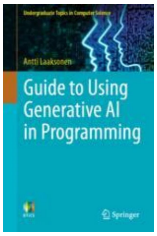
Review of [1]

Guide to Using Generative AI in Programming

Antti Laaksonen



Springer, 2026
183 pages, $49.99 Softcover, $39.99 eBook

Review by
Nicholas Tran
Department of Mathematics and Computer Science
Santa Clara University

Antti Laaksonen's *Guide to Using Generative AI in Programming* is a concise primer on using generative AI tools in programming. Written for undergraduate programmers who seek to improve their coding efficiency, the book explains the capabilities and limitations of generative AI tools for programming and demonstrates how they can be used to generate, analyze, test, and debug code in twelve short chapters that sketch out techniques through examples. The book also illustrates how to learn programming actively using generative AI as a tutor and as an assessment tool. Acknowledging the rapidly evolving power of generative AI, the book ends with an ambivalent discussion of the future of programming. When the author asked a chatbot to respond to the statement that though AI "may have some programming skills, but human programmers are still superior," the chatbot replies that "AI can enhance and support, but not replace, the unique strengths of human programmers." Whether this is an example of AI sycophancy is left as an exercise for the reader.

1 Summary of Contents

Chapter 1 gives an overview of the contents and organization of the book and provides evidence of the book's human authorship. **Chapter 2** reviews the evolution of tools and resources available to programmers up to the arrival of generative AI. **Chapter 3** introduces general-purpose chatbots and editor-integrated assistants based on large language models that can generate code. It outlines best practices for using these tools, including how to write effective prompts and the need to verify the accuracy of the generated code. It ends with a brief discussion of the key concepts in the technology of generative AI such as language models, tokens and embeddings, neural networks, and context and attention.

Chapters 4 through 7 describe how to use generative AI tools in four main programming activities. **Chapter 4** explains how generative AI can be better than traditional search engines in retrieving documentation (especially from different sources) when provided additional context and examples. **Chapter 5** illustrates prompting techniques to generate code that matches more precisely the programmer's needs. A particularly useful tip is to ask the AI to generate code based on examples or even drawings and to automate routine tasks with shell scripts. **Chapter 6** discusses

the use of AI tools to inspect code for logical errors, to generate unit tests, and to simulate user interactions to test user interfaces. One interesting example is the use of AI to generate a large number of test cases to compare the output of an algorithm against that of a brute-force solution for the same problem. **Chapter 7** demonstrates how AI tools can effectively explain the purpose of a given piece of code; in contrast, these tools are no better than traditional tools in reformatting code or improving its style. Converting code to another language or refactoring code to improve its structure are also tasks that AI tools can perform better than traditional tools, although the result should be verified for correctness.

Chapter 8 reinforces the caution made in the earlier chapters about accuracy and reliability of AI-generated code by discussing the limitations of generative AI tools, especially the tendency to "hallucinate" or generate made-up information to please the user.

Chapter 9 shows that well-known software engineering principles such as precise specification, and top-down modular design are still important when using generative AI tools to develop a complex project. As an example, the chapter walks through the development of a Tetris-like word game as a web application using HTML and JavaScript. The example shows that imprecise specifications of the intuitive concept of "rotation" led to buggy versions of the game, but once the correct definition and a good modular design was provided, the AI tools were able to generate code that worked correctly on the first try.

Chapter 10 shows that generative AI tools can be used effectively as a personal tutor to explain well-known concepts, to help debug code, and to create exercises to test the user's understanding, but it also warns about the risk of passive learning, where the user simply verifies the AI's solution without engaging in the active problem-solving process. As an example, the chapter demonstrates how to learn to program in the language Prolog.

Chapter 11 discusses the use of generative AI tools for teaching from the instructor's perspective. It gives various methods for recognizing AI-generated solutions and shows how to use AI as a source of inspiration for creating exercises.

Chapter 12 ends with a discussion of the future of programming in the age of generative AI.

2 My Opinion

Undergraduate programmers at all levels will learn something useful from this book (especially Chapter 10), and its no-nonsense style makes it easy to absorb the material. Their instructors will find Chapter 11 useful as well; although the book does not discuss GitHub Copilot at length, I personally find it convenient (in a disturbing way) in preparing exams and their solutions.

The reader should take heed of the book's warning to take advice from generative AI with a healthy dose of skepticism. Inspired by the book, I asked ChatGPT to explain the JFLAP code of a DFA accepting binary strings starting with 00 and ending with 1 and of a Turing machine accepting $\{0^n1^n : n \geq 0\}$. Its answers are generally correct but not perfect: several iterations were needed to get the simplest description of the DFA, and a warning of a nonexistent bug was issued for the Turing machine. ChatGPT and Claude were much less successful with my request to generate a two-tape Turing machine to perform binary addition, giving solutions that stopped immediately after the first move on simple inputs.

Currently free plans for high-quality generative AI tools impose a daily access limit that is too low for many applications described in the book.