What defines a correct program? What education makes a good programmer? The answers to these questions depend on whether programs are seen as mathematical entities, engineered socio-technical systems or media for assisting human thought.

Chapter 1
Introduction

Cultures of Programming shows how programming concepts and methodologies emerged and developed from the 1940s to the present and interprets key historical moments as interactions between five different cultures of programming.

Teacher: The author of this book believes that we all have different ways of thinking about programming. I wonder if that is really the case. Maybe we can just find out whether we agree or disagree about programming?

Xenophon: I do not see how we could disagree on this. Programming is the process of developing software.

Socrates: This is a painfully limited perspective! Programming is a tool for understandings the world.

Archimedes: Those are lofty visions, but in reality, most programming is done to solve a business problem that is identified through careful analysis of business needs. Methodologies make understanding of the problem, obtained through programming, key part of the iterative development lifecycle.

Socrates: You keep treating programming as a boring commercial utility. It is better seen as a kind of literacy. Programming forces you to think about the world in a clear structured way that you otherwise do not experience. It teaches a new way of thinking.

Pythagoras: I'm all for treating programming as a new kind of literacy, but only because it is really a form of applied mathematics. It teaches mathematical thinking. Programming is a process

Diogenes: [...]

Teacher: [...]

Archimedes: [...]

Pythagoras: [...]

Socrates: [...]

Diogenes: [...]

Xenophon: [...]

Diogenes: [...]

Socrates: [...]

Pythagoras: [...]

Socrates: [...]

over the world to teach programming to kids.

Xenophon: When *Diogenes* started talking about hackers, I thought the discussion was becoming a bit odd, but using live musical performances as the most notable example of programming is just crazy! How is that using computers for anything useful?

Socrates: Do you not find education and exploring creative potential of computers for a new kind of art useful? If you want to question what kind of use of computers is useful, I would worry more about the ways in which they get used by large corporations!

In the first part of the exhibition, we look at the basic beliefs of the five different cultures of programming—mathematical culture, hacker culture, engineering culture, managerial culture and humanistic culture.

Teacher: Do you have anything specific in mind, *Socrates*?

Socrates: Consider the system for screening job applications built by Amazon that journalists uncovered in 2015. This was supposed to identify good candidates and Amazon trained it on resumes received in the past 10 years. It turned out that the algorithm was heavily biased against women and would penalize resumes including phrases such as "women's chess club captain".

Teacher: This might be an interesting case to talk about. Perhaps the plurality of our views on programming will let us better identify what the issues with programming are and find a way of addressing those.

Pythagoras: I agree the Amazon system is unacceptable. But this is not an issue with the algorithm itself. The issue is that it was used poorly. Presumably, the algorithm just learned a bias that was already present in the training data set.[8]

Socrates: What I find more worrying is that we often do not understand the effects of programs that we create. Consider the numerous AI chatbots that have learned to imitate the inflammatory and racist language of their users or their training datasets![1]

Teacher: We will get to issues with programming soon enough, but I wanted to start with important concepts and positive examples. Can we please get back to those?

Archimedes: As I said, I do not think there is a single achievement. What matters is that we are getting better at programming and are able to bring value to the society. A good example is the Agile movement, which takes as central the idea that business people and developers need to work more closely together.

Socrates: Phrases like "we value individuals and interactions over processes and tools" from the Agile manifesto sound nice. But the Agile movement also subordinates programming into a support role for commercial enterprises. Like earlier software development methodologies, it is a mechanism for control. Not to mention the fact that all 17 authors of the Agile manifesto [...] the perspective they can offer is inevitably narrow.

Xenophon: [...] the issue you have with control. If you want to build anything [...] still need to have some programmers and then you obviously also need some form of team structure or "control" if you wish.

Socrates: [...] is a misguided view, but first I'm curious to hear what *Xenophon* finds to be an example of paradigmatic programming achievement.

Xenophon: An example? The development of the Apollo guidance computer (Figure ??), which helped to land a man on the moon. The development had its difficulties, but those were resolved thanks to rigorous definition of requirements, followed by careful coding and rigorous testing.

Pythagoras: You are playing it safe! Nobody can disagree that landing a man on the moon is an impressive achievement, but even the Apollo guidance software had bugs.

Xenophon: That is correct, but those bugs were well-documented and the crew knew how to operate the computer to avoid their effects. Flying with the bugs simply had a lower risk than attempting to fix them at the last minute.[3]

Pythagoras: This is why I find the present state of computer programming unsatisfactory. A program is a formal entity and so you can use formal mathematical methods to show that a piece of software is correct. We should build software that is provably without bugs rather than coming up with post-hoc workarounds!

Diogenes: Has anybody actually done this in practice? What is your example?

Pythagoras: Formal verification is challenging, but there are some good examples such as the formally verified [...] systems that are robust against, including one that controlled an unmanned flight of the AH-6 helicopter.[4] But my example of a paradigmatic achievement would be the Algol programming language, which pioneered the [...] formally analysed and made all the follow-up work thinkable.

Teacher: Our examples so far include the Agile movement, the Algol language and the Apollo guidance system. *Diogenes* and *Socrates*, would you agree that they are good examples of the great achievements in programming?

Rooted in disciplines such as electrical engineering, business management, mathematics or psychology, the different cultures of programming have exchanged ideas and given rise to novel programming concepts and methodologies.

Socrates: Training data is one issue, but not the only one. This is a perfect example of why you need to think about programming [...] algorithms are designed by humans, built by humans and used by humans. A human bias can enter the scene at any point during programming [...]

Pythagoras: According to a dictionary definition,[9] an algorithm is "a set of mathematical instructions or rules that [...] computer, will help to calculate an answer to a problem." A set of mathematical instructions is a mathematical entity. An algorithm cannot be any more biased than a multiplication or a monoid!

Diogenes: You are wrong. You can have an instruction `if (gender="M") salary+=1000`, which increases the salary [...]

Pythagoras: But this is mixing data with your algorithms! What I mean by an algorithm is a fully generic procedure [...] does not contain any hard-coded information about a specific problem in this way.

Diogenes: Even if [...] practical human knowledge. In an artificial neural network, you have to set the number of layers, propagation functions, hyperparameters [...] source of bias. What worries me about algorithms like neural networks is that it is very hard to gain the necessary practical experience to [...]

They have also clashed about the nature of programming and those clashes remain at the core of many questions about programming today.

Archimedes: You can build systems that guarantee fairness, but not by relying on unreliable practical experience. For systems that make decisions about individuals, you can use counterfactual fairness,[10] which ensures that the result given by the algorithm is the same regardless of the demographic group of the individual.

Xenophon: I have to admit, this discussion is beyond me. I can see that there are more and less problematic algorithms, but which one should I use if I need to conform with the right to explanation required by the EU GDPR regulation? The industry needs to agree on standards that we can adhere to in order to avoid such problems.

Socrates: A regulation might solve your problem in the short-term. In the long-term, programmers needs liberal arts education that includes social sciences, arts and philosophy. Much of the discussion we're just having is already present in Heidegger's work "The question concerning technology" from 1954!

Teacher: We are getting back to arguing about the nature of programs. But I'm still curious if we can use our different perspectives in a more productive way.

Xenophon: I don't see how I could have a productive conversation with anyone who ignores the business context in which programming is typically done.

Diogenes: It will always be counter-productive to just talk. What matters is code.

Pythagoras: [...] tion. If we shift our attention from the nature of programming to concrete programming concepts, we may be able to find ways through which our different views can contribute to the development of programming.

In the second part, we illustrate how cultures exchanged ideas on programming languages, types and object-oriented programming and how they disagreed about the nature of the discipline and questions of beauty and art.

Cultures of Programming

The Development of Programming Concepts and Methodologies

Tomas Petricek