Learning to Program with LLMs: How University Students Use LLMs in an Introductory Programming Course

Introduction

The potential of Large Language Models (LLMs) to enhance programming education has attracted widespread interest among researchers and educators (Finnie-Ansley et al., 2022). Prior research has actively investigated the usability and pedagogical potential of LLMs in programming education from educators' perspective, such as LLM-powered tool design and curriculum development around LLM capabilities (see Jury et al., 2024; Vadaparty et al., 2024). To create more targeted and pedagogically supportive LLM-based learning experiences, it is crucial to gain a deeper understanding of students' authentic practices when using LLMs for programming learning (Hellas et al., 2024). These use patterns not only reflect students' learning focus and priorities, but also illuminate areas where further instructional support is most needed. However, few studies have systematically examined how students interact with LLMs in authentic learning contexts (Ma et al., 2024; Scholl & Kiesler, 2024). Moreover, existing research seldom probes into students' reflections like intentions and challenges underlying specific uses, failing to adequately reveal the nuanced processes through which students engage with LLMs to support their programming learning.

Against this backdrop, the present study employed in-depth interviews and qualitative content analysis to explore how students meaningfully engage with LLMs in an authentic introductory programming learning context, aiming to provide practical insights to inform the design of more tailored and effective LLM-based learning experiences.

This study addresses the following research question:

How do students use LLMs to support their programming learning in an introductory programming course?

Literature Review

LLMs are widely considered to have the potential to significantly impact programming education in higher education (Finnie-Ansley et al., 2022), especially in introductory programming contexts. Several studies have demonstrated LLMs' capabilities in solving programming problems (Finnie-Ansley et al., 2022), generating code explanations (Narayanan et al., 2024), and enhancing error messages (Leinonen et al., 2023)—features that can serve as powerful means to support students' learning. With LLMs' widespread availability (Vadaparty et al., 2024) and promising potential to support programming education, students have begun actively embracing the use of LLMs in their learning processes (Ma et al., 2024; Scholl & Kiesler, 2024).

Many educators and researchers have explored how LLMs can be effectively integrated into introductory programming courses from the perspectives of educators, with a focus on LLM usability, interface design, and pedagogical potential. For instance, some have developed LLM-powered tools to generate worked examples (Jury et al., 2024), others have designed GPT-4-based interactive homework assistants (Zamfirescu-Pereira et al., 2023). There are also efforts to restructure courses around LLMs by shifting emphasis away from syntax and code writing from scratch (Vadaparty et al., 2024). These developments highlight the need for empirical insights into students' authentic LLM usage to inform the design of more targeted and personalized LLM-supported learning experiences (Hellas et al., 2024).

However, there has been limited investigation of students' authentic LLM usage practices. Recent literature examining the LLM usage in programming education has primarily focused on evaluating their instructional effectiveness (Katona & Gyonyoru, 2025; Lyu et al., 2024). Only a

few studies have looked into how students interact with LLMs, identifying common use cases such as code explanation, optimization, and correction (Ma et al., 2024; Scholl & Kiesler, 2024). However, these studies offer only surface-level categorizations without systematic analysis, limiting the comprehensive understanding of how such LLM engagement supports learning. Moreover, much of this research relies on self-reported survey data or chat logs, which detach students' LLM usage behaviors from specific learning contexts and limit insights into the underlying learning intentions, concrete strategies, and challenges, thereby failing to capture the nuanced ways in which LLMs are meaningfully leveraged. These limitations underscore a critical need for research that systematically and deeply explores how students meaningfully engage with LLMs in authentic contexts—providing insights essential for designing effective LLM-based tools and interventions.

Methods

This study took place in a university-level introductory Python programming course for education majors. Students engaged in guided learning through weekly readings, videos, and pre-class coding exercises, followed by in-class instructor-led quizzes and concept reviews. A final programming project was required. Students were encouraged to use LLMs to support learning throughout the course. A customized programming chatbot, *Coding Buddy*, built on OpenAI's GPT platform with a ChatGPT-like interface, was made accessible to students via a shared link. In the interviews, though other LLM-powered tools were occasionally mentioned, most reported uses centered on Coding Buddy and ChatGPT.

Six novice students who actively used LLMs throughout the course were recruited for semi-structured interviews. Each interview began with a central guiding question—*How did you use LLMs to support your programming learning in the course?* —to anchor conversation on LLM

usage, followed by adaptive probes to elicit students' reflections on specific LLM uses (such as their intentions, challenges, and dilemmas) (Gill et al., 2008). This approach was adopted to capture the nuanced, context-sensitive nature of LLM engagement. All interviews were audio-recorded with participants' informed consent and subsequently transcribed verbatim.

Interview transcripts were then analyzed using inductive qualitative content analysis (Elo & Kyngäs, 2008), with each complete participant response to a question serving as the unit of analysis. Initially, the first author (coder 1) repeatedly reviewed the transcripts to develop a preliminary coding scheme grounded in the research questions and emergent patterns. This scheme was iteratively refined through open coding of a subset of transcripts. To enhance analytic rigor, a second coder was trained using a detailed coding manual outlining procedures, category definitions, and unitization rules. Two rounds of independent coding and iterative discussion yielded a Cohen's Kappa value of 0.91. Though primarily inductive, our analysis remained open to theoretical constructs during interpretation. One emergent code was interpreted in alignment with the concept of *template*, capturing students' development of pattern-like code chunks that serve as foundational building blocks for structuring the programming knowledge base (Clancy & Linn, 1999; Robins et al., 2003). The finalized scheme was then applied to the remaining transcripts by coder 1.

Findings

The analysis identified two overarching categories in which students leveraged LLMs for support: (1) *Programming Knowledge Development* and (2) *Coding Practice*. Within the first category, students used LLMs to facilitate conceptual understanding and to support the development of templates. In the second category, students employed LLMs to assist with code planning and code implementation. The following sections detail how learners interacted with LLMs across these dimensions (codes). Selected examples of representative sub-codes are

discussed in the main text, while the complete coding scheme and a full set of illustrative examples are provided in Appendix A.

LLM Usage for Supporting Programming Knowledge Development

The first major category of LLM use was *Programming Knowledge Development*, including two codes: *Conceptual Understanding Support* and *Template Development*.

Students mentioned frequently using LLMs to support their conceptual understanding of foundational programming knowledge, such as concepts, syntax and data structures. Most learners frequently sought clearer explanations of abstract programming knowledge, prompting LLMs to provide "simplified" and contextualized interpretations (*Knowledge Explanation*). However, some expressed skepticism and reported verifying LLM-generated content by "double-checking with Google". Some students used LLMs to help consolidate knowledge through quizzes, practice, and reflection on mistakes (*Knowledge Consolidation*). As one participant described, "I ask the LLMs about the questions I got wrong in quizzes". Additionally, students used LLMs to seek out supplemental knowledge and background information beyond the curriculum, particularly on topics they were "not familiar with," to support the understanding of the learning materials (*Knowledge Supplementation*).

Beyond conceptual understanding, several students employed LLMs to develop templates—reusable pattern-like code chunks that are foundational building blocks of their programming knowledge base, which can be efficiently retrieved and applied to coding tasks (Clancy & Linn, 1999; Robins et al., 2003). Some learners prompted LLMs to generate example code structures for specific tasks and then reproduced or altered the code to practice and internalize the patterns (*Model Structure Generation and Practice*). As one explained, "I asked ChatGPT to give a first version, then try to type everything myself without looking, and compare my version

with its output to see the differences." Some requested multiple alternative implementations of the same task from LLMs, which helped them expand their understanding of different approaches and build a mental library of templates (*Seeking Multiple Solutions*).

LLM Usage for Supporting Coding Practice

The second major category focused on *Coding Practice*, which included two codes: *Code Planning* and *Code Implementation*.

Several participants described using LLMs during the planning stages of their projects for idea generation and feasibility checks. For some, LLMs served as a source of inspiration—providing project ideas and resource recommendations when students "had no clue" (*Idea Generation*). Others used LLMs to assess the feasibility of their directions and plans (*Feasibility Check*). In one case, LLM feedback helped the learner "scale back from overly ambitious features", recalibrating the project scope.

Beyond planning, students described extensive use of LLMs to support code implementation. Several participants relied on LLMs for key environment setup tasks, such as generating GitHub SSH keys (*Environment Setup*). Students also leveraged LLMs' capacity to retain and organize prior interactions for project management (*Project Management*). Features such as "chat history" and folder organization enabled continuity by clustering project-specific discussions for easy retrieval.

A particularly salient theme under code implementation was using LLMs as external "super brains" for knowledge management, including personalized knowledge storage and intelligent knowledge retrieval (*Knowledge Management*). Several students organized explanations, code snippets, and insights within single threads or folders, making the LLM tools personalized "memory databases" that enabled "quick retrieval of information". For knowledge retrieval,

students prompted LLMs to "intelligently" surface relevant content from earlier conversations, which helped them recall context-specific knowledge and receive personalized advice on how to apply it in new coding situations.

Students' use of LLMs to facilitate their code writing varied, including prompting them to provide the code structure outlining each step, directly implementing code sections, and refining their codes (*Code Writing*). Notably, several participants reported intentionally avoiding direct coding solutions unless constrained by time. Many participants asked LLMs to explain the overall structure or specific sections of code to help them understand the code (*Code Explanation*). Debugging was also commonly reported (*Debugging*). When encountering syntax or logic errors, students turned to LLMs for diagnosis and repair.

Discussion and Significance

In line with prior research (Ma et al., 2024; Scholl & Kiesler, 2024), our findings revealed that students used LLMs in various ways, such as code explanation, generation, and conceptual clarification. Extending beyond previous work, we observed more advanced, learning-centered uses. Some students employed LLMs as scaffolding tools to support the development of templates—pattern-like chunks that abstract programming knowledge and serve as foundational building blocks for an expert-like knowledge base (Clancy & Linn, 1999; Robins et al., 2003). Such use patterns underscore LLMs' potential not only to provide surface-level assistance, but also to promote deeper learning strategies, highlighting opportunities to better align LLM support with students' higher-order learning processes through instructional design.

While our findings regarding concerns about hallucinations and over-reliance are consistent with prior research (e.g., Scholl & Kiesler, 2024), our analysis revealed that students were not merely concerned—they actively exercised agency to address these challenges. One

student, for example, instructed the LLM to withhold direct answers unless she was “desperate” under time pressure. Another verified explanations through external sources like Google to ensure accuracy. Such strategies reflect a deliberate effort to navigate the tension between the double-edged immediate efficiency of LLMs and the pursuit of meaningful learning. Our findings challenge binary views of LLMs as either inherently beneficial or detrimental to learning and point to the importance of supporting students in fostering reflective, strategic use—ultimately enhancing their AI literacy and agency in future human-AI interactions (Ng et al., 2022).

The significance of this study lies in its contribution to a more comprehensive and nuanced understanding of how students integrate LLMs into their authentic learning workflows. It moves beyond broad claims of effectiveness by unpacking the specific ways in which students used LLMs to support programming learning. These insights have implications for the design of LLM-supported learning environments and instructional strategies in introductory programming courses.

References

Clancy, M. J., & Linn, M. C. (1999). Patterns and pedagogy. ACM SIGCSE Bulletin, 31(1), 37–42. https://doi.org/10.1145/384266.299673

Finnie-Ansley, J., Denny, P., Becker, B. A., Luxton-Reilly, A., & Prather, J. (2022). The Robots Are Coming: Exploring the Implications of OpenAI Codex on Introductory Programming. *Proceedings of the 24th Australasian Computing Education Conference*, 10–19. https://doi.org/10.1145/3511861.3511863

Gill, P., Stewart, K., Treasure, E., & Chadwick, B. (2008). Methods of data collection in qualitative research: Interviews and focus groups. *British Dental Journal*, *204*(6), 291–295. https://doi.org/10.1038/bdj.2008.192

Hellas, A., Leinonen, J., & Leppänen, L. (2024). *Experiences from Integrating Large Language Model Chatbots into the Classroom* (arXiv:2406.04817). arXiv. http://arxiv.org/abs/2406.04817

Jury, B., Lorusso, A., Leinonen, J., Denny, P., & Luxton-Reilly, A. (2024). Evaluating LLM-generated Worked Examples in an Introductory Programming Course. *Proceedings of the 26th Australasian Computing Education Conference*, 77–86. https://doi.org/10.1145/3636243.3636252

Katona, J., & Gyonyoru, K. I. K. (2025). Integrating AI-based adaptive learning into the flipped classroom model to enhance engagement and learning outcomes. *Computers and Education: Artificial Intelligence*, *8*, 100392. https://doi.org/10.1016/j.caeai.2025.100392

Leinonen, J., Hellas, A., Sarsa, S., Reeves, B., Denny, P., Prather, J., & Becker, B. A. (2023). Using Large Language Models to Enhance Programming Error Messages. *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, 563–569. https://doi.org/10.1145/3545945.3569770

Lyu, W., Wang, Y., Chung, T. (Rachel), Sun, Y., & Zhang, Y. (2024). Evaluating the Effectiveness of LLMs in Introductory Computer Science Education: A Semester-Long Field Study. *Proceedings of the Eleventh ACM Conference on Learning @ Scale*, 63–74. https://doi.org/10.1145/3657604.3662036

Ma, B., Chen, L., & Konomi, S. (2024). *Enhancing Programming Education with ChatGPT: A Case Study on Student Perceptions and Interactions in a Python Course* (arXiv:2403.15472). arXiv. https://doi.org/10.48550/arXiv.2403.15472

Narayanan, A. B. L., Oli, P., Chapagain, J., Hassany, M., Banjade, R., Brusilovsky, P., & Rus, V. (2024). Explaining code examples in introductory programming courses: LLM vs humans. *Ai for Education: Bridging Innovation and Responsibility at the 38th Aaai Annual Conference on Ai*.

Ng, D. T. K., Leung, J. K. L., Su, M. J., Yim, I. H. Y., Qiao, M. S., & Chu, S. K. W. (2022). AI literacy in K-16 classrooms. Springer.

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and Teaching Programming: A Review and Discussion. *Computer Science Education*, *13*(2), 137–172.

Scholl, A., & Kiesler, N. (2024). *How Novice Programmers Use and Experience ChatGPT when Solving Programming Exercises in an Introductory Course*. 1–9.

Vadaparty, A., Zingaro, D., Smith Iv, D. H., Padala, M., Alvarado, C., Gorson Benario, J., & Porter, L. (2024). CS1-LLM: Integrating LLMs into CS1 Instruction. *Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1*, 297–303. https://doi.org/10.1145/3649217.3653584

Zamfirescu-Pereira, J. D., Qi, L., Hartmann, B., DeNero, J., & Norouzi, N. (2023). Conversational Programming with LLM-Powered Interactive Support in an Introductory Computer Science Course. *NeurIPS'23 Workshop on Generative AI for Education*. https://openreview.net/forum?id=TAy2YVVV6q

Appendices

Appendix A. The Coding Scheme for Students' LLM Use

Category	Code	Sub-Code	Definition	Example Quotes
Programming Knowledge Development (PKD)	Conceptual Understanding Support (CUS)	Knowledge Explanation (KE)	Asking LLMs to explain foundational knowledge.	".. sometimes I'm not familiar with [the concept] and AI can help me, like, really understand it with like simplified words." "The most prevalent usage of mine is to fact-check the definition of some function, because sometimes, when I'm confused with what professor has taught us." "It's also hard to describe abstract coding concepts clearly in class, but I can provide context and code to ChatGPT, so it really helps me understand concepts and foundations." "When we started with the videos from [Professor C], I wouldn't understand anything. So what I basically did was go back to Coding Buddy. Whenever [Professor C] said something that I didn't get. I would go and ask like, can you please explain whatever concept like if I were a high school student."
		Knowledge Consolidation (KC)	Using LLMs to help consolidate knowledge by different ways, including reviewing quizzes, asking for exercises, and reflecting on mistakes.	"And we also have like, quiz in the in the class, and sometimes I got it wrong. I will ask the Coding Buddy about the questions I got wrong." "And in terms of debugging, it will also not only provide you with like a more doable way, but also it'll elaborate some key knowledge that it thinks you should know."

		Knowledge Supplementation (KS)	Asking LLMs to provide supplementary knowledge/information to aid understanding.	"...because we have a lot of reading materials and they have a lot of, like, background information I'm not familiar with, sometimes I'll ask Coding buddy or other AI chatbot about it." "They serve like, a supplementary information source in addition to like, what the course provides."
	Template Development (TD)	Model Structure Generation and Practice (MSGP)	Prompting LLMs to generate model structure (e.g., example code) for specific programming tasks, which they then used for hands-on practice to become familiar with underlying structural patterns. This helped learners gradually internalize these patterns and build a library of reusable templates.	"I will ask ChatGPT to give a first version, and I will look through the version and try to type everything by myself without looking at the answer, and then I will compare between the answer and what I put in, and see what the difference is." "I'm just creating a kaleidoscope, but I don't know how to start. Like, I want to hover my mouse on the screen, and the circle will change with my mouse…I think it's very complicated, and I don't know the style. So I will ask GPT to create a demo, and I will refer to its structure. Like 'can you give me an example of dots, like all the dots in chaos but repetitive way?', and I change it to circles and squares to some different shape. Also add some features like buttons too. It just serves as a scaffolding template, and you just build upon that….Actually it's quite important….once I've practiced something, I can reuse it repeatedly."
		Seeking Multiple Solutions (SMS)	Asking LLMs to generate multiple implementations of the same task, enabling them to	"It showed me two ways to do the same function so I could compare them." "And I will also ask ChatGPT to give multiple versions of the same section and compare across different versions to see, 'Oh, why is it doing that?' and what's the difference."

			explore alternative approaches, expand their template accumulation, and refine their understanding of different templates.	"…within each function block, you can also ask: what different ways can achieve the same function? And explain what the differences are between each function. And anything I don't understand, I'll just ask until I do."
Coding Practice (CPR)	Code Planning (CP)	Idea Generation (IG)	Asking LLMs to provide initial project ideas or resource suggestions during project planning.	"It gave me a few project ideas when I had no clue where to start." "Yeah, especially when I had no idea at all, I would describe my purpose and ask what I should do." "I needed to highlight, the text in my project and I also googled it. But it seems that no one has such experience in developing such code. So at a very first stage, there's totally blanking [in] my mind. So I just described my purpose and ask LLM, like, what should I do? And then it provided me with a list of libraries."
		Feasibility Check (FC)	Asking LLMs to suggest initial plans and/or resources, analyzing the "pros and cons" as aids for decision-making; Asking LLMs to assess and improve the feasibility of initial plans.	"I just described my purpose and ask LLM, like, what should I do? And then it provided me with a list of libraries. I asked it to explain the pros and cons of each library. So that is like more time efficient." "[classmate A] and I didn't know what was realistic. So we asked Coding Buddy about our ideas. It helped us scale back from overly ambitious features and come up with something feasible for our skill level."

	Code Implementation (CI)	Environment Setup (ES)	Using LLMs to help with environment setup.	"So when we got to the first week, I realized like a couple of my classmates did have experience, and I was like starting from scratch. So I didn't even know like where could I find my terminal? What was the difference between, like the VS code and my terminal, like, I have the most basic questions. So I started using coding Buddy to solve for those basically like, how do I get here? What do we need to do to download this? What do I need to start." "Maybe it helps me set up some environments. Like [when] learning GitHub. AI helped me with SSH keys, which is not covered in course content."
		Project Management (PM)	Using the folder feature of LLMs to store, organize, or retrieve project-related information/ guidance/codes.	"I think they have a new function like in the chat history. They have a folder, so you can drag and drop all the conversation history related to a specific project, so you can retrieve that later as well… I forgot the exact time when it came out. But yeah, I think it's been there for a while. Not a lot, because usually the assignments are pretty small projects. You can always wrap up everything in a single conversation history. But if you're doing a large project, it's a good way to manage everything."
		Knowledge Management (KM)	Using LLMs to store previous knowledge; Rely on LLMs to intelligently retrieve relevant knowledge from personalized knowledge database and provide personalized knowledge	"I keep my past chats to track what I've learned so I don't repeat questions" "And AI, sort of like this big language model can, like store all of your prior knowledge, and can apply them in a very smart way." "And right now, I think ChatGPT updated and they have, like, a file solution, I can use all the old conversations in one file, and it will sort of generate into, like a memory database, just like, I can pull out these information quickly" "And I'm gonna, typically, I use like, 'combining with my previous knowledge and considering this assignment. Can you tell me about the Python knowledge, or, like, the right type of knowledge. I need to

		application advice.	finish this assignment.' I'll do that kind of thing." "Sometimes when I didn't close the chat box, it could retrieve the project I asked before and use it as an example [when explaining how the functions would be applied in coding languages]." "[LLMs] help me pull out my memory and sort of like, sort things out and give me some advice."
	Code Writing (CW)	Asking LLMs to provide the structure and outlining steps to complete the code writing task; Ask LLMs to implement specific code sections within each code block for finishing assignments or projects; Asking LLMs to refine codes.	"It gives me a clear outline of how to build the function step by step." "Sometimes when I was desperate, I just asked it to give me the answer." "I will ask AI tools to help me refine or better organize my codes. Like to make it more concise."
	Code Explanation (CE)	Asking LLMs to explain overall code structures and/or each step; Asking LLMs to explain specific code parts.	"… [LLMs] give me a clear view of all the logic and structures" "I ask GPT, maybe, like, explain each line so I know how it works, like, one by one."

	Debugging (DB)	Asking LLMs to support with the debugging of the codes.	"…the features I use most is the debug[ging]... [When] I try many times I still cannot figure out the solution. So I will ask, What's wrong with my codes? And I will tell the AI tools, like, I want to achieve this. I want to achieve this result. So help me with it." "VS Code. Yeah, it just pop up errors. And I don't know, because I'm like, so careless. I'm a very careless person. I'm just, like, copy paste to Coding buddy and 'tell me Where's the wrong? Why is it constantly telling me there's still a bug,' and it will just offer me the solution." "[The feature I frequently use was] debugging. I paste terminal error messages into ChatGPT. It explains why the error happened and offers multiple solutions."