A Study on Personalized Programming Learning for High School Students Supported by Large Language Models

Chun-Lin Chou[1], Chin-Chun Chan[2], Pei-Hsuan Tung[3], Yu Chieh Wu[4], Chiu Pin Lin[5]
Tsing Hua University
*sherri92123@gmail.com

Abstract: *This study investigates the impact of an LLM-based AI teaching assistant on personalized programming learning for high school students and examines its effects on learning achievement, self-efficacy, learning attitudes, and AI acceptance. Using a single-group pretest-posttest design, thirty students from an information technology course at a senior high school in Hsinchu, Taiwan participated in a twenty-week experiment. The first ten weeks involved traditional instruction; the subsequent ten weeks introduced an LLM-based AI teaching assistant providing personalized explanations, code diagnosis, error feedback, and conceptual guidance based on individual proficiency levels. Research instruments included a programming achievement test, self-efficacy scale, learning attitude scale, and AI acceptance scale. Quantitative data were analyzed using descriptive statistics, paired-sample t-tests, and Wilcoxon signed-rank tests. Qualitative data from student interviews, AI interaction logs, and teacher observations were analyzed through thematic analysis and cross-validation.*

Keywords: LLM, personalized learning, AI teaching assistant, programming education, self-efficacy

1. Introduction

This study integrated an LLM-based AI teaching assistant into a Python elective course at a Taiwanese senior high school to supplement traditional instruction. The study explores how this system provides personalized learning support according to students' varying proficiency levels and to analyze its impact on students' programming learning achievement, self-efficacy, learning attitudes, and AI acceptance. The research objectives are as follows:

1. To investigate whether the AI teaching assistant can improve high school students' learning achievement in Python courses.
2. To explore the impact of the AI teaching assistant on students' self-efficacy, learning attitudes, and AI acceptance.
3. To analyze the inquiry process of student-AI teaching assistant interactions and teacher observation results.

2. Literature Review

2.1 Programming Learning Theory and Learning Difficulties

Programming is regarded as a highly challenging task in information education. Beginners must simultaneously understand syntax, control flow, and problem abstraction, which easily causes high cognitive load (Qian & Lehman, 2017). They often experience unstable mental models and accumulation of misconceptions in core concepts such as variables and loops (Robins et al., 2003; Kaczmarczyk et al., 2016). This variance in learning achievement is closely related to psychological motivation, with self-efficacy being an important factor affecting performance (Ramalingam et al., 2004; Lishinski, 2016). Without debugging strategies and immediate feedback, individuals with low self-efficacy are prone to anxiety and avoidance (Klassen & Klassen, 2018) and tend to accumulate misconceptions (Shute et al., 2016). According to Bandura's (1986) theory and Vygotsky's (1978) scaffolding learning perspective, learners can cross the Zone of Proximal Development (ZPD) with external support. With technological development, Large Language Models can serve as a new form of intelligent scaffolding, supporting students to gradually construct understanding by reducing task complexity (Reiser, 2004) and providing structured guidance (Kasneci et al., 2023).

2.2 Prompt Design and Theoretical Foundation of AI Teaching Assistants

Research indicates that specific prompt patterns foster this scaffolding by engaging learners in explanation and reflection (White et al., 2023). In programming education, such structured guidance is vital for helping novices transition from trial-and-error to systematic hypothesis generation, which facilitates the establishment of stable mental models (Ihantola, et al., 2025). Furthermore, prompt engineering—conversing with AI to articulate computational steps—serves as a valuable pedagogical activity that aids in error resolution and problem decomposition (Denny et al., 2023). Strategies like the "Alternative Approaches" pattern further promote knowledge transfer by enabling students to clarify logical relationships through solution comparison (White et al., 2023). Consequently, AI teaching assistants function across five instructional dimensions: guiding logical understanding, clarifying syntax to reduce cognitive load, offering alternative examples, encouraging self-explanation, and supporting systematic debugging. This reflects the evolution of generative AI from a mere answer provider into a scaffolding tool that supports the learning process.

3. Research Method

3.1 Research Procedure

This study adopted a single-group pretest-posttest quasi-experimental design with 32 tenth-grade students from a public senior high school in Hsinchu enrolled in a Python Programming elective course. The twenty-week study was divided into two stages. During weeks 1-10, traditional instruction through lectures and practical exercises established students' basic programming syntax and logical abilities. At week 10, students were divided into high-achievement (H) and low-achievement (L) groups based on median test scores. During weeks 11-20, a pretest was administered, followed by introduction of a personalized LLM-based AI teaching assistant on Discord. Students from each achievement level were randomly assigned to either 'generic prompt' or 'guided prompt' conditions, forming four groups of 8 students each: high-achievement generic (H1), high-achievement guided (H2), low-achievement generic (L1), and low-achievement guided (L2). This design examined the interactive effect of prior ability and AI guidance strategies (direct answers vs. scaffolded guidance) on learning achievement.

Data collection included pre- and post-intervention programming achievement test, alongside scales measuring programming achievement tests and scales assessing programming self-efficacy, learning attitudes, learning motivation, anxiety, and AI acceptance. Finally, quantitative data were integrated with qualitative data from semi-structured interviews to analyze changes in students' cognitive performance, problem-solving strategies, and learning attitudes.

3.2 Learning Achievement Test.

The learning achievement test initially consisted of 25 items (15 basic, 10 advanced) covered core concepts of Python programming, including variables and operations, flow control, data structures, functions, string processing, and debugging, to assess students' understanding and application abilities in programming knowledge. The researchers conducted item analysis based on pilot test results, calculating each item's difficulty index (p value) and discrimination index (D value), deleting or revising items with insufficient discrimination ($D < 0.30$), and adjusting item stems and options. After revision, the formal achievement test retained 24 items with a total score of 12 points, a time limit of 60 minutes, and was used as the learning achievement test for Stage 1 and Stage 2 according to the course schedule to compare students' learning performance before and after AI teaching assistant intervention.

3.3 Attitude Scales

This study employed three revised scales, all using a five-point Likert scale (1 = strongly disagree, 5 = strongly agree), to collect psychological and attitudinal changes during students' programming learning process, totaling 27 items revised to minimize respondent burden. For attitude and psychological scales, this study employed three five-point Likert scales that have undergone reliability and validity testing for assessment:

1. Programming Self-Efficacy Scale: Adapted from Ramalingam and Wiedenbeck (1998), consisting of 10 items measuring dimensions including comprehension, logical control, and problem-solving, with reliability Cronbach's α ranging from .92 to .98.

2. Programming Attitude Scale: Adapted from Wiebe et al. (2003), consisting of 8 items measuring learning attitude, perceived value, and confidence toward programming, with reliability α approximately .91.

3.AI Education Acceptance Scale: Adapted from Davis (1989), consisting of 9 items covering perceived ease of use, perceived usefulness, and behavioral intention, with reliability α ranging from .87 to .94.

3.4 AI Teaching Assistant System and Prompt Design

3.4.1 Generic and Guided Prompt design

This study developed an AI teaching assistant system based on the Discord platform and Large Language Models (LLM), designing two sets of prompt templates according to Cognitive Load Theory (Sweller, 2010).

• Generic Prompt: Adopts task-oriented design, emphasizing conciseness and technical accuracy in responses, directly providing answers or examples to student questions without proactively extending guidance.

• Guided Prompt: Role-set as a guide, restricting direct provision of answers. The system provides five assistance functions based on scaffolding theory, guiding students to recognize needs and construct problem-solving approaches.

Based on relevant literature, this study summarizes the five major functions and theoretical foundations of guided prompts as follows:

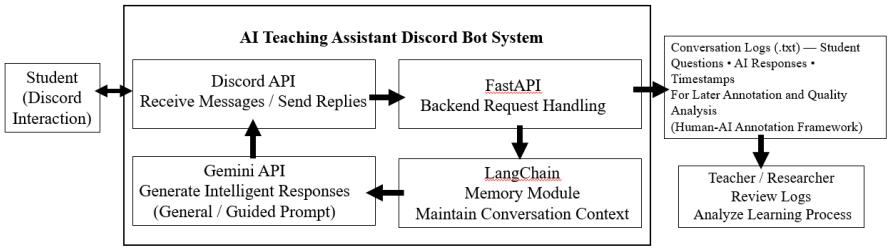
1. Logical Explanation: Step-by-step explanation of execution flow, helping to establish mental models and reduce working memory load (Sweller, 2010; Robins et al., 2003).
2. Syntax Explanation: Real-time indication of errors and provision of rule explanations, supporting individualized debugging processes (Ihantola et al., 2010; Keuning et al., 2018).
3. Alternative Approach: Providing different implementation methods to compare advantages and disadvantages, promoting style judgment and knowledge transfer (Guzdial, 2015; Kölling et al., 2003).
4. Add Comments: Requiring explanation of code in natural language, promoting externalization of thinking and strengthening metacognition (Schraw & Moshman, 1995; Tenenberg & Murphy, 2005).
5. Code Review: Conducting systematic error diagnosis and revision suggestions, improving debugging efficiency (Ala-Mutka, 2005; Keuning et al., 2018).

3.4.2 AI Teaching Assistant Discord Bot System Architecture

We designed the AI Teaching Assistant Discord Bot System provides personalized programming education. Students interact via Discord platform, where messages are processed through FastAPI backend services. LangChain maintains conversation context and memory across interactions, enabling coherent dialogue continuity. The Gemini API generates pedagogically-appropriate responses using customized prompts—either general prompt or guided prompt for students requiring scaffolded support. All interactions are automatically logged with timestamps, creating comprehensive datasets for educational research. Teachers and researchers analyze these conversation logs through a Human-AI Annotation Framework, identifying learning patterns and refining instructional strategies.

Figure 1

AI Teaching Assistant Discord Bot System Architecture



4. Results Analysis

This section presents the quantitative analysis results of two intervention modes—'generic prompt' or 'guided prompt'—on students' learning achievement, programming self-efficacy, learning attitudes, and AI acceptance scales, and explains the findings of this study.

4.1 Learning Achievement Performance

4.1.1 Wilcoxon Signed-Rank Test

To examine the impact of AI teaching assistant intervention on overall student programming ability, this study employed a paired-sample Wilcoxon signed-rank test to analyze differences in students' pretest and posttest scores. Statistical results showed that all students (N = 32) had significantly higher average scores on the posttest (M = 8.28, SD = 2.36) than pretest scores (M = 6.12, SD = 1.96), with test results reaching a highly significant level (Z = 5.95, p < .001). This result confirms that regardless of students' initial ability, introducing an LLM-based AI teaching assistant system as classroom and after-class support can effectively improve their programming problem-solving performance.

Table 1.

Paired-sample t-test results for overall students' pretest and posttest.

Test Stage	*n*	*M*	*SD*	*t*	*df*	*p*
Pretest	32	6.12	1.96	5.95	31	< .001*
Posttest	32	8.28	2.36			

*Note. * p < .05 indicates significant difference.*

4.1.2 Four-Group Subdivision Analysis

Subgroup analysis revealed that while all groups exhibited upward trends, only the low-achievement guided group (L2) achieved statistically significant improvement (p=.0078). The results indicate that while LLM-based AI assistants effectively enhance overall programming achievement(p<.001), guided prompts provide critical scaffolding that is particularly effective for improving learning achievement of low-achievement students.

Table 2.

Four-group AI intervention pre-post difference analysis (Wilcoxon signed-rank test)

Group	*N*	Pre Mean	Post Mean	Wilcoxon (T, p)	Mean Diff.	Significance
H1	8	7.63	8.50	T = 5.5, p = .1719	+0.875	ns
H2	8	5.38	7.88	T = 0.0, p = .0625	+2.500	ns
L1	8	6.13	7.25	T = 3.0, p = .1562	+1.125	ns
L2	8	5.38	9.50	T = 0.0, p = .0078	+4.125	Sig.

Note. ns = not significant, Sig. = significant.

4.2 Scale Analysis

4.2.1 Descriptive Statistical Analysis

Descriptive analysis revealed that high-achievement groups (H1, H2) possessed higher programming self-efficacy and learning attitudes, reflecting stable confidence and engagement. In contrast, low-achievement groups (notably L2) exhibited higher AI acceptance, indicating greater trust and willingness to utilize the AI teaching assistant.

Table 3.

Descriptive statistics of four groups on three scales (percentage standardized).

Group	Self-Efficacy (CPSES)	Attitude (ATT)	AI Acceptance (AIA)
H1	67.50% ± 18.91	82.19% ± 15.03	73.33% ± 6.29

H2	75.75% ± 15.54	75.94% ± 16.58	86.11% ± 15.70
L1	59.25% ± 24.12	69.38% ± 27.96	76.67% ± 24.37
L2	71.75% ± 6.18	68.12% ± 14.38	89.17% ± 11.66

4.2.2 Prompt Group Comparison

Students were further divided into generic prompt group (H1+L1) and guided prompt group (H2+L2) according to prompt type to compare differences in performance on the three scales. Analysis results showed that the guided prompt group (U = 75.5, p = .0495) was significantly higher than the generic prompt group on both programming self-efficacy and AI acceptance scales, indicating that structured guidance prompt design helps students build learning confidence and enhances their trust and willingness to use the AI teaching assistant. While no significant difference was found in learning attitudes, the guided group exhibited a more stable performance trend.

Table 4.

Comparison of generic and guided groups on three scales (Mann-Whitney U test).

Scale	Generic M±SD	Guided M±SD	Generic Mdn	Guided Mdn	*U*	*p*	Sig.
Self-Efficacy	31.69±10.68	36.88±5.81	30.0	35.0	75.5	.0495	Sig.
Attitude	30.31±9.07	28.81±6.21	32.0	29.5	158.0	.2648	ns
AI Acceptance	33.75±7.78	39.44±6.05	33.0	41.5	73.0	.0386	Sig.

Note. ns = not significant, Sig. = significant.

4.3 Dialogue Record Analysis

4.3.1 Scoring Criteria.

This study combined the ICAP Framework (Chi & Wylie, 2014) and Bloom's Taxonomy (Bloom, 1956) to construct an 'Inquiry Depth' analysis framework, classifying student questions into six levels according to their cognitive level and learning intention as the scoring basis for dialogue analysis. This framework sequentially reflects students' inquiry behaviors from non-learning-oriented interaction and surface-level questioning to behaviors demonstrating explanation, reflection, and knowledge transfer capabilities from low to high, effectively presenting students' level of cognitive engagement throughout the programming learning process.

Table 5.

Inquiry depth scoring rubric.

Level	Name	Description
IRR	Irrelevant Message	Pure greetings, thanks, links, system notifications, etc., unrelated to learning inquiry
D1	Surface/Answer-Seeking	Only asks for results, lacks context and attempts
D2	Clarification/Location	Can describe errors or context, but has not yet attempted
D3	Trial and Revision	Specifically describes modifications or experimental process, operational level
D4	Explanation and Hypothesis	Proposes causal speculation or principle explanation, and plans verification
D5	Reflection and Transfer	Summarizes experience, generalizes strategies, or transfers to new problems

4.3.2 Human-Machine Annotation Process and Results

This study collected a total of 2,904 dialogue messages between students and the AI teaching assistant, an LLM performed automatic annotation based on the inquiry depth criteria.To verify the reliability, researchers randomly selected approximately one-tenth of the sample (N = 290) for manual annotation by two human raters. The three-way annotation results (teacher, researcher, and LLM) were tested for consistency using Cohen's κ, with results showing κ values for each

comparison combination ranging between 0.65 and 0.73, reaching 'substantial agreement' level. This result indicates that LLM can stably simulate human judgment in inquiry depth annotation tasks, demonstrating reliability as an auxiliary analysis tool for educational research.

Table 6.

Three-way annotation results.

Comparison Group	κ Value	95% CI	Interpretation
Teacher vs Researcher	0.651	[0.580, 0.718]	Substantial
Teacher vs LLM	0.734	[0.670, 0.793]	Substantial
Researcher vs LLM	0.694	[0.673, 0.714]	Substantial

4.3.3 Interaction Level Analysis.

Using 1,308 "question-answer interaction units" (Q-A pairs) as the primary analysis unit, this study utilized the highest inquiry level per pair as the representative score. A total of 1,308 interaction units were analyzed, with inquiry depth distribution statistics compiled according to student groupings (H1, H2, L1, L2). Results showed that student groups using guided prompts (H2, L2) had higher proportions of high-level inquiry (D3-D5) than generic prompt groups (H1, L1), while their proportion of irrelevant messages (IRR) was significantly reduced, indicating that interaction content was more focused on learning tasks.

Table 7.

Interaction sample distribution.

Group	Total N	IRR	D1	D2	D3	D4	D5	High-Level (D3+D4)
H1	305	12.7%	29.1%	31.1%	13.9%	12.4%	0.7%	13.2%
H2	344	7.6%	34.1%	27.0%	13.7%	17.1%	0.5%	17.5%
L1	333	12.0%	27.9%	32.1%	15.1%	12.8%	0.0%	12.8%
L2	326	8.0%	21.1%	31.6%	25.1%	13.9%	0.4%	14.3%

4.3.4 Result Interpretation

Further comparison of inquiry depth distribution among different student groups yields the following trends:

(1) High-achievement students under guided prompt support (H2) showed significantly higher proportions of high-level inquiry compared to those not using guided prompts (H1), indicating that structured prompts help promote their reasoning and reflection.

(2) Low-achievement students under guided prompts (L2) also showed higher proportions of operational and explanatory level interactions, indicating that guided prompts help them progress from surface-level questioning to the experimentation and revision stage.

Overall, prompt design can produce positive effects on students of different achievement levels, with particularly significant effects on deepening inquiry for high-achievement students.

5. Conclusion and Discussion

5.1 Impact of AI Teaching Assistant on Learning Achievement

Research results show that LLM-based AI assistant students overall showed significant improvement in Python programming learning achievement, including program comprehension, error correction, and problem decomposition. This result indicates that the immediate feedback and sequential prompts provided by the AI teaching assistant can function as scaffolding in the learning process, helping students correct errors in real-time and deepen conceptual

understanding. Notably, low-achievement students showed greater improvement after AI teaching assistant intervention, indicating that AI teaching assistants have significant potential in remedial learning and reducing learning gaps, highlighting AI's potential for narrowing learning gaps, while high-achievement students utilized the too l for advanced applications.

5.2 Impact of AI Teaching Assistant on Students' Self-Efficacy, Learning Attitudes, and AI Acceptance

The results show that after introducing the AI teaching assistant, students' programming self-efficacy, learning attitudes, and AI acceptance all showed overall positive changes. The "evaluation-free" nature of the AI assistant fostered a **psychologically safe space**, reducing anxiety and encouraging iterative trial-and-error and self-correction.

5.3 Student-AI Teaching Assistant Interaction Inquiry Depth and Teacher Observations

In dialogue process analysis, this study revealed that inquiry depth is driven by prompt design; guided prompts effectively transitioned low-achievement students from answer-seeking to experimentation (D3) and promoted higher-level reflection (D4-D5) in high-achievement peers. Teacher observations also indicated that the AI teaching assistant can effectively share immediate guidance needs in class, allowing teachers to devote more energy to high-level instructional guidance and promoting the development of students' self-directed learning.

References

Ala-Mutka, K. M. (2005). A survey of automated assessment approaches for programming assignments. Computer Science Education, 15(2), 83–102. https://doi.org/10.1080/08993400500150747

Bandura, A. (1986). Social foundations of thought and action: A social cognitive theory. Prentice-Hall, Inc.

Bloom, B. S., Engelhart, M. D., Furst, E. J., Hill, W. H., & Krathwohl, D. R. (1956). *Taxonomy of educational objectives: The classification of educational goals. Handbook I: Cognitive domain*. David McKay Company.

Chi, M. T. H., & Wylie, R. (2014). The ICAP framework: Linking cognitive engagement to active learning outcomes. Educational Psychologist, 49(4), 219–243. https://doi.org/10.1080/00461520.2014.965823

Davis, F. D. (1989). Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. *MIS Quarterly*, *13*(3), 319–340. https://doi.org/10.2307/249008

Denny, P., Kumar, V., & Giacaman, N. (2023). Conversing with Copilot: Exploring prompt engineering for solving CS1 problems using natural language. In Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1 (pp. 1136–1142). Association for Computing Machinery. https://doi.org/10.1145/3545945.3569823

Guzdial, M. (2015). *Learner-centered design of computing education: Research on computing for everyone*. Morgan & Claypool Publishers. https://doi.org/10.2200/S00684ED1V01Y201511HCI033

Ihantola, P., Ahoniemi, T., Karavirta, V., & Seppälä, O. (2010). Review of recent systems for automatic assessment of programming assignments. In *Proceedings of the 10th Koli Calling International Conference on Computing Education Research* (pp. 86–93). https://doi.org/10.1145/1930464.1930480

Kaczmarczyk, L. C., Petrick, E. R., East, J. P., & Herman, G. L. (2010). Identifying student misconceptions of programming. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education (SIGCSE '10)*, 107-111. https://doi.org/10.1145/1734263.1734299

Kasneci, E., Sessler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., ... & Kasneci, G. (2023). ChatGPT for good? On opportunities and challenges of large language models for education. Learning and Individual Differences, 103, 102274. https://doi.org/10.1016/j.lindif.2023.102274

Keuning, H., Jeuring, J., & Heeren, B. (2018). A systematic literature review of automated feedback generation for programming exercises. ACM Transactions on Computing Education, 19(1), Article 3. https://doi.org/10.1145/3231711

Kölling, M., Quig, B., Patterson, A., & Rosenberg, J. (2003). The BlueJ system and its pedagogy. *Computer Science* Education, 13(4), 249–268. https://doi.org/10.1076/csed.13.4.249.17496

Lishinski, A. (2016). Cognitive, affective, and dispositional components of learning programming. In Proceedings of the 2016 ACM Conference on International Computing Education Research (pp. 261–262). Association for Computing Machinery. https://doi.org/10.1145/2960310.2960338

Padurean, V.-A., Denny, P., & Singla, A. (2025). BugSpotter: Automated generation of code debugging exercises. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE TS 2025)*. Association for Computing Machinery. https://doi.org/10.1145/3641554.3701974

Qian, Y., & Lehman, J. D. (2017). Students' misconceptions and other difficulties in introductory programming: A literature review. *ACM Transactions on Computing Education (TOCE)*, 18(1), 1-24. https://doi.org/10.1145/3077618, V., LaBelle, D., & Wiedenbeck, S. (2004). Self-efficacy and mental models in learning to program. In Proceedings of the 9th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (pp. 171–175). Association for Computing Machinery. https://doi.org/10.1145/1007996.1008042

Reiser, B. J. (2004). Scaffolding complex learning: The mechanisms of structuring and problematizing student work. The Journal of the Learning Sciences, 13(3), 273–304. https://doi.org/10.1207/s15327809jls1303_2

Ramalingam, V., & Wiedenbeck, S. (1998). Development and Validation of Scores on a Computer Programming Self-Efficacy Scale and Group Analyses of Novice Programmer Self-Efficacy. *Journal of Educational Computing Research*, *19*(4), 367-381. https://doi.org/10.2190/C670-Y3C8-LTJ1-CT3P (Original work published 1998)

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. Computer Science Education, 13(2), 137–172. https://doi.org/10.1076/csed.13.2.137.14200

Schraw, G., & Moshman, D. (1995). Metacognitive theories. *Educational Psychology Review*, 7(4), 351-371. https://doi.org/10.1007/BF02212307

Sweller, J. (2010). Element interactivity and intrinsic, extraneous, and germane cognitive load. Educational Psychology Review, 22(2), 123–138. https://doi.org/10.1007/s10648-010-9128-5

Tenenberg, J., & Murphy, L. (2005). Knowing what I know: An investigation of undergraduate knowledge and self-knowledge of data structures. *Computer Science Education*, *15*(4), 297–315. https://doi.org/10.1080/08993400500307677

Vygotsky, L. S. (1978). Mind in society: The development of higher psychological processes. Harvard University Press.

Wiebe, E. N., Williams, L., Yang, K., & Miller, C. (2003). Computer Science Attitude Survey (Report No. NCSU CSC TR-2003-1). Department of Computer Science, North Carolina State University.

White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J., & Schmidt, D. C. (2023). A prompt pattern catalog to enhance prompt engineering with ChatGPT. *arXiv preprint arXiv:2302.11382*. https://doi.org/10.48550/arXiv.2302.11382