Computational Thinking and Coding for Every Student

Discussion Guide

Jane Krauss

Kiki Prottsman

Computational Thinking and Coding for Every Student offers both a rationale and practical steps for introducing computer science in school. The authors recommend discussing your ideas and experiences with fellow readers to foster professional learning and personal reflection. This discussion guide is intended as a starting point for collegial conversations. Connect with other teachers who are starting their journey. Whenever you come across a question that really makes you think, share your response on social media and let everyone share in your epiphany!

Discuss on social media!

- FB: www.facebook.com/groups/CodingInClass
- Twitter: @CTandCoding
- G+: https://goo.gl/R9Q429
- See the companion site for more: resources.corwin.com/ComputationalThinking

PREFACE

Skipped the preface, did we? Many readers skip past the front matter of a book and dive right into the first chapter. This might be a good time to go back and read it. It's short, so you and your colleagues (who also skipped it) can read it together! In the preface, the authors summarize the state of computer science education (it's booming) and describe the arc of the text. Think of the preface as a brief summary of both the context for and the context of the book.

1. The preface presents the recent shift from little to lots of momentum for computer science in U.S. schools. Discuss: What was your impetus to give computer science (or at least this book) a try? What piqued your interest? In what ways were your impressions about computer science confirmed or changed?

_____

_____

_____

2. As you get ready to dive into this book, what are your greatest hopes about introducing computer science? Your greatest concerns?

CHAPTER 1. AN INTRODUCTION TO COMPUTER SCIENCE

1. In Chapter 1 you explored the processes for solving a sudoku puzzle in a methodical way. Did you get a sense of how a solution could be derived by applying computational thinking?

2. Algorithms are all around us. Discuss another activity or task you accomplish by carefully executing a set of steps. What instructions might you lay out if you emulated the process in a computer app?

3. You likely have a better sense now of what computer science is and is not. Pretend you are speaking to a parent or your school administrator. How would you describe the difference between learning to use computers and learning computer science?

CHAPTER 2. WHY KIDS SHOULD HAVE THE OPPORTUNITY TO LEARN

1. In Chapter 2 the authors present computer science both as a fundamental literacy and as a way for students to practice resiliency in problem solving. What lessons in the core curricula do you teach that draw on students' reasoning and logic? What parallels do you draw between those and learning experiences in computer science?

2. The authors relate Seymour Papert's association of computers to mud pies as both being mediums to think with. How does your role as teacher change when your students are learning in open-ended and exploratory ways?

3. Chapter 2 concludes by pulling back from the classroom experience to consider the larger societal case for why more youth should follow a computing education and career pathway. What of this rationale resonates with you? Do you see introducing CS as a supplement to your program, or as an obligation?

CHAPTER 3. TRY YOUR HAND AT CODING

1. Whew! Because you worked through the exercises in Chapter 3, you are now more experienced in computer science than most teachers! Your computational thinking got a workout, too. Each exercise ended with journal questions–prompts to help you examine the experience. Refer to your notes and discuss your experiences and impressions. Did you have similar experiences? Or did some activities present greater challenges or result in greater rewards for some of you than others?

2. Pair programming was recommended as a metacognitive strategy for evaluating and improving reasoning. Did you try it with these exercises? If so, in what ways did thinking aloud together affect the experience?

CHAPTER 4. GETTING STARTED IN THE CLASSROOM

1. As with any student experiences involving computers, educators need to pay attention to students' physical and social well-being when teaching computer science. Discuss the advice in Chapter 4, and describe what resonated with you the most. What explicit actions will you take now that you might not have considered before?

2. Another practical concern raised in Chapter 4 was accessing appropriate technology for a fulsome computing experience for your students. Think about your teaching and learning environment in terms of technology access. In what ways does it fall short, and what workarounds might you try? If adjustments involve others, how will you state your case?

3. Authors offered suggestions for getting started with computing. Consider your technology access and personal readiness and discuss the curriculum choices you plan to make.

CHAPTER 5. DOS AND DON'TS OF TEACHING COMPUTER SCIENCE

1. Discuss which "dos and don'ts" advice was most useful to each of you. What other "dos and don'ts" came to mind?

2. In what ways do any of these "dos and don'ts" apply to other subjects you teach?

3. Do any of these "dos and don'ts" conflict with practices you should or shouldn't do in other classes?

CHAPTER 6. ACTIVITIES THAT FOSTER COMPUTATIONAL THINKING

1. Computational thinking is an approach to problem solving that backs up a few steps to encompass problem *finding* and problem *posing* as well. Have you thought about problem solving as a stage in a bigger process before now? With your fellow readers, think of a problem-solving activity in your regular curriculum that could be more robust if you didn't hand students a problem to solve but instead set up the conditions by which they become aware of and begin framing a problem to solve.

2. There is great interplay between the elements of computational thinking one applies to the task of programming a computer. That said, teasing CT apart helps get at important features of the process. In Chapter 6, the authors recommend that you get comfortable calling out CT practices by name. Are these "pillars" making sense to you? Would you refer to them within computer science learning? Where are they evident in other subjects you teach?

3. Look at the everyday examples of each "pillar" of computational thinking in the table at the end of Chapter 6. With your reading buddies, challenge yourselves to think of several more examples for each pillar. Can you come to agreement on one best example for each?

CHAPTER 7. DECOMPOSITION

1. The authors state, "Decomposition is breaking a problem down into smaller, more manageable parts." Which of the activities in Chapter 7 is most suited to your students and would best get across this universal approach to problem solving?

2. Imagine you are explaining decomposition as a problem-solving method to your students. How would you describe the process, and what examples might you give that would resonate with the interests and life experiences of the particular age group you teach?

CHAPTER 8. PATTERN RECOGNITION (WITH PATTERN MATCHING)

1. Reflect on the "pattern matching" activities and lesson in Chapter 8, and discuss in what ways "pattern matching" is about extrapolation and paying attention to salient cues. How might pattern matching be put to use in examining routines in long division, trends in history, structures in music composition, or "tells" in a game of poker?

2. How does pattern matching help pave the way for abstraction (described in the next chapter)?

CHAPTER 9. ABSTRACTION

1. Abstraction is used extremely often, even though we normally don't call it out as such in everyday life. Can you think of times when you use abstraction effortlessly?

2. Say you were going to explain the process of making cookies first to a forty-year-old, then to a four-year-old. How would your abstraction differ? With whom do you think you would use the most abstraction? Why?

3. Can you relate the idea of abstraction back to computer science? How might it help make your work easier if you were trying to create one function that added $x + 5$, one that added $x + 2$, and one that added $x + 7$ (where x is a number given as input by the user)?

CHAPTER 10. AUTOMATION

1. The authors mention in Chapter 10 that automation isn't always about running things on machines. How might automation make something easier, even if you still have to do it by hand?

2. Algorithms and automation often go together. Can you think of a reason that you might need one without the other?

3. Refer back to that abstracted algorithm for creating cookies presented earlier in the discussion guide. Now, imagine you were going to translate

it for a bakery system. How would the algorithm be different if you were sharing instructions with adults versus children? What might that algorithm look like if you were trying to prepare it for automation?

CHAPTER 11. ACTIVITIES THAT FOSTER SPATIAL REASONING

1. Your introduction to spatial reasoning started with a story from Seymour Papert's childhood. He said, ""Gears, serving as models, carried many otherwise abstract ideas into my head." Describe a time in your own learning where associating a new or abstract concept to something in the physical world helped you reach understanding.

2. The authors posit that spatial thinkers aren't born; rather, they are made through sufficient experience. What advice in this chapter about "spatializing" your teaching are you likely to try? Why does this kind of activity have merit over other ideas?

CHAPTER 12. MAKING WITH CODE

1. The authors offer examples of inventive student work. They also say, "Making isn't about *stuff*. It's not even about the *space*. More than anything, making is culture and design thinking." Discuss ways you could infuse the maker spirit into your school program.

2. Together with your reading partners, go to Twitter and search the hashtag #makerED. What are maker-educators talking about? In what ways do their interests (or concerns) resonate with you?

3. The authors discuss pros and cons of making in school as well as issues around equitable access. If the topic of making came up in a staff meeting at school, what defense for, or argument against, making would you make?

CHAPTER 13. DESIGNING A CURRICULUM CONTINUUM ACROSS K–12

1. In Chapter 13 the authors discriminate between computer science and digital literacy. Some teachers believe graphic design or word processing fall into the category of computer science. What elements of computer science can you imagine children could learn in classes like this (or even music and physical education) if teachers were to infuse CS vocabulary and concepts into these non-CS classes?

2. Clearly, if students can learn CS from "unplugged" lessons, then computer science involves more than programming. In what way do you think students who have learned CS have an advantage in their other studies over students who have not?

CHAPTER 14. IMPORTANT IDEAS ACROSS ALL GRADES

1. Some students fight the idea of working as part of a team. How would you explain the benefits of pair programming to reluctant students to help them understand it's an aid to learning, and not a punishment?

2. Some teachers feel they aren't doing their job if they spend time standing back and observing, rather than actively teaching and helping. What are some of the pitfalls of jumping in to assist students too soon? What might be the benefits of holding back direct help and instead providing guiding questions and resources so students help themselves? What percentage of each style of helping do you do in your teaching? What percentage would you like this to be?

3. What ideas do you have to share to make stronger learners in a computer science environment? What can you do to promote equity in CS? If you have a study group, share some of your favorites with one another. Otherwise, head over to our Facebook page to have a meaningful discussion.

CHAPTER 15. THE ELEMENTARY PATHWAY

1. This chapter discusses unique challenges when teaching computer science to elementary school students. In your experience, are there any pertinent developmental milestones that the authors failed to consider? What would you suggest others take into account when trying to teach CS to young students?

2. Sometimes students in grades K–5 pick up concepts faster than adults do. (Think about how this is true with languages or complex remote controls.) How would you take advantage of their agile thinking in your classroom?

3. When you teach computer science and computational thinking to young students, it changes the way they look at problems in other subjects as well. What are some of the perks and pitfalls of having children learn to think like computer scientists?

CHAPTER 16. THE MIDDLE SCHOOL PATHWAY

1. Middle school students are a rare breed. Not yet adults, but no longer wanting to be treated like children, this age band can be difficult to bring on board with new ideas. What are some of your best techniques for encouraging buy-in when introducing new topics to middle school students? How would you present the computer science value proposition?

2. Middle school classrooms show a huge disparity in the computer science background of their students. How might you address wide differences in a way that strengthens everyone, rather than treating some students as remedial and others as advanced?

3. In what ways can you tailor your instruction to appeal to the "What's in it for me" nature of middle schoolers? Can you use this mechanism to encourage students to be positive with one another (both in class and online)?

CHAPTER 17. THE HIGH SCHOOL PATHWAY

1. High school students are starting to open their eyes to the world around them and investigate not only ways that they can get help from their community but also ways that they can be of help. Do you think your high schoolers would respond better to an activity that you have preplanned to benefit an actual end user in their neighborhood or to an activity that they are allowed to design themselves?

2. Many students will be able to easily leap into text-based coding at this age, but some will struggle. Block-based coding is a simple way of building a foundation, but it has its limitations. How can you utilize different skill sets to keep all students moving in an upward direction without anyone feeling like they are hindered by the learning pace of others?

3. When working in groups at this age, you will find that some students position themselves so they don't need to do any coding at all. In what ways might you structure your projects so that everyone gets to experience "taking the wheel," while keeping coding anxiety to a minimum?

CHAPTER 18. ADAPTING LESSONS FOR YOUR CLASS

1. With a new subject such as computer science, you might be uncomfortable straying from teaching a lesson the way it is written. If you find that you need some tweaks to an otherwise great lesson plan, what do you plan to do? Share your resources with our online community on Facebook. Also, Tweet away!

2. Sometimes lessons need more than a little alteration for a particular group of students. Choose one of the lessons from the book (or find one online) and with your reading partners, practice "fixing" it for your needs. Share your adapted lesson with our online community so they can benefit from your work, as well!

3. What might you do when you recognize that you have the perfect place to squeeze in a computer science project but cannot find a lesson that even comes close to hitting the mark for your particular grade or subject? Will you attempt to create one on your own? Will you ask for help from our community? Would you feel comfortable assigning your class the task of coming up with a project and then assigning them to complete that project?

CHAPTER 19. WHAT PEOPLE ARE DOING AND HOW THEY ARE DOING IT WELL

1. Chapter 19, with stories and testimonials from the field, was meant to inspire. Which story or testimonial spoke to you the most? What was it about the teacher, learner, or community experience that made it meaningful?

2. If you were to host an event to foster community support for your school's computer science program, what would you do? What would you be asking of the community? (Beyond monetary support, could this include volunteer time, tours of industry, or participation in career fairs?) How would you present the value proposition to make the most of a well-timed and well-advertised fundraiser?

3. Do you have any students with testimonials? Have you heard any from your local community or any online communities that you belong to? We would love to have you share those with us! Please post any outstanding student experiences (positive or otherwise) to our online group. Please share on social media to your worldwide neighborhood of educators and make computer science education better for everyone!