

Analyzing Chat Protocols of Novice Programmers Solving Introductory Programming Tasks with ChatGPT

Andreas Scholl ¹, Daniel Schiffner ², and Natalie Kiesler ³



Abstract: Large Language Models (LLMs) have taken the world by storm, and students are assumed to use related tools at a great scale. In this research paper we aim to gain an understanding of how introductory programming students chat with LLMs and related tools, e.g., ChatGPT-3.5. To address this goal, computing students at a large German university were motivated to solve programming exercises with the assistance of ChatGPT as part of their weekly introductory course exercises. Then students (n=213) submitted their chat protocols (with 2335 prompts in sum) as data basis for this analysis. The data was analyzed w.r.t. the prompts, frequencies, the chats' progress, contents, and other use pattern, which revealed a great variety of interactions, both potentially supportive and concerning. Learning about students' interactions with ChatGPT will help inform and align teaching practices and instructions for future introductory programming courses in higher education.

Keywords: ChatGPT-3.5, large language models, students, interaction pattern, application, chat protocols, introductory programming, higher education

1 Introduction

The increasing need for competent computing graduates proficient in programming, software development, and related technical competencies [Ca17] is one of the factors exacerbating pressure on higher education institutions to offer high quality, competency-based education [Ra21]. However, the latter requires extensive resources, mentoring, and, for example, formative feedback for learners, especially in introductory programming classes [Je22; Lo24]. This is due to the fact that novices experience a number of challenges in the process, which have been subject to extensive research in the past decades [Du86; Lu18; SS86]. Among them are cognitively demanding competencies [Ki20; Ki24], such as problem understanding, designing and writing algorithms, debugging, and understanding error messages [Du86; ER16; Ki20; Lu18; SS86]). Educators' expectations towards novice learners and what they can achieve in their first semester(s) seem to be too high and unrealistic [Lu16; Lu18; WCL07]. Moreover, the student-educator ratio in introductory programming classes keeps increasing in German higher education institutions, thereby limiting resources to provide feedback and hints, and adequately address heterogeneous prior knowledge and diverse educational biographies [Pe16; SB22].

1 Nuremberg Tech, Computer Science, Keßlerplatz 12, 90489 Nuremberg, Germany,
andreas.scholl@th-nuernberg.de,  <https://orcid.org/0009-0009-1275-1374>

2 DIPF | Leibniz Institute for Research and Information in Education, Information Centre Education, Rostocker
Straße 6, 60323 Frankfurt am Main, Germany, d.schiffner@dipf.de,  <https://orcid.org/0000-0002-0794-0359>

3 Nuremberg Tech, Computer Science, Keßlerplatz 12, 90489 Nuremberg, Germany,
natalie.kiesler@th-nuernberg.de,  <https://orcid.org/0000-0002-6843-2729>

Fortunately, educational technologies based on Large Language Models (LLMs) have the potential to support novice learners of programming seeking help, if carefully used. LLMs and related tools have been shown to pass introductory programming tasks and courses [Ge23; KS23a; Sa23], enhance programming error messages [Le23a; Le23b; Ma23; Sa22], and generate formative feedback [ADS23; AKS24; BK23; KKL24; RKJ23] for learners. It is therefore not surprising that the emergence of LLMs triggered an extensive discussion within the computing education community regarding learning objectives, curricula, assessments and ethical questions [Pr23b]. Due to the novelty of LLMs and their recent broad availability, most studies investigated LLMs from the educator's perspective, hypothesizing about (students') application scenarios (e.g., [A124; Jo24]). The student perspective (e.g., their trust or attitude towards LLMs [Am24; RHG24]), let alone their actual use of LLMs in a classroom setting, has only been addressed in a few very recent studies (e.g., [Li24b]). Understanding students' use of LLMs and respective interactions, however, is crucial to support them and adequately instruct them regarding the critical and reflective application of LLMs in their studies.

Therefore, the present work investigates how students in an introductory programming class actually work with LLMs, such as ChatGPT, when doing their coursework. Specifically, it is guided by the research question (RQ): *How do students chat with ChatGPT in the context of introductory programming course assignments?* To answer this question, an exercise sheet was developed as part of an introductory programming course at Goethe University in the winter term 2023/24. Students were instructed to solve programming tasks with the assistance of ChatGPT-3.5 (i.e., the freely available version at the time, December 2023), and to submit their chat protocols, which is the data basis of this work. The quantitative and qualitative analysis of the chat protocols provides the first insights of its kind into students' authentic interactions with the LLM in a curricular course setting. The contribution of this paper is thus a collection of students' interaction with an LLM, such as ChatGPT. These insights have implications for educators considering the use of LLMs in their introductory programming courses, as they can help inform the development of new forms of instructions with the goal of fostering a conscious and critical use.

2 Related Work

In 2019, a literature review of 146 articles on artificial intelligence applications in higher education [Za19] pointed out the rapidly increasing relevance of Artificial Intelligence (AI) in computing education and related fields. The review identified four main areas of interest: “adaptive systems and personalization, assessment and evaluation, profiling and prediction and intelligent tutoring systems” [Za19, p. 11]. Notably, Zawacki-Richter et al. recognized “the dramatic lack of critical reflection” [Za19, p. 21] of the challenges and risks of AI tools, including pedagogical and ethical considerations.

Five years later, in 2024, Generative AI (GenAI) and related tools have already taken the world by storm [Pr23b; Pr24], and implications for higher education have become the focus

of the discourse in many disciplines, including computing education research [Ma24]. This development is due to the broad availability of GenAI tools, which gained popularity with the launch of OpenAI's ChatGPT in late November 2022. Its impressive performance in, for example, introductory programming tasks and exams [Ge23; KS23a; Sa23] initiated discussions on LLMs' potential and how it would affect computing curricula, learning objectives, but also teaching, learning, and assessing in general [Be23; Ki23; Pr23a; Pr23b]. Prather et al. further emphasize the fast pace of GenAI tools and their developments, indicating the need for ongoing research to keep up with recent technological advances. This is especially true if research data (e.g., for benchmarking these tools) are not available [KS23b; Pr23b].

General concerns and limitations of LLMs and GenAI tools comprise the accuracy and reliability in educational settings, as ChatGPT is susceptible to (re-)producing biased or inaccurate information [Gi24]. The tool's knowledge base, which may lack recent information, can further lead to inaccuracies. Hallucinations add to this problem. Finally, it can bypass plagiarism detection tools. All of these aspects challenge the integrity of academic work and the reliability of AI as an educational tool, indicating the need for action by educational institutions [Gi24; Pr23b; Zh22a].

In the context of introductory programming education, studies revealed the potential of GenAI tools for several application scenarios. Among them are the effective generation of code explanations [Ma23; Sa22], enhanced programming error messages [Le23b], but also the analysis of student code with the goal of fixing students' errors [Ph23; Zh22b], or providing (elaborate) feedback [AKS24; KLK24]. All of these studies focus on students' (potential) use of LLMs.

For example, MacNeil et al. investigated students' perceptions towards automatically generated line-by-line code explanations by OpenAI's Codex and GPT-3 as part of an e-book. A majority of students evaluated the generated explanations as helpful [Ma23]. Similarly, Leinonen et al. found that students rate code explanations generated by GPT-3 better on average than explanations from their peers. They also note that students showed no aversion towards LLM-generated feedback, and that students prefer line-by-line code explanations [Le23a]. It is therefore not surprising that LLMs are being integrated into educational tools and environments to generate novice-friendly explanations tailored to each error [Ta24], without revealing code solutions [Ka24], or to provide guardrails [Li24a].

Related to that is the use of LLMs to analyze students' solutions and fix errors, or provide various types of elaborated feedback [Na06; Sh08] for novice programmers [KJH18]. Phung et al. investigated the use of LLMs to fix syntax errors in Python programs and developed a technique to receive high precision feedback. Other studies qualitatively explored the feedback generated by ChatGPT [ADS23; AKS24; KLK24]. Kiesler et al. characterized the feedback generated by ChatGPT-3.5 in response to authentic student solutions for introductory programming tasks. They found stylistic elements, textual explanations of the cause of errors and their fix, illustrating examples, meta-cognitive and motivational

elements, but also misleading information, uncertainty in the model’s response, and requests for more information by the LLM. Azaiz et al. [ADS23] noted difficulties of GPT-3.5 with the formatting of its output, recognizing correct solutions, and it hallucinated errors. Roest et al. [RKJ23] conclude that the feedback generated by the GPT-3.5-turbo model seems to lack sufficient detail towards the end of an assignment. Nonetheless, a recent qualitative evaluation of GPT4 Turbo’s feedback shows notable improvements, as the outputs are more structured, consistent, and always personalized [AKS24].

While some of the aforementioned studies utilize authentic student data (e.g., students’ solutions as an input to LLMs), others directly involve students to survey their perspective, or usability of LLMs. One example is the study by Prather et al. [Pr23c] who investigated students’ use of GitHub Copilot in an introductory programming assignment. Their observations and interviews explore students’ perceptions regarding the (usability) challenges and potential benefits of this technology, resulting in design implications. Similarly, Vaithilingam et al. explored the usability of LLMs and their code generation abilities from the student perspective, concluding that Copilot’s design should be improved [VZG22]. Jayagopal et al. explored the learnability of program synthesizers (including Copilot) by observing and interviewing students, which resulted in a set of lessons learned regarding system design. They also found the possibility to write a prompt to be perceived as more exciting and less confusing by students, compared to tools requiring format specifications [JLC22]. In another usability study, Barke et al. presents a grounded theory analysis of programmers’ interactions with Copilot, based on observations of 20 experienced programmers. They distinguish users in acceleration mode (i.e., Copilot is used to consciously achieve a goal faster) and exploration mode (i.e., Copilot is used to explore options). Moreover, they show that over-reliance on Copilot can reduce the task completion rate, and cognitive load can result from having to choose from multiple suggestions [BJP23].

In another study, Denny et al. [De23] investigated first year students’ interactions (n=54) with *Promptly* in the context of a Python class. The tool utilizes the concept of “Prompt Problems”, aiming at teaching students how to prompt LLMs based on predefined problems. It automatically evaluates (i.e., executes and tests) code generated by an LLM (text-davinci-003 and gpt-3.5-turbo). In their analysis, however, Denny et al. focus on the quantification of students’ interactions with Promptly (e.g., number of words per prompt, prompts per problem) but not the follow-up interactions. Moreover, students’ responses to reflective questions were used as a data basis.

Even though many of these studies have investigated potential applications of LLMs by computing students, and specifically, novice programmers [AKS24; KLK24; Le23a; Ma23], they are conducted from an educator’s perspective, meaning all use cases had been predefined. So, they are not necessarily authentic. The presented usability studies [BJP23; JLC22; Pr23c; VZG22] were based on observations and interviews, and did not focus on novice programmers and their application of LLMs and related tools. The only study on students’ interactions [De23] emphasizes quantitative aspects, and does not analyze follow-up prompts and interaction patterns.

In addition, few very recent studies so far have been dedicated to computing students' trust in LLMs [Am24], and their attitude towards related tools [RHG24]. Even fewer studies investigated students' use of LLMs based on actual interaction data. Liu et al. investigated students' use of ChatGPT-3.5 as virtual teaching assistants in a classroom setting. However, they evaluated the tools' effectiveness (which was not based on students' interaction patterns) [Li24b]. Grande et al. evaluated the student perspectives on using ChatGPT for an assignment on professional ethics, but they used the students' submissions to the task as a data basis [GKR24]. As a result, there is a lack of authentic interaction data of students using an LLM, i.e., their complete chat protocols, and their (qualitative) evaluation in the context of an introductory programming class.

3 Methodology

To answer the RQ *How do students chat with ChatGPT in the context of introductory programming course assignments?*, the study leverages empirical data collected from students (n=213) enrolled in an introductory programming course. Students were asked to complete a series of programming exercises with the assistance of ChatGPT-3.5, and to submit their chat protocols as part of the assignment. In this section, we introduce the course context, instructions, the selected tasks, and the methodology applied for data analysis.

3.1 Course Context Used for Data Gathering

The context of this study was an introductory programming course for first-year computing students (n=790) at Goethe University Frankfurt (Germany) during the winter term 2023/24. The majority were computer science students, with some individuals from other majors who pursue computer science as a minor. The course is designed for novice learners or programmers, with no prerequisites. The class was accompanied by a Moodle course with learning materials. It comprised a 2-hour lecture per week for all 790 students, and a 2-hour tutorial session in groups of 20-30 students. A crucial components of the tutorial are the weekly or bi-weekly exercise sheets with programming tasks. Students are awarded points for (mostly individually) submitting their solutions, which contribute towards the exam held at the end of the semester.

For the present study, a specific exercise sheet was developed for the tutorials in the week of December 6, 2023. Students were expected to submit their individual solutions two weeks later, group work was not permitted. Students were further instructed to “complete the tasks using ChatGPT via the free version” (3.5) on the web interface, and to submit “all prompts and responses” as paired entries in a spreadsheet via the Moodle course. However, no specific instructions were provided on how to interact with ChatGPT-3.5, except for a reference to OpenAI's guide on prompt engineering [Op23]. This approach was deliberately chosen to avoid influencing students' interactions with ChatGPT-3.5. The exercise sheet

was available in both German and English to accommodate the diverse student body of the course. Using ChatGPT for this exercise sheet was voluntary, but two extra points were offered as an incentive. Moreover, students had been introduced to the present study's objective and the process during the lecture preceding the tutorial.

3.2 Selected Tasks

The exercise sheet addressed the concept of *recursion*, *functions*, *lists*, *conditionals*, *string manipulation*, and *documentation*. The exercise comprised two main tasks, while each one included sub-tasks. The first task, consisting of four sub-tasks, presented code snippets with recursive elements for several operations: (1a) summation of the digits of a number, (1b) reversing a list, (1c) performing multiplication, and (1d) computing the Ackermann function. Students were asked to read and interpret the given code snippets with the goal of determining the output of the code, the number of function calls, and identifying the implemented type of recursion.

The second task presented a more complex challenge, namely to implement a function that determines the number of “happy strings” within all sub-strings of a given string. A “happy string” was defined as a string that can either be rearranged into (or already is) a repetition of some string. To solve this task, students were required to (2a) test a given string for the “happy” property, and (2b) test all possible sub-strings of the given string. Students were encouraged to adopt a recursive approach by awarding additional points.

Prior to their integration into the exercise sheet, the authors evaluated ChatGPT-3.5's capability to solve the selected tasks. For task 1, ChatGPT demonstrated a high success rate in identifying the correct functions but frequently miscalculated the number of calls, often overlooking the base case. Task 2 proved to be more challenging for ChatGPT-3.5. Although the responses included the necessary steps for problem solving, it never included a correct solution (we tried 10 regenerations). This may have been due to a degree of ambiguity, and, for example, the uncommon term “happy string” in the task description.

3.3 Data Analysis

To evaluate students' interactions with ChatGPT-3.5, and how they seek assistance from the GenAI tool, the collected chat protocols were quantitatively and qualitatively analyzed. For the **quantification** of students' use patterns, the following aspects were investigated:

- *Number of prompts per student*: The total count of prompts given to ChatGPT-3.5, offering insights into the extent of interaction from students with the GenAI tool for the given tasks.
- *Number of words per prompt*: Identifying the verbosity or conciseness of students' prompts, to help characterize students' descriptions of the problem or task (and its complexity).

- *Follow-up interactions per student*: This characteristic aims to capture the iterative nature of problem-solving interactions with ChatGPT-3.5. It is supposed to help identify students' use patterns.

In addition, the chat protocols were **qualitatively** analyzed [Ma00] by two of the authors to identify the type of follow-up interactions, the contents or issues addressed by the students' prompts, and the overall (interaction) strategy students seem to pursue or apply.

- *Solution requests (SR)*: Determines whether the student's prompt explicitly seeks a direct solution, shedding light on the intention behind the query and the expected outcome of the interaction (e.g., if only the task description is prompted). Frequencies emphasize the extend to which students apply this kind of request.
- *Type of follow-up interaction*: Categorizes whether a prompt is a standalone query (STA), a follow-up to a previous prompt (PRE), a response to a ChatGPT answer (RES), or a correction of a previous response (COR). We further add the frequencies of these types. Each prompt was categorized into only one of these four types.
- *Issue or problem solving step addressed by the prompt*: This classification identifies the issue addressed by the prompt. These categories include common problem solving steps and well-known issues for novice learners based on the literature [Du86; ER16; Ki20; Lu18; SS86]: problem understanding (PU), conceptual understanding (CU), code generation (CG), debugging (DE), runtime analysis (RA), syntax/style (SY), documentation (DO), test cases (TC), and other categories (OT). It should be noted that a single prompt can reflect multiple issues within the problem solving process.
- *Categories describing students' interactions*: Describes how students navigate and modify their prompts based on the generated output. Categories reflecting these interaction pattern were inductively built based on the material.

4 Results

A total of 360 students successfully completed the exercise sheet. Of these, 305 students engaged with ChatGPT-3.5 as part of the exercise. Due to the absence of the required template usage or submission in non-processable file formats, the data from 92 students could not be included in the final analysis. Therefore, the final dataset comprises 213 students, providing a collection of 2335 prompts for detailed examination. In total, we received 1668 German and 426 English prompts (The data is available online [SK24]).

Students engaged in approximately 10.96 prompts on average. The median during all interactions is 7 prompts per student. Based on these numbers, we categorize students into three groups, as shown in Table 1: Group A, comprising individuals who use 0-5 prompts; Group B, including those who use 6-11 prompts; and Group C, consisting of students who use 12 or more prompts. The distinction yields almost equal sized groups.

Regarding the evaluation of word counts per prompt, it should be noted that the number of words for task 1 alone is 146, and 175 for task 2. This partially explains the decrease in

Range of prompts per student	No. of Students	No. of Prompts	Avg word count	Solution Requests (SR)	Follow-Up Interactions			
					STA	PRE	RES	COR
A: 0 to 5	66	262	73.74	213	172	57	25	8
B: 6 to 11	79	612	51.49	416	240	205	111	56
C: 12 and more	68	1461	40.46	603	358	462	322	319

Tab. 1: Quantification of students' interactions with ChatGPT-3.5, including some of the qualitative aspects of their follow-up interactions.

the average word count from Group A to Group B by 30%, followed by a further decline from Group B to Group C by 21% to 40.46 words on average per prompt. Furthermore, the number of solution requests (SR) shows a declining trend across the groups. In Group A, approximately 81% of prompts are SR, which decreases to 68% in Group B and 41% in Group C (see Table 1).

The nature of follow-up interactions (FIs) also varies among the groups (see Table 1). In Group A, the majority of interactions had no follow-up (STA, in 172 cases). In many cases, the prompts resembled the task descriptions, which had been used as input. At the same time, 90 instances of FIs were observed, comprising 57 follow-ups to a previous prompt (PRE), 25 responses to a generated answer by ChatGPT (RES), and 8 corrections of responses (COR), indicating a correction rate of approximately 3%.

Group B reveals an increase of FIs, with 372 instances of FIs, and somewhat fewer instances (240) without an FI (STA). Notably, there is a greater number of students referring back to their own prompts (PRE, 205 cases), while 56 corrections of a previous response (COR) were identified (9%). In contrast, Group C demonstrates a significant increase of FIs, with the vast majority of follow-ups (1103 is sum). In Group 3, the correction rate (22%) was particularly high, with 319 correction instances (COR).

The analysis of student prompts relating to programming problem-solving categories reveals several trends, summarized in Table 2. For example, the proportion of initial prompts aimed at understanding the problem (PU), including solution requests, is smaller in longer conversations. In group A, these prompts constitute 76% of the interactions, dropping to 50% in group B, and further declining to 34% in Group C. Conversely, there is an increase of prompts requesting help in debugging (DE) in longer conversations. This category sees a rise from 6% in group A to 17% in group B, and a further increase to 24% in group C. The number of prompts referring to runtime analyses (RA) is higher for group A, accounting for 15% of interactions. This may be due to fewer number of prompts in group A and runtime analysis being an explicit part of Task 1. In terms of concept understanding (CU), code generation (CG), and documentation (DO), there is a slight increase from group A to group B and C, while the percentage difference between group B and C is minimal. Additionally, the analysis shows a relatively stable percentage of prompts related to syntax/style (SY) from group A to C, and a relatively consistent ratio for test case prompts (TC) to overall prompts observed across group A, B, and C (about 10%).

Range of prompts per student	No. of Prompts	Problem-solving category								
		PU	CU	CG	DE	RA	SY	DO	TC	OT
A: 0 to 5	262	198	25	34	16	40	3	24	26	3
B: 6 to 11	612	303	87	121	101	55	10	83	58	10
C: 12 and more	1461	497	185	273	354	115	36	186	143	29

Tab. 2: Distribution of students’ prompts across the various problem-solving categories (multiple categories per student and prompt).

After qualitatively analyzing the prompts, we were able to identify two common patterns of interactions with the GenAI tool: (1) the *Task Description Prompts* pattern, and (2) the *Prompts in Own Words* pattern. For (1), students seem to use the given task description at least as part of their initial prompt. In the other pattern (2), students rephrase the task description or immediately focus on specific parts of the problem statement. We provide a summary of both pattern and sub-pattern in Table 3 and Table 4, whereas the example ID refers to the student ID in the chat protocols. The order of the prompting patterns in the tables is supposed to reflect on the increasing specificity. Table 3 summarizes respective use pattern for *Task Description Prompts*, showing different approaches to generate a solution using ChatGPT after having used the task description as an initial prompt.

Example ID	Follow-ups after initial task description prompt:
1012	No or few additional instructions.
1044	Giving direct orders / requests including style, documentation or corrections.
1199	Short, but extensive prompting for explanations to gain understanding.
1135	Specific prompts requesting explanations and corrections.
1070	Testing of initially generated code by the LLM. Providing incorrect (console) output followed by “Correct the code” instructions. No additional input or adaptation.
1195	Testing of initially generated code by the LLM. Providing incorrect (console) output followed by “Correct the code” instructions, resulting in disappointment.
1006	Testing of initially generated code by the LLM. Providing incorrect (console) output followed by “Correct the code” instructions. Followed by instruction to restart, and adaptation of instruction.
1156	Requesting (conceptual) explanations. Follow-up to create text and not bulletpoints.

Tab. 3: Examples for the *Task Description Prompts*-pattern encountered in the dataset.

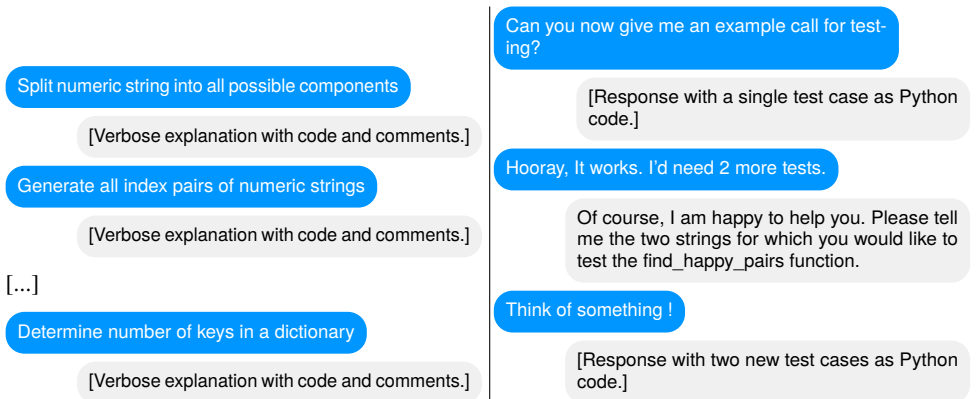
For the *Prompt in Own Words* pattern (summarized in Table 4), students exhibit different approaches to achieve the correct solution or responses to their questions. These pattern seem more directed towards a specific aspect of the problem, or occurred only after the student had created a solution.

In addition to these two patterns, we observed some other interesting aspects of students’ interactions. This includes switching the language from German to English as soon as the GenAI tool produced English outputs. Some prompts started in English, but students

Example ID	Prompts and follow-ups in students' own words
1014	Demanding explanations for specific concepts, e.g. indices, dictionaries.
1058	Adding own solution and asking specific questions.
1106	Providing additional task constraints. Task description is provided as a follow up. Requests specific aspects, e.g., test cases, documentation, while providing examples.

Tab. 4: Examples of *Prompts in Own Words*-pattern found in the dataset.

asked to create a documentation in German. Also, as part of the OT categorization, we found interjections, e.g., “Hooray”, “Thank You!”, which resemble human-to-human communication (ID 1037). In Figure 1, we present two exemplary excerpts from students and their ChatGPT interactions. Figure 1a illustrates that the student did not write complete sentences, but assigned explicit tasks to be executed. Figure 1b shows a human-to-human interaction with the tool.



(a) Excerpt from chat protocol 1014 with brief and direct commands. (b) Excerpt from chat protocol 1037 resembling human-to-human interaction.

Fig. 1: Excerpts of two example chat protocols illustrating different interaction patterns (translated into English by the authors).

5 Discussion

The analysis of students' chat protocols indicates diverse interactions with GenAI tools, such as ChatGPT, when being applied as part of an introduction programming course in higher education. For example, students engaged differently with ChatGPT in terms of the number of prompts used, their length, and regarding (the content of) follow-up interactions.

One group of students ($n=66$) only used 0-5 prompts while working on the selected tasks. This group may not have been thoroughly engaged in the interaction with ChatGPT. There is also a large number of solution requests (SR) present in this group. This may be an indicator for students looking for a quick solution to the given problems.

The second group of students ($n=79$) used ChatGPT differently. For example, we see an increasing number of prompts in this group. Still, many students requested the correct solution (SR). What is interesting though is that there are many follow-up interactions. This might indicate that either the initial response of ChatGPT was not considered good enough, or that the generated response actually fostered an interest in (an aspect of) the topic or problem. The increasing number of follow-ups on previous prompts (PRE) and responses to a ChatGPT answer (RES) disambiguate this group from students who only submitted 0-5 prompts.

The third group of students ($n=68$) used the GenAI tool most frequently, but for different purposes. For example, a smaller percentage of solution requests was identified. Especially due to the high number of corrections (COR), students seemed to be more reflective of the generated answers when trying to generate valid solutions. Especially interesting is the increase in responses to a ChatGPT answer (RES), which show that students actually used the tool as a “learning partner” instead of a simple query tool. This is also supported by the high numbers of prompts requesting help for understanding (PU) and debugging (DE).

The qualitative data analysis further reveals a difference in the categorization of problem-solving steps, the identified follow-up interactions, and prompting patterns (*Task Description Prompts* and *Prompts in Own Words*). While the former are well known especially in the context of feedback generation [KJH18; CLK24], the latter represent entirely new categories aiming to describe students’ interactions and prompting pattern with ChatGPT. Educators can use these patterns as a basis for designing instructions on how to (not) use LLMs and related tools, and to reflect on their limitations and potential benefits.

To answer our research question *How do students chat with ChatGPT in the context of introductory programming course assignments*, we conclude that there is a certain variety and range of students’ applications of the tool. There are at least three very different behaviors present. One seems to focus on problem solving by generating a single prompt to generate the best answer possible. Other students revealed a more “peer-like” interaction with the GenAI tool. They used more follow-up prompts, focused on problem solving step-by-step, tried to use the tool to identify and correct errors, and communicated respectively (we thus observed signs of anthropomorphism towards ChatGPT in this study). The third recognized use pattern is more straight forward and focuses on code generation, but utilizes the option of having an interactive tool. We assume that there are even more facets to these behaviors.

6 Limitations

A limitation of this study is the context of the data collection, which was a single class taking place at one university. Even though the high number of students participating in the study strengthens its representative character, the results may not be transferable to other educational contexts. Another limitation is that due to their knowledge of the study, students may have interacted differently with the tool than they would normally have in the context of an assignment, which is known as observer's paradox [RD39].

7 Conclusions and Future Work

This work presents the first-of-its kind exploratory study on students' use of ChatGPT as a tool to support problem-solving in the early stages of programming education at university level. To investigate students interactions, an exercise sheet had been designed and students were instructed to use ChatGPT, but not how to do so. Students' chat protocols were quantitatively and qualitatively analyzed to characterize students' use and interaction patterns. The high participation rate ($n=213$) allows us to draw several conclusions. For example, the results reveal three groups of students interacting differently with the tool. Among these groups are students seeking an immediate solution, but also students seeking help with problem understanding, conceptual knowledge, debugging, syntax error or style and documentation. Some students also used the chat more extensively than others, thereby resembling a "studdy buddy" or "virtual tutor". The chat protocols further revealed new patterns of interactions, e.g., students starting with *Task Description Prompts* and specifying their requests later, and those using *Prompts in Own Words* with more specific requests. However, there is a huge diversity in students' use of ChatGPT.

Implications for teaching and learning are manifold. For example, the diversity of use patterns reflects the heterogeneity of students' needs for support, and feedback. GenAI tools have the potential to offer that degree of personalization [AKS24]. Regardless, educators need to provide the necessary tools and advice on how to use them successfully, e.g., prompt patterns [Wh23]. Moreover, it is important to communicate the tools' limitations and potential pitfalls to students so they do not end up frustrated. The computing education community also needs to continue their discussion on the competencies required for successful graduates, and the role of GenAI tools in that process.

As our approach has been executed with a general LLM (ChatGPT-3.5), it would be interesting to see how a specialized AI tool for coding would increase its support for learners as part of future work. Moreover, a replication study at another institution or country would further strengthen the results. With the world facing new challenges and new tools every day, the potential in using AI for learning computer science is not yet fully understood. And to quote ChatGPT-3.5: "In this perpetual cycle of learning and teaching, computers enlighten students who, in turn, empower computers, perpetuating the beautiful recursion of knowledge."

References

- [ADS23] Azaiz, I.; Deckarm, O.; Strickroth, S.: AI-enhanced Auto-Correction of Programming Exercises: How Effective is GPT-3.5? *International Journal of Engineering Pedagogy (iJEP)* 13 (8), pp. 67–83, 2023, doi: 10.3991/ijep.v13i8.45621.
- [AKS24] Azaiz, I.; Kiesler, N.; Strickroth, S.: Feedback-Generation for Programming Exercises With GPT-4, 2024, doi: 10.48550/arXiv.2403.04449, arXiv: 2403.04449 [cs.AI].
- [Al24] Al-Hossami, E.; Bunescu, R.; Smith, J.; Teehan, R.: Can Language Models Employ the Socratic Method? Experiments with Code Debugging. In: *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1. SIGCSE 2024*, ACM, Portland, OR, USA, pp. 53–59, 2024, doi: 10.1145/3626252.3630799.
- [Am24] Amoozadeh, M.; Daniels, D.; Nam, D.; Kumar, A.; Chen, S.; Hilton, M.; Srinivasa Ragavan, S.; Alipour, M. A.: Trust in Generative AI among Students: An exploratory study. In: *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1. ACM*, New York, pp. 67–73, 2024, doi: 10.1145/3626252.3630842.
- [Be23] Becker, B. A.; Craig, M.; Denny, P.; Keuning, H.; Kiesler, N.; Leinonen, J.; Luxton-Reilly, A.; Prather, J.; Quille, K.: *Generative AI in Introductory Programming*. 2023, URL: <https://csed.acm.org/wp-content/uploads/2023/12/Generative-AI-Nov-2023-Version.pdf>.
- [BJP23] Barke, S.; James, M. B.; Polikarpova, N.: Grounded Copilot: How Programmers Interact with Code-Generating Models. *Proc. ACM Program. Lang.* 7 (OOPSLA1), 2023, doi: 10.1145/3586030.
- [BK23] Bengtsson, D.; Kaliff, A.: Assessment Accuracy of a Large Language Model on Programming Assignments, 2023, URL: <https://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-331000>.
- [Ca17] Camp, T.; Adrion, W. R.; Bizot, B.; Davidson, S.; Hall, M.; Hambrusch, S.; Walker, E.; Zweben, S.: *Generation CS: the growth of computer science*, New York, 2017, doi: 10.1145/3084362.
- [De23] Denny, P.; Leinonen, J.; Prather, J.; Luxton-Reilly, A.; Amarouche, T.; Becker, B. A.; Reeves, B. N.: Promptly: Using Prompt Problems to Teach Learners How to Effectively Utilize AI Code Generators, 2023, arXiv: 2307.16364 [cs.HC].
- [Du86] Du Boulay, B.: Some difficulties of learning to program. *Journal of Educational Computing Research* 2 (1), pp. 57–73, 1986, doi: 10.2190/3LFX-9RRF-67T8-UVK9.
- [ER16] Ebert, M.; Ring, M.: A presentation framework for programming in programing lectures. In: *Proc. EDUCON*. IEEE, pp. 369–374, 2016.
- [Ge23] Geng, C.; Zhang, Y.; Pientka, B.; Si, X.: Can ChatGPT Pass An Introductory Level Functional Language Programming Course?, 2023, arXiv: 2305.02230 [cs.CY].
- [Gi24] Gill, S. S.; Xu, M.; Patros, P.; Wu, H.; Kaur, R.; Kaur, K.; Fuller, S.; Singh, M.; Arora, P.; Parlikad, A. K.; Stankovski, V.; Abraham, A.; Ghosh, S. K.; Lutfiyya, H.; Kanhere, S. S.; Bahsoon, R.; Rana, O.; Dustdar, S.; Sakellariou, R.; Uhlig, S.; Buyya, R.: Transformative effects of ChatGPT on modern education: Emerging Era of AI Chatbots. *Internet of Things and Cyber-Physical Systems* 4, pp. 19–23, 2024, doi: 10.1016/j.iotcps.2023.06.002.
- [GKR24] Grande, V.; Kiesler, N.; Rodriguez, M. A. F.: Student Perspectives on Using a Large Language Model (LLM) for an Assignment on Professional Ethics. In: *Proceedings of the 2024 Conference on Innovation and Technology in Computer Science Education V. 2. ITiCSE 2024*, ACM, Milan, Italy, 2024, doi: 10.1145/3649217.3653624.

- [Je22] Jeuring, J.; Keuning, H.; Marwan, S.; Bouvier, D.; Izu, C.; Kiesler, N.; Lehtinen, T.; Lohr, D.; Peterson, A.; Sarsa, S.: Towards Giving Timely Formative Feedback and Hints to Novice Programmers. In: Proceedings of the 2022 Working Group Reports on Innovation and Technology in Computer Science Education. ITiCSE-WGR '22, ACM, Dublin, Ireland, pp. 95–115, 2022, doi: 10.1145/3571785.3574124.
- [JLC22] Jayagopal, D.; Lubin, J.; Chasins, S. E.: Exploring the learnability of program synthesizers by novice programmers. In: Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology. Pp. 1–15, 2022.
- [Jo24] Joshi, I.; Budhiraja, R.; Dev, H.; Kadia, J.; Ataullah, M. O.; Mitra, S.; Akolekar, H. D.; Kumar, D.: ChatGPT in the Classroom: An Analysis of Its Strengths and Weaknesses for Solving Undergraduate Computer Science Questions. In: Proc. of the 55th ACM Technical Symposium on Computer Science Education V. 1. ACM, New York, pp. 625–631, 2024, doi: 10.1145/3626252.3630803.
- [Ka24] Kazemitabaar, M.; Ye, R.; Wang, X.; Henley, A. Z.; Denny, P.; Craig, M.; Grossman, T.: CodeAid: Evaluating a Classroom Deployment of an LLM-based Programming Assistant that Balances Student and Educator Needs. arXiv preprint arXiv:2401.11314, 2024.
- [Ki20] Kiesler, N.: Towards a Competence Model for the Novice Programmer Using Bloom’s Revised Taxonomy – An Empirical Approach. In: Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education. ITiCSE '20, ACM, Trondheim, Norway, pp. 459–465, 2020, doi: 10.1145/3341525.3387419.
- [Ki23] Kiesler, N.: Beyond the Textbook: Rethinking Students’ Competencies in the LLM Era, Generative AI: Implications for Teaching and Learning, Uppsala, Sweden, 2023, doi: 10.13140/RG.2.2.28355.37922/1.
- [Ki24] Kiesler, N.: Modeling Programming Competency: A Qualitative Analysis. Springer International Publishing, Cham, 2024, ISBN: 978-3-031-47148-3.
- [KJH18] Keuning, H.; Jeuring, J.; Heeren, B.: A Systematic Literature Review of Automated Feedback Generation for Programming Exercises. ACM Trans. Comput. Educ. 19(1), 2018, doi: 10.1145/3231711.
- [KLK24] Kiesler, N.; Lohr, D.; Keuning, H.: Exploring the Potential of Large Language Models to Generate Formative Programming Feedback. In: 2023 IEEE Frontiers in Education Conference (FIE). Pp. 1–5, 2024, doi: 10.1109/FIE58773.2023.10343457.
- [KS23a] Kiesler, N.; Schiffner, D.: Large Language Models in Introductory Programming Education: ChatGPT’s Performance and Implications for Assessments, 2023, doi: 10.48550/arXiv.2308.08572, arXiv: 2308.08572 [cs.SE].
- [KS23b] Kiesler, N.; Schiffner, D.: Why We Need Open Data in Computer Science Education Research. In: Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1. ITiCSE 2023, ACM, Turku, Finland, pp. 348–353, 2023, doi: 10.1145/3587102.3588860.
- [Le23a] Leinonen, J.; Denny, P.; MacNeil, S.; Sarsa, S.; Bernstein, S.; Kim, J.; Tran, A.; Hellas, A.: Comparing Code Explanations Created by Students and Large Language Models. In: Proc. ITiCSE. Pp. 124–130, 2023, doi: 10.1145/3587102.3588785.
- [Le23b] Leinonen, J.; Hellas, A.; Sarsa, S.; Reeves, B.; Denny, P.; Prather, J.; Becker, B. A.: Using Large Language Models to Enhance Programming Error Messages. In: Proc. SIGCSE. ACM, 2023, doi: 10.1145/3545945.3569770.

- [Li24a] Liffiton, M.; Sheese, B. E.; Savelka, J.; Denny, P.: CodeHelp: Using Large Language Models with Guardrails for Scalable Support in Programming Classes. In: Proceedings of the 23rd Koli Calling International Conference on Computing Education Research. Koli Calling '23, ACM, Koli, Finland, 2024, doi: 10.1145/3631802.3631830.
- [Li24b] Liu, R.; Zenke, C.; Liu, C.; Holmes, A.; Thornton, P.; Malan, D. J.: Teaching CS50 with AI: Leveraging Generative Artificial Intelligence in Computer Science Education. In: Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1. SIGCSE 2024, ACM, New York, pp. 750–756, 2024, doi: 10.1145/3626252.3630938.
- [Lo24] Lohr, D.; Kiesler, N.; Keuning, H.; Jeuring, J.: “Let Them Try to Figure It Out First” – Reasons Why Experts (Do Not) Provide Feedback to Novice Programmers. In: Proceedings of the 2024 Conference on Innovation and Technology in Computer Science Education V. 1. ITiCSE 2024, ACM, Milan, Italy, 2024, doi: 10.1145/3649217.3653530.
- [Lu16] Luxton-Reilly, A.: Learning to Program is Easy. In: Proc. ITiCSE. Pp. 284–289, 2016, doi: 10.1145/2899415.2899432.
- [Lu18] Luxton-Reilly, A.; Simon; Albluwi, I.; Becker, B. A.; Giannakos, M.; Kumar, A. N.; Ott, L.; Paterson, J.; Scott, M. J.; Sheard, J.; Szabo, C.: Introductory Programming: A Systematic Literature Review. In: Proc. ITiCSE. ACM, New York, pp. 55–106, 2018, doi: 10.1145/3293881.3295779.
- [Ma00] Mayring, P.: Qualitative content analysis forum qualitative sozialforschung. Forum: qualitative social research 1 (2), 2000.
- [Ma23] MacNeil, S.; Tran, A.; Hellas, A.; Kim, J.; Sarsa, S.; Denny, P.; Bernstein, S.; Leinonen, J.: Experiences from Using Code Explanations Generated by Large Language Models in a Web Software Development E-Book. In: Proc. SIGCSE TS. Pp. 931–937, 2023, doi: 10.1145/3545945.3569785.
- [Ma24] MacNeil, S.; Leinonen, J.; Denny, P.; Kiesler, N.; Hellas, A.; Prather, J.; Becker, B. A.; Wermelinger, M.; Reid, K.: Discussing the Changing Landscape of Generative AI in Computing Education. In: Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2. ACM, New York, p. 1916, 2024, doi: 10.1145/3626253.3635369.
- [Na06] Narciss, S.: Informatives Tutorielles Feedback: Entwicklungs- und Evaluationsprinzipien auf der Basis instruktionspsychologischer Erkenntnisse. Waxmann Verlag, Münster, 2006.
- [Op23] OpenAI: Prompt Engineering, <https://platform.openai.com/docs/guides/prompt-engineering>, Accessed: 2024-04-08, 2023.
- [Pe16] Petersen, A.; Craig, M.; Campbell, J.; Tafilovich, A.: Revisiting why students drop CS1. In: Proc. Koli Calling. Pp. 71–80, 2016, doi: 10.1145/2999541.2999552.
- [Ph23] Phung, T.; Cambronero, J.; Gulwani, S.; Kohn, T.; Majumdar, R.; Singla, A.; Soares, G.: Generating High-Precision Feedback for Programming Syntax Errors using Large Language Models, 2023, arXiv: 2302.04662 [cs.PL].
- [Pr23a] Prather, J.; Denny, P.; Leinonen, J.; Becker, B. A.; Albluwi, I.; Caspersen, M. E.; Craig, M.; Keuning, H.; Kiesler, N.; Kohn, T.; Luxton-Reilly, A.; MacNeil, S.; Petersen, A.; Pettit, R.; Reeves, B. N.; Savelka, J.: Transformed by Transformers: Navigating the AI Coding Revolution for Computing Education: An ITiCSE Working Group Conducted by Humans. In: Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 2. ACM, New York, pp. 561–562, 2023, doi: 10.1145/3587103.3594206.

- [Pr23b] Prather, J.; Denny, P.; Leinonen, J.; Becker, B. A.; Albluwi, I.; Craig, M.; Keuning, H.; Kiesler, N.; Kohn, T.; Luxton-Reilly, A.; MacNeil, S.; Petersen, A.; Pettit, R.; Reeves, B. N.; Savelka, J.: The Robots Are Here: Navigating the Generative AI Revolution in Computing Education. In: *Proceedings of the 2023 Working Group Reports on Innovation and Technology in Computer Science Education*. ACM, New York, pp. 108–159, 2023, doi: 10.1145/3623762.3633499.
- [Pr23c] Prather, J.; Reeves, B. N.; Denny, P.; Becker, B. A.; Leinonen, J.; Luxton-Reilly, A.; Powell, G.; Finnie-Ansley, J.; Santos, E. A.: “It’s Weird That It Knows What I Want”: Usability and Interactions with Copilot for Novice Programmers. *ACM Trans. Comput.-Hum. Interact.* 2023, doi: 10.1145/3617367.
- [Pr24] Prather, J.; Leinonen, J.; Kiesler, N.; Gorson Benario, J.; Lau, S.; MacNeil, S.; Norouzi, N.; Opel, S.; Pettit, V.; Porter, L.; Reeves, B.; Savelka, J.; Smith IV, D. H.; Strickroth, S.; Zingaro, D.: How Instructors Incorporate Generative AI into Teaching Computing. In: *Proceedings of the 2024 Conference on Innovation and Technology in Computer Science Education V. 2. ITiCSE 2024*, ACM, Milan, Italy, 2024, doi: 10.1145/3649405.3659534.
- [Ra21] Raj, R.; Sabin, M.; Impagliazzo, J.; Bowers, D.; Daniels, M.; Hermans, F.; Kiesler, N.; Kumar, A. N.; MacKellar, B.; McCauley, R.; Nabi, S. W.; Oudshoorn, M.: Professional Competencies in Computing Education: Pedagogies and Assessment. In: *Proceedings of the 2021 Working Group Reports on Innovation and Technology in Computer Science Education*. ACM, New York, pp. 133–161, 2021, doi: 10.1145/3502870.3506570.
- [RD39] Roethlisberger, F. J.; Dickson, W. J.: *Management and the Worker*. Harvard University Press, Cambridge, 1939.
- [RHG24] Rogers, M. P.; Hillberg, H. M.; Groves, C. L.: Attitudes Towards the Use (and Misuse) of ChatGPT: A Preliminary Study. In: *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1. SIGCSE 2024*, ACM, Portland, OR, USA, pp. 1147–1153, 2024, doi: 10.1145/3626252.3630784.
- [RKJ23] Roest, L.; Keuning, H.; Jeuring, J.: Next-Step Hint Generation for Introductory Programming Using Large Language Models, 2023, arXiv: 2312.10055 [cs.CY].
- [Sa22] Sarsa, S.; Denny, P.; Hellas, A.; Leinonen, J.: Automatic Generation of Programming Exercises and Code Explanations Using Large Language Models. In: *Proc. ICER*. ACM, 2022, doi: 10.1145/3501385.3543957.
- [Sa23] Savelka, J.; Agarwal, A.; Bogart, C.; Sakr, M.: Large Language Models (GPT) Struggle to Answer Multiple-Choice Questions about Code, 2023, arXiv: 2303.08033 [cs.CL].
- [SB22] Strickroth, S.; Bry, F.: The Future of Higher Education is Social and Personalized! Experience Report and Perspectives. In: *Proc. CSEDU*. Vol. 1, pp. 389–396, 2022, doi: 10.5220/0011087700003182.
- [Sh08] Shute, V. J.: Focus on formative feedback. *Review of Educational Research* 78 (1), 2008.
- [SK24] Scholl, A.; Kiesler, N.: Data: Analyzing Chat Protocols of Novice Programmers Solving Introductory Programming Tasks with ChatGPT, 2024, doi: 10.17605/OSF.IO/WBKQV, URL: osf.io/wbkqv.
- [SS86] Spohrer, J. C.; Soloway, E.: Novice mistakes: Are the folk wisdoms correct? *Communications of the ACM* 29 (7), pp. 624–632, 1986, doi: 10.1145/6138.6145.
- [Ta24] Taylor, A.; Vassar, A.; Renzella, J.; Pearce, H.: dcc –help: Transforming the Role of the Compiler by Generating Context-Aware Error Explanations with Large Language Models. In: *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1. ACM*, New York, pp. 1314–1320, 2024, doi: 10.1145/3626252.3630822.

- [VZG22] Vaithilingam, P.; Zhang, T.; Glassman, E. L.: Expectation vs. Experience: Evaluating the Usability of Code Generation Tools Powered by Large Language Models. In. ACM, New York, pp. 1–7, 2022.
- [WCL07] Whalley, J.; Clear, T.; Lister, R.: The many ways of the Bracelet project. BACIT, 2007.
- [Wh23] White, J.; Fu, Q.; Hays, S.; Sandborn, M.; Olea, C.; Gilbert, H.; Elnashar, A.; Spencer-Smith, J.; Schmidt, D. C.: A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT, 2023, arXiv: 2302.11382 [cs.SE].
- [Za19] Zawacki-Richter, O.; Marin, V. I.; Bond, M.; Gouverneur, F.: Systematic review of research on artificial intelligence applications in higher education—where are the educators? *International Journal of Educational Technology in Higher Education* 16 (1), pp. 1–27, 2019.
- [Zh22a] Zhai, X.: ChatGPT User Experience: Implications for Education. 2022, DOI: <http://dx.doi.org/10.2139/ssrn.4312418>.
- [Zh22b] Zhang, J.; Cambronero, J.; Gulwani, S.; Le, V.; Piskac, R.; Soares, G.; Verbruggen, G.: Repairing Bugs in Python Assignments Using Large Language Models. arXiv preprint arXiv:2209.14876, 2022.