Questions Before Answers? A Comparison of Codebase Chatbots and Tutorials for Codebase Learning

Shaokang Jiang [1], Jimmy Koppel [2] and Michael Coblenz [1]

[1] *University of California San Diego, La Jolla, USA*
[2] *UpToSpeed*

Abstract

LLM based chatbot is common in helping and providing guidance to programmers while programming. However, it is not clear how effective they are in comparison to human-written tutorials, especially when facing with a large unfamiliar codebase. We conducted a between-subject experiment with 15 experienced software engineers to compare the effectiveness of LLM chatbots and human-written tutorials in helping people learn and get started with a new codebase. We found that users assisted by LLM chatbots were more confused and less confident, while those using tutorials had a better understanding of the code structure but struggled with low-level coding challenges. We also found that the programmer's personal experience with the programming language is more important than any other factors, including the experience with LLM systems and the framework, in determining the productivity. *Keywords*: LLM. Empirical studies of programmers. Productivity. Learnability. Usability analysis.

1 Introduction

LLM-based chatbots are usually an extension of modern IDEs, allowing people to seek answers or solutions using natural language leveraging LLM. Some chatbots also have the capability of indexing the entire codebase, such as Cody (fig. 1). While coding, programmers can ask questions about the codebase, the code, and how to complete tasks. During interaction, programmers need to design a question, type it in an input box, and then the chatbot will provide an answer in a chat flow. The chatbot can also provide code snippets and sometimes even full code completion. The history messages are accumulated in the chat flow, allowing people to ask follow-up questions.

On the one hand, an LLM chatbot with codebase support could be potentially useful for learning large codebases, as it can extract underlying information and provide high-level explanations, especially useful while software engineers are migrating to other similiar platforms. On the other hand, communicating with an LLM chatbot often requires well-designed prompts, and the information provided by the chatbot can sometimes be misleading.

Human-written tutorials are widely used in the computer science education community [1], [2]. They are usually structured in a step-by-step manner, providing a high-level overview and guiding learners through a series of training tasks. Their effectiveness has been validated by the education community [3]. On the one hand, human-written tutorials are usually well-structured, making sure learners can understand the codebase from a high-level fast. On the other hand, they may lack personalized guidance and may not be able to provide real-time help when learners are stuck.

Given these possibilities, we conducted an experiment with 15 junior-level or higher software engineers to understand how they interact with the two systems. We asked participants to complete a programming task on a 2000 line Django codebase with the help from either a human-written tutorial group or with access to an LLM chatbot group. Following this, we conducted semi-structured interviews. As an exploratory effort, we also conducted an additional experiment using the same protocol with Cursor, an IDE equipped with AI support for autocompletion and LLM chat with the capability of indexing the entire codebase.

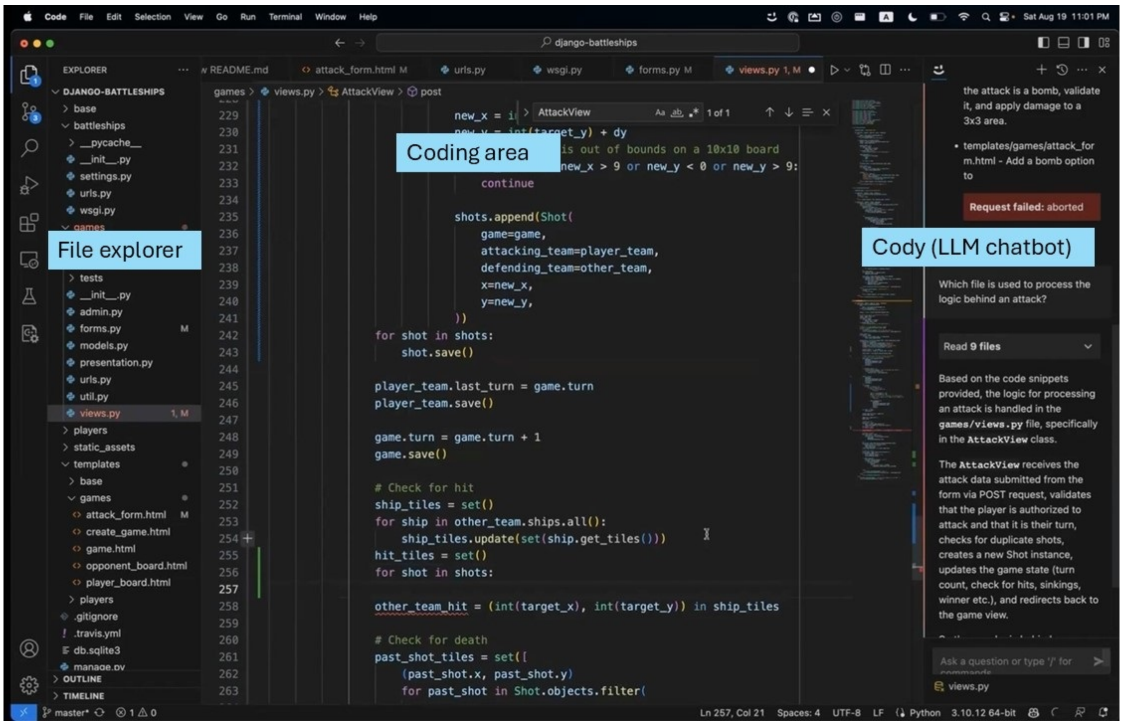
Previous studies have found LLM tools to improve the productivity of programmers [4], but they did not measure the effectiveness of getting started with an unfamiliar codebase or the confusion level of programmers. They also did not compare the effectiveness of LLM chatbots with human-written tutorials. Our study aims to fill this gap.

We considered several research questions. Answering them will help explore the benefits of each approach, identify situations where one is more useful than the other, and compare the effectiveness

PLATEAU
13th Annual Workshop at the Intersection of PL and HCI

DOI: 10.35699/1983-3652.yyyy.nnnnn

Organizers:
Sarah Chasins, Elena Glassman, and Joshua Sunshine



```
            new_x = i
            new_y = int(target_y) + dy
            is out of bounds on a 10x10 board
            new_x > 9 or new_y < 0 or new_y > 9:
                continue

            shots.append(Shot(
                game=game,
                attacking_team=player_team,
                defending_team=other_team,
                x=new_x,
                y=new_y,
            ))
    for shot in shots:
        shot.save()

    player_team.last_turn = game.turn
    player_team.save()

    game.turn = game.turn + 1
    game.save()

    # Check for hit
    ship_tiles = set()
    for ship in other_team.ships.all():
        ship_tiles.update(set(ship.get_tiles()))
    hit_tiles = set()
    for shot in shots:

    other_team_hit = (int(target_x), int(target_y)) in ship_tiles

    # Check for death
    past_shot_tiles = set([
        (past_shot.x, past_shot.y)
        for past_shot in Shot.objects.filter(
```

the attack is a bomb, validate it, and apply damage to a 3x3 area.

- templates/games/attack_form.html - Add a bomb option to

Request failed: aborted

Which file is used to process the logic behind an attack?

Read 9 files

Based on the code snippets provided, the logic for processing an attack is handled in the `games/views.py` file, specifically in the `AttackView` class.

The `AttackView` receives the attack data submitted from the form via POST request, validates that the player is authorized to attack and that it is their turn, checks for duplicate shots, creates a new Shot instance, updates the game state (turn count, check for hits, sinkings, winner etc.), and redirects back to the game view.

Figure 1. A sample use of Cody.

of LLM chatbots and human-written tutorials in helping experienced software engineers learn and get started with a new codebase.

- RQ1: Does the LLM chatbot affect the productivity and learning process of experienced programmers compared with human-written tutorial?

  H1: Programmers using the LLM chatbot will start working with the codebase faster than those using a human-written tutorial.

  H2: Programmers using the LLM chatbot will have faster task completion times than those using a human-written tutorial.
- RQ2: How does a LLM chatbot affect programmers' confusion level?

  H3: Programmers using the LLM chatbot will be less confused.
- RQ3: How does the previous experiences with LLM chatbot-based systems, familiarity with Python and Django correlate with the productivity and learning process of programmers?

In addition to those research questions, we also conducted a post-study semi-structured interview and open-coded the results to understand the participants' thoughts, experience and suggestions to the LLM Chatbot, tutorial, and LLM support IDE.

Contributions Surprisingly, we did not find a significant improvement in task completion time for programmers using the LLM chatbot. We also didn't find a correlation between the experience with LLM chatbot-based systems and productivity. Instead, we found that the programmer's personal experience with the programming language itself is more important than any other factor in determining productivity. We also found that participants using the LLM chatbot exhibited a higher level of confusion compared to the tutorial group.

Implications The implications include contributing ideas for future solutions to reduce the time experienced programmers spend learning new codebases by leveraging LLM-generated interactive tutorials. The results of the study are helpful in providing guidance on designing new tools to improve the onboarding experience for companies with large codebases. New tools need to leverage LLMs in generating step-by-step interactive tutorials, considering the user's personal background, with potential generalizability to the entire programming community, including novices.

2 Related work

Human-written tutorials are widely used by the education community and have been validated for their usefulness [2], [3]. Brown et al.'s [1] research highlights the importance of using step-by-step guidance. Tools relevant to tutorials and new tool designs have also been studied. Interactive tutorials that require learners to perform tasks while learning have also been proven to be useful [5]–[7]. Video-based tutorials have also been shown to be useful by previous works [8], [9].

In contrast, our study focuses on a more advanced format of the content, facing with junior or senior software engineers and is intended to compare between the group using human-written tutorial and the group using LLM chatbot.

Ross et al. [4] found that conversational interaction with LLM models can make programming more productive. However, they didn't measure the effectiveness of getting started with an unfamiliar codebase. Previous studies [10] also showed the effectiveness of using chatbots in learning, but these were not AI-supported.

Several studies [11]–[13] assessed the learning effect of using AI chatbots in CS1 education and found these tools helpful for learning. However, Frankford et al. [14] suggested that general-purpose chatbots inhibit learning for CS1 students. In contrast, we focus on experienced programmers learning a medium-sized codebase.

Nam et al. [15] and Jonan [16] found that LLM chatbots with codebase support and prompts can help programmers understand the codebase better. In contrast, we focus on a general-purpose AI tool with only codebase support and measure the effectiveness of understanding a larger codebase.

Previous studies measured the best time to use LLM-generated chatbots and in which scenarios they are most useful [17]. In contrast, we compare the effectiveness of LLM chatbots and human-written tutorials. Barke et al. [18] classified the interaction mode with the autocomplete feature of Copilot. In contrast, we measure the effectiveness of the LLM chatbot in the process of understanding a codebase.

Gardella et al. [19] found that AI autocompletion tools reduce pressure while programming for novice programmers. In contrast, we focus on experienced programmers and measure the AI chatbot's effectiveness in the learning process.

3 Method

We conducted a between-subjects experiment with 15 participants remotely. For the human-written tutoial group, participants were required to complete a tutorial before proceeding to the main task. For the LLM chatbot group, participants went directly to the main task and were allowed to use Cody, a popular LLM chatbot with codebase support. Both groups could choose the coding environment they were comfortable with and were allowed to access the internet. There were no time limits for the entire study, and participants could pause at any point they wished. The study consisted of two parts, as follows:

Exploratory lab study: Finish a task Participants were asked to continue talking while completing a task. To ensure we could evaluate programmers' interactions with an unfamiliar codebase in a reasonable time, we selected an implementation of a popular board game to make sure everyone was familiar with the context, and with around 2000 lines of code to ensure people could finish it in a reasonable amount of time. To improve external validity and fully utilize the power of Cody, the codebase was an existing game on GitHub[1]. Our task required participants to add a new feature called bomb, which hits a range of squares, to the game, which involved modifying some front-end code and adding back-end logic, while also establishing connections between them. Experimenters did not answer any task-related questions in this part. Completion was determined by the experimenters by checking whether the front-end worked appropriately and the back-end logic was correctly implemented. There were no time limits for this part, but participants were allowed to give up at any time, and its completion would be recorded as incomplete. Participants were also not required to take care of code quality.

1 https://github.com/RuairiD/django-battleships

Post-study semistructured interview After the task was completed, participants were asked to complete a post-study interview. The interview consisted of their thoughts on the task, and their opinions on the system. More specifically, we asked participants the following questions: 1) What are your thoughts on the coding experience you just had? 2) Can you show me or explain what you did or thought during the [specific] operation you performed earlier? 3) Can you talk more about your experience using Cody/Tutorial? and 4) Is there anything else you would like to share?

3.1 Experiment description

All participants ran our setup script before the start of the experiment. Participants selected the IDE they liked. However, due to limitations with Cody, participants in the LLM chatbot group were required to use VSCode. Most participants used VSCode, while only two used Vim. During the experiment, participants were permitted to use Copilot to autocomplete the code, but they were not allowed to use Copilot Chat. For those who did not have Copilot, experimenters provided basic syntax help and functions similar to the Copilot system.

Tutorial Participants in the tutorial group were required to complete a human-written tutorial before starting the main task. The tutorial contains three parts: 1) an introduction video to the codebase; 2) a written high level introduction clarifying how the codebase structures matches with MVC framework, and the entry point of some higher wrapper files; and 3) a step-by-step walkthrough tasks, asking participants

3.2 Data collection and pre-analysis

During the entire experiment, we used Zoom to record everything on the participants' screens, including their audio and faces. We used a mixed-methods approach to analyze the data, details as shown below:

3.2.1 Task completion time

We extracted this data manually from recording, measuring the time taken to complete the task. The task completion time was calculated from the moment the participant finished reading the task prompt and setting up the relevant environment to the moment they completed testing and the experimenters agreed with their result. We used the time taken to complete the task as the primary measure of productivity.

3.2.2 First typing time

We also recorded the time taken from the moment the participant finished reading the task prompt and setting up the relevant environment to the moment they started the first meaningful typing, meaning the first operation that is not randomly typing or deleting, determinating from their screen recording and voice. We used this as a measure of the time spent in getting start with the codebase.

3.2.3 Confusion score from transcript

As suggested by Sidney and Art [20], when leaners struggling with errors will lead to confusion. We transcribed the audio from the task part using Assembly AI, and then we extracted the confusion score using a top-ranking model for text-based emotion extraction on Hugging Face[2]. The model is trained on a human-labeled dataset on Reddit data by Dorottya et al. [21]. We then determined the confusion score based on a threshold of 0.5 as recommended by the model author.

3.3 Participants

We recruited 15 participants from a professional software engineering online forum. Participants were randomly assigned to the LLM Chatbot group (the group using Cody) or to the Tutorial group upon arrival. All participants signed the appropriate consent form before the start of the study.

2 `SamLowe/roberta-base-go_emotions` model:https://huggingface.co/SamLowe/roberta-base-go_emotions

All participants are Male. Participation was voluntary, the estimated time was two hours, and no compensation was provided.

Table 1. Experience for Python and Django was categorized as: Beginner, Intermediate, Proficient, Advanced, Expert.

Person	**Type**	**Python**	**Django**	LLM chatbot
P1[3]	Tutorial	Intermediate	Novice	Novice
P3	Tutorial	Beginner	Beginner	Expert
P6	Tutorial	Advanced	Intermediate	Novice
P8	Tutorial	Beginner	Novice	Intermediate
P9	Tutorial	Proficient	Novice	Novice
P10	Tutorial	Expert	Novice	Novice
P12	Tutorial	Intermediate	Novice	Expert
P15	Tutorial	Expert	Novice	Intermediate
P13	Cursor	Proficient	Beginner	Novice
P2	Chatbot	Expert	Intermediate	Expert
P4	Chatbot	Expert	Novice	Expert
P5	Chatbot	Proficient	Novice	Intermediate
P7	Chatbot	Advanced	Beginner	Novice
P11	Chatbot	Proficient	Proficient	Novice
P14	Chatbot	Beginner	Novice	Intermediate

We asked participants about their previous experience with Python and Django and manually clustered their responses into six levels: Novice, Beginner, Intermediate, Proficient, Advanced, and Expert. We classified individuals with significant professional, long-term, or recent experience as Experts. Those lacking recent or extensive experience were categorized as Proficient. Individuals with hobby experience, strong personal interest, and multiple personal projects were classified as Advanced. Those who occasionally used the language or whose usage was limited to a specific field were classified as Intermediate. Participants who had only studied Python in a class were assigned to the Beginner category. Finally, those with no experience were classified as Novice.

Previous studies have shown the importance of the prompt while interacting with LLM chatbots [22]. Therefore, we asked participants about their experience with LLM chatbots and categorized them into three groups: Novice, Intermediate, and Expert. Those who use LLM models daily were classified as Experts. Participants with some experience but not with the entire system were categorized as Intermediate. Those with no experience or who had only tried LLM chatbots once were classified as Novices.

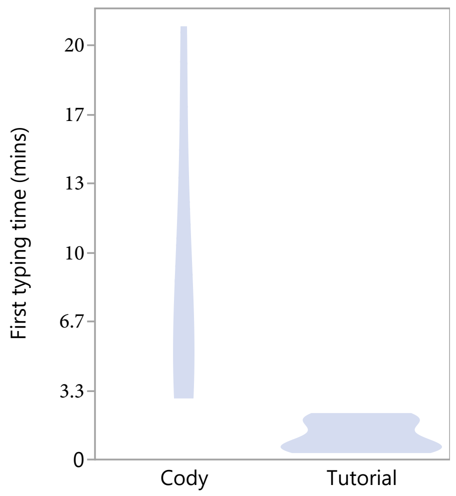
To confirm both group get a similiar distribution between groups, a Fisher's exact test was conducted and shows that the two groups have no observed significant difference in terms of their experience with Python ($p \approx 1.0$) , Django ($p \approx 0.84$), and LLM chatbots($p \approx 1.0$).

4 Results

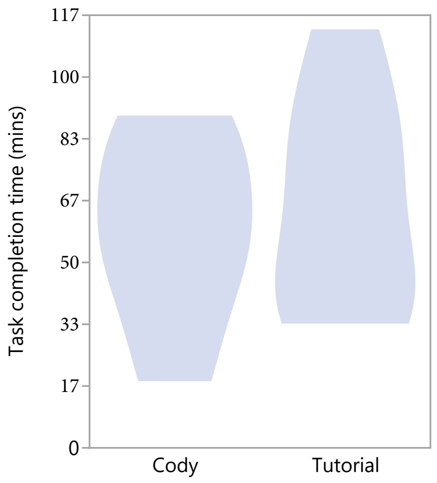
4.1 RQ1: Does the LLM chatbot affect the productivity and learning process of experienced programmers compared with human-written tutorial?

H1: Programmers using the LLM chatbot will start working with the codebase faster than those using a human-written tutorial.

If the LLM chatbot is able to provide good guidance on where to start, it is possible that programmers with Cody access can start working with the codebase faster. One key measurement of the pace at which people learn the codebase is the time taken to start the first meaningful typing. We



(a) Tutorial group spent significantly less time in getting started with the codebase ($p \approx 0.0034$, N = 13)



(b) No significant difference in task completion time between the two groups ($p \approx 0.34$, N = 13)

manually collected this data from the screen recording. The results are shown in fig. 2a. A Wilcoxon rank-sum test shows that the tutorial group (Median = 0.83 mins, M = 1.2 mins) spent significantly less time than the Cody group (Median = 6.1 mins, M = 8.8 mins) in getting started with the codebase ($p \approx 0.0034$). The time taken to start the first meaningful typing for the participant using Cursor is 15 minutes, higher than the average of the other two groups.

H2: Programmers using the LLM chatbot will have faster task completion times than those using a human-written tutorial.

If LLM chatbot is able to provide a good guidance, it is possible that programmers with Cody access can complete the task faster. One key measurement of the productivity is the task completion time. We collected this from the screen recording manually. An ANOVA test $[F(1,9) \approx 0.17, p \approx 0.69]$ shows that there is no significant difference ($p \approx 0.69$) in task completion time between the tutorial group (Median = 62 mins, M = 65 mins) and the Cody group (Median = 56 mins, M = 58 mins). The task completion time for the person with Cursor is 133 minutes, higher than the average of the other two groups.

4.2 RQ2: How does a LLM chatbot affect programmers' confusion?

H3: Programmers using the LLM chatbot will be less confused.

As suggested by Sidney and Art [20], when leaners struggling with errors will lead to confusion. If the LLM chatbot is able to provide good guidance and participants find it useful, it is possible that programmers with Cody access will be less confused. Using the audio from the task part, we measured the confusion level based on the transcripts by utilizing a pre-trained roberta model, `SamLowe/roberta-base-go_emotions`. As suggested by the author, we selected a threshold of 0.5 to determine whether the transcript demonstrated confusion. The results show that among the 13 participants, none of the participants in the tutorial group were confused, while one participant in the Cody group was confused. The person from the Cursor group was also not confused.

By comparing the raw score of the model, an ANOVA test $[F(1,11) \approx 0.41, p \approx 0.41]$ shows that there is no significant difference in the confusion score between the tutorial group (Median = 0.28, M = 0.31) and the Cody group (Median = 0.20, M = 0.25). The confusion score for the person with Cursor is 0.27, higher than the average of the Cody group but fewer than the average of tutorial group.

4.3 RQ3: How does the previous experiences with LLM chatbot-based systems, familiarity with Python and Django correlate with the productivity and learning process of programmers?

As suggested by Zamfirescu-Pereira et al. [22], knowing how to construct response to the LLM chatbot can determine the effectiveness of using the chatBot system. So, we are interested in whether the previous experience with LLM chatbot-based systems is correlated with the productivity and learning process of programmers, as measured by the task completion time and the time taken for people to start with the codebase. Because participants' experience with chatBot, Python, and Django are ordinal, we used Kendall's τ correlation, which is a statistic used to measure the ordinal association between two measured quantities.

Surprisingly, a Kendall's τ correlation shows no correlation between the level of LLM experience and the task completion time ($p \approx 0.30, \tau \approx -0.45, N = 5$). A Kendall's τ correlation also shows no correlation between the level of LLM experience and the time taken to start with the codebase ($p \approx 0.30, \tau \approx 0.24, N = 6$).

We are further interested at whether the familiarity with Python and Django is correlated with the productivity and learning process of programmers. A Kendall's τ correlation shows a significant correlation between the level of Python experience and the task completion time ($p \approx 0.0050, \tau \approx -0.69$), suggesting that the more experienced programmers are with Python, the faster they can complete the task.

A Kendall's τ correlation shows no correlation between the level of Python experience and the time taken to start with the codebase ($p \approx 0.66, \tau \approx -0.098$). A Kendall's τ correlation shows no correlation between the level of Django experience and the task completion time ($p \approx 0.86, \tau \approx -0.044$). A Kendall's τ correlation shows no correlation between the level of Django experience and the time taken to start with the codebase ($p \approx 0.67, \tau \approx 0.098$).

Generally speaking, the result is telling us that the programmer's personal experience with the programming language is more important than any other factors, including the experience with LLM chatbot-based systems and the framework, in determining the productivity.

4.4 Opinions on LLM Chatbot (Cody)

We transcribed the interview recordings with Assembly AI and then performed an open-coded analysis of the interviews provided by the participants. Two out of seven participants generally think Cody makes programmers complete tasks faster. We identified several themes that emerged from the data as follows:

Feel confused, resulting in lack of confidence and less productivity Five out of seven participants in the Cody group mentioned that they got confused when using Cody. They suggested confusion about which parts of the codebase to modify to complete the task, an inability to fully understand the codebase, and difficulty in identifying the reasons behind failures. They also suggested that a debugger could be helpful. For instance, P4 suggests, "And then once I'm checking something in, I want someone to be able to say, why did you add this line? And it can come down to something like, this is the best way I thought of doing it, but I really never want to have a line in my code. It's like, why is this here?"

Due to this confusion and lack of understanding, participants expressed a lack of confidence in the code they wrote and their ability to learn enough to be productive and make modifications in the future. For instance, P14 suggests, "And I'm not so sure that all edge cases are covered, but it seems like they probably are because this is pretty simple."

Participants also suggested that due to the confusion introduced by Cody, they needed more time to understand the codebase and perform modifications. P11 mentioned, "I think it took way longer for me to do this one... it was hard to know what to do and where."

Certain response content is helpful, but think before using A sample usage of Cody is shown in fig. 1. Participants suggested several structures currently provided by Cody are helpful, including some fixing suggestions, step-by-step guidance for finishing a task, the synthesized answer format, and listing multiple possible answers. As P5 mentions, "[Cody] gives multiple answers and you kind of pick and choose. Instead of, like, if I go to, say, Stack Overflow."

Participants also suggested that the best way of interacting with its structure is to use it as guidance, not rely on its answer and never fully trust its answers, especially for newly designed, not publicly available API systems. In extreme cases, it may provide misleading information: "I think that's just a form. Like, Cody used the word web form. It was a separate thing," as P4 suggested.

Conflicting on the usefulness to novices Participants are conflicted on whether Cody is useful for novices. Some participants suggest that Cody is helpful for novices, as even though Cody may provide misleading information, novices may not even be able to realize it; as P4 suggests, "the cost of correcting a hallucination would be better than, like, not using Cody." However, participants also suggest that only programmers with a large amount of knowledge can use Cody effectively, as P14 suggests, "There's lots of, like, implied knowledge in this. I don't think it would be useful pretty much at all for someone that has a lot less experience than me."

Introduce vulnerabilities and garbage code Even though we asked participants to ignore code quality, two out of seven mentioned that Cody introduced some garbage code into their codebase, necessitating multiple iterations to fix it. As P11 suggested, "the fact that I'm sort of duplicating methods here and there is obviously, like, pretty ugly and especially this." In some extreme cases with one participant, the experimenter reported that Cody introduced a vulnerability into the codebase that may significantly increase runtime due to frequent and expensive database operations.

However, participants also suggested that Cody's suggestions inspire people to think of reusing existing components, as P4 mentioned. This could potentially increase the code quality, but only one person reported this.

Effective for low-level guidance, lacks high-level structure Five out of seven participants suggested that Cody is effective for providing low-level guidance, including syntax-based help, and is helpful in understanding small scope structures, such as algorithms, and pointing out the direct place to modify for low-level tasks. As P11 suggested: "It was actually really good at pointing out exactly where I needed to update"

However, participants also suggested that Cody lacks high-level introductions, such as a general overview of the codebase or guidance on the functionality of a file in the entire system. This lack of high-level context led to confusion about where to start and which parts of the codebase to focus on. As P11 suggests, "it's kind of hard to know what to do and where."

4.5 Opinions on human-generated Tutorial

Good at providing high-level structure but lacking low-level details Participants suggested that the tutorial lacks low-level details, such as code syntax, relevant information, basic information, or relevant information to the current framework. Participants did have questions about these aspects. Like P6 suggested, "I definitely got annoyed by the types. Like the true false of the text form thing used by".

As for the high-level structure, participants generally agree it is useful in understanding the codebase faster. However, four out of six participants pointed out that the tutorial is only helpful if the programmer has basic knowledge of the coding language, Python in this case. As P9 suggested, "you need to be able to write Python, otherwise you wouldn't in practice ever be modifying this code."

Personalized guidance would be useful Several participants suggested that providing personalized guidance is useful, such as using the language/structure/concepts that participants are familiar with. As P12 suggested, "I think it was very good with the instruction I got that explained the kind of MVC model in Django helped me."

However, several participants also suggested that the tutorial is missing enough personalized details, including the content being hard for participants to follow, lacking user background, and failing to provide necessary low-level information, such as how two components are connected together: "the tutorial did not teach me all the Django that I wanted to know."

Messages from experts boost confidence and save time Several participants appreciated the knowledge provided by experts in the tutorial, such as debugging techniques and tricky points in connecting the backend with the view in Django. Participants suggested that these messages and teaching materials from experts gave them confidence and saved time. As P6 suggested, "I would go back to the tutorial as kind of a checklist to ensure I touched the three places I needed to touch."

4.6 Opinions on LLM support IDE (Cursor)

As an exploratory effort, we are just reporting the themes from the only participant in the Cursor group. The participant suggested that the tool caused confusion, lacked real-world examples and personalized guidance, and recommended using familiar real-world metaphors. Additionally, there was subtle encouragement of copy and paste.

4.7 Discussion

Overall, both human-generated tutorials and LLM chatbots have their own advantages. The LLM chatbot is able to inspire thinking more about the structure, which is supposed to be useful for self-motivated learning with specific prompts. However, it is hard to provide the correct answer directly. As P14 suggests: "I use it actually as a tutorial," using it as high-level guidance for a brief tutorial can be helpful.

Generally speaking, both groups of people get confused somewhere during programming based on interview results, but the LLM-guided group of people represents a higher level of confusion and usually don't know what they are doing or the reason behind their actions. As P4 suggested, "I might do things like, you know, add lines that I don't know what they do or, like, ..." However, the tutorial group feel confused about more specific questions, for example, how the view connects with the modules. As P3 suggested, "I had a really hard time not doing it properly and just trying to get forward because [...] on the website it says depth over pattern matching [...]". The confusion score assessment from the transcript is consistent with the interview results, indicating that the LLM chat group experienced a higher level of confusion.

Regarding the lack of personalized details in the tutorial, since these tutorials are written by humans, it is common to encounter this issue. Embedding the power of LLMs may help mitigate the problem. It is surprising to see that the participant using Cursor also felt the guidance lacked personalized details, but the degree of this issue can be different. The Cursor participant referred more to the lack of their own personal experience, while for the tutorial group, it was more about general concepts, such as sample syntax.

5 Limitations

The training tutorial beforehand for the tutorial group uses the exact same codebase as what they are going to work on. This might give the tutorial group a faster initial response time, especially influencing the first meaningful typing time. Previous studies show that tutorials made by experts are usually of higher quality than those made by the general public [23]. The tutorial used in this study was made by the experimenter, which might not be as high quality as one made by an expert. This can negatively influence the results of the tutorial group.

The task choice of a board game might be easy to understand for participants who have played it before. This might negatively impact people who have never played it before. Additionally, having all male participants may lead to a different understanding and may not be inclusive of a diverse community. All participants are professionals, which might not be representative of the general population. Novices might find LLM chatbots and tutorial instructions more useful and feel less confused.

The study asked participants to add a bomb feature. Considering ethical problems, the Cody system might occasionally have blocked responses that contained the word "bomb," which might have influenced the results and made participants feel more confused due to reading incomplete responses. We asked participants to ignore code quality during development. This might make the task easier for some participants and harder for those who believe in writing clean code.

Truong [24] suggests that measuring confusion using models fine-tuned from go-emotion can be unreliable; however, most of our conclusions about confusion are based on interview data, so we do not expect the results to change significantly.

6 Future Work

Both LLM chatbots and human-written tutorials have their own advantages and disadvantages. Future work can focus on combining the two to provide a more effective and personalized learning experience by considering well-formatted responses, including step-by-step instructions, considering more user personal background, and being more specific in the generated responses. We also found that the chatbot usually provides a broad range of suggestions. Building tools that utilize this ability to inspire people to think more, combined with some human input interactive learning tools, can also be helpful.

7 Conclusion

Although LLM chatbot is thought to make people clear and can get started with the codebase faster, we found that it is not the case. Participants using the LLM chatbot were more confused and less confident, while those using tutorials had a better understanding of the code structure but struggled with low-level coding challenges. In addition, we also found that the programmer's personal experience with the programming language is more important than any other factors, such as the experience with LLM chatbot-based systems and the framework, in determining the productivity. And, the tutorial group spent significantly less time in getting started with the codebase, showcasing the effectivenss of human-written tutorials in providing high-level structure.

References

[1] N. C. Brown and G. Wilson, "Ten quick tips for teaching programming," *PLoS computational biology*, vol. 14, no. 4, e1006023, 2018.

[2] W. Taffe, "Writing in the computer science curriculum," *Writing across the Curriculum*, vol. 1, no. 1, pp. 17–22, 1989.

[3] D. M. Layton, "The role of the tutorial system in enabling students' academic success," *South African Journal of Higher Education*, vol. 29, no. 4, pp. 198–210, 2015.

[4] S. I. Ross, F. Martinez, S. Houde, M. Muller, and J. D. Weisz, "The programmer's assistant: Conversational interaction with a large language model for software development," in *Proceedings of the 28th International Conference on Intelligent User Interfaces*, ser. IUI '23, Sydney, NSW, Australia: Association for Computing Machinery, 2023, pp. 491–514, ISBN: 9798400701061. DOI: 10.1145/3581641.3584037. [Online]. Available: https://doi.org/10.1145/3581641.3584037.

[5] R. Shen, D. Y. Wohn, and M. J. Lee, "Comparison of learning programming between interactive computer tutors and human teachers," in *Proceedings of the ACM Conference on Global Computing Education*, ser. CompEd '19, Chengdu,Sichuan, China: Association for Computing Machinery, 2019, pp. 2–8, ISBN: 9781450362597. DOI: 10.1145/3300115.3309506. [Online]. Available: https://doi.org/10.1145/3300115.3309506.

[6] C. Hulls, A. Neale, B. Komalo, V. Petrov, and D. Brush, "Interactive online tutorial assistance for a first programming course," *IEEE Transactions on Education*, vol. 48, no. 4, pp. 719–728, 2005. DOI: 10.1109/TE.2005.858400.

[7] O. Alasmari, J. Singer, and M. B. Ada, "Online coding tutorial systems: A new category of programming learning platforms," in *2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC)*, 2024, pp. 2222–2227. DOI: 10.1109/COMPSAC61105.2024.00356.

[8] Ł. Tomczyk, M. L. Mascia, and F. D. Guillen-Gamez, "Video tutorials in teacher education: Benefits, difficulties, and key knowledge and skills," *Education Sciences*, vol. 13, no. 9, p. 951, 2023.

[9] A. Kadriu, L. Abazi-Bexheti, H. Abazi-Alili, and V. Ramadani, "Investigating trends in learning programming using youtube tutorials," *International Journal of Learning and Change*, vol. 12, no. 2, pp. 190–208, 2020.

[10] P. Bii, "Chatbot technology: A possible means of unlocking student potential to learn how to learn," *Educational Research*, vol. 4, no. 2, pp. 218–221, 2013.

[11] S. Groothuijsen, A. van den Beemt, J. C. Remmers, and L. W. van Meeuwen, "Ai chatbots in programming education: Students' use in a scientific computing course and consequences for learning," *Computers and Education: Artificial Intelligence*, vol. 7, p. 100 290, 2024, ISSN: 2666-920X. DOI: https://doi.org/10.1016/j.caeai.2024.100290. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666920X24000936.

[12] C. W. Okonkwo and A. Ade-Ibijola, "Python-bot: A chatbot for teaching python programming.," *Engineering Letters*, vol. 29, no. 1, 2020.

[13] S. Groothuijsen, A. van den Beemt, J. C. Remmers, and L. W. van Meeuwen, "Ai chatbots in programming education: Students' use in a scientific computing course and consequences for learning," *Computers and Education: Artificial Intelligence*, vol. 7, p. 100 290, 2024.

[14] E. Frankford, C. Sauerwein, P. Bassner, S. Krusche, and R. Breu, "Ai-tutoring in software engineering education," in *Proceedings of the 46th International Conference on Software Engineering: Software Engineering Education and Training*, ser. ICSE-SEET '24, Lisbon, Portugal: Association for Computing Machinery, 2024, pp. 309–319, ISBN: 9798400704987. DOI: 10.1145/3639474.3640061. [Online]. Available: https://doi.org/10.1145/3639474.3640061.

[15] D. Nam, A. Macvean, V. Hellendoorn, B. Vasilescu, and B. Myers, "Using an llm to help with code understanding," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24, Lisbon, Portugal: Association for Computing Machinery, 2024, ISBN: 9798400702174. DOI: 10.1145/3597503.3639187. [Online]. Available: https://doi.org/10.1145/3597503.3639187.

[16] J. Richards and M. Wessel, *What you need is what you get: Theory of mind for an llm-based code understanding assistant*, 2024. arXiv: 2408.04477 [cs.SE]. [Online]. Available: https://arxiv.org/abs/2408.04477.

[17] C. Yu, "Unlocking the full potential of ai chatbots: A guide to maximizing your digital companions," in *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*, ser. FSE 2024, Porto de Galinhas, Brazil: Association for Computing Machinery, 2024, pp. 680–682, ISBN: 9798400706585. DOI: 10.1145/3663529.3664457. [Online]. Available: https://doi.org/10.1145/3663529.3664457.

[18] S. Barke, M. B. James, and N. Polikarpova, "Grounded copilot: How programmers interact with code-generating models," *Proc. ACM Program. Lang.*, vol. 7, no. OOPSLA1, Apr. 2023. DOI: 10.1145/3586030. [Online]. Available: https://doi.org/10.1145/3586030.

[19] N. Gardella, R. Pettit, and S. L. Riggs, "Performance, workload, emotion, and self-efficacy of novice programmers using ai code generation," in *Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1*, ser. ITiCSE 2024, Milan, Italy: Association for Computing Machinery, 2024, pp. 290–296, ISBN: 9798400706004. DOI: 10.1145/3649217.3653615. [Online]. Available: https://doi.org/10.1145/3649217.3653615.

[20] S. D'Mello and A. Graesser, "Dynamics of affective states during complex learning," *Learning and Instruction*, vol. 22, no. 2, pp. 145–157, 2012, ISSN: 0959-4752. DOI: https://doi.org/10.1016/j.learninstruc.2011.10.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0959475211000806.

[21] D. Demszky, D. Movshovitz-Attias, J. Ko, A. Cowen, G. Nemade, and S. Ravi, *Goemotions: A dataset of fine-grained emotions*, 2020. arXiv: 2005.00547 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2005.00547.

[22] J. Zamfirescu-Pereira, R. Y. Wong, B. Hartmann, and Q. Yang, "Why johnny can't prompt: How non-ai experts try (and fail) to design llm prompts," in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, ser. CHI '23, Hamburg, Germany: Association for Computing Machinery, 2023, ISBN: 9781450394215. DOI: 10.1145/3544548.3581388. [Online]. Available: https://doi.org/10.1145/3544548.3581388.

[23] M. Lount and A. Bunt, “Characterizing web-based tutorials: Exploring quality, community, and showcasing strategies,” in *Proceedings of the 32nd ACM International Conference on The Design of Communication CD-ROM*, ser. SIGDOC ’14, Colorado Springs, CO, USA: Association for Computing Machinery, 2014, ISBN: 9781450331838. DOI: 10.1145/2666216.2666221. [Online]. Available: https://doi.org/10.1145/2666216.2666221.

[24] V. Truong, “Textual emotion detection–a systematic literature review,” 2024.