后完成整个问题（最低质量的代码输出）到只提出最少的问题（最高的效率）。其他研究报告说，AI 工具通过更容易访问文档和为特定 API 准确生成代码来帮助软件开发过程 [Pinto et al., 2024]。

3 框架

专业技能习得 “在做中学” 哲学已被许多学习框架提出，例如 Kolb 的体验学习循环和基于问题的学习（PBL）[Kolb, 2014, Schmidt, 1994]。这些框架将完成现实世界任务与学习新概念和发展新技能联系起来。体验学习也专门在高等教育软件工程课程中得到探索，以模仿在专业环境中解决问题 [Gonzalez-Huerta et al., 2020]。在最简单的形式中，我们将 AI 工具辅助建模为采取与没有 AI 不同的学习路径。我们假设在开发过程中使用 AI 工具生成代码实际上相当于走捷径来完成任务，而没有明显的学习阶段。

AI 编码使用模式 以前的工作发现人类以许多不同的方式将 AI 用于编码：从问答到编写代码，再到调试 [Poitras et al., 2024, Wang et al., 2020, Pinto et al., 2024]。在我们的框架中，使用 AI 辅助的不同方式代表了采取的不同学习路径，以实现完成任务的目标。我们在本工作的定性分析中分析了这些不同的使用模式（第 6 节）。

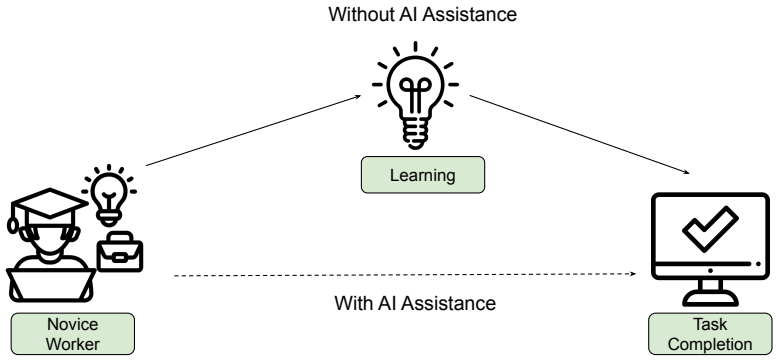

图 2: 随着 AI 辅助在工作场所变得越来越普遍，新手工人可能在没有相同学习成果的情况下完成任务。我们的实验旨在调查需要新技能的任务完成过程，以了解 AI 辅助对编码技能形成的影响。

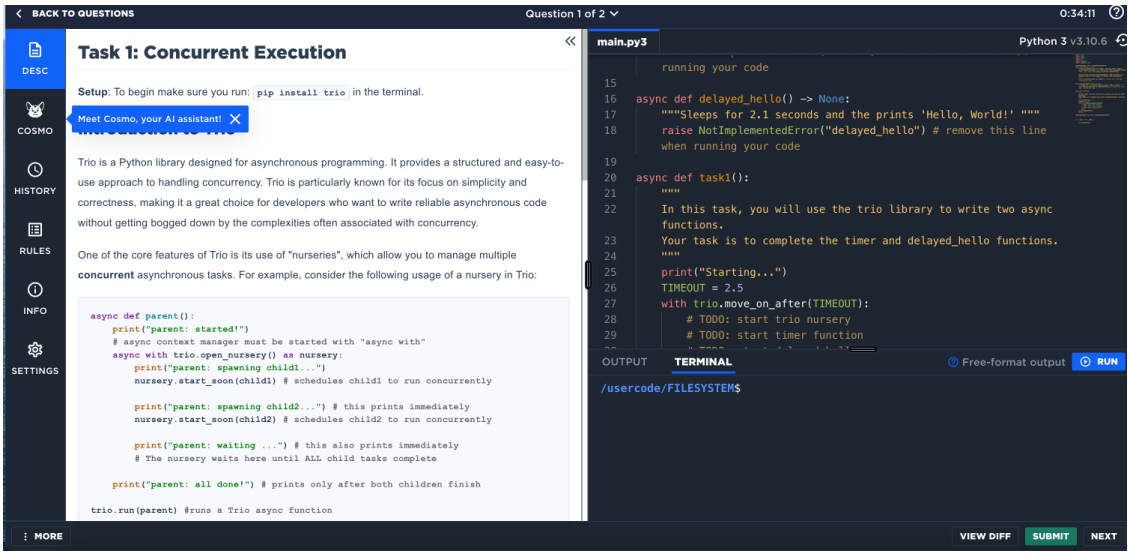

图 3: 实验界面：我们使用在线面试平台来运行我们的实验。处理条件参与者被提示使用 AI 助手。

研究问题 基于此背景，我们专注于在职学习：工人必须获得新技能才能完成任务的环境。我们试图了解 AI 对生产力和技能形成的影响。我们询问 AI 辅助是呈现即时生产力与长期技能发展之间的权衡，还是 AI 辅助呈现增强两者的捷径。我们的研究问题如下：

- **RQ1**：当需要新技能时，AI 辅助是否能提高任务完成生产力？
- **RQ2**：使用 AI 辅助如何影响这些新技能的发展？

4 方法

4.1 任务选择：使用 Trio 学习异步编程

我们为初级软件工程师在工作岗位上可能遇到的几种不同技能制作了任务原型：从数据分析到绘图。我们围绕 Python Trio 库设计了一个实验，[1] 该库旨在异步并发和输入输出处理（I/O）。这个库不如 `asyncio` 出名（根据 `StackOverflow` 问题的数量），并且涉及仅精通 Python 之外的新概念（例如，结构化并发）。它还明确设计得易于使用——使其特别适合学习实验。

[1] 参见文档：`https://trio.readthedocs.io/en/stable/`

我们设计并测试了五个使用 Trio 库进行异步编程的任务，这是一项在处理大规模数据或软件系统时经常在专业环境中学习的技能。我们创建的任务包括问题描述、启动代码以及对完成任务所需的 Trio 概念的简要描述。这些任务旨在通过简短的自学教程来模仿学习使用新库或新软件工具的过程。例如，在软件工程师的入职材料中，通常有如何使用内部库的描述以及使用新库构建技能的小任务。在几次试点研究之后，我们在主要研究中使用了前两个任务；在初始测试期间，每个任务花费 10-20 分钟。第一个任务是编写一个计时器，在其他函数运行时每过一秒打印一次。这个任务介绍了 Trio 中的 nursery、启动任务和并发运行函数的核心概念。第二个任务涉及实现一个记录检索函数，该函数可以处理 Trio 库中的缺失记录错误。这个任务介绍了错误处理和用于存储结果的内存通道等概念。这两个任务是独立的；我们提供了足够的说明和使用示例，以便参与者可以在没有另一个任务的情况下完成一个任务。

我们使用带有 AI 助手聊天界面的在线面试平台（图 3）进行我们的实验。AI 条件下的参与者被提示使用 AI 助手来帮助他们完成任务。此助手使用的基础模型是 GPT-4o，该模型被提示为智能编码助手。AI 助手可以访问参与者当前版本的代码，并且可以直接生成两个任务的完整、正确的代码。

4.2 评估设计

基于对计算机科学教育评估的先前荟萃分析 [Cheng et al., 2022]，我们确定了四种用于评估编码技能掌握程度的问题类型。回到我们最初开发和维护监督自动化所需的技能的动机，其中一些领域的熟练程度可能比其他领域更重要，以便监督 AI 生成的代码。我们考虑的四种问题类型如下。

- **调试** 识别和诊断代码中错误的能力。这项技能对于检测 AI 生成的代码何时不正确以及理解其失败原因至关重要。
- **代码阅读** 阅读和理解代码功能的能力。这项技能使人类能够在部署之前理解和验证 AI 编写的代码。
- **代码编写** 编写或选择正确编码方式的能力。随着 AI 编码工具的进一步集成，低级代码编写（如记住函数的语法）将不如高级系统设计重要。
- **概念性** 理解工具和库背后的核心原则的能力。概念性理解对于评估 AI 生成的代码是否使用了符合库应如何使用的适当设计模式至关重要。

我们研究中的两个任务涵盖了 Trio 库的 7 个核心概念。我们设计了一个测验，其中包含涵盖这 7 个概念的调试、代码阅读和概念性问题。我们排除了代码编写问题，以减少语法错误对评估的影响；这些错误可以通过 AI 查询或网络搜索轻松更正。我们基于项目反应理论在用户测试和初步研究中测试了 5 个版本（表 2）的测验。例如，我们确保所有问题与总体测验分数充分相关，每个问题都有适当的平均分数，并且问题被拆分使得问题之间不存在局部项目依赖（即，参与者无法通过查看其他问题来推断问题的答案）。我们使用的最终评估包含 14 个问题，总共 27 分。我们在运行实验之前提交了我们研究中测验的评分标准。

4.3 研究设计

我们使用受试者间随机实验来测试在编码技能形成过程中使用 AI 的效果。每个参与者首先在编码平台上完成热身编码任务，其中需要在字符串列表周围添加边框。这个 Python 编码问题在此编码平台的用户中平均需要 4 分钟才能完成。这个编码问题中没有异步概念。没有参与者在完成热身阶段时可以使用 AI。我们使用此阶段来校准参与者的 Python 熟悉程度，并帮助参与者熟悉界面。

下一阶段是 Trio 任务阶段，参与者在同一个编码平台上最多有 35 分钟的时间使用 Trio 完成两个编码任务。在此阶段，AI 辅助条件（处理组）的参与者可以通过基于聊天的 AI 助手（图 3）获得编码帮助。所有参与者都被指示尽可能快地完成任务。

完成 Trio 任务后，参与者完成了评估阶段，他们在其中参加我们在上一节中描述的测验，并在测验后完成一项包含人口统计和经验问题的调查。

在我们的主要研究中，52 名参与者完成了任务，对照组和处理组各 26 名。对于所有试点研究和主要研究，我们只招募自我报告拥有超过一年 Python 经验、至少每周编写一次 Python 代码、至少尝试过几次 AI 编码辅助并且从未使用过 Trio 库的参与者（表 1）。

我们使用编码平台在参与者编码时收集他们的击键，以及在编码条件下他们与 AI 编码助手交互的记录。我们使用 Google 表单在编码任务之前和编码任务之后收集参与者的调查回复。这些任务总共最多需要 1 小时 15 分钟，平均持续时间为 58.5 分钟。参与者通过第三方众包平台招募，并获得 150 美元的固定报酬。

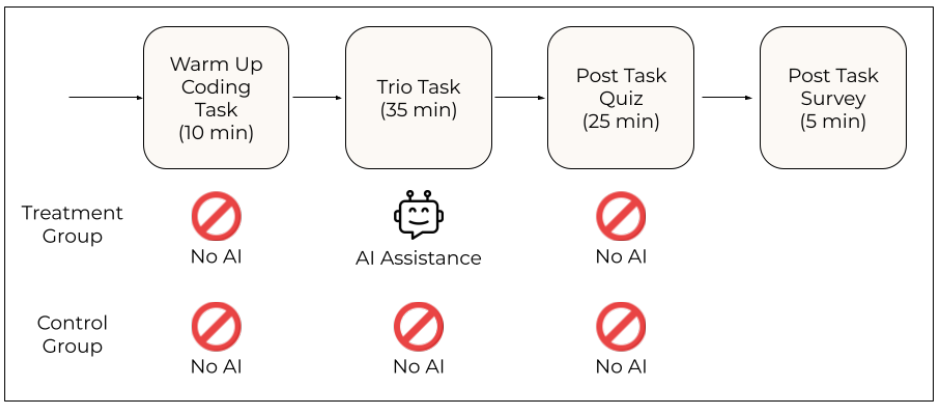

图 4: 学习任务和理解检查的概述。所有参与者都完成了一个不需要 Trio 知识的热身编码任务。在主要 Trio 任务期间，处理组的参与者可以使用 AI 辅助来回答问题或生成代码。所有参与者在理解检查中都不允许使用 AI。

	处理组 (%)	对照组 (%)	差异 (%)
编码经验年限			
1-3 年	2 (0.077)	2 (0.077)	0
4-6 年	10 (0.385)	9 (0.346)	1 (0.038)
7+ 年	14 (0.538)	15 (0.577)	1 (0.038)
Python 使用频率			
定期 / 经常	18 (0.692)	16 (0.615)	2 (0.077)
每日 / 广泛	8 (0.308)	10 (0.385)	2 (0.077)
每任务异步测验分数			
0-2 (0-40%)	5 (0.192)	5 (0.192)	0
3-4 (60-80%)	18 (0.692)	15 (0.577)	3 (0.115)
5 (100%)	3 (0.115)	6 (0.231)	3 (0.115)
先前使用 Python Asyncio	18 (0.692)	20 (0.769)	2 (0.077)
任务前编码时间	6.5 分钟	8 分钟	1.5 分钟

表 1: 主要研究参与者的平衡表（n=52）。

试点	平台	参与者	任务数	挑战
A	P1	n=39	5	**不合规：** 35% 不 AI 条件下的参与者使用 AI 辅助来复制说明并粘贴结果。参与者还自我报告在不 AI 条件下使用 AI 辅助。
B	P1	n=107	5	**持续不合规：** 即使被警告严格的 No AI 要求，参与者仍继续使用 AI 进行编码和测验。25% 的参与者使用 AI，并且参与者平台无法进行屏幕录制。
C	P2	n=20	5	**局部项目依赖：** 通过观看屏幕录制，我们观察到参与者在问题之间来回滚动以猜测正确答案。
D	P2	n=20	2	**Python 语法延迟：** 在 35 分钟的时间限制内，只有 60% 的对照组（不 AI 条件）参与者完成了两项任务。屏幕录制显示几名参与者在 Python 语法问题上挣扎，例如 try/except 块和字符串格式化。这些延迟与 Trio 库无关。

表 2: 不同数据提供商、任务和评估设计的试点研究摘要。

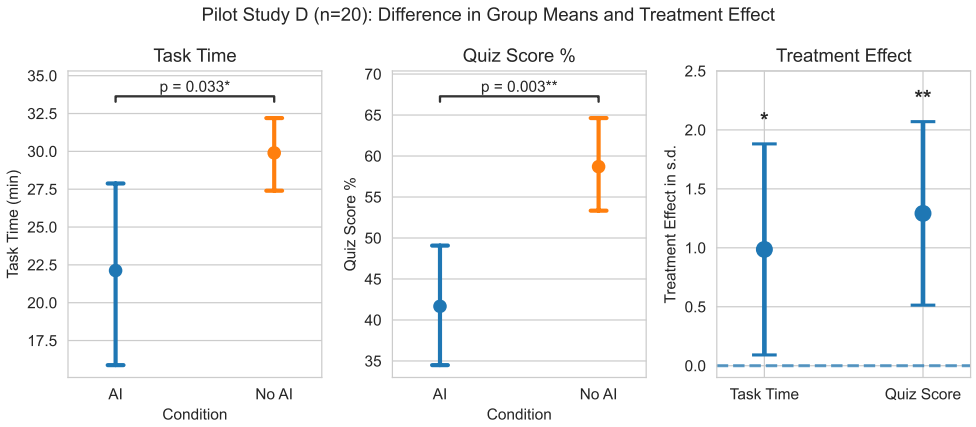

图 5: 试点研究 D 中对照组（No AI）和处理组（AI Assistant）之间总体任务时间和测验分数的平均值差异。误差条代表 95% CI。显著性值对应于处理效应。* p<0.05，**<0.01，***<0.001

5 结果

5.1 试点研究

不合规我们在运行完整研究之前进行了 4 项试点研究（表 2）。前两项试点研究是在不同的众包平台（P1）上进行的。在这个平台上，我们观察到高不合规率（35%），既在任务期间也在测验期间（即，参与者在对照组中使用 AI 来完成编码任务或使用 AI 来完成评估。我们通过编码平台记录中用户复制说明或将代码粘贴到编辑器时观察到不合规行为。我们测试了不同的机制来确保对照组（No AI）中的参与者不为任务使用 AI。然而，尽管有更明确的说明，对照组中仍有大约 25% 的参与者使用 AI。我们与第二个众包平台（P2）进行了两项试点研究，每项研究有 20 名参与者。利用参与者进展的屏幕录制，我们验证了参与者在对照组中也没有使用 AI，也没有为测验使用 AI。

局部项目依赖在试点研究 C 中，我们在测验中观察到了*局部项目依赖*：参与者会比较问题并根据其他问题中提供的代码片段识别答案。这促使我们将测验拆分为几个不同的页面，其中每个页面上的问题不会为其他问题提供提示。基于屏幕录制，我们观察到这在试点 D 中减少了局部项目依赖。此外，我们将任务总数从五个减少到两个。这种变化使我们能够更好地从第一个任务中分离学习，同时消除一个混淆变量：AI 条件下的参与者仅仅因为他们在规定时间内完成了更多任务而看到更多概念。为了使测验与此修改保持一致，我们将测验问题调整为仅涵盖前两个任务。

任务完成的障碍在试点研究 D 中，我们纳入了 20 名参与者。我们发现 AI 和不 AI 条件之间的任务完成时

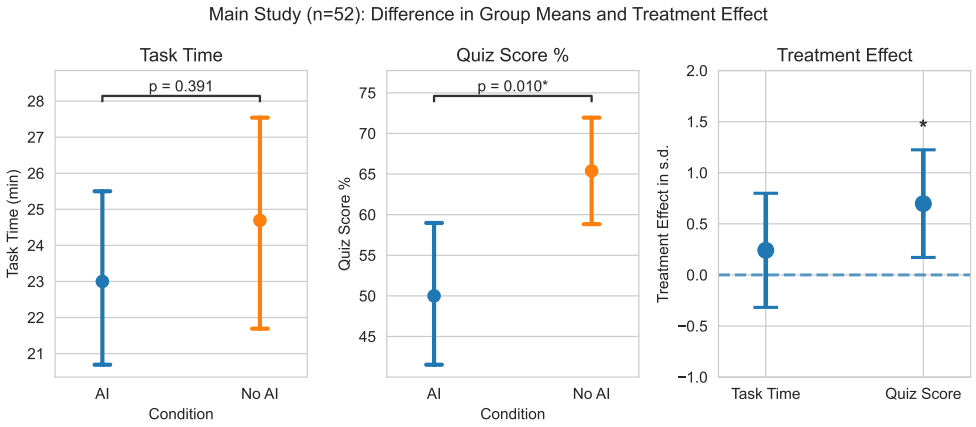

图 6: 主要研究中对照组（No AI）和处理组（AI Assistant）之间总体任务时间和测验分数的平均值差异（n=52）。误差条代表 95% CI。显著性值对应于处理效应。* p<0.05，**<0.01，***<0.001

间和测验分数存在显著差异（图 5）。当我们回顾屏幕录制时，对照组（不 AI）中的参与者在与 Trio 无关的 Python 语法上挣扎，例如 `try/except` 块和字符串格式化。在 35 分钟的时间限制内，对照组（不 AI 组）的任务完成率仅为 60%，而处理组（AI 组）的完成率为 90%。由于我们的重点不是 Python 语法，我们为主要研究添加了关于字符串格式化和 `try/except` 块的语法提示。

图 5 展示了对两个结果测量的处理效应：任务时间和测验分数。处理组（AI）更快地完成了 Trio 任务（Cohen's d=1.11，p=0.03），表明任务效率有所提高。然而，AI 组在知识测验中的表现明显更差（Cohen's d=1.7，p=0.003），表明学习保留率降低。对于我们完整的功效分析和研究的预注册，[2] 我们假设保守效应量 d = 0.85（观察到的学习效应的一半），以解释试点研究中典型的潜在效应量膨胀。

5.2 主要研究

5.2.1 参与者

为了招募 50 名参与者，我们将研究发送给 58 名众包工作者。参与者在以下属性之间保持平衡（通过单独的招募调查记录）：编码经验年限、Python 经验年限、先前使用 Python Asyncio 库、过去一年中 Python 的使用频率以及异步编程熟悉度分数（5 个问题的多项选择概念检查）。参与者在任务完成后收集的人口统计细分汇总在图 17 中，以避免刻板印象的威胁。我们研究中的大多数参与者拥有学士学位，年龄在 25 到 35 岁之间，并且要么是自由职业者，要么是专业软件开发人员。53 名参与者完成了研究的所有三个部分。遵循我们的预注册取消资格标准，1 名参与者在测验中留下四个空白问题后被取消资格，因为他们没有意识到测验有多个部分，随后时间用完了。

5.2.2 结果

图 6 显示，虽然使用 AI 完成我们的编码任务并没有显著改善任务完成时间，但通过完成任务获得的技能形成水平，通过我们的测验来衡量，显着降低（Cohen d=0.738，p=0.01）。处理组和对照组的平均值之间存在 4.15 分的差异。对于 27 分的测验，这转化为 17% 的分数差异或 2 个等级分。将热身任务时间作为协变量控制，处理效应仍然显着（Cohen's d=0.725，p=0.016）。

以前的工作对 AI 是帮助还是阻碍编码生产力提出了褒贬不一的结果 [Peng et al., 2023, Becker et al., 2025]；我们的研究与以前的结果不同，因为它旨在研究 AI 如何影响在需要新知识的任务执行过程中的技能形成。虽然我们确实观察到 AI 组中新程序员的平均完成时间稍短，但由于 1-3 年参与者组的小组规模（n=4），任务时间的差异并不显着。对照组（No AI）的 26 名参与者中有 4 名未能在 35 分钟的限制内完成第二项任务，而 AI 条件下的每位参与者都完成了第二项任务。我们的结果没有 conclusively 发现使用 AI 在此任务中加速或减速。

[2]预注册：`https://osf.io/w49e7`

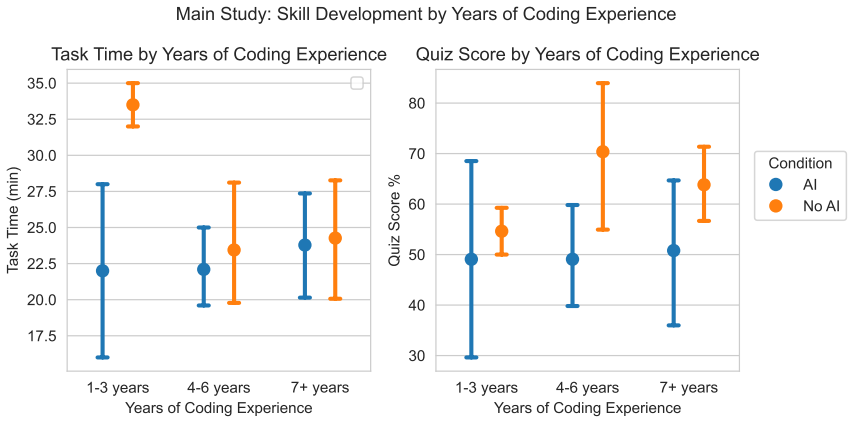

图 7: 按编码经验年限划分的任务完成时间和测验分数。误差条代表 95% CI。对照组（No AI）的平均测验分数在所有编码经验水平上都更高。

在所有先前的编码经验水平上，用户在对照组（不 AI）中的平均得分高于处理组（AI 辅助）（图 7）。这表明我们的任务选择和任务设计并不关键地取决于参与者的经验水平，而是为每个经验组提出了要获得的新技能。

概念组分析在探索性数据分析（未预注册）中，测验分数被分解为子区域和问题类型（图 8）。测验中的每个问题恰好属于一个任务（例如，任务 1 或任务 2）和一种问题类型（例如，概念性、调试或代码阅读）。对于两个任务，处理组和对照组的测验分数之间存在差距。在不同类型的问题中，最大的分数差距出现在调试问题中，最小的分数差距出现在代码阅读问题中。这个结果是预期的，因为处理组和对照组可能通过任务有类似的代码阅读接触，但没有 AI 辅助的对照组在任务期间遇到更多错误，并变得更有能力进行调试。

任务体验在进一步的探索性数据分析中，我们还发现了参与者完成研究体验的方式存在差异。对照组（No AI）报告了更高的自我报告学习（7 分制），而两组都报告了在完成任务时的高度享受（图 9）。在任务难度方面，图 10 显示，虽然处理组（AI 辅助）的参与者发现任务比对照组更容易，但两组都发现任务后测验同样具有挑战性。

6 定性分析

虽然生产力和测验分数的总体统计数据有助于揭示 AI 辅助如何影响新学习任务的高级趋势，但对每个参与者如何完成学习任务的更深入分析使我们能够更好地理解参与者的异质性。在我们定性分析的初始编码阶段，我们手动标注了主要研究中 51 名参与者的屏幕录制。[3]我们将标注分组为与任务进度事件相关的几个主要概念，例如错误、AI 交互、AI 查询和任务完成（表 5）。这种分析使我们不仅能够理解整体生产力和学习，还能理解在我们研究的每个任务中如何使用 AI。我们将这些标注的记录公开供未来研究使用。[4]

分析参与者之间的这些概念或常见模式有助于补充我们对这个新库中技能形成和任务完成的定量观察。具体来说，以下轴显示了参与者和条件之间的差异：

- **AI 交互时间**：AI 条件缺乏显着加速的原因可以通过一些参与者使用 AI 的方式来解释。几名参与者花费了大量时间与 AI 助手交互，总共花费高达 11 分钟撰写 AI 查询（图 12）。
- **查询类型**：研究参与者在仅概念性问题、仅代码生成以及概念性、调试和代码生成查询的混合之间有所不同。专注于询问 AI 助手调试问题或确认其答案的参与者花费更多时间完成任务（图 18）。
- **遇到错误**：对照组（不 AI）的参与者遇到更多错误；这些错误包括语法错误和 Trio 错误（图 14）。遇到更多错误并独立解决错误可能改善了 Trio 技能的形成。

[3]AI 条件中一名参与者的屏幕录制不可用。

[4]有关标注过程的详细信息可以在第 B 节中找到，标注的记录可以在 `https://github.com/safety-research/how-ai-impacts-skill-formation` 中找到

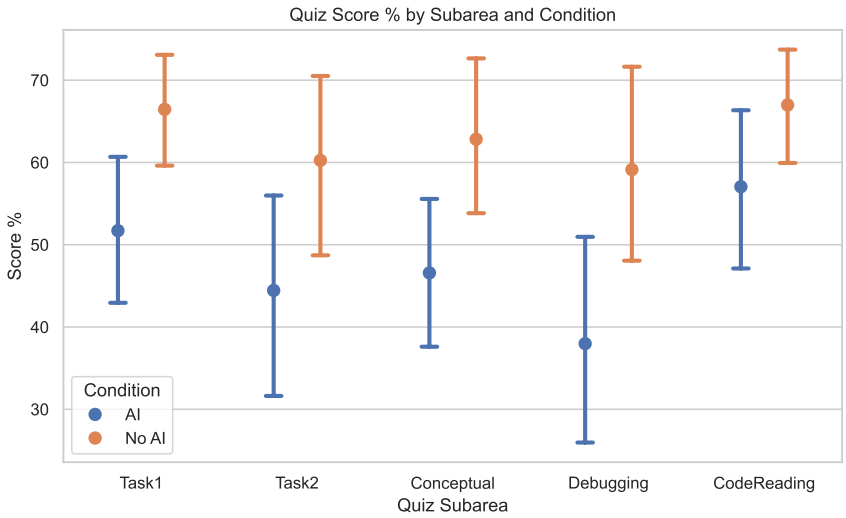

图 8: 与每个任务和技能区域相关的问题类型的分数细分。调试问题揭示了处理组和对照组之间平均测验分数的最大差异。

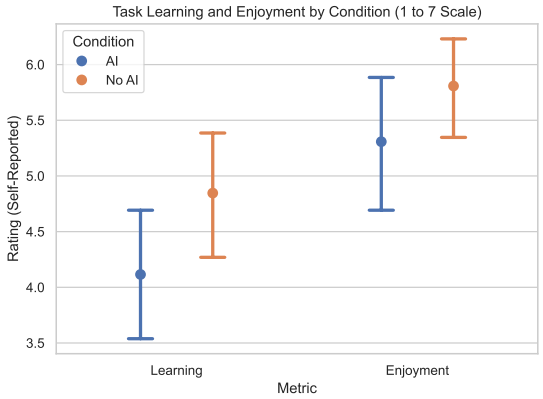

图 9: 我们在研究期间按条件自我报告的享受和学习程度。

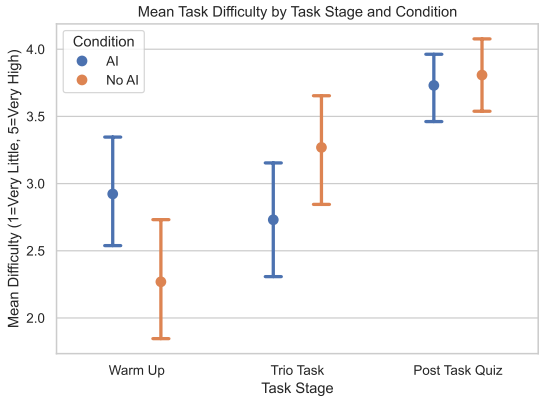

图 10: 在我们研究的不同阶段按条件自我报告的任务难度。

- **主动时间**：使用 AI 减少了主动编码时间的数量。编码的时间转移到与 AI 交互和理解 AI 生成的时间（图 16）。

使用这些轴，我们基于查询类型、查询数量、每项任务查询和主动时间开发了六种 AI 交互模式的类型学。由于这种分类，这六种模式对完成时间和技能形成（即测验分数）产生不同的结果。图 11 总结了每种模式和平均任务结果。我们可以将交互模式分为两类：低分交互模式和高分交互模式；高分模式通常涉及更多的认知努力和更少的 AI 依赖。虽然每个行为模式集群都很小，但低分集群和高分集群之间的差异是显着的。

低分交互模式低分模式通常涉及严重依赖 AI，要么通过代码生成，要么通过调试。这些组的平均测验分数低于 40%。表现出这些交互模式的参与者表现出较少的独立思考和更多的认知卸载 [Lee et al., 2025]。

- **AI 委托**（n=4）：这个组的参与者完全依赖 AI 编写代码和完成任务。这个组完成任务最快，在此过程中遇到很少或没有错误。
- **渐进式 AI 依赖**（n=4）：这个组的参与者首先提出 1 或 2 个问题，最终将所有代码编写委托给 AI 助手。这个组的测验分数很差，主要是因为没有掌握第二项任务中的任何概念。
- **迭代式 AI 调试**（n=4）：这个组的参与者依赖 AI 来调试或验证他们的代码。这个组向 AI 助手提出

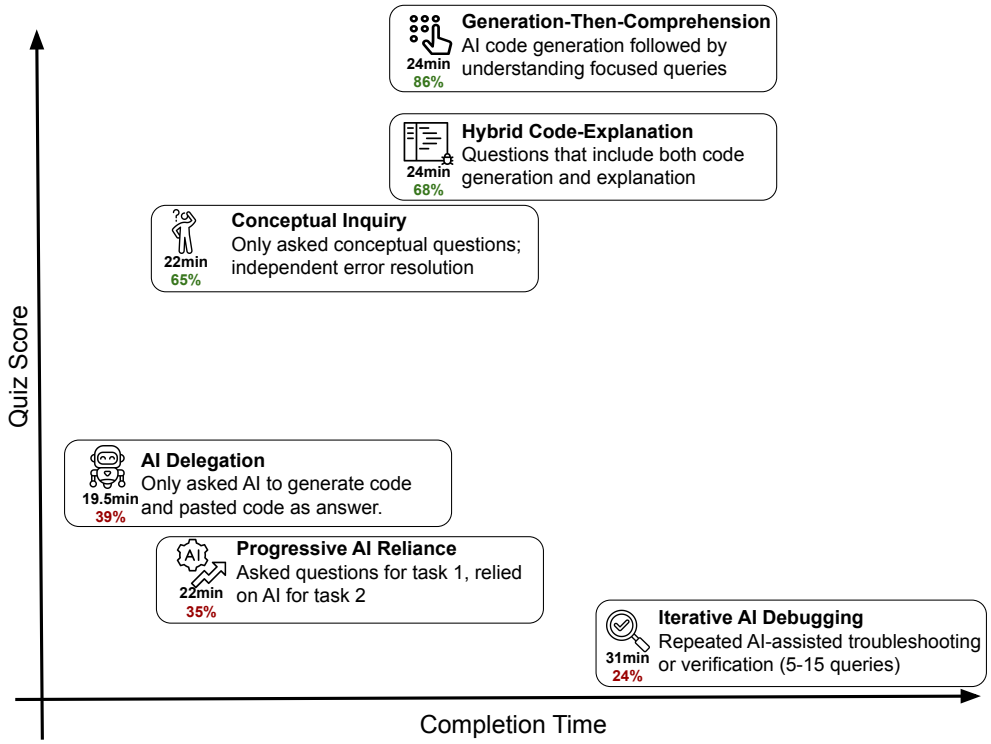

图 11: 我们研究中处理组（AI）条件下的 6 种 AI 交互角色，平均完成时间和测验分数。

了更多的查询，但依靠助手来解决问题，而不是阐明自己的理解。因此，他们的测验分数很差，完成两项任务的速度相对较慢。

高分交互模式高分交互模式是行为集群，其中平均测验分数为 65% 或更高。这些集群中的参与者使用 AI 进行代码生成、概念查询或两者的组合。

- **生成后理解**（n=2）：这个组的参与者首先生成代码，然后手动复制或将代码粘贴到他们的工作中。生成他们的代码后，他们然后向 AI 助手提出后续问题以提高理解。这些参与者在使用 AI 时并不是特别快，但在测验中表现出了高水平的理解。重要的是，这种方法看起来与 AI 委托组几乎相同，但另外使用 AI 来检查他们自己的理解。
- **混合代码解释**（n=3）：这个组的参与者编写混合查询，他们在其中要求代码生成以及生成代码的解释。阅读和理解他们要求的解释需要更多时间。
- **概念性询问**（n=7）：这个组的参与者只提出概念性问题，并依靠他们提高的理解来完成任务。虽然这个组遇到了许多错误，但他们也独立解决了这些错误。平均而言，这种模式是高分模式中最快的，在 AI 委托模式之后总体上第二快。

6.1 AI 交互

交互时间 与以前发现 AI 辅助对编码有显着提升或加速的工作相反 [Peng et al., 2023, Cui et al., 2024]，如果我们只查看处理组和对照组之间的总完成时间，我们的结果不会显示出生产力的显着改善。通过分析 AI 条件下的参与者如何完成任务，缺乏改善生产力的原因是由于与 AI 助手交互所花费的时间。处理组的一些参与者花费了大量时间（长达 11 分钟）与 AI 助手交互。例如，通过打字或思考要打什么。我们通过标记用户开始输入查询（标注为 *AI 交互*事件）和 AI 助手产生答案（标注为 *AI 查询*事件）之间的时间来捕获与 AI 交互所投入的时间。

由于参与者可以询问 AI 助手尽可能多的问题，少数参与者提出了超过五个问题，并在此 35 分钟的作业中

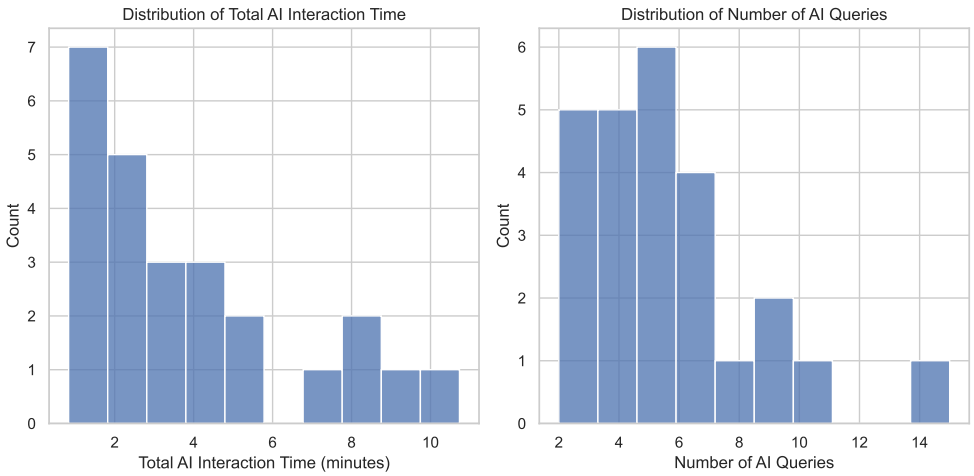

图 12: 总 AI 交互时间和 AI 查询数量的分布。在任务期间花费超过 6 分钟与 AI 交互的参与者导致处理组（AI 辅助）不比对照组（No AI）快得多。

花费长达六分钟撰写单个查询（图 12）。[5]由于 AI 条件下的中位完成时间只有 19 分钟，花费高达 6 分钟撰写单个查询占与 AI 助手交互所花费的总时间的很大一部分。虽然这种效应可能是由于我们任务的持续时间较短，但 Becker et al. 也在较长的任务中发现了专家编码员的减速效应，当时等待 AI 编写的代码的参与者可能会分心。

然而，从技能形成的角度来看，撰写查询所花费的时间可能有助于更好地理解任务，从而更好地获得技能。屏幕录制显示参与者在思考要向 AI 助手询问什么（例如，重新阅读说明和重写查询）。因此，一些参与者花费了几分钟来撰写单个查询。因此，虽然这种时间成本在基于聊天的助手中比代理编码助手更突出，但在不需要撰写查询的代理或自动完成设置中，知识的损失可能甚至更大。由于与 AI 辅助的交互时间较短，完成时间的更显着差异可能会转化为对技能形成的更大的负面影响。当我们查看单个查询时，并非所有查询都涉及显着的思考和时间。因此，我们分析单个查询以更好地了解参与者如何从新技能中学习。

AI 查询 我们将用户输入到 AI 助手中的查询分为 5 大类：解释、生成、调试、能力问题和感谢（表 3）。最常见的查询类型是解释（q=79）；用户请求有关 trio 库的更多信息、有关异步操作的详细信息以及高级概念介绍。处理组 25 名参与者中有 21 名提出了解释问题；这反映了我们参与者的高度参与。第二常见的是要求生成代码的查询（q=51）；一些参与者要求完成整个任务，而其他参与者要求实现特定功能。只有 25 名中的 16 名或三分之二的参与者使用 AI 来生成代码。其中 4 名参与者只要求代码生成，没有提出其他类型的问题。事实上，8 名得分最低的参与者中有 3 名要求 AI 在没有要求解释的情况下生成代码，这表明如果 AI 组中的所有参与者都仅使用 AI 生成代码，与对照组相比的技能形成差异甚至会更大。

常见的第三类查询是调试（q=9）。我们的任务设计得很简单，但参与者仍然遇到了各种错误（第 6.2 节）。这是一个更广泛的查询类别，包括直接粘贴到 AI 助手中的错误以及要求 AI 助手确认编写的代码是正确的。更高比例的调试查询与较慢的完成时间（图 18）和较低的测验分数（图 19）相关。这表明在学习新任务时依赖 AI 进行调试（例如，重复要求 AI 在不理解的情况下检查和修复事物）与较少的学习有关。

尽管我们只招募了以前使用过 AI 助手的参与者，但仍有问题（q=4）关于助手是否可以看到现有代码以及助手是否知道特定库。作为对这些问题的回应，AI 辅助澄清了他们可以看到代码和说明。几名参与者在任务正确完成后还表达了对助手的赞赏。即使以额外的任务时间为代价，这些感谢的表达也反映了人机交互中的礼貌 [Druga et al., 2017, Ribino, 2023] 也出现在 AI 编码辅助的背景下。

Adopting AI Advice: Pasting vs Manual Code Copying Another pattern that differs between participants is that some participants directly paste AI-written code, while other participants manually typed in (i.e., copied) the the AI generated code into their own file. The differences in this AI adoption

[5]参与者被指示尽快完成任务，并获得参与费的固定报酬。参见第 4.3 节。

Query Type	Example Query
Explanation (q=79)	*"can trio.sleep use partial seconds?"*
	"Can you remind me what the different trio async operations are?"
	"Looks good, can you give me a really brief overview of the general idea behind all of this?"
Generation (q=51)	*"given this instruction to trio, can you implement the missing bits of main.py?"*
	"complete get_user_data"
	"implement delayed_hello(). It should simply sleep for 2.1 seconds upon which it prints 'Hello World!' "
Debugging (q=9)	*"Does that look right? If so let's move on to delayed_hello()"*
	"I'm having issues getting my code to work. I'm getting a notimplementederror for delayed_hello"
	Pasted Error (e.g., *"Traceback (most recent call last): File "/user-code/FILESYSTEM/main.py3", line 81, in... "*)
Capabilities Question (q=4)	*"Can you see the current question?"*
	"So what can you do for me here? Can you write code directly into the file?"
	"Are you aware of how trio works? Are there parallels in its execution model to another library I'd be more familiar with like asyncio"
Appreciation (q=4)	*"Great job, we got the expected output on the first try."*
	"Looks like it worked, thanks!"
	"Trueeee!"

表 3: Examples of different types of queries received by AI assistant and counts of each type of query. 11 queries have multiple (two) labels.

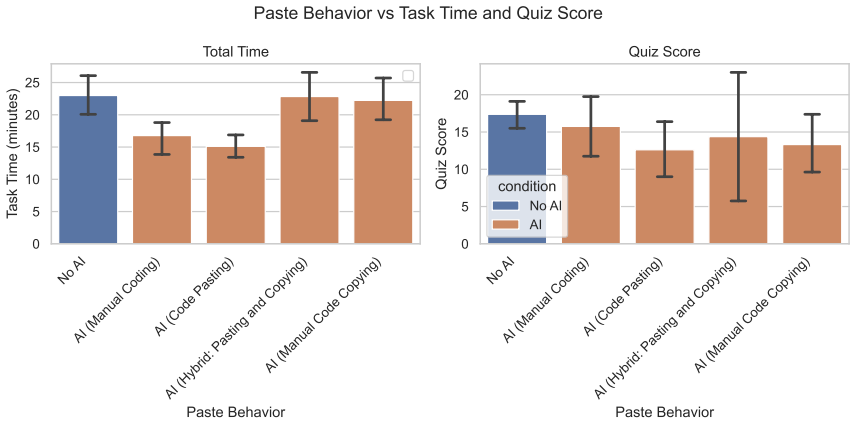

图 13: Decomposing AI Coding Behavior: Participants using AI by directly pasting outputs experience the most significant speed ups while participants who manually copied the AI-generated output were similar in pace to the control (No AI) group.

	AI	No AI
总计	1.0 (0.0-3.0)	3.0 (2.0-5.0)
任务 1	0.0 (0.0-2.0)	2.0 (0.5-3.0)
任务 2	0.0 (0.0-1.0)	2.0 (0.5-2.0)

表 4: 按条件每个参与者遇到的错误数量。值为中位数（Q1–Q3）。

style correlate with completion time. In Figure 13, we isolate the task completion time and compare how the method of AI adoption affects task completion time and quiz score. Participants in the AI group who directly pasted ($n = 9$) AI code finished the tasks the fastest while participants who manually copied ($n = 9$) AI generated code or used a hybrid of both methods ($n = 4$) finished the task at a speed similar to the control condition (No AI). There was a smaller group of participants in the AI condition who mostly wrote their own code without copying or pasting the generated code ($n = 4$); these participants were relatively fast and demonstrated high proficiency by only asking AI assistant clarification questions. These results demonstrate that only a subset of AI-assisted interactions yielded productivity improvements.

For skill formation, measured by quiz score, there was no notable difference between groups that typed vs directly pasted AI output. This suggests that spending more time manually typing may not yield better conceptual understanding. Cognitive effort may be more important than the raw time spent on completing the task.

6.2 遇到错误

参与者在处理组和对照组条件下遇到和解决错误的方式存在显着差异。在平台上，参与者可以经常使用运行按钮或终端运行他们的代码。通常，大多数参与者在尝试完成大部分问题后第一次运行代码，并且只在更改后才再次运行代码。当我们观看任务进度的屏幕录制时，我们记录了每个参与者遇到的每个错误。

错误频率 AI 组遇到的错误少于对照组：处理组的中位参与者在整个任务中只遇到一个错误，而对照组的中位数是三个错误。表 4 显示了错误分布的差异。AI 组的大多数参与者能够在第一次运行代码时完成任务。相反，在对照组中，大多数参与者在完成每个任务的过程中遇到了几个错误。在完成两项任务时没有遇到错误的 12 名参与者中，只有 2 名在对照组。

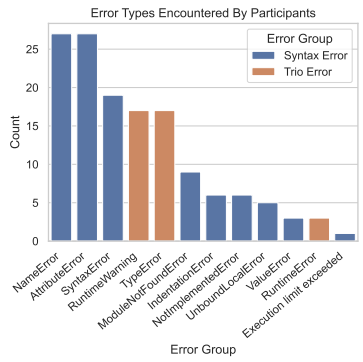

图 14: 参与者按错误类型遇到的所有错误的计数。

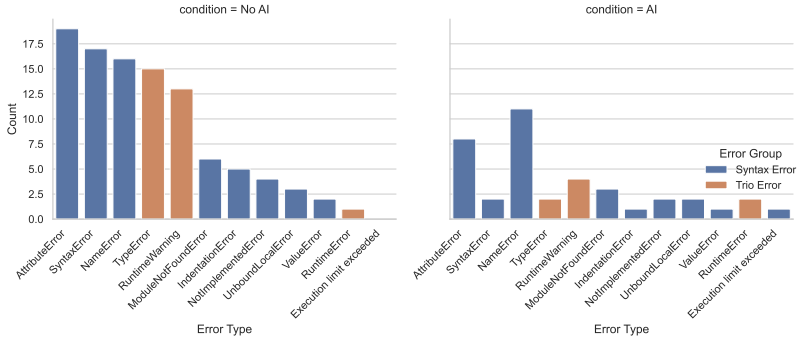

图 15: 按参与者条件计算的错误数量：AI（处理）和 No AI（对照）。对照组遇到了更多与关键 Trio 概念相关的错误（例如，`TypeError` 和 `RuntimeWarning`）。

错误和 Trio 技能 并非所有错误在我们的研究中对技能发展都具有相同的权重。某些错误需要对 Trio 库有更深入的理解，这可能会解释学习结果的差异。图 14 显示，最常见的错误与 Trio 库没有直接关系：`NameError` 和 `AttributeError` 通常是变量名或函数名上的拼写错误，可以快速更正。其他错误与 Trio 直接相关：当协程从未被等待时会出现 `RuntimeWarning`，当 trio 函数获取协程对象而不是异步函数时会出现 `TypeError`。这些错误强制理解 trio 库如何处理协程和使用 await 关键字的关键概念，这些在评估中进行了测试。虽然 AI 条件下的参与者也遇到错误（图 15），但遇到的 Trio 相关错误要少得多。

对于对照组的参与者，更高频率的遇到错误导致对代码发生的事情以及如何使用正在展示的新库进行更多的批判性思考。此外，特别与 Trio 库相关的错误的频繁出现确保了在完成任务的过程中获得这些特定概念。这两个差异表明，遇到错误可能在编码技能的形成中发挥重要作用。此外，我们对保持调试技能重要性的最初动机可能依赖于在不依赖 AI 的情况下获得这些技能。

6.3 主动编码时间的转移

虽然我们在主要分析中衡量的结果是总任务时间的生产力，但实际花费在主动编码上的时间量说明了一个更清晰的模式。图 16 显示，AI 条件下的参与者在任务上花费的主动时间少得多。这种从编码到阅读和理解的转移也在以前的工作中发现 [Becker et al., 2025]。当我们查看测验分数时，对照组在不使用 AI 的情况下以更高的总主动时间获得高测验分数。在每个条件内，更高的主动时间与更低的测验分数相关，这是因为更有经验的程序员在主动编码上花费更少的时间，同时与新手程序员相比拥有更好的基础知识。

6.4 参与者反馈

四分之一的参与者在完成任务和测验后留下了反馈。在对照组（No AI）中，参与者评论说他们发现任务很有趣，任务说明很好地帮助发展了对 Trio 的理解。在处理组（AI 辅助）中，参与者评论说他们希望他们在任务期间更多地关注 Trio 库的细节，要么通过阅读生成的代码，要么通过更深入地生成解释。具体来说，参与者报告感到“懒惰”，并且“（他们的）理解中仍然存在很多差距”。参与者反馈的情绪表明，即使任务说明和测验问题在组之间相同，对照组的体验更积极（表 6 和表 7 提供了所有参与者的所有参与者反馈）。

7 讨论

我们的主要发现是，使用 AI 完成需要新技能（即新 Python 库的知识）的任务会减少技能形成。在随机对照试验中，参与者被分配到处理条件（使用 AI 助手、网络搜索和说明）或对照条件（仅使用网络搜索和说明完成任务）。我们在使用 AI 辅助的参与者中测量的概念理解、代码阅读和调试技能的侵蚀表明，获得新技能的工人在学习过程中应该谨慎对待他们对 AI 的依赖。在使用 AI 的参与者中，我们发现高分交互模式（65%-86% 测验分数）与低分交互模式（24%-39% 测验分数）之间的技能形成结果存在明显分歧。高分者只向 AI 提出概念性问题，而不是代码生成，或者要求解释伴随生成的代码；这些使用模式表明了高水平的认知参与。

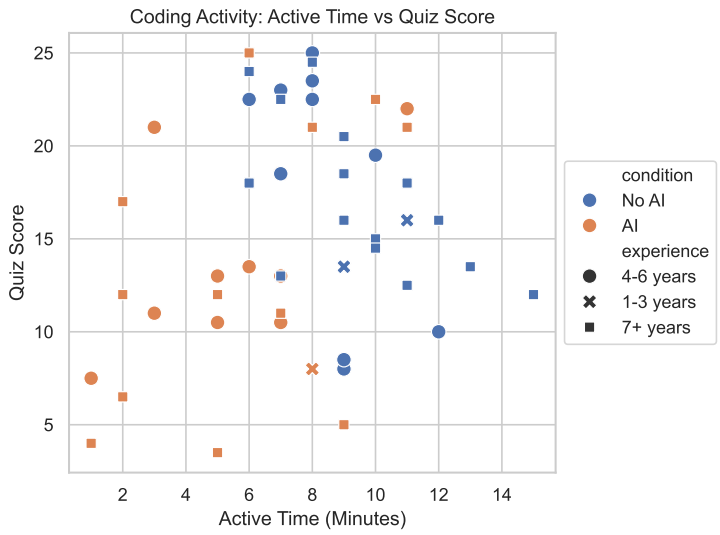

图 16: 主动编码时间与测验分数：主动编码时间表示实际花费编码的时间，通常只是总任务时间的很小一部分。No AI 条件的参与者在主动编码上花费了更多时间，并获得了更高的测验分数。

与我们最初的假设相反，我们没有在我们的主要研究中观察到任务完成的显着性能提升。虽然使用 AI 提高了任务的平均完成时间，但效率的提高在我们的研究中并不显着，尽管 AI 助手在被提示时可以生成完整的代码解决方案。我们的定性分析表明，我们的发现很大程度上是由于参与者在任务期间决定如何使用 AI 的异质性。有一组参与者依赖 AI 生成所有代码，从未提出概念性问题或解释。这个组的完成速度比对照组快得多（19.5 分钟 vs 23 分钟），但这组只占处理组参与者的 20% 左右。AI 组中提出大量查询（例如，15 个查询）、花费很长时间撰写查询（例如，10 分钟）或要求后续解释的其他参与者提高了平均任务完成时间。这些对比鲜明的 AI 使用模式表明，用新知识或技能完成任务不一定产生与只需要现有知识的任务相同的生产力收益。

总的来说，我们的结果表明，如果没有保持认知参与，激进地将 AI 纳入工作场所可能会对工人的专业发展产生负面影响。鉴于时间限制和组织压力，初级开发人员或其他专业人士可能会依赖 AI 尽可能快地完成任务，代价是真正的技能发展。此外，我们发现测验分数的最大差异在于调试问题。这表明，随着公司转向更多的人类监督的 AI 代码编写，如果他们的技能形成最初受到使用 AI 的抑制，人类可能不具备验证和调试 AI 编写的代码所需的技能。

7.1 未来工作

我们的工作是理解人机协作过程中 AI 辅助对人类影响的第一步。我们希望这项工作将激励解决以下限制的未来工作：

- **任务选择**：本研究侧重于使用基于聊天的界面的单个任务。这应该是认知卸载的下限，因为代理 AI 编码工具需要甚至更少的人类参与。在我们的工作中，没有思考依赖 AI 的用户在评估中表现最差；完全代理工具会产生类似的效果。未来工作应该研究代理编码工具对学习成果和技能发展的影响。
- **任务长度**：理想情况下，技能形成需要数月到数年的时间。我们在一小时期间测量了特定 Python 库的技能形成。未来工作应通过 AI 采用影响的纵向测量来研究现实世界的技能发展。
- **参与者现实性**：虽然我们研究中的参与者是专业或自由职业程序员，但学习该库的动力与实际工作所需的动力不同。未来研究应旨在研究真实公司中新手工人的技能习得。
- **提示技能**：我们收集了自我报告的对 AI 编码工具的熟悉程度，但我们实际上并不衡量提示技术的差异。我们工作的扩展还将测试超出自我报告的提示流利程度。
- **评估设计**：我们的研究通过综合测验衡量技能形成。其他研究可以使用完成另一项任务或将编码设计为替代评估策略。

- **人类辅助**：我们不包括反事实，即接受人类援助如何影响技能形成。由于人类援助和反馈发生在不同的环境中（例如，教室、结对编程、代码审查），未来工作可以比较 AI 与人类在所有这些环境中的反馈对技能形成的影响。

对于软件工程或任何其他行业的新手工人，我们的研究可以被视为尽管 AI 工具普遍存在，但仍需要有意的技能发展的价值的一小部分证据。我们的研究表明，即使在掌握过程中可能会遇到障碍（例如错误），但在遇到学习机会时部署认知努力以掌握新工具的好处。认知努力可以由 AI 协助；除了我们描述的模式之外，主要的 LLM 服务还提供学习模式（例如，ChatGPT 学习模式，Claude Code 学习/解释模式）。最终，为了适应 AI 时代的技能发展，需要对 AI 对工人的影响有更广阔的视野。新 AI 经济中的参与者不仅要关心从 AI 获得的生产力提升，还要关心随着新 AI 工具的激增，专业知识发展的长期可持续性。

8 致谢

我们要感谢 Ethan Perez、Miranda Zhang 和 Henry Sleight 通过 Anthropic Safety Fellows Program 使这个项目成为可能。我们还要感谢 Matthew Jörke、Juliette Woodrow、Sarah Wu、Elizabeth Childs、Roshni Sahoo、Nate Rush、Julian Michael 和 Rose Wang 提供实验设计反馈。

我们要感谢 Jeffrey Shen、Aram Ebtekar、Minh Le、Mateusz Piotrowski、Nate Rahn、Miles McCain、Jessica Zhu、Alex Wang、John Hewitt、Rosanne Hu、Saffron Huang、Kyle Hsu、Sanjana Srivastava 和 Jennifer Leung 进行任务测试和反馈。

参考文献

David Autor, Frank Levy, and Richard Murnane. The skill content of recent technological change: an empirical exploration, 2001.

Joel Becker, Nate Rush, Elizabeth Barnes, and David Rein. Measuring the impact of early-2025 ai on experienced open-source developer productivity. *arXiv preprint arXiv:2507.09089*, 2025.

Hannah Bleher and Matthias Braun. Diffused responsibility: attributions of responsibility in the use of ai-driven clinical decision support systems. *AI and Ethics*, 2(4):747–761, 2022.

Samuel R Bowman, Jeeyoon Hyun, Ethan Perez, Edwin Chen, Craig Pettit, Scott Heiner, Kamilė Lukošiūtė, Amanda Askell, Andy Jones, Anna Chen, et al. Measuring progress on scalable oversight for large language models. *arXiv preprint arXiv:2211.03540*, 2022.

Erik Brynjolfsson, Danielle Li, and Lindsey Raymond. Generative ai at work. *The Quarterly Journal of Economics*, page qjae044, 2025.

Zana Buçinca, Maja Barbara Malaya, and Krzysztof Z Gajos. To trust or to think: cognitive forcing functions can reduce overreliance on ai in ai-assisted decision-making. *Proceedings of the ACM on Human-computer Interaction*, 5(CSCW1):1–21, 2021.

Qingwan Cheng, Angela Tao, Huangliang Chen, and Maira Marques Samary. Design an assessment for an introductory computer science course: A systematic literature review. In *2022 IEEE frontiers in education conference (FIE)*, pages 1–8. IEEE, 2022.

Jonathan H Choi and Daniel Schwarcz. Ai assistance in legal analysis: An empirical study. 2023.

Zheyuan Kevin Cui, Mert Demirer, Sonia Jaffe, Leon Musolff, Sida Peng, and Tobias Salz. The effects of generative ai on high skilled work: Evidence from three field experiments with software developers. *Available at SSRN 4945566*, 2024.

Fabrizio Dell'Acqua, Edward McFowland III, Ethan R Mollick, Hila Lifshitz-Assaf, Katherine Kellogg, Saran Rajendran, Lisa Krayer, François Candelon, and Karim R Lakhani. Navigating the jagged technological frontier: Field experimental evidence of the effects of ai on knowledge worker productivity and quality. *Harvard Business School Technology & Operations Mgt. Unit Working Paper*, (24-013), 2023.

Stefania Druga, Randi Williams, Cynthia Breazeal, and Mitchel Resnick. " hey google is it ok if i eat you?" initial explorations in child-agent interaction. In *Proceedings of the 2017 conference on interaction design and children*, pages 595–600, 2017.

Michael Gerlich. Ai tools in society: Impacts on cognitive offloading and the future of critical thinking. *Societies*, 15(1):6, 2025.

Javier Gonzalez-Huerta, Jefferson Seide Molléri, Aivars Šablis, and Ehsan Zabardast. Experiential learning approach for software engineering courses at higher education level. *arXiv preprint arXiv:2012.14178*, 2020.

Kunal Handa, Alex Tamkin, Miles McCain, Saffron Huang, Esin Durmus, Sarah Heck, Jared Mueller, Jerry Hong, Stuart Ritchie, Tim Belonax, et al. Which economic tasks are performed with ai? evidence from millions of claude conversations. *arXiv preprint arXiv:2503.04761*, 2025.

Suhas Kannam, Yuri Yang, Aarya Dharm, and Kevin Lin. Code interviews: Design and evaluation of a more authentic assessment for introductory programming assignments. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1*, pages 554–560, 2025.

Zachary Kenton, Noah Siegel, János Kramár, Jonah Brown-Cohen, Samuel Albanie, Jannis Bulian, Rishabh Agarwal, David Lindner, Yunhao Tang, Noah Goodman, et al. On scalable oversight with weak llms judging strong llms. *Advances in Neural Information Processing Systems*, 37:75229–75276, 2024.

Artur Klingbeil, Cassandra Grützner, and Philipp Schreck. Trust and reliance on ai—an experimental study on the extent and costs of overreliance on ai. *Computers in Human Behavior*, 160:108352, 2024.

David A Kolb. *Experiential learning: Experience as the source of learning and development*. FT press, 2014.

Hao-Ping Lee, Advait Sarkar, Lev Tankelevitch, Ian Drosos, Sean Rintel, Richard Banks, and Nicholas Wilson. The impact of generative ai on critical thinking: Self-reported reductions in cognitive effort and confidence effects from a survey of knowledge workers. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pages 1–22, 2025.

Jack B Longwell, Ian Hirsch, Fernando Binder, Galileo Arturo Gonzalez Conchas, Daniel Mau, Raymond Jang, Rahul G Krishnan, and Robert C Grant. Performance of large language models on medical oncology examination questions. *JAMA Network Open*, 7(6):e2417641–e2417641, 2024.

Brooke N Macnamara, Ibrahim Berber, M Cenk Çavuşoğlu, Elizabeth A Krupinski, Naren Nallapareddy, Noelle E Nelson, Philip J Smith, Amy L Wilson-Delfosse, and Soumya Ray. Does using artificial intelligence assistance accelerate skill decay and hinder skill development without performers' awareness? *Cognitive Research: Principles and Implications*, 9(1):46, 2024.

Negar Maleki, Balaji Padmanabhan, and Kaushik Dutta. Ai hallucinations: a misnomer worth clarifying. In *2024 IEEE conference on artificial intelligence (CAI)*, pages 133–138. IEEE, 2024.

Shakked Noy and Whitney Zhang. Experimental evidence on the productivity effects of generative artificial intelligence. *Science*, 381(6654):187–192, 2023.

Nicholas Otis, Rowan Clarke, Solene Delecourt, David Holtz, and Rembrand Koning. The uneven impact of generative ai on entrepreneurial performance. 2024.

Zelin Pan, Zhendong Xie, Tingting Liu, and Tiansheng Xia. Exploring the key factors influencing college students' willingness to use ai coding assistant tools: An expanded technology acceptance model. *Systems*, 12(5):176, 2024.

Sida Peng, Eirini Kalliamvakou, Peter Cihon, and Mert Demirer. The impact of ai on developer productivity: Evidence from github copilot. *arXiv preprint arXiv:2302.06590*, 2023.

Gustavo Pinto, Cleidson De Souza, Thayssa Rocha, Igor Steinmacher, Alberto Souza, and Edward Monteiro. Developer experiences with a contextualized ai coding assistant: Usability, expectations, and outcomes. In *Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering-Software Engineering for AI*, pages 81–91, 2024.

Eric Poitras, Brent Crane, and Angela Siegel. Generative ai in introductory programming instruction: Examining the assistance dilemma with llm-based code generators. In *Proceedings of the 2024 on ACM Virtual Global Computing Education Conference V. 1*, pages 186–192, 2024.

Siddhartha Prasad, Ben Greenman, Tim Nelson, and Shriram Krishnamurthi. Generating programs trivially: student use of large language models. In *Proceedings of the ACM Conference on Global Computing Education Vol 1*, pages 126–132, 2023.

Omer Reingold, Judy Hanwen Shen, and Aditi Talati. Dissenting explanations: leveraging disagreement to reduce model overreliance. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 21537–21544, 2024.

Patrizia Ribino. The role of politeness in human–machine interactions: a systematic literature review and future perspectives. *Artificial Intelligence Review*, 56(Suppl 1):445–482, 2023.

Henk G Schmidt. Problem-based learning: An introduction. *Instructional science*, pages 247–250, 1994.

Judy Hanwen Shen and Carlos Guestrin. Societal impacts research requires benchmarks for creative composition tasks. *arXiv preprint arXiv:2504.06549*, 2025.

Alex Tamkin, Miles McCain, Kunal Handa, Esin Durmus, Liane Lovitt, Ankur Rathi, Saffron Huang, Alfred Mountfield, Jerry Hong, Stuart Ritchie, et al. Clio: Privacy-preserving insights into real-world ai use. *arXiv preprint arXiv:2412.13678*, 2024.

Helena Vasconcelos, Matthew Jörke, Madeleine Grunde-McLaughlin, Tobias Gerstenberg, Michael S Bernstein, and Ranjay Krishna. Explanations can reduce overreliance on ai systems during decision-making. *Proceedings of the ACM on Human-Computer Interaction*, 7(CSCW1):1–38, 2023.

Dakuo Wang, Elizabeth Churchill, Pattie Maes, Xiangmin Fan, Ben Shneiderman, Yuanchun Shi, and Qianying Wang. From human-human collaboration to human-ai collaboration: Designing ai systems that can work together with people. In *Extended abstracts of the 2020 CHI conference on human factors in computing systems*, pages 1–6, 2020.

Wei Wang, Huilong Ning, Gaowei Zhang, Libo Liu, and Yi Wang. Rocks coding, not development: A human-centric, experimental evaluation of llm-supported se tasks. *Proceedings of the ACM on Software Engineering*, 1(FSE):699–721, 2024.

Emma Wiles, Lisa Krayer, Mohamed Abbadi, Urvi Awasthi, Ryan Kennedy, Pamela Mishkin, Daniel Sack, and François Candelon. Genai as an exoskeleton: Experimental evidence on knowledge workers using genai on new skills. *Available at SSRN 4944588*, 2024.

Suqing Wu, Yukun Liu, Mengqi Ruan, Siyu Chen, and Xiao-Yun Xie. Human-generative ai collaboration enhances task performance but undermines human' s intrinsic motivation. *Scientific Reports*, 15(1):15105, 2025.

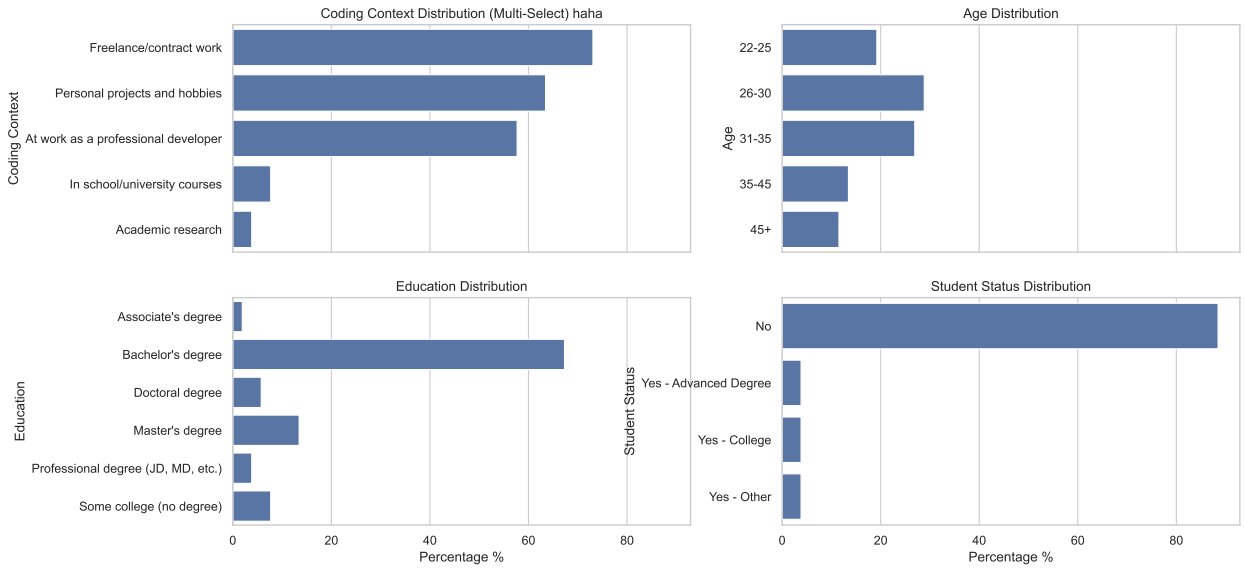

图 17: 主要研究的参与者分布，在任务后收集以避免刻板印象威胁。大多数参与者是专业程序员。

A 参与者详细信息

A.1 伦理审查

该方案由 Anthropic 的内部审查员进行了审查和批准。参与者在本研究中没有接触到任何风险。有理由预期从本研究中获益的是学习一项新的软件（Python）技能。我们不保证或承诺参与者将从本研究中获得任何特定的学习益处。我们在预筛选阶段收集了参与研究的知情同意。我们给予参与者在任何时候撤回同意而不受惩罚的权利。即使他们未通过注意检查，未完全完成任务或错误完成任务，他们仍将获得补偿。测验响应存储在 Google Drive 中，编码击键存储在编码平台中。所有存储的信息都完全匿名；只有数据平台可以使用 ID 来识别参与者以进行付款。我们进一步删除数据平台标识符，以注释两组之间的编码模式。

B 定性分析数据和详细信息

B.1 标注程序

51/52 名参与者上传了他们在热身任务、主要编码任务和测验中的工作屏幕录制。我们观看了所有参与者（25 名 AI 条件，25 名不 AI 条件）主要编码任务的录制。我们记录以下事件的时间戳：

我们还根据这些事件代码记录了参与者如何在这些条件下使用 AI 的一般主题。

B.2 数据可用性

我们在以下 URL 提供每个参与者的标注记录：`https://github.com/safety-research/how-ai-impacts-skill-formation`。

B.3 参与者反馈详细信息

我们在表 6 和表 7 中包含了所有参与者的反馈。由于 AI 组的平均完成时间更快，AI 组留下了更多评论，因为他们在任务结束时感觉有更多时间。

事件	描述	附加信息
任务开始	用户打开每个任务	
AI 交互	用户开始在 AI 窗口中输入	交互描述
AI 查询	用户收到来自 AI 的答案	查询
网络搜索	用户查询搜索引擎	查询
粘贴（直接）	用户粘贴 AI 助手的输出	
代码复制	用户使用 AI 输入编写代码	
错误	代码运行时产生错误	错误信息
界面错误	开发环境或 AI 助手错误	
任务完成	实现正确的输出	
任务提交	用户提交任务	代码完成

表 5: 为每个主任务的视频录制手动标注的事件。

条件	反馈
AI	如果我可以直接查找追加数组的语法，第一个测验（打印 *s）会容易得多。我在做任务时忘记了这一点 - 通常，在谷歌上搜索需要五秒钟，不需要模型辅助，但由于我无法离开标签页，我被卡住了。
AI	通过使用 AI 助手，我觉得我变得懒惰了。我没有像 otherwise 那样仔细地阅读 Trio 库介绍和代码示例。
AI	很酷的项目，希望一切顺利！
AI	我不介意更深入地与助手合作，真正理解并证明 trio 库的细节。我觉得我对它有了相当不错的概述，但我的理解中仍然存在很多空白。
AI	我从热身中感到很愚蠢，但希望另一个项目展示了我能做的事情。抱歉，如果我浪费了你的时间。
AI	这既有趣又具有挑战性。热身令人困惑，因为任务似乎有一些问题，但整个过程是一个有趣的学习体验。
AI	我希望我花一点时间理解 Cosmo 的解释！
AI	我很慢。我认为时间限制让我以不代表我正常工作流程的方式行事，特别是在构建心理模型与获得代码进展所花费的时间比例上。我希望能比我在当下允许自己的方式更深入地理解 trio，因为我知道我没有时间。我认为我的注意力很分散，因为我试图匆忙操作。我意识到我对 Trio 的工作方式与其他库相当相似，但实际上并不像我想象的那样熟悉。这之后我的感觉是，执行模型可能存在差异，但我没有真正深入挖掘以理解它们。令我惊讶的是，像理解 ‘start_soon‘ 方法这样简单的事情..... 就像我在更深层次的理解方面没有学到任何东西。感谢你让我参与！
AI	这很有趣！我希望我在使用 Cosmo 编码时更多地关注 trio 语法

表 6: AI 条件参与者的反馈

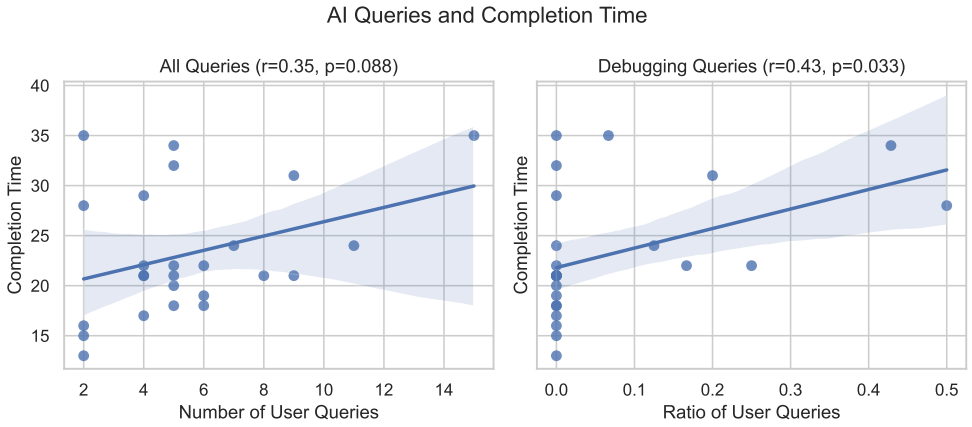

图 18: 随着查询数量的增加，总完成时间也增加，询问大量调试查询的用户也倾向于花费更多时间来完成任务。

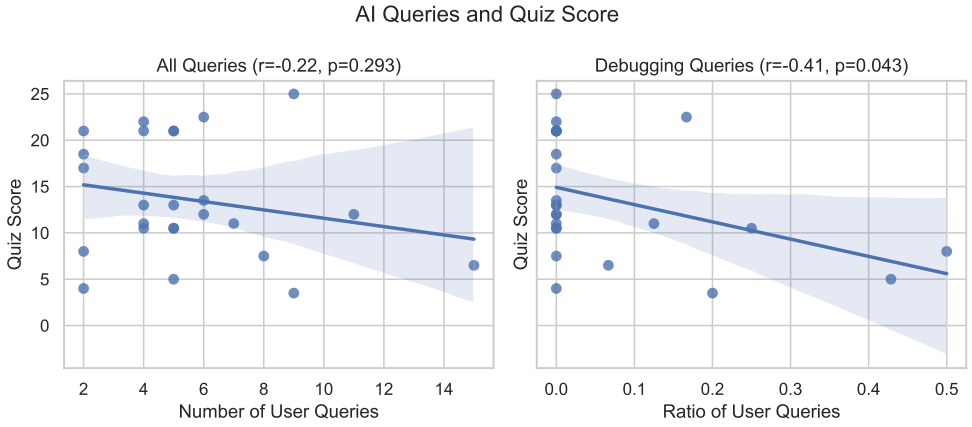

图 19: 总查询数量与测验分数之间没有明显的模式。然而，严重依赖 AI 进行调试的用户往往测验分数较低。

C 评估详细信息

C.1 评估设计

问题类型我们讨论了我们使用的三种问题类型：概念性理解、代码阅读和调试，在第 4.2 节中。

知识类别评估涵盖了 Trio 库的 7 个核心概念：

1. **Async 和 await 关键字**：何时在异步函数中使用 await 关键字。例如：“当在异步函数中使用 await 关键字时会发生什么？”
2. **启动 Trio 函数**：基本的 Trio 用法，包括如何生成任务以及具有不同持续时间的生成任务的行为。
3. **Trio 中的错误处理**：理解错误传播模式以及如何在子任务中捕获错误。例如“当子任务在 Trio 中引发未处理的异常时，父任务会发生什么？”
4. **协程：**调用异步函数时，如何调试协程从未等待的错误。
5. **使用 Trio 的内存通道**：理解 `start_soon` 不返回任何内容以及如何在并行运行多个任务时使用字典、列表和其他内存通道来收集数据。

条件	反馈
No AI	这很有趣，但在某些系统上录制方面可能很麻烦，并引起一些焦虑，特别是如果你搞砸了录制就无法回去。
No AI	学习异步编程很有趣，我以前从未遇到过。如果我可以访问我在第 2 部分完成的编码任务以在测验中进行参考，我认为我可以做得更好，但我仍然尽力了。我时间用完了，因为调试问题对我来说很具挑战性。
No AI	这很有趣
No AI	编程任务非常有趣，并且在帮助我理解 Trio 如何工作方面做得很好，尽管我以前从未使用过它。我在这个测验上花费了太多时间，但这是由于我的时间管理。即使我没有在第一部分花费太多时间，我认为在 30 分钟的窗口内我仍然会很紧张。

表 7: No AI（对照）条件参与者的反馈

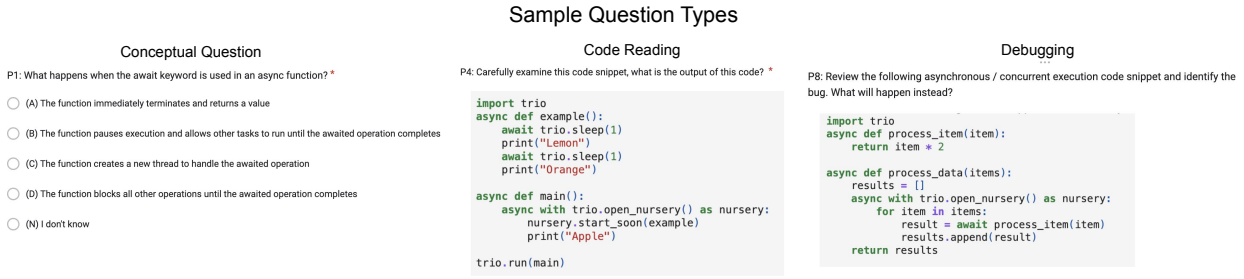

图 20: 我们评估中的示例问题类型。我们设计评估来测试三种不同的软件技能：概念性理解、代码阅读和代码编写。

6. **打开和关闭 Trio nursery**：理解异步上下文管理器以及如何使用它们。例如，一个调试问题，其中代码片段中的 nursery 启动正确。
7. **顺序执行与并发执行：**并发任务的预期行为。例如 “阅读以下代码并识别每个任务何时开始和完成”

D 任务详细信息

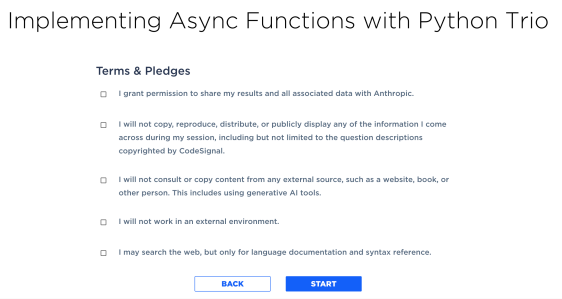

图 21: 对照组参与者采取的承诺：参与者同意不使用 AI 辅助。

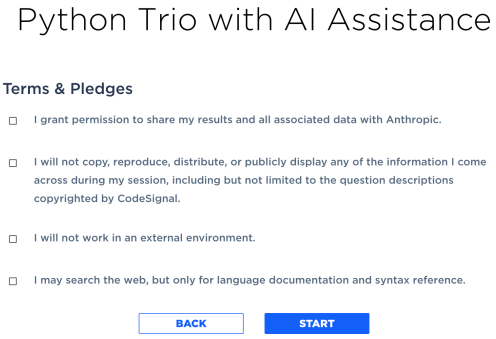

图 22: 处理组参与者采取的承诺。

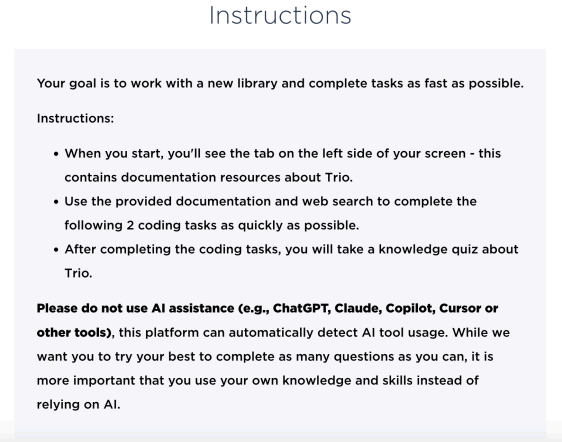

图 23: 给予对照组参与者的说明。我们严重强调不使用 AI 工具。

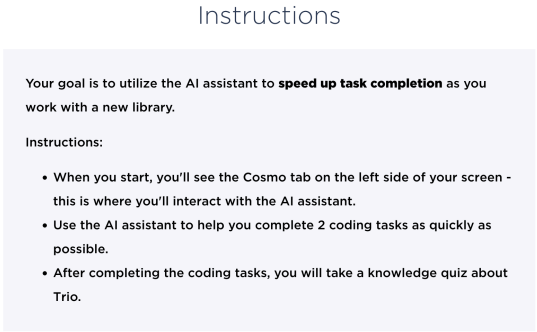

图 24: 给予处理组参与者的说明。该组被鼓励使用 AI 助手尽快完成任务。

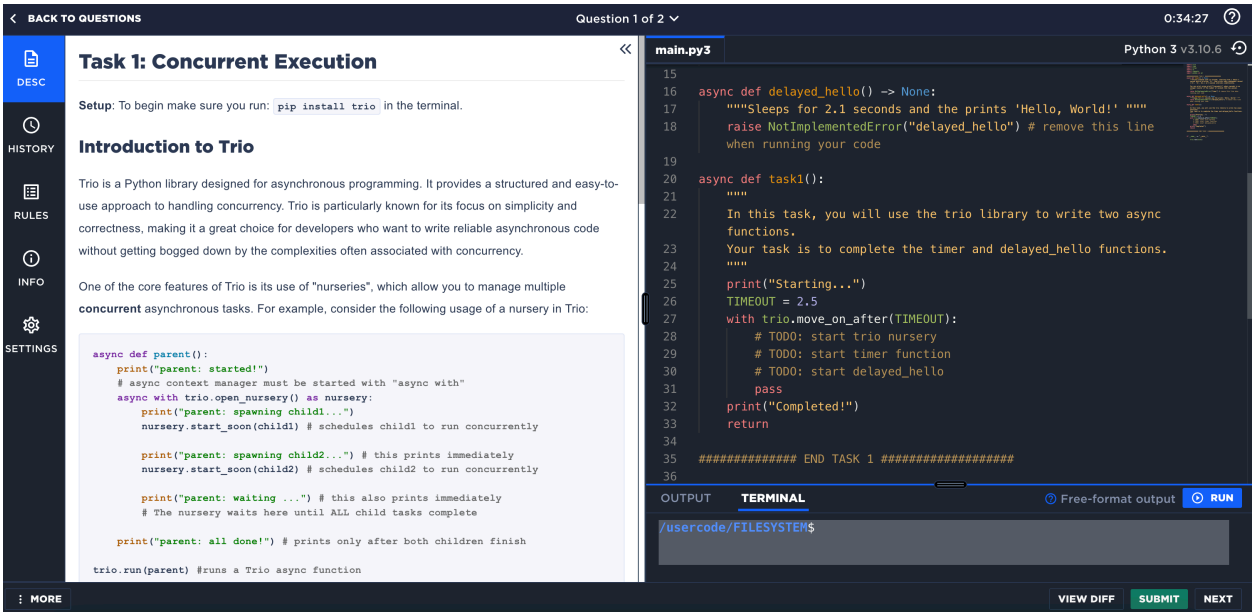

图 25: 对照条件下任务平台的屏幕截图。说明在左侧，编码编辑器在右侧。

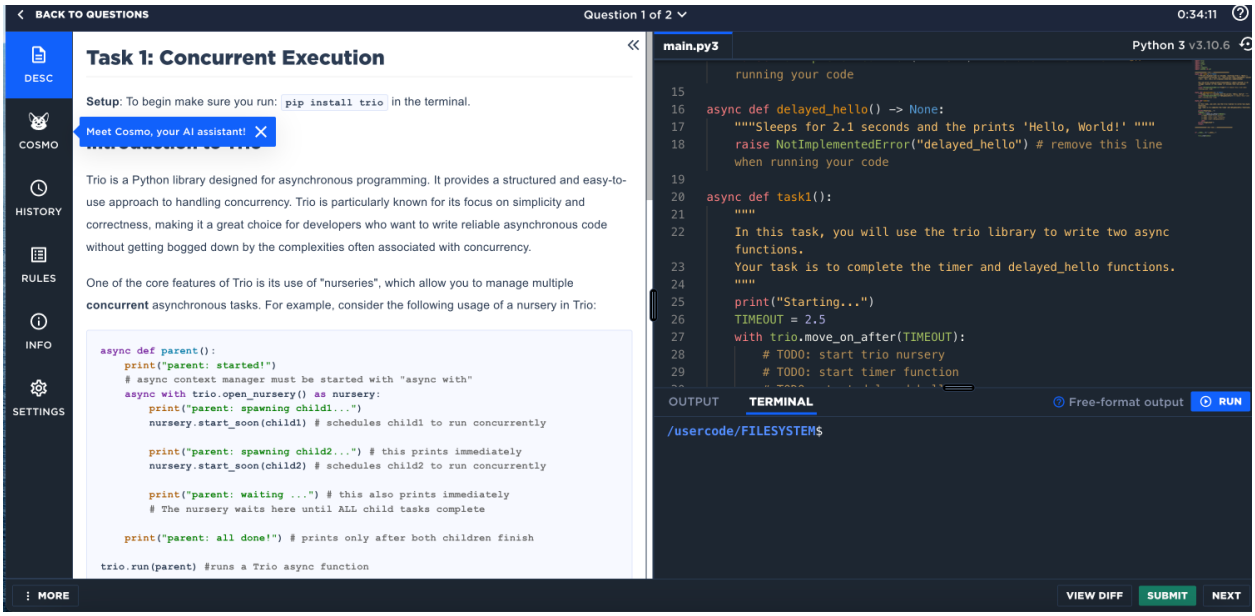

图 26: AI 条件下任务平台的屏幕截图。说明在左侧，编码编辑器在右侧。左侧工具平面上有一个使用 AI 助手的提示。

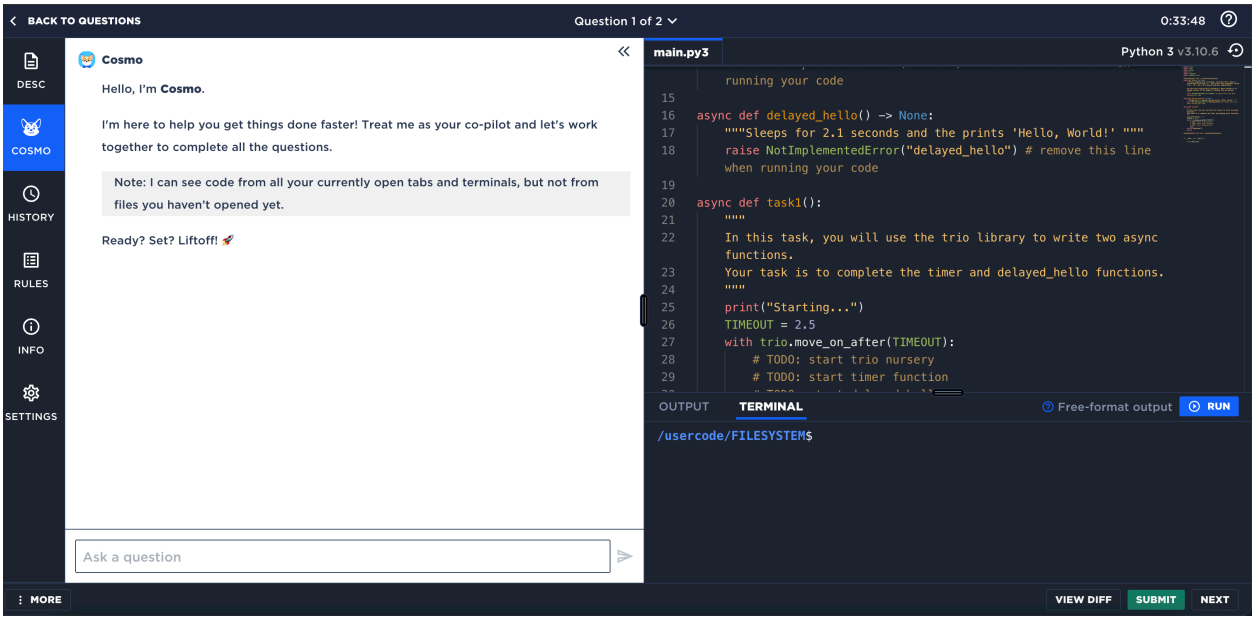

图 27: 与 AI 助手交互时任务平台的屏幕截图。