ISSN: 2473-4901
v10 n6172

Student Perceptions of Learning through Original and AI-Generated Python Programs from a Software Quality Perspective

Mark Frydenberg
mfrydenberg@bentley.edu

Anqi Xu
axu@bentley.edu

Jennifer Xu
jxu@bentley.edu

Computer Information Systems Department
Bentley University, Waltham, MA

Abstract

This study explores student perceptions of learning to code by evaluating AI-generated Python code. In an experimental exercise given to students in an introductory Python course at a business university, students wrote their own solutions to a Python program and then compared their solutions with AI-generated code. They evaluated both solutions using a software quality assessment framework, focusing on the correctness, efficiency, understandability, consistency, and maintainability, which provided a guide to evaluating code beyond simply correctness of the solution. Research examines how students perceive and utilize generative AI, considering their motivations, outcomes, and experiences. Findings suggest that while students see significant potential in using AI tools to enhance their coding process and appreciate the efficiency and compactness of the AI-generated code, they often prefer their own solutions due to familiarity and features used. This research aims to inform future studies on student application of AI tools in learning to code and provides educators with a model for evaluating AI's impact on student learning.

Keywords: Python, Coding, Generative AI, Software Quality Assessment

1. INTRODUCTION

Learning to code is an important skill for future business professionals (Learn Computer Science. Change the World., 2024; Shein, 2014) and introductory students often find it challenging to master a programming language as well as develop the critical thinking skills necessary to assess the quality of their code. The arrival of generative artificial intelligence (AI) introduced large language models such as ChatGPT, Microsoft Copilot, and Google Gemini, which are capable of writing code in multiple programming languages.

This paper explores student perceptions of learning from AI-generated Python programs. The authors describe a study where students enrolled in an introductory Python course at a business-focused university first solved a programming exercise on their own, and then compared their solutions with code generated by AI tools. Students evaluated both solutions using a software quality assessment framework (Boehm et al., 1976) which focused on correctness, efficiency, understandability, consistency, and maintainability of the code. By examining how students perceive and utilize generative AI, focusing on their motivations, outcomes, and experiences, this research can inform future studies on the use of AI tools by students learning to code and provide educators with a model for evaluating its impact on student learning.

As students continue to use AI platforms as learning support tools, these research questions arise:

RQ1: How do students perceive learning from code they write themselves compared to code generated by AI?

RQ2. How can characteristics of software quality provide a framework for students to evaluate their own code and that generated by AI tools?

RQ3: What are students' overall opinions of using AI as a learning tool?

RQ4: What factors impact those perceptions?

2. LITERATURE REVIEW: STUDENT PERCEPTIONS ON THE USE OF GENERATIVE AI IN COMPUTING EDUCATION

Since ChatGPT and other generative AI models first emerged in November 2022, educators have been exploring ways to integrate them as learning tools in the computing classroom (Denny et al., 2024; Ma et al., 2024; Phung et al., 2023). Large language models "are now capable of producing code automatically and have demonstrated impressive performance on problems that are typical in introductory programming courses (Denny et al., 2024, p. 296)." For students learning to code, using AI can be a temptation (to get the correct answers on demand) as well as a tool to better understand their own solutions.

Recent studies on student perceptions of AI in coding courses have found that students are concerned about the validity and accuracy of results produced (Chan & Hu, 2023; Zastudil et al., 2023; Zhang et al., 2024), and on becoming dependent on those results to succeed. "One of the main issues is over-reliance on AI, which may hinder people's growth, skills, and intellectual development over time (Aruleba et al., 2024, p. 11)."

Although concerns about academic integrity, plagiarism (Chan & Hu, 2023; Tala et al., 2024), and data privacy have been increasing due to the personalized and immediate support that generative AI platforms provide, using AI tools can also promote creativity and assist in brainstorming new ideas (Ma et al., 2024; Tala et al., 2024), and knowing how to use AI properly can help with employability (Feldman & Anderson, 2024).

In a study of graduate and undergraduate economics students, Tala et al. (2024) explored perceptions of generative AI tools and found that "students with more advanced digital skills are more inclined to use AI for content generation. (Tala et al., 2024, p. 83)."

Instructors are finding new ways to incorporate generative AI tools into their classes (Choudhuri et al., 2024) and those which are most successful have students use AI's output as the basis for further problem solving.

One study (Ma et al., 2024) explored how beginning students perceived using ChatGPT to learn how to code in Python. They used ChatGPT as a programming partner, asking for help with concepts, code verification, and debugging and optimizing their code. Students found that the explanations from ChatGPT were helpful in explaining concepts and debugging code, but consistent with other results, were concerned about over-reliance on AI tools.

In a study regarding how students in an introductory Java class perceive feedback generated by AI platforms (Zhang et al., 2024), students received AI-generated feedback about their code. One version included the code in the prompt, the other did not. Students evaluated the feedback provided to determine whether AI having access to their solutions impacted the quality of the feedback provided. They found that when AI analyzed their code, the feedback was more useful.

Many students will provide the description of a problem as it appears in their assignment or textbook, directly to generative AI platforms with the hope of obtaining code that solves the problem, but this is not always sufficient to obtain results that align with what is taught in the classroom. AI tools often generate code using programming constructs that students may not have learned yet. Recognizing that "the ability to engineer effective prompts is now an essential skill for computing students (Denny et al., 2024, p. 297)," they introduce "Prompt Problems," an exercise where students solve programming problems by formulating natural language prompts which guides AI platforms to generate code to solve the problem.

The following perceptions emerged from our literature review on the use of generative AI tools in computing education:

Generative AI tools can:

- give useful, personalized feedback when reviewing student's code
- provide helpful feedback when debugging code
- be helpful when checking homework
- confuse students by providing solutions that do not align with concepts taught in class

Students are most concerned about:

- knowing or being able to determine if solutions from generative AI are complete and accurate
- relying on generative AI too much
- academic integrity

3. METHODOLOGY

This study involved 81 participants from six sections of an introduction to programming course at a New England business-focused university. Of the 81 participants, 54 completed the survey at the end of the study. The course materials, assignments, and exams were common among all sections, which were taught by four different instructors during the Fall 2023 semester. Students voluntarily participated and received extra credit points toward their course grade for their participation.

The instructors provided a programming problem to students to complete (see Appendix A) that was modeled after a programming assignment they completed earlier in the semester. The problem was designed in a way that the solution could either be implemented using several basic decision statements (various forms of if/else and if/elif/else) or using more efficient data structures (e.g., lists and dictionaries) to manage the data and logic of the program.

Students were asked to evaluate their own code and the AI-generated code based on five software quality elements (Boehm et al., 1976), and then make an overall assessment of their approach versus AI's approach to solving the problem.

Software Quality Assessment

Software quality can be assessed from many different perspectives using various metrics. In a seminal paper, Boehm et al. (1976) identify eleven quality indicators of computer programs, including understandability, completeness, conciseness, portability, consistency, maintainability, testability, usability, reliability, structuredness, and efficiency. The quality of software is directly affected by the quality of these individual program components.

As software has been increasingly complex, the number of quality indicators has also grown significantly. Based on an extensive literature review, a more recent study compiles an inventory of 48 software quality metrics, which are grouped into six dimensions: functionality, reliability, usability, efficiency, maintainability, and portability (Miguel et al., 2014).

Since the project in our course involves students creating one Python program whose solution can be implemented in about 100 lines, we have selected the following five quality indicators for student reflection and analysis:

<u>Correctness/Completeness:</u> Does the program provide all the correct output given the input?

<u>Efficiency:</u> Are the data about the application (e.g., pricing information) maintained using efficient data structures? An efficient data structure can reduce the use of control structures (e.g., loop and if statements).

<u>Understandability/ Conciseness:</u> Is the program easy to read and understand? Is it long and overly complicated? Are comments included and helpful? Is the program so concise that it is hard to understand?

<u>Consistency:</u> Is the program consistent in its use of naming variables, indentation, and formatting?

<u>Maintainability/Structuredness:</u> A program is maintainable if it is modular, does not duplicate steps, and is written in such a way that if business circumstances change, updates to the code to reflect those changes are minimal. For example, does the solution break the problem down into smaller modules or functions? If a business decides to change prices of their

products, would that require a change to many lines of code?

The authors felt that using all 15 metrics would be overwhelming to students, so we selected eight and consolidated them to five that were particularly relevant to new coders when evaluating short Python programs. Metrics such as device independence, legibility, and augmentability as described by Boehm et al. (1976) are less applicable to this assignment.

We focused on important skills that novice programmers need to develop, such as how to evaluate whether a solution is correct. The course also tries to teach students to write maintainable code; even in smaller, simpler programs, code must be understandable and concise, so it is easier to modify and debug. Consistency is important so that students learn to write code that is readable, with meaningful variable names so it is self-documenting. Teaching students to write code that is modular and well-structured from the start encourages good programming habits that will be useful later in their studies as their programs become more complex.

These qualities also best align with the course goals and objectives, which include defining algorithmic solutions and designing modular programs to implement those solutions, identifying test cases to test and debug code to ensure it runs properly, and efficiently representing data values using appropriate data structures.

In addition to their written programs and a report evaluating the software quality of their and AI's solutions, students also completed a short survey (see Appendix B) which asked about their experiences using generative AI tools as a partner in learning to code, and the perceived usefulness of the feedback that they received while interacting with generative AI tools.

Empirical Analysis

To address RQ3 regarding factors influencing students' perceptions of AI tools, this study employed an empirical analysis using linear regression. Specifically, analysis focused on three key questions in the survey:

- "I found the AI-generated solutions to be clear, concise, and relevant to the assignment."
- "I trust the solutions AI-generated to be correct and accurate."
- "Reviewing code generated by AI tools increased my confidence in writing code myself."

These questions served as dependent variables in our analysis, measured by five-point Likert scales ranging from 1 ("Strongly Disagree") to 5 ("Strongly Agree").

The regression analysis includes two independent variables: the aggregate scores of students' responses to the five quality dimensions for AI-generated code and for their own code, respectively. Specifically,

$$AIQuality = \sum_{i=1}^{5} QualityForAI_i \qquad (1)$$

$$StudentCodeQuality = \sum_{i=1}^{5} QualityForStudent_i \qquad (2)$$

where $QualityForAI_i$ represents students' responses to the five quality dimensions for AI-generated code including Correctness / Completeness, Efficiency, Understandability / Conciseness, Consistency, and Maintainability / Structuredness.

Similarly, $QualityForStudent_i$ are students' answers to the five quality dimensions for their own generated code. The analysis also includes several control variables:

- $StudentPreference$. This variable represents the student's preference for using AI-generated code or their own code.
- $StudentMajor$. We categorize students' majors as either technology-related or non-technology-related majors.
- $Quiz1$. Students did two quizzes during the semester. This variable records the score of the first quiz.
- $Quiz2$. Students' score of the second quiz.
- $Gender$. Student's gender.
- $Section$. Students participating in the survey were enrolled in six class sections taught by four different instructors. We include this variable to exclude confounding effects caused by educational differences.

The regression model is provided in Equation (3).

$$\begin{aligned}&StudentPerception\\&= \alpha_0 + \alpha_1 AIQuality\\&+ \alpha_2 StudentCodeQuality\\&+ \alpha_3 StudentPreference\\&+ \alpha_4 StudentMajor + \alpha_5 Quiz1 + \alpha_6 Quiz2\\&+ \alpha_7 Gender + \alpha_8 Section + \varepsilon\end{aligned} \quad (3)$$

4. RESULTS

Survey Findings

Figures 1 and 2 show responses to survey questions asking student opinions on their experience of using AI tools and the impact of generative AI tools on their learning. While responses were measured by five-point Likert scales ranging from 1 ("Strongly Disagree") to 5 ("Strongly Agree"), in Figures 1 and 2, Strongly Agree and Agree values were combined, as were Disagree and Strongly Disagree, to simplify presenting the results using a scale of Disagree/Neutral/Agree. Students mostly agreed that using generative AI tools was enjoyable, and that they found the results to be clear, concise, and relevant. Concomitantly, most students disagreed that they could trust the AI-generated solutions. This could be because of the complexity of the program they were writing.

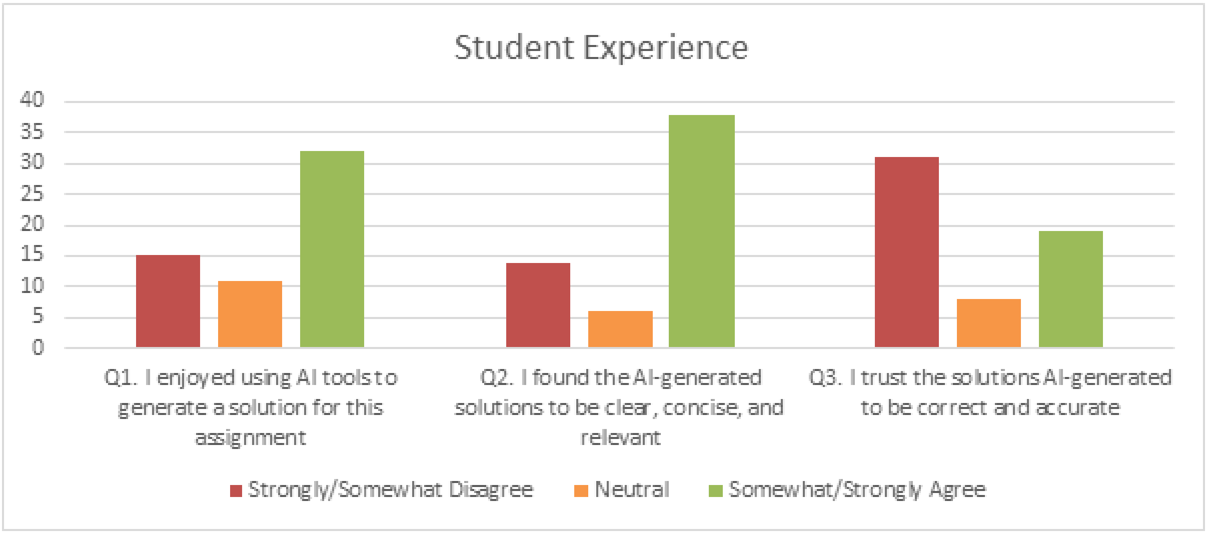

Figure 1. Student Experience with AI

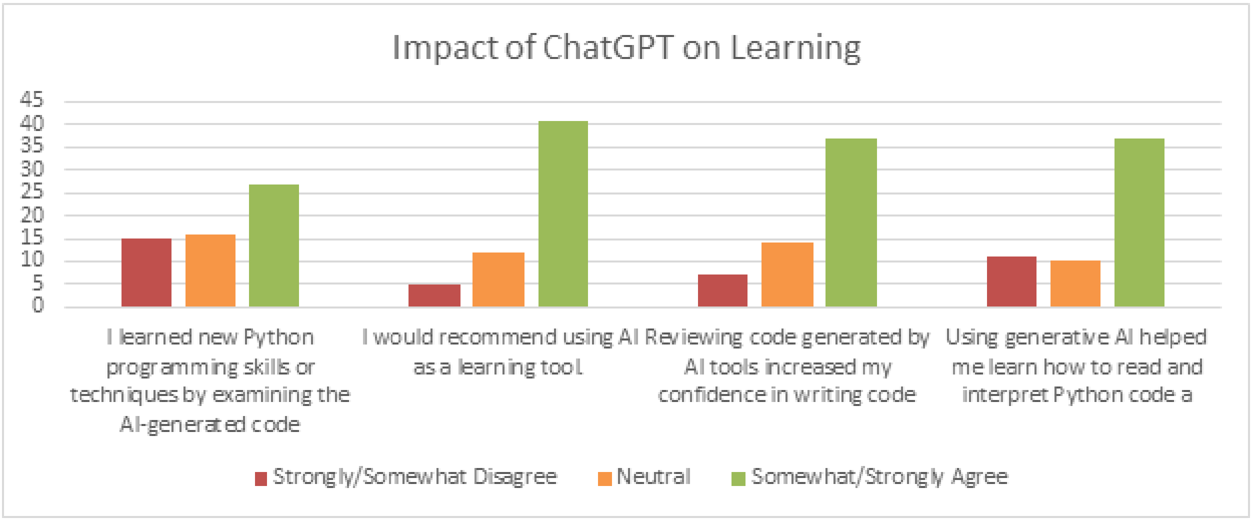

Figure 2. Impact of Generative AI on Learning

As for impact of generative AI on learning, students generally agreed that they learned new Python skills or techniques, that they recommend using AI as a learning tool, that reviewing AI-generated code increased confidence in writing their own code, and in learning how to read and interpret Python code.

The most popular generative AI tool that students used was ChatGPT (85%), followed by Google Gemini/Bard (6%). Students used Grok (1%), Microsoft Copilot (1%), and other tools (6%) to complete the project.

Empirical Analysis Findings
After removing the incomplete student responses from the 81 participants, 54 valid data points for empirical analysis remained. Table 1 presents the results for the empirical analysis using a linear regression regarding the effects of the independent variable (i.e., the AI software quality and the student-generated code quality) on students' perception of AI tools.

Dependent variables are Clarity, Trust, and Confidence as shown in Columns (1), (2), and (3) of Table 1. Column (1 / Clarity) reports the impact of the independent variables on students' perception of AI tools' clarity, conciseness, and relevance to the assignment based on the survey question "I found the AI-generated solutions to be clear, concise, and relevant to the assignment."

Column (2 / Trust) reports the impact of the independent variables on the extent to which they trust AI-generated solutions, based on the survey question "I trust the solutions AI-generated to be correct and accurate."

Column (3 / Confidence) reports the impact of the independent variables on how AI can help improve students' confidence based on responses to the survey question "Reviewing code generated by AI tools increased my confidence in writing code myself."

Results show that the overall quality of AI-generated code is positively related with students' perception of AI tools' clarity, conciseness, and relevance. In the shaded row, $AIQuality$ has a significant positive relationship to clarity. The value of 0.480 suggests that higher AI quality leads to perceptions that AI solutions are clearer, concise, and relevant. However, the quality of students' own generated code has no impact on this perception.

Similar results also apply to students' perception of AI tool's correctness and accuracy. A significant positive relationship (0.454, p<0.05), indicates that higher $AIQuality$ increases trust in the accuracy of AI-generated solutions.

Students also believe that "Reviewing code generated by AI tools increased my confidence in writing code myself." The significant positive relationship (0.248, p<0.05) suggests that higher $AIQuality$ boosts confidence in writing code after reviewing AI-generated code. Only $Quiz2$ scores have a small positive impact on confidence in writing code. As Quiz 2 was the more difficult of the two quizzes in the course, this suggests the perception that students who did well on the quiz were more confident that AI tools can further help them improve their code-writing abilities. Results show no significant differences between different sections, genders, or majors, in terms of how clear, concise, and relevant the AI solutions are, on trust in AI solutions, or on confidence that AI can be helpful when writing code.

Dependent Variables: Clarity, Trust, Confidence
Method: Linear Regression
Included observations: 54
Standard errors are in parentheses.

Variable	**(1)**	**(2)**	**(3)**
	Clarity	**Trust**	**Confi-dence**
$AIQuality$	0.480**	0.454*	0.248*
	(0.160)	(0.174)	(0.121)
$StudentCodeQuality$	-0.016	-0.120	0.163
	(0.203)	(0.220)	(0.153)
$StudentPreference$ (AI vs Student)	-0.177	-0.503	-0.433
	(0.481)	(0.522)	(0.362)
$Quiz1$	-0.013	-0.052	-0.035
	(0.044)	(0.047)	(0.033)
$Quiz2$	-0.014	-0.007	0.049*
	(0.027)	(0.029)	(0.020)
N	54	54	54

Notes: *p < 0.05, **p < 0.01, ***p < 0.001.

Table 1. Empirical Analysis Results

Qualitative Evaluation
Students noted that several attempts were often necessary to have the AI-generated code produce desired results. One student commented, "The AI was able to get the correct output after a lot of prompting it. Even after a lot of prompting, it was unable to successfully… line the columns up." Additional shortcomings in the AI-generated solutions were related to lack of error handling and validating user inputs.

One student said in his overall assessment:

> "Both the AI-generated code and my manually developed code have their merits. The AI code is notable for its straightforward and clean approach, while my code is distinguished for its modular structure and potentially more efficient handling of user choices. The choice between the two would depend on the specific needs and preferences of the developer or the project. The AI code appears to be a well-rounded solution without the need for significant prompt refinement. It gives a good framework where the user would have a good start. Although it has some errors and some areas made need a bit more tweaking, the overall code seems to have a good start."

This gap between what students learn in the classroom and the often more advanced solutions generated by AI underscores the need for critical thinking skills, as students must be able to evaluate the quality and accuracy of AI-generated code.

5. DISCUSSION

This study focuses on perceived benefits and drawbacks that students have when using generative AI tools to develop code. While we encourage students to use AI tools as part of their learning, equally important is their ability to evaluate the results that AI generates to determine their trustworthiness. In the context of a coding course, this may often involve learning new features or constructs of a language not covered in class, to be able to critically evaluate the output and refine their prompts or the code generated to further develop their own coding skills.

We return to our research questions:

RQ1: How do students perceive learning from code they write themselves compared to code generated by AI?

Both students and instructors agree that AI tools can be a valuable learning tool, and instructors need to adapt their assignments to find ways to integrate these tools into the classroom that develop critical thinking and problem-solving skills.

They see AI tools to supplement and enhance their coding efforts, but still recognize the value of being able to write their own original code. Many prefer their own code solutions for some tasks because their code is more familiar and easier to follow, regardless of whether their code is more efficient.

RQ2. How can characteristics of software quality provide a framework for students to evaluate their own code and that generated by AI tools?
The five quality indicators provide guidelines for students to evaluate their code and that generated by AI.

Correctness/Completeness: Students can test both their code and AI generated with various combinations of input values, and make sure that the output values match.

Efficiency: Students can examine both solutions to determine if they use similar approaches to storing data, and if not, try to figure out if one leads to more efficient code than the other. For example, using dictionaries can often reduce the number of lines of code required to represent the same information using if statements and other control structures.

Understandability/ Conciseness: Students can compare both solutions to determine "readability" and the extent to which comments are helpful.

Consistency: Students can review their code for using consistent naming conventions, white space, and other formatting guidelines. While most IDE's automatically indent, and AI tends to produce consistent code that follows standard coding conventions, students can use AI-generated code as a model when evaluating their own code.

Maintainability/Structuredness: Students can modularize their code into different functions, making it easier to maintain, and compare their solution with the AI-generated code.

When evaluating AI-generated code, students can apply their own standards – is the AI-generated code "too perfect?" For example, AI-generated code tends to use long descriptive variable names and include more comments than most beginning students might write on their own. Students can determine if they are necessary or in excess.

RQ3: What are students' overall opinions of using AI as a learning tool?
Many students appreciate the efficiency of AI-generated code and noted that it was often more concise than the code they would write themselves. This is often because as beginning coders, it takes practice and experience to recognize when to use Python's advanced features such as terse if and list comprehensions to create more concise code.

RQ4: What factors impact those perceptions?
Students found that for complex problems such as the one assigned, AI-generated code could not always accommodate the many different possible cases for data validation and error handling.

Limitations
The authors acknowledge that this study has two major limitations. First, it was performed once during the Fall 2023 semester with a group of 81 students from across 6 sections of the course taught by 4 instructors. While all assignments were synchronized, variation teaching style and class formats exist.

Second, the assignment was offered as optional/ extra credit at the end of the semester, so students had different motivations for participating. Students who did not need the extra credit to boost their grades completed it for the sake of learning, while other students may have been motivated to complete the assignment specifically because of the possibility of boosting their grades with extra credit.

Future Research
Understanding how students use ChatGPT and other generative AI tools for coding assistance at different stages of their learning process would be useful. A future study might conduct a similar learning scenario twice during the semester, first when students are beginners, and then later when topics are more advanced, to see how and if they still find generative AI helpful.

Conclusions
Anecdotal evidence suggests that learners use generative AI tools at the beginning of their coding journeys in a tutoring capacity to master conceptual topics of Python (sequence, selection, repetition, functions, logic) while they use them as more of a reference tool or personal assistant (coding agent) when coding more applied topics such as data structures or interacting with data analytics libraries.

In the latter case, students' use generative AI to save them time by providing syntax for specific data queries or visualization features ("find all of the customers from Massachusetts with orders exceeding $50,000", "place the legend of the chart at the top right", "make the wedges of the pie chart in four different shades of blue") rather than explaining fundamental programming concepts.

This exercise asked students to develop a Python solution to a programming problem, compare their solution with one generated by AI tools, and evaluate the results from a software quality perspective. The process revealed insights into students' perceptions around using AI for learning, from which educators can leverage generative AI tools in ways that encourage critical thinking and enhance learning outcomes. This study contributes to the body of knowledge that shows that generative AI will play an increasingly important role in computing education.

7. ACKNOWLEDGMENTS

The authors acknowledge Prof. Jacqueline Masloff of Bentley University for creating the "Build a Bike" assignment described in Appendix A, and for encouraging her students to participate in this study.

8. REFERENCES

Aruleba, K., Sanusi, I. T., Obaido, G., & Ogbuokiri, B. (2024). Integrating ChatGPT in a Computer Science Course: Students Perceptions and Suggestions (Version 1). https://doi.org/10.48550/ARXIV.2402.01640

Boehm, B. W., Brown, J. R., & Lipow, M. (1976). Quantitative evaluation of software quality. In *Proceedings of the 2nd International Conference on Software Engineering.*

Chan, C. K. Y., & Hu, W. (2023). Students' voices on generative AI: Perceptions, benefits, and challenges in higher education. *International Journal of*

Educational Technology in Higher Education, 20(1), Article 1. https://doi.org/10.1186/s41239-023-00411-8

Choudhuri, R., Liu, D., Steinmacher, I., Gerosa, M., & Sarma, A. (2024). How far are we? The triumphs and trials of generative AI in learning software engineering. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering* (pp. 1–13). https://doi.org/10.1145/3597503.3639201

Code.Org. (2024). Learn computer science. Change the world. [Website]. Retrieved August 25, 2024, from https://code.org/

Denny, P., Leinonen, J., Prather, J., Luxton-Reilly, A., Amarouche, T., Becker, B. A., & Reeves, B. N. (2024). Prompt problems: A new programming exercise for the generative AI era. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education* (Vol. 1, pp. 296–302). https://doi.org/10.1145/3626252.3630909

Feldman, M. Q., & Anderson, C. J. (2024, June). Non-expert programmers in the generative AI future. In *Proceedings of the 3rd Annual Meeting of the Symposium on Human-Computer Interaction for Work* (pp. 1-19). https://doi.org/10.1145/3663384.3663393

Kim, Y., Park, Y., & Choi, J. (2017). A study on the adoption of IoT smart home service: Using Value-based Adoption Model. *Total Quality Management & Business Excellence*, 28(9–10), 1149–1165. https://doi.org/10.1080/14783363.2017.1310708

Ma, B., Chen, L., & Konomi, S. (2024). Enhancing programming education with ChatGPT: A case study on student perceptions and interactions in a Python course (Version 3). https://doi.org/10.48550/ARXIV.2403.15472

Miguel, J. P., Mauricio, D., & Rodriguez, G. (2014). A Review of Software Quality Models for the Evaluation of Software Products. *International Journal of Software Engineering & Applications*, 5(6), 31–53. https://doi.org/10.5121/ijsea.2014.5603

Petrovska, O., Clift, L., Moller, F., & Pearsall, R. (2024). Incorporating generative AI into software development education. In *Proceedings of the 8th Conference on Computing Education Practice* (pp. 37–40). Association for Computing Machinery (ACM). https://doi.org/10.1145/3568812.3603476

Rihidima, L. V. C., Abdillah, Y., & Rahimah, A. (2022). Adoption of Cash on Delivery Payment Method in E-commerce Shopping: A Value-based Adoption Model Approach. *Jurnal Manajemen Teori Dan Terapan: Journal of Theory and Applied Management*, 15(3), 347–360. https://doi.org/10.20473/jmtt.v15i3.38964

Shein, E. (2014). Should everybody learn to code? Communications of the ACM, 57(2), 16–18. https://doi.org/10.1145/2557447

Tala, M. L., Müller, C. N., Nastase, I. A., State, O., & Gheorghe, G. (2024). Exploring university students' perceptions of generative artificial intelligence in education. *Amfiteatru Economic,* 26(65), 71-88. https://doi.org/10.24818/EA/2024/65/71

Yilmaz, R., & Karaoglan Yilmaz, F. G. (2023). The effect of generative artificial intelligence (AI)-based tool use on students' computational thinking skills, programming self-efficacy, and motivation. *Computers and Education: Artificial Intelligence*, 4, 100147. https://doi.org/10.1016/j.caeai.2023.100147

Zastudil, C., Rogalska, M., Kapp, C., Vaughn, J., & MacNeil, S. (2023). Generative AI in computing education: Perspectives of students and instructors. In *2023 IEEE Frontiers in Education Conference (FIE)* (pp. 1–9). Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/FIE58773.2023.10343467

Zhang, Z., Dong, Z., Shi, Y., Price, T., Matsuda, N., & Xu, D. (2024). Students' perceptions and preferences of generative artificial intelligence feedback for programming. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 38, No. 21, Article 21). https://doi.org/10.1609/aaai.v38i21.30372

ISSN: 2473-4901
v10 n6172

Appendix A. BUILD A BIKE Description

In this assignment you are going to write a program to help people interested in buying a bike for commuting to school or work or just for fun and exercise. The program will allow them to configure a bicycle that will meet their needs. Bike buyers may not know what kind of bike is best suited for them so the program will ask them if they bike on paved roads and bike paths; pavement and natural surfaces; and dirt roads and trails. Their selection will then determine a recommendation of the best bike for them.

The program displays the type of bike that is recommended and ask if they want to continue with that selection. If they do, then depending on the type of bike they select, they will be asked for the size of bike, frame material, and handlebars they prefer. The base prices for the models and the additional costs for options are:

Bike Model	Price
Mountain Bike (M)	$1550.00
Hybrid Bike (H)	$1150.00
Road Bike (R)	$1000.00

Frame Size Options	Price
Small (S)	$1000.00
Medium (M)	$1500.00
Large (L)	$2000.00

Frame Material Options	Price
Aluminum (A)	$200.00
Carbon Fiber (C)	$750.00
Steel (S)	$350.00
Aluminum/Carbon Mix (M)	$800.00

Handlebar Options	Price
Flat (F)	$0.00
Riser for Hybrid Bike (R)	$0.00
Riser for Mountain Bike (R)	$50.00
Riser for Road Bike (R)	$75.00
Drop (D)	$50.00

For each model, this chart shows recommended surface types and options for frame and handlebars:

Bike Model	Recommended Surface Type	Frame Materials	Handlebars
Mountain Bike (M)	Dirt roads and trails	Aluminum Carbon Fiber Aluminum/Carbon Fiber Mix	Flat Riser
Hybrid Bike (H)	Pavement and natural surfaces	Aluminum	Riser
Road Bike (R)	Paved roads and bike paths	Aluminum Carbon Fiber Steel	Drop Riser

Requirements
The program starts by asking the user what type of road surface they prefer to bike on; this will determine what kind of bike is best suited for them. Then, regardless of bike type, they will be asked for the size of the frame (this is based on a person's physical size). The next two questions about the frame material and handlebars will be determined by the type of bike that is selected. After all the selections are made, the output with the options selected and the pricing will be displayed in a formatted table. A sample menu might look like this:

```
Build a Bike
What kind of biking do you do?
        A - Dirt road and trails
        B - Pavement and natural surfaces
        C - Paved roads and bike paths
        Enter your choice: A
We recommend a Mountain Bike.
```

Next, display the options available for the recommended bike:

```
Your bike is available in size [S]mall, [M]edium, or [L]arge.
You can choose [A]luminum, [C]arbon Fiber, or Aluminum/Carbon [M]ix.
You can choose [F]lat or [R]iser handlebars.
Enter your choice(s): saf
```

Determine the **bike's product code** by creating a string of the letters shown in parentheses in the tables of bikes and options. For example, the code for a Mountain bike, Small frame, aluminum/carbon Mix, with Flat handlebars is SMF (The type of bike does not need to be repeated). The product code is based on the options shown in the order presented.

Summarize the bicycle configuration and product code with the customer's chosen options in a report. Include the title "**{modelname} {code} Configuration and Price**" where **{modelname}** is the name of the bike model and **{code}** is the code created above. Display results in a formatted table.

```
*****************************************************
     Mountain Bike SAF Configuration and Price:
Type of Bike:     Mountain Bike            $1,550.00
Bike Size:        Small                    $1,000.00
Frame Material:   Aluminum                 $  200.00
Handlebars:       Flat Handlebars          $    0.00
=====================================================
Total Price:                               $2,750.00
```

See the end of the assignment for several sample user interactions.

ADDITIONAL REQUIREMENTS:

- When only one configuration is possible, the program should not display options for which the user has no choice.
- Accept both upper- and lower-case letters for each menu option.
- Format all prices with two decimal places, dollar signs, and commas for values greater than $999.99.
- No input validation needed; you can assume all user input satisfied the requirement.
- Use any knowledge **we covered in class** to finish the program.

YOUR EXTRA CREDIT ASSIGNMENT

Part 1. Write the program yourself using any Python techniques that we learned in class to solve this problem. Do not use any AI tools to assist.

Part 2. Using an AI tool of your choice, provide the problem description for AI to generate a solution in the form of a Python program. Review and run the code generated by AI to verify whether it works, and if it is not correct, determine how to modify your prompt(s) to obtain (hopefully) better results.

Write a report in which you describe:

- The name of the generative AI tool you used.
- Share the link to the conversation you have with AI. Provide the URL or copy and paste your entire conversation as an appendix to your report.

Compare your solution with the one(s) provided by the AI tools you used. Comment on:

- the overall approach that you took compared with that of your generated AI solution. What approach did you take? (how did you represent the data, or navigate through the different choices?) What approach did AI take?
- Did the generated AI code use features of Python that you needed to learn about to understand?
- After reviewing both, which one do you prefer for solving this problem?
- Did you need to refine your prompts to improve the Generative AI solution?

When comparing your solution with the one(s) produced by AI, consider the following aspects regarding the quality of the code. These metrics are based on a seminal paper on computer software quality evaluation (Boehm et al. 1976)[1].

1. **Correctness/Completeness**: Run the AI-generated code on various test cases. Does the code generate correct outputs given different inputs? Does the code provide all the required output?
2. **Efficiency**: Are the data about the application (e.g., pricing information) maintained using efficient data structures? An efficient data structure can reduce the use of control structures (e.g., loop and if statements).
3. **Understandability/ Conciseness**: Is the code easy to read and understand? Is it long and overly complicated? Are comments included and helpful? Is the code so concise that it is hard to understand?
4. **Consistency**: Consistency in code makes it easier to maintain, debug, and for others to collaborate. Is the code consistent in its use of naming variables, indentation, and formatting?
5. **Maintainability/Structuredness**: Code is maintainable if it is modular, does not duplicate steps, and is written in such a way that if business circumstances change, updates to the code to reflect those changes are minimal. For example, does the solution break the problem down into smaller modules or functions? if the business decides to change the prices of their products, would that require a change to many lines of the code?

Submit both your Python code file and your Word doc containing your evaluation of the generated AI code.

RUBRIC

#	**Criteria**	**Points**
1	Ask for the user's name and display the user's name in the Welcome prompt	2
2	Compute the calculations from the values provided in the Introduction and as described in the "Calculations" section	10
3	Formatting output with correct number of decimals, separator line of equal signs, new lines.	5
4	Comments and good programming style	1
5	Use of symbolic constants and commented out code	2
	Total	**20**

[1] Boehm, B. W., J. R. Brown, & M. Lipow. (1976). Quantitative evaluation of software quality. *Roceedings of the 2nd International Conference on Software Engineering.*

Sample User Interactions

Sample Run 1

```
Build a Bike
What kind of biking do you do?
        A - Dirt road and trails
        B - Pavement and natural surfaces
        C - Paved roads and bike paths
        Enter your choice: a
We recommend a Mountain Bike.
Your bike is available in size [S]mall, [M]edium, or [L]arge.
You can choose [A]luminum, [C]arbon Fiber, or Aluminum/Carbon [M]ix.
You can choose [F]lat or [R]iser Handlebars.
Enter your choice(s): saf

*****************************************************
     Mountain Bike SMAF Configuration and Price:
Type of Bike:     Mountain Bike             $1,550.00
Bike Size:        Small                     $1,000.00
Frame Material:   Aluminum                  $  200.00
Handlebars:       Flat Handlebars           $    0.00
=====================================================
Total Price:                                $2,750.00
```

Accept both upper- and lowercase inputs.

Sample Run #2

```
Build a Bike
What kind of biking do you do?
        A - Dirt road and trails
        B - Pavement and natural surfaces
        C - Paved roads and bike paths
        Enter your choice: B
We recommend a Hybrid Bike.
Your bike is available in size [S]mall, [M]edium, or [L]arge.
Enter your choice(s): m
*****************************************************
      Hybrid Bike MHAR Configuration and Price:
Type of Bike:     Hybrid Bike               $1,150.00
Bike Size:        Medium                    $1,500.00
Frame Material:   Aluminum                  $  200.00
Handlebars:       Riser Handlebars          $    0.00
=====================================================
Total Price:                                $2,850.00
```

Accept both upper- and lowercase inputs.

Appendix B. Survey

Participant Information

Your Name: ____

In which section of CS 230 are you enrolled?

Which Generative AI tool did you use to generate a solution to this program?

○ ChatGPT ○ Google Bard ○ Grok ○ Copilot ○ Other ___________

Provide a link to your AI chat session if possible.

Analysis

Describe your overall approach to solving this problem, as compared to your AI-generated solution. For example, what Python statements or data structures did you use? How did the AI-generated solution accomplish these tasks?

Did the AI generated solution use features of Python not covered in class that you needed to learn on your own?

Describe the prompts you entered for an AI-generated solution. Did you need to refine them to get the desired results, and if so, how?

Evaluation

Criteria	Your Code	AI Generated Code
Correctness/Completeness Run the code on various test cases. Does the code generate correct outputs given different inputs? Does the code provide all the required output?		
Efficiency Are the data about the application (e.g., pricing information) maintained using efficient data structures? An efficient data structure can reduce the use of control structures (e.g., loop and if statements).		
Understandability/ Conciseness Is the code easy to read and understand? Is it long and overly complicated? Are comments included and helpful? Is the code so concise that it is hard to understand?		
Consistency Consistency in code makes it easier to debug, and for others to collaborate. Is the code consistent in its use of naming variables, indentation, and formatting?		
Maintainability/Structuredness Code is maintainable if it is modular, does not duplicate steps, and is written in such a way that if business circumstances change, updates to the code to reflect those changes are minimal. For example, does the solution break the problem down into smaller modules or functions? If the business decides to change the prices of their products, would that require a change to many lines of the code?		

In this section we ask you to review your solution and the solution generated by AI based on several considerations often used when evaluating software.

After reviewing both solutions, which do you prefer and why?

Opinion

To what extent do you disagree or agree with these statements?
(Scale: Strongly Disagree, Disagree, Neutral, Agree, Strongly Agree)

Student Experience

1. I enjoyed using AI tools to generate a solution for this assignment
2. I found the AI-generated solutions to be clear, concise, and relevant to the assignment
3. I trust the solutions AI-generated to be correct and accurate

Impact of Generative AI on Learning

4. I learned new Python programming skills or techniques by examining the AI-generated solutions
5. I would recommend using AI as a learning tool.
6. Reviewing code generated by AI tools increased my confidence in writing code myself
7. Using generative AI helped me learn how to read and interpret Python code as a solution for this assignment