Observing How Students Program with an LLM-powered Assistant: Quantifying Visual Expertise Through Eye-Tracking

Anonymous ACL submission

Abstract

The proliferation of language models is revolutionizing Human-AI Interaction, offering users a conversational interface to accomplish various tasks and access information. Understanding how these models affect the way students learn the skill of computer programming remains an unstudied area of research. This paper presents an experiment designed to investigate the interaction dynamics of university students with varying computer programming abilities when utilizing ChatGPT, as an AI-assistive tool to accomplish coding tasks. Eye-tracking technology was employed to capture gaze patterns and visual attention during their interactions with the language model. For this study, data was collected from 26 university students with a range of programming experience (from Sophomore to Ph.D.-level). More experienced programmers spent 3x more time focusing on the programming IDE over the ChatGPT UI, compared to their less experienced peers (as measured by fixations $p < 0.05$), while novice programmers fixated equally on both interfaces, but were 5.5x faster at completing the tasks with reduced levels of complex visual attention (as measured by saccades $p < 0.05$) indicating an over-reliance on LLM outputs. This work provides an avenue for the development of systems that can assess programmer's focus and attention as they problem solve.

1 Introduction

The advent and public availability of mainstream Large Language Models (LLMs), at either very low-cost or no-cost has trivialized their use. Popular LLMs such as ChatGPT are being prompted daily by active users; with an estimated uptake of 100 million monthly active users in January of 2023, just two months after its launch, making it the fastest-growing consumer application in history (Hu, 2023). The proliferation of LLMs has resulted in uses for both professional and personal miscellaneous tasks. Whether the LLM prompt is mundane or complex, the average person seems to be prioritizing its use as an alternative to web search (Ibrahim et al., 2023). With the rapid rise in LLM usage, an understudied area of research is the impact of LLMs in higher education and its use by students in the learning process (Zumwalt et al., 2014; Maldonado et al., 2023).

LLMs in higher education: The latest studies suggest that ChatGPT and similar models perform equally and sometimes better than university students in a diverse set of academic courses (Ibrahim et al., 2023). However, in the case where the student uses an LLM model for learning, it is unclear to what extent the output provided by such models are assimilated and understood by the user (Jeon and Lee, 2023). Moreover, LLMs are now capable of generating code (Acher et al., 2023), and this ability is expected to change the ways and techniques programming students learn and interact with programming tasks (Yilmaz and Yilmaz, 2023). Thus, the aim of this study is to understand how a student navigates the task of completing a programming exercise given the availability of LLM-powered tools like ChatGPT. We perform this assessment using eye-tracking technology to measure visual attention (Mansoor et al., 2023).

Visual expertise acquisition: Eye-tracking has been widely used to uncover the process of learning visually (Gegenfurtner and van Merriënboer, 2017; Davies, 2018), across different domains, including medicine (Zammarchi and Conversano, 2021), strategic and algorithmic thinking (Reingold and Sheridan, 2011), natural language processing (Salicchi et al., 2021; Barrett et al., 2016; Bolotova et al., 2020), and programming (Mansoor et al., 2023). This study lays the foundation for examining visual expertise acquisition in students utilizing AI aids to complete technical coding tasks, a growing domain of research within Human-AI Interaction. (Langner et al., 2023; MacKenzie, 2012).

2 Research Question

Our aim is to quantify the visual attention behavior of university students as they use an LLM-powered assistant (ChatGPT) to accomplish programming tasks. To this end, we explore how eye-tracking trends differ (if at all), conditioned on students' proficiency in programming.

2.1 Hypothesis

Our hypothesis is that increased programming proficiency (as measured by academic seniority) would lead to reduced time spent in solving programming tasks (as measured by decreased eye-tracking metric values).

2.2 Contributions

To the best of our knowledge, this is the first paper to quantify visual expertise acquisition of individuals as they complete programming tasks with the aid of an LLM-powered assistant.

3 Methods

The experiment was reviewed and approved by the institution's IRB committee. To support extensions and reproducibility of this work, the collected data is made publicly available[1].

3.1 Experimental Tasks

Students were required to solve a set of four programming tasks for the experiment. Tasks covered key syntactic and semantic constructs of programming, with each task specifically focused on: looping constructs, data structures, creative problem solving, as well as testing algorithms using sample input/output (see Appendix 1); these tasks were selected based on the Computational Thinking (CT) model defined in (Moon et al., 2020; Shute et al., 2017). The CT model is defined as the conceptual foundation needed to solve problems effectively and efficiently (Shute et al., 2017). The experiment followed a within-subjects design (i.e. each subject completes all four programming tasks), furthermore, subjects were required to complete all tasks in the same order and sequentially, without any time limit. This setup provided an equitable opportunity to all participants irrespective of programming experience, providing richer insights into learning behavior by eliminating temporal pressures (Moon et al., 2020).

[1] link to data to be made available on publication

3.2 Participant Recruitment

University students were recruited to participate in the study, with a focus on programming-related majors including Computer Science and Engineering. Participants were rewarded with a 15 USD Amazon gift card at the end of their engagement in the study.

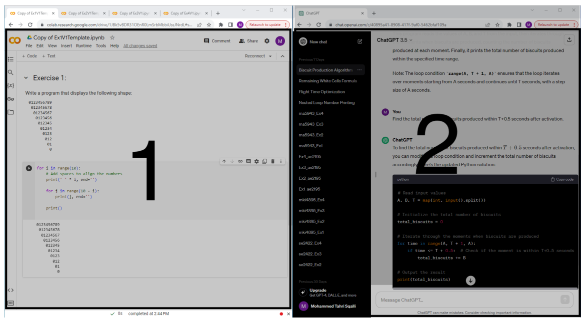

Figure 1: Pre-defined Areas of Interest (AOIs) for the ChatGPT experiment. AOI 1: Google Colab Integrated Development Environment (IDE). AOI 2: ChatGPT interface

3.3 Experimental Setup

The participants were asked to record their answers on Google Colab, an online Python Notebook IDE. The experiment screen was split into two halves containing the Google Colab IDE (on the left half of the screen) versus the ChatGPT interface (on the right half of the screen). These halves were designated as two areas of interests (AOIs) based on which the calculations of the eye-tracking variables were made (Figure 1). The experiment was designed to allow for capturing information that distinguishes between LLM-outputs and coding implementations (as opposed to an experimental setup with Github Copilot which is a more complex LLM-user interaction to study). Screen recordings as well as eye-tracking, mouse tracking, textual input, the language model's responses were recorded. A post-experiment semi-structured interview and questionnaire were conducted to capture demographic information (major, academic year) as well as participants' observations and reflections of the experiment.

The experiment employed a screen-based eye tracker, SmartEye AI-X, with a frequency of 60 Hz, with iMotions software version 9.3 to record and capture participants' gaze patterns. Prior to the start of each experiment, the eye tracking system was calibrated with the participants' eye. The threshold for the calibration accuracy was $\pm$ 5Hz.

3.4 Eye-tracking Features

We measure three key eye-tracking features: fixation counts, dwell-time, and saccade counts (Hutton, 2019). *Fixation counts* refer to the number of times a person's eyes stops or fixates on the AOI, which indicates what is being focused on. *Dwell-time* measures the amount of time a person spends gazing and/or fixating on the AOI, this is reported as the percentage of time a user spends on a given AOI during a programming task, and aids in assessing attention-levels. *Saccade counts* refers to the number of times a person displays a rapid and involuntary eye movement that occurs between fixations, which measures how often individuals shift their attention from one area to another (within an AOI).

3.5 Statistical Tests

To evaluate whether the participant groups and AOIs have a statistically significant difference with the three recorded eye-tracking features, we perform a two-way Analysis of Variance (ANOVA) (Lowry, 2014). To further evaluate any observed statistical significance, we perform the Tukey's Honestly Significant Difference (HSD) post-hoc test to identify which specific groups had statistically significant differences in group means while accounting for multiple comparison tests (Type I error) (Heckert et al., 2002).

4 Results

We collected data from 26 university students: 5 were sophomore, 5 were junior, 10 were senior, and 6 were in post-graduate programs (including Ph.D). To perform comparative analysis, participants were split into two categories: 'novice' category (sophomore and junior students), and 'experienced' category (seniors and post-graduate students). A total of 104 programming tasks were recorded (26 participants x 4 tasks), with 91 tasks solved correctly out of the 104, with the majority of errors submitted by novice students (12 out of the 13 errors). On average, participants completed a task between 5 minutes and 48 seconds to 7 minutes and 7 seconds, with experienced programmers spending on average 5.5x more time on a given task compared to novice programmers (Table 1). We observed that hallucinations where generated 8 times by ChatGPT: 3 for experienced programmers and 8 for novice programmers; only experienced programmers noticed the hallucinations.

Table 1: Time to complete programming tasks.

Task	Novice Mean (SD)	Experienced Mean (SD))	Overall Mean (SD)
1	04:22 (01:02)	07:14 (01:49)	05:48 (02:56)
2	02:46 (05:55)	13:54 (11:08)	08:20 (15:45)
3	02:26 (04:18)	09:22 (10:12)	07:07 (13:43)
4	03:28 (01:09)	09:38 (03:33)	06:03 (05:02)
Errors	12	1	13
Hallu-cination	5	3	8

Table 2: Descriptive statistics for fixation count, dwell time (%), and saccades count across skill groups (novice vs. experienced) and AOIs (ChatGPTUI vs. Colab IDE).

Group	Fixation (Count) Mean (SD)	Dwell Time (%) Mean (SD)	Saccade (Count) Mean (SD)
Novice			
ChatGPT UI	156 (176)	41% (25%)	349 (527)
Colab IDE	144 (134)	54% (27%)	229 (233)
Experienced			
ChatGPT UI	249 (216)	23% (12%)	279 (274)
Colab IDE	686 (510)	72% (13%)	860 (609)
ANOVA p-val	< 0.05	9.3e-2	< 0.05
Tukey p-adjusted			
AOI	6.5e-3	-	1.8e-2
Skill	8.2e-4	-	2.1e-2
Skill x AOI	1.8e-2	-	9.0e-3

To compare the difference in eye-tracking trends between the two proficiency groups (novice vs. experienced) and their AOIs (ChatGPT UI vs. Google Colab IDE), we performed a two-way ANOVA for the three eye-tracking features, where the AOI and participant proficiency group were the independent variables and a given eye-tracking feature was the dependent variable (Table 2). The independent variables were found to have a statistically significant effect for two eye-tracking features: the fixation counts ($p < .05$) and the saccade counts ($p < .05$).

Next, we performed the Tukey's (HSD) post-hoc test to identify which specific groups had statistically significant differences in group means while accounting for multiple comparison tests. We found that the p-adjusted values of multiple comparisons were statistically significant (p-adj < 0.05) for both the fixation counts (p-adj = 6.5e-3 and 8.1e-3) and the saccade counts (p-adj = 9.0e-3 and 2.1e-3) across AOIs and participant proficiency (skill) groups, respectively. There was also a statistically significant difference in the eye-tracking aforementioned measures across sub-groups (intersection of AOIs and skills).

5 Discussion

Experienced programmers focus on IDE over LLM-assistance: Our results indicate that eye-tracking behavior differs according to programming proficiency, based on the observed statistically significant differences in fixation counts and saccade counts, and is in line with previous findings (Jessup et al., 2021). Our results also indicate that more experienced participants, on average, fixated relatively more on the Colab IDE over the ChatGPT UI (686 vs 249 counts) whereas their less experienced peers fixated an almost equivalent amount across the two AOIs, Colab IDE and ChatGPT UI, respectively (144 vs. 156 counts).

Experienced programmers were more diligent: A similar pattern emerges when considering dwell time; overall, the more experienced programmers spent approximately three times more attending to the Colab IDE (72%) over the ChatGPT UI (23%). This indicates that more experienced programmers allocate relatively more effort to programmatically solving the task within the IDE. We hypothesized that experienced programmers would have reduced eye-tracking measures and would spend less time solving tasks, but observed the opposite behavior; on average, experienced programmers had higher fixations and saccades, and spent 5.5x *more* time solving the task than novices, opposite to what we had hypothesized or has been suggested in the literature (Peitek et al., 2022). This implies that experienced programmers were more diligent in their approach to solving the task, by fixating more, spending more time overall on the task, and focusing their effort on the IDE rather than the outputs of the LLM-assistant. An underlying motivation for this behavior was captured in relayed by an experienced programmer: "ChatGPT is useless in programming tasks. I feel like you will have to care for it, and walk it through the correct answer! I used it only when I had to look for the syntax.". This sentiment translated behaviorally (i.e. dwell time) whereby visually expert programmers preferred to rely more on their programming skills rather than the LLM output.

Novices seemed to be over-reliant on LLM outputs: Additionally, novice programmers, on average, accorded a significantly lower number of saccade counts within the Colab IDE (279 counts) compared to more experienced programmers (860 counts), that is, they shifted their attention less often from one area to another (within the AOI) - see figure 5. This reduced saccade count implies that novice programmers were not as thoroughly investigating the diversity of information that was presented to them while they programmed their solution, furthermore, they may have been more reliant on copying the outputs of the LLM to solve the task since they completed tasks on average 2 to 11 minutes faster (5.5x) than more experienced programmers, and with the majority of solutions correct.

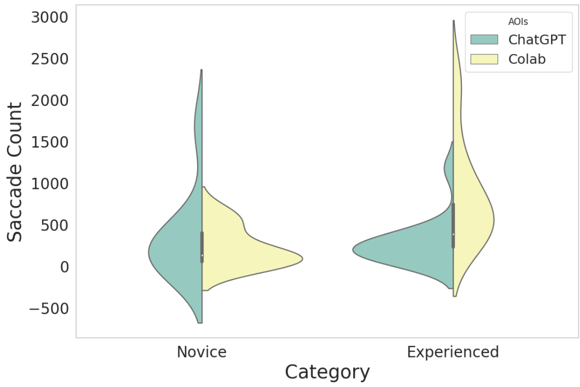

Figure 2: Violin plot of saccade counts for novice and experienced participants across the two AOIs. On average, experienced programmers had higher saccades while using their IDE implying a richer interaction.

Hallucinations yielded cognitive burdens: There were 3 cases where LLM generated answers contained hallucinations that were identified by participants, all from the experienced group. The identification of hallucinations resulted in an increased amount of time spent solving the task; approximately 1 standard deviation above the mean completion time of the respective task (24 mins 5 secs [task 2], 20 mins 50 secs [task 3], and 14 mins 34 secs [task 3]). The increased time spent solving the problem resulted in increased saccade counts (527 on average) indicating an increased cognitive load. In the interview with a participant who identified a hallucination, they shared: "I identified a mis-representation in the sample input/output that I asked ChatGPT to generate for exercise 3. I noticed that something was odd in the calculation provided. This is when I had to re-check everything that it gave as an output. I had to look for the step that went wrong. I realized that although the logic and the code were correct, ChatGPT did not calculate the output correctly". The observed increase in time and saccade counts is supported by prior work observing changes in eye-tracking behavior while debugging code (Melo et al., 2017).

References

Mathieu Acher, Jose Galindo Duarte, and Jean-Marc Jezequel. 2023. On programming variability with large language model-based assistant. In *Proceedings of the 27th ACM International Systems and Software Product Line Conference-Volume A*, pages 8–14.

Maria Barrett, Joachim Bingel, Frank Keller, and Anders Søgaard. 2016. Weakly supervised part-of-speech tagging using eye-tracking data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 579–584.

Valeria Bolotova, Vladislav Blinov, Yukun Zheng, W Bruce Croft, Falk Scholer, and Mark Sanderson. 2020. Do people and neural nets pay attention to the same words: studying eye-tracking data for non-factoid qa evaluation. In *Proceedings of the 29th ACM international conference on information & knowledge management*, pages 85–94.

Alan Davies. 2018. *Examining Expertise Through Eye Movements: A Study of Clinicians Interpreting Electrocardiograms*. Ph.D. thesis, The University of Manchester.

Andreas Gegenfurtner and Jeroen J. G. van Merriënboer. 2017. Methodologies for studying visual expertise. *Frontline Learning Research*, 5(3):1–13.

N Alan Heckert, James J Filliben, C M Croarkin, B Hembree, William F Guthrie, P Tobias, and J Prinz. 2002. Handbook 151: Nist/sematech e-handbook of statistical methods.

Krystal Hu. 2023. ChatGPT sets record for fastest-growing user base - analyst note — reuters.com. [Accessed 10-09-2023].

SB Hutton. 2019. Eye tracking methodology. *Eye movement research: An introduction to its scientific foundations and applications*, pages 277–308.

Hazem Ibrahim, Fengyuan Liu, Rohail Asim, Balaraju Battu, Sidahmed Benabderrahmane, Bashar Alhafni, Wifag Adnan, Tuka Alhanai, Bedoor AlShebli, Riyadh Baghdadi, Jocelyn J. Bélanger, Elena Beretta, Kemal Celik, Moumena Chaqfeh, Mohammed F. Daqaq, Zaynab El Bernoussi, Daryl Fougnie, Borja Garcia de Soto, Alberto Gandolfi, Andras Gyorgy, Nizar Habash, J. Andrew Harris, Aaron Kaufman, Lefteris Kirousis, Korhan Kocak, Kangsan Lee, Seungah S. Lee, Samreen Malik, Michail Maniatakos, David Melcher, Azzam Mourad, Minsu Park, Mahmoud Rasras, Alicja Reuben, Dania Zantout, Nancy W. Gleason, Kinga Makovi, Talal Rahwan, and Yasir Zaki. 2023. Perception, performance, and detectability of conversational artificial intelligence across 32 university courses. *Scientific Reports*, 13(1).

Jaeho Jeon and Seongyong Lee. 2023. Large language models in education: A focus on the complementary relationship between human teachers and chatgpt. *Education and Information Technologies*, pages 1–20.

Sarah Jessup, Sasha M Willis, Gene Alarcon, and Michael Lee. 2021. Using eye-tracking data to compare differences in code comprehension and code perceptions between expert and novice programmers.

Moritz Langner, Peyman Toreini, and Alexander Maedche. 2023. Leveraging eye tracking technology for a situation-aware writing assistant. In *Proceedings of the 2023 Symposium on Eye Tracking Research and Applications*, pages 1–2.

Richard Lowry. 2014. Concepts and applications of inferential statistics.

I MacKenzie. 2012. Human-computer interaction: An empirical research perspective.

Liam Richards Maldonado, Azza Abouzied, and Nancy W. Gleason. 2023. ReaderQuizzer: Augmenting research papers with just-in-time learning questions to facilitate deeper understanding. In *Computer Supported Cooperative Work and Social Computing*. ACM.

Niloofar Mansoor, Cole S Peterson, Michael D Dodd, and Bonita Sharif. 2023. Assessing the effect of programming language and task type on eye movements of computer science students. *ACM Transactions on Computing Education*.

Jean Melo, Fabricio Batista Narcizo, Dan Witzner Hansen, Claus Brabrand, and Andrzej Wasowski. 2017. Variability through the eyes of the programmer. In *2017 IEEE/ACM 25th International Conference on Program Comprehension (ICPC)*, pages 34–44. IEEE.

Jewoong Moon, Jaewoo Do, Daeyeoul Lee, and Gi Woong Choi. 2020. A conceptual framework for teaching computational thinking in personalized oers. *Smart Learning Environments*, 7(1).

Norman Peitek, Annabelle Bergum, Maurice Rekrut, Jonas Mucke, Matthias Nadig, Chris Parnin, Janet Siegmund, and Sven Apel. 2022. Correlates of programmer efficacy and their link to experience: A combined eeg and eye-tracking study. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 120–131.

Eyal M. Reingold and Heather Sheridan. 2011. *Eye movements and visual expertise in chess and medicine*. Oxford University Press.

Lavinia Salicchi, Alessandro Lenci, and Emmanuele Chersoni. 2021. Looking for a role for word embeddings in eye-tracking features prediction: does semantic similarity help? In *Proceedings of the 14th International Conference on Computational Semantics (IWCS)*, pages 87–92.

Valerie J. Shute, Chen Sun, and Jodi Asbell-Clarke. 2017. Demystifying computational thinking. *Educational Research Review*, 22:142–158.

Ramazan Yilmaz and Fatma Gizem Karaoglan Yilmaz. 2023. Augmented intelligence in programming learning: Examining student views on the use of chatgpt for programming learning. *Computers in Human Behavior: Artificial Humans*, 1(2):100005.

Gianpaolo Zammarchi and Claudio Conversano. 2021. Application of eye tracking technology in medicine: A bibliometric analysis. *Vision*, 5(4):56.

Ann C. Zumwalt, Arjun Iyer, Abenet Ghebremichael, Bruno S. Frustace, and Sean Flannery. 2014. Gaze patterns of gross anatomy students change with classroom learning. *Anatomical Sciences Education*, 8(3):230–241.

Ethics Statement

The institutional review board approval for this study was granted by the ethics board of [Anonumized] under the IRB Protocol Reference Number [Anonmyzied] prior to the commencement of the study.

Appendix 1: Programming tasks

In order to solve the four programming tasks, students had to be skilled at the below concepts:

1. *Looping constructs*: The problem requires the student's understanding of 'for' loops and 'nested for' loops

2. *Data structures manipulations*: The problem requires the student's familiarity with data structures in order to solve a searching and sorting problem.

3. *Creative problem solving using algorithms*: This problem contains a form of ChatGPT hallucination, where the student is required to verify the correctness of the reasoning provided by ChatGPT.

4. *Testing sample input/output*. This problem requires the students to analyze a provided sample input and output for a programming exercise to have a complete understanding of the algorithmic problem.

Below are the four programming exercises:

1. Write a program that displays the following shape:

```
0123456789
 012345678
  01234567
   0123456
    012345
     01234
      0123
       012
        01
         0
```

2. There are three airports A, B and C, and flights between each pair of airports in both directions. A one-way flight between airports A and B takes P hours, a one-way flight between airports B and C takes Q hours, and a one-way flight between airports C and A takes R hours. Consider a route where we start at one of the airports, fly to another airport and then fly to the other airport. What is the minimum possible sum of the flight times? Make sure to test your imput and output for the correct answer.

3. There are H rows and W columns of white square cells. You will choose h of the rows and w of the columns, and paint all of the cells contained in those rows or columns. How many white cells will remain? It can be proved that this count does not depend on what rows and columns are chosen. Constraints:

$$1 \leq H, W \leq 20$$

$$1 \leq h \leq H$$

$$1 \leq w \leq W$$

4. A biscuit making machine produces B biscuits at the following moments: A seconds, $2A$ seconds, $3A$ seconds, and each subsequent multiple of A seconds after activation.

   Find the total number of biscuits produced within $T + 0.5$ seconds after activation.

   Constraints: All values in input are integers. $1 \leq A, B, T \leq 20$

   Input:
   Input is given from Standard Input in the following format: $A\ B\ T$

   Output:
   Print the total number of biscuits produced within $T + 0.5$ seconds after activation.

Appendix 2: Questionnaire

article

Section 1 of 5

Acquiring Programming Skills in the Era of ChatGPT - An Eye Tracking and Attention Study

Welcome to our post-study questionnaire! We're eager to learn more about your programming journey, both before and after your interaction with ChatGPT. This survey aims to delve into your programming awareness and skills, examining how your confidence and abilities have evolved with and without the assistance of programming tools. Your valuable insights will help us better understand the impact of ChatGPT on students' programming trends. Thank you for participating in this important study!

6 Demographics

1. **Email:** ____________________________
2. **netID:** ____________________________
3. **Gender:** ___________ Female ___________ Male
4. **Age:** ___________
5. **Which University year are you at?** ___________ First ___________ Second ___________ Third ___________ Fourth ___________ Fifth
6. **What is your expected graduation year?** ___________ 2024 ___________ 2025 ___________ 2026 ___________ 2027 ___________ Other: ___________

Section 2 of 5

Section 2: Task Difficulty

This section inquires about the difficulty of the coding process in general as well as throughout the experiment.

1. Please rate the difficulty of the on-screen programming tasks you just participated in?
___________ Difficult 1 2 3 4 5 Easy
2. How would you rate the difficulty of the programming task WITHOUT the use of assistant coding forums and websites OTHER THAN ChatGPT? (like not using StackOverflow, Coding forums, or just Google Searches)
___________ Difficult 1 2 3 4 5 Easy
3. How would you rate the difficulty of the programming task WITH the use of assistant coding tools OTHER THAN ChatGPT? (like using StackOverflow, Coding forums, or just Google Searches)
___________ Difficult 1 2 3 4 5 Easy
4. Compared to your usual way of coding, how would you rate the difficulty of the programming task WITH the use of ChatGPT?
___________ Difficult 1 2 3 4 5 Easy
5. Generally, how confident do you typically feel when you are programming? i.e., Before running your code for the first time, do you feel confident enough that you implemented the correct algorithm?
___________ Not confident 1 2 3 4 5 Very confident

Section 3 of 5

Section 3: Programming Training Background

This section seeks to know more about your background related to programming, i.e.: For how long and how did you first learn about programming.

1. Are you a student training to acquire the skill of programming?
___________ Yes ___________ No
2. Which category best describes your usual job/position?
___________ Computer Science Major ___________ Engineering Major (Programming Courses is part of the Curriculum) ___________ Non-Engineering Major (Programming Courses is an Elective) ___________ Self-Taught ___________ Other: ___________
3. How long have you been familiar with programming and its concepts?

___________ < 6 months ___________ 6 months - 1 year ___________ 1 - 2 years ___________ 2 - 3 years ___________ 4 years ___________ + 4 years (Before University)

4. How often do you program (part of class or for fun)?
___________ Never ___________ Daily ___________ Weekly ___________ Monthly ___________ Other: ___________

5. How many credit hours of programming-related courses have you received as part of your undergraduate degree?
___________ None ___________ 1 - 5 ___________ 11 - 20 ___________ 21-30 ___________ Other: ___________

6. What format did your programming training take?
___________ Formal lecture/seminar ___________ Workshop/Lab ___________ Online activities/courses ___________ Taught by friends outside class ___________ By building a side-project ___________ None of the above ___________ Other: ___________

Section 4 of 5

Section 4: Personal Coding/Programming System

This section tries to find whether you use or follow a system when programming.

1. Do you use a system (trick, hack, or tool) when programming?
___________ Yes ___________ No ___________ Don't know

2. Do you refer to a system when programming? Briefly name the system, methodology, or framework, and give a brief description of how it works.
____

3. Which statement best describes the origin of your system?
___________ I was taught this system as part of a class on programming ___________ I learned this system from other friends ___________ I developed this system myself ___________ Other: ___________

4. Has the system you have used changed over time?
___________ Yes ___________ No ___________ Don't know

5. How did your system change?
____

6. Why did your system change?
____

7. Which aspect of programming do you pay most attention to when writing code?
___________ The algorithm ___________ If it runs, it is already not that bad... ___________ The programming methodology ___________ Code Optimization ___________ Applying the data structures I know to solve the problem ___________ The clarity and quality of Input/Output ___________ How much time I spent on one program/piece of code ___________ Whether someone will judge my programming skills when reading my code ___________ That it does what it is supposed to do, and that is it ___________ Other: ___________

8. How often do you apply coding best practices (commenting the code, using naming conventions, proper indentation, testing all the side cases) when programming?
___________ Never ___________ Occasionally ___________ Sometimes ___________ Often ___________ Always ___________ Don't know ___________ Other: ___________

Section 5 of 5

Section 5: Personal Opinion about AI-based Coding Assistants

This section inquires about your opinion about the usefulness and preferences for AI-based coding assistants.

1. How useful do you find automated programming outputs? (Like the ones generated by ChatGPT)

___________ Useless 1 2 3 4 5 Useful

2. How accurate do you think automated coding tools are? (Like GitHub Copilot or ChatGPT, or integrated IDE coding and debugging assistants)
___________ Inaccurate 1 2 3 4 5 Accurate

3. Do you enjoy the process of programming?
___________ Yes ___________ No
___________ Maybe ___________ I don't know
___________ Other: ___________