

Designing Purpose-first Instruction for Supporting End-user Programmer Learning Goals

Mehmet Arif Demirtaş

University of Illinois Urbana-Champaign
mad16@illinois.edu

Abstract

Computing knowledge is valuable to people besides just software developers. People from various disciplines use computing to create programs that support their personal interests and address domain-specific tasks. These learners, often called *end-user programmers*, are most interested in learning about the goals that can be achieved with programming and creating programs that satisfy these goals using high-level tools. In contrast, conventional computing education materials prioritize writing code from scratch by emphasizing data structures, language syntax, and software development practices. As a result, computing education fails to align with the needs of end-user programmers.

My research aims to improve computing education for end-user programmers through two synergistic goals. First, I explore the effectiveness of instructional materials that emphasize the purpose of code with students who may be interested in becoming end-user programmers. Second, I work with instructors to understand how they teach computing in diverse disciplines. Informed by instructor values and practices, I plan to design systems for creating instructional material aimed at end-user programmers.

1. Introduction and Background

Computing can be valuable for everyone: from artists (Chung & Guo, 2024) to scientists (O'Brien, 2025), professionals across various disciplines use computing in ways that differ both from each other and from the conventional learning goals in CS courses. These professionals are described as *end-user programmers*, as they develop programs for goals that support personal interests or tasks in their disciplines (Ko et al., 2011). End-user programmers often use opportunistic programming strategies (Brandt, Guo, Lewenstein, & Klemmer, 2008), such as following trial-and-error strategies, modifying similar code examples instead of writing code from scratch, and implementing functionalities without focusing on maintainability and scalability. These programmers are interested in using programming to create artifacts with complex functionality rather than studying theoretical knowledge that might be necessary for building robust and scalable programs (Ali & Dasgupta, 2025).

However, existing computing instruction values software development endpoints over end-user programming practices. Formal instructional resources are designed with minimal input from students and educators working with computing in non-CS contexts, and emphasize syntax over applications (Becker & Fitzpatrick, 2019). Some computing education researchers argue that top-down approaches that emphasize the purpose of the code at a high level are less effective for learning, compared to bottom-up approaches that focus on fundamental skill development (Thorgeirsson, Ewen, & Su, 2025). This is also reflected by end-user programmers' preference for informal learning environments, such as online resources, over formal instructional materials (Dorn & Guzdial, 2010).

In my work, I aim to improve computing education for end-user programming goals by investigating two main research questions:

- **RQ1:** What instructional techniques help students who are interested in end-user programmer learning goals?
- **RQ2:** What practices do instructors in various disciplines employ for supporting end-user programmer learning goals?

I explore the first research question through classroom studies with non-CS majors, who are likely to pursue end-user programmer learning goals (Hur & Cunningham, 2024). I explore the second research question through human-centered design projects with instructors to learn from practices they have adopted and design systems that can spread these practices to larger audiences.

2. Research Progress

2.1. RQ1: Evaluating instructional techniques for end-user programmer learning goals

2.1.1. Supporting students with purpose-first tutorials (SIGCSE 2026)

In this work (Demirtas, Zheng, & Cunningham, 2026), we built on *purpose-first programming* (Cunningham, Ericson, Agrawal Bejarano, & Guzdial, 2021) to design three programming tutorials for non-major students. These tutorials (covering SQL for database queries, Python for web scraping, and Python for HTTP requests) were developed for students with minimal programming background.

After using these tutorials in the hands-on lab sessions for an introductory programming course, we observed that students felt significantly more confident in their ability to use programming for domain-specific tasks, as shown by comparisons of pre/post surveys. Moreover, students were able to discuss potential use cases of each programming tool and discuss challenges and tradeoffs of these tools in their open-ended responses to several activities. Our findings suggested that instruction that focuses on code's *purpose* can be effective in courses with non-major students who are aligned with end-user programmer learning goals, especially for supporting affective outcomes and for building conceptual understanding.

2.1.2. Comparing top-down and bottom-up approaches to computing instruction (in progress)

While our first study showed promising results for certain affective outcomes, we noted that similar top-down approaches that prioritize code's purpose have been hypothesized to be ineffective for learning in recent literature (Thorgeirsson et al., 2025). In our ongoing work, we investigate this further by comparing a purpose-first tutorial on a data visualization library (Seaborn) as an example of top-down instruction to a popular online tutorial covering the same library through a bottom-up approach. The purpose-first tutorial is organized so that students start by seeing examples of authentic problems that are solved with the data visualization library, which are then broken down into subgoals and functions. In contrast, the bottom-up tutorial covers each function subsequently. Our objective was to compare whether there were any differences in learning outcomes between students who were randomly assigned to complete one of two tutorials. Our preliminary findings indicate that students from both groups performed on a statistically identical level according to equivalence testing on assessments of coding ability, syntax knowledge, and conceptual understanding. Our initial results indicate that top-down instruction can be as effective as bottom-up instruction in the context of learning a new library, while also having a higher alignment with what end-user programmers look for in learning resources.

2.2. RQ2: Learning from practices employed by instructors

2.2.1. Supporting instructors as they create inclusive computing instruction (CHI 2025)

Our work with students demonstrates the potential value of authentic examples in instruction. However, curating these examples can be time-consuming for instructors. In this work (Jain, Demirtas, & Cunningham, 2025), we designed a system in conversation with computing instructors to alleviate the workload of finding examples of authentic applications for various programming libraries.

Our design process started with interviews with computing instructors. We observed that their workflows can be improved through automation: to find authentic programming applications, they needed to collect examples across many sources, identify code lines common between examples, and interpret and decide which examples would be useful for students. We proposed an AI-supported workflow to identify common code lines so that instructors could focus their expertise on interpreting and refining the generated draft for students. We explored this hypothesis through design workshops with seven computing instructors, revealing design considerations on how instructors want AI support to be integrated into their workflows. Informed by these workshops, we developed a system for identifying common patterns in authentic programming examples, and instructors were more productive and experienced lower cognitive demands with our system compared to a baseline condition.

The key contribution of our work is to show that AI support in educational contexts can be beneficial when it follows practices already valued by instructors. Given that there is a major push to incorporate AI systems in education, we provide a framework for centering instructor voices in such systems. For instance, many participants praised the interactions we provided for quickly navigating examples at a high level while explicitly complaining about the chat interfaces common in many GenAI tools, showing the importance of interactions that align with instructor needs.

2.2.2. Co-designing curriculum design tools with non-CS instructors (in progress)

Our prior work has shown that instructors have practices that can be supported effectively with automated support tools. In this work, we plan to identify additional practices that have proved to be efficient in supporting end-user programmer learning goals. Most importantly, we are planning to interview instructors who are teaching computing in non-CS domains to amplify voices that are outside the CS sphere of influence. These interviews will inform co-design sessions for tools that support the development of instruction that follows the best practices collected from non-CS instructors, which, in turn, could support broader adoption of effective instructional techniques for end-user programmers.

In our initial findings from the interviews for this project, we noted that instructors value live coding and showcasing authentic examples and use cases of tools they are covering in their courses. For most instructors, ensuring that students can apply the knowledge they acquire in the course in their own projects is a key goal, and their instructional tasks are designed with this goal in mind. Therefore, our initial findings seem to support the importance of purpose-first instruction for end-user programmers.

3. Conclusion & Next Steps

In my past work, I focused on showing the importance of emphasizing code's purpose for learners interested in end-user programming. My ongoing and proposed research will focus on identifying concrete practices for highlighting code's purpose, and co-designing tools with instructors to support this mode of instructional design. I am seeking feedback on what these practices could entail, as well as methodological advice for conducting co-design sessions that empower instructors as the main stakeholders.

4. References

- Ali, M., & Dasgupta, S. (2025). "even though i went through everything, i didn't feel like i learned a lot": Insights from experiences of non-computer science students learning to code. In *Proceedings of the 2025 chi conference on human factors in computing systems*. New York, NY, USA: Association for Computing Machinery. Retrieved from <https://doi-org.proxy2.library.illinois.edu/10.1145/3706598.3713624> doi: 10.1145/3706598.3713624
- Becker, B. A., & Fitzpatrick, T. (2019). What do cs1 syllabi reveal about our expectations of introductory programming students? In *Proceedings of the 50th acm technical symposium on computer science education* (pp. 1011–1017).
- Brandt, J., Guo, P. J., Lewenstein, J., & Klemmer, S. R. (2008, May). Opportunistic programming: How rapid ideation and prototyping occur in practice. In *Proceedings of the 4th international workshop on End-user software engineering* (pp. 1–5). Leipzig Germany: ACM. doi: 10.1145/1370847.1370848
- Chung, A. M., & Guo, P. J. (2024). Perpetual teaching across temporary places: Conditions, motivations, and practices of media artists teaching computing workshops. In *Proceedings of the 2024 acm conference on international computing education research-volume 1* (pp. 374–388).
- Cunningham, K., Ericson, B. J., Agrawal Bejarano, R., & Guzdial, M. (2021, May). Avoiding the Turing Tarpit: Learning Conversational Programming by Starting from Code's Purpose. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (pp. 1–15). Yokohama Japan: ACM.
- Demirtas, M. A., Zheng, C., & Cunningham, K. (2026). Teaching Authentic Programming Applications to Novices: Purpose-first Tutorials in a General Education Computing Course. In *Proceedings of the 57th ACM Technical Symposium on Computer Science Education V.1* (Vol. 1, pp. 274–280). Association for Computing Machinery. Retrieved 2026-04-23, from <https://dl.acm.org/>

doi/10.1145/3770762.3772574

- Dorn, B., & Guzdial, M. (2010). Discovering computing: perspectives of web designers. In *Proceedings of the sixth international workshop on computing education research* (p. 23–30). New York, NY, USA: Association for Computing Machinery. Retrieved from <https://doi.org/10.1145/1839594.1839600> doi: 10.1145/1839594.1839600
- Hur, J., & Cunningham, K. (2024, August). Profiling Conversational Programmers at University: Insights into their Motivations and Goals from a Broad Sample of Non-Majors. In *Proceedings of the 2024 ACM Conference on International Computing Education Research - Volume 1* (Vol. 1, pp. 293–311). New York, NY, USA: Association for Computing Machinery. doi: 10.1145/3632620.3671123
- Jain, Y., Demirtas, M. A., & Cunningham, K. I. (2025). Plaid: Supporting computing instructors to identify domain-specific programming plans at scale. In *Proceedings of the 2025 chi conference on human factors in computing systems*. New York, NY, USA: Association for Computing Machinery. Retrieved from <https://doi-org.proxy2.library.illinois.edu/10.1145/3706598.3713832> doi: 10.1145/3706598.3713832
- Ko, A. J., Abraham, R., Beckwith, L., Blackwell, A., Burnett, M., Erwig, M., ... Wiedenbeck, S. (2011, April). The state of the art in end-user software engineering. *ACM Comput. Surv.*, 43(3). Retrieved from <https://doi-org.proxy2.library.illinois.edu/10.1145/1922649.1922658> doi: 10.1145/1922649.1922658
- O'Brien, G. (2025). How scientists use large language models to program. In *Proceedings of the 2025 chi conference on human factors in computing systems*. New York, NY, USA: Association for Computing Machinery. Retrieved from <https://doi-org.proxy2.library.illinois.edu/10.1145/3706598.3713668> doi: 10.1145/3706598.3713668
- Thorgeirsson, S., Ewen, T., & Su, Z. (2025). What Can Computer Science Educators Learn From the Failures of Top-Down Pedagogy? In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1* (pp. 1127–1133). ACM. Retrieved 2026-04-17, from <https://dl.acm.org/doi/10.1145/3641554.3701873> doi: 10.1145/3641554.3701873