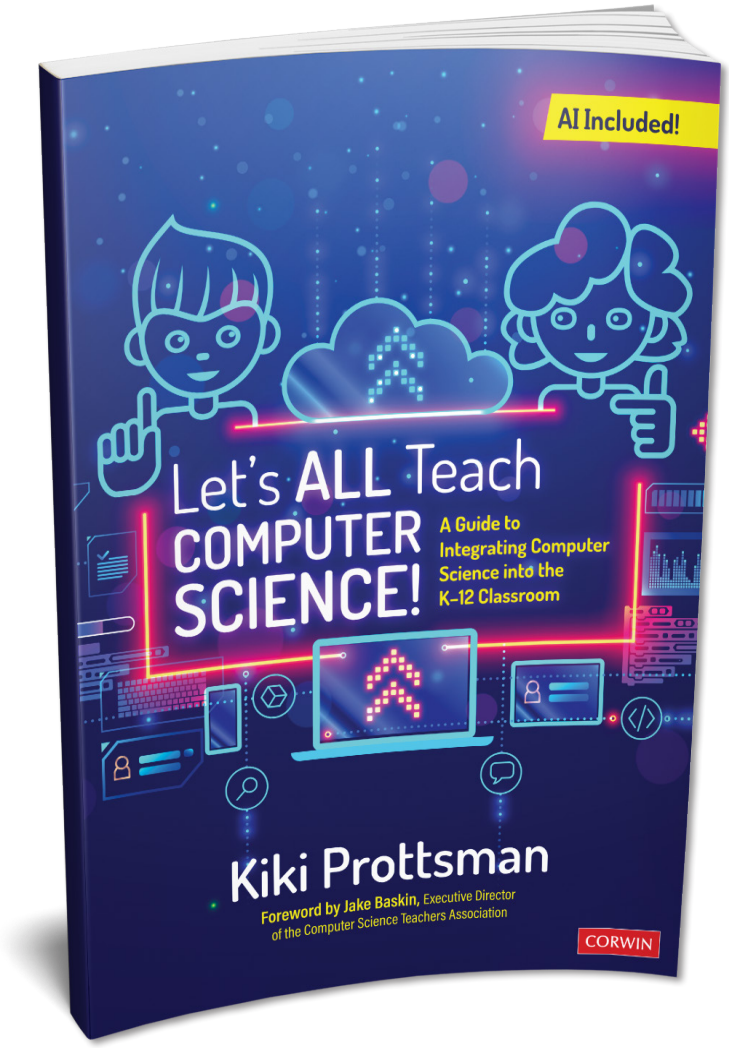

Thank you

FOR YOUR INTEREST IN CORWIN

Please enjoy this complimentary excerpt from Let's All Teach Computer Science!.

LEARN MORE about this title!

CORWIN

The Marvelous Art of Computer Science Integration

Let's delve deeper into how this fusion between computer science and other subjects takes place. Picture a math class where students not only learn traditional equations but also explore the creation of step-by-step recipes to help peers solve similar problems in the future. Those are called **algorithms,** and those are fundamental to computer science.

In a history lesson, students might use digital tools to look at historical trends and compare the way prices changed in certain areas when hit by drought or tragedy. That's data analytics, another very popular piece of computer science.

Art, biology, choir … any class can look at the way computer science is affecting the industry and bring small, entertaining lessons into everyday education. While these additions are capable of being massive and complicated, they don't have to be complex for students to develop foundations for the future.

Keep in mind that integration isn't about overshadowing your core subject, but about making harmonious enhancements so students naturally absorb CS concepts and vocabulary. It's about selecting relevant computer science ideas that naturally fit into your existing curriculum. The goal is to make integration so seamless that the focus stays on your material, and students view CS as a multitool for future problems.

Integration should be customizable according to educator preferences and student needs. Someday, districts might start dictating exactly how and when each non-CS class should hit various computer science standards, but until that day comes, you get to choose how much digital pizzazz you want to add to your teaching repertoire. That freedom may bring some level of uncertainty, but a jam-packed toolkit will help make the process easier for you and your students.

You may find that you prefer a collection of one-off coding activities, allowing you to provide a minidigital adventure for your classroom. Or you might want an extensive, year-long journey where computer science becomes the trusted sidekick in all of your lessons, like Watson for Holmes. Either way, it's no mystery that computer science has the capability of enhancing subjects when introduced properly.

Why Integration?

Computer science is one of the fastest growing fields in any developed nation. CS exists within every field, from medicine to agriculture to theater—and jobs in computer science tend to pay disproportionately well when compared to other options that require little more than a bachelor's degree.[1]

Even if we have no interest in educating students to become professional computer scientists, the skills that are taught in CS classes have been shown to bolster students in relation to both executive function[2] and standardized test scores.[3] This fact amplifies my point that computer science education is more than a path to the tech sector. It's also a method for enhancing a student's educational growth by providing lifelong support in all of their scholastic adventures. That's why I promote it, and that's why I'm writing this book.

It should come as no surprise that much of the current computer science landscape is excessively academic. As a subject, it springs from one of the most complex and logic-filled systems to be developed by humankind, which exacerbates the idea that it's not to be tampered with, except by an exclusive braintrust. This myth creates an unnecessary elitism within the field, breeding a world where technology and programs fail to represent the people who use them.

Until recently, most children weren't introduced to computer science until college, and even then the subject was reserved for students who intentionally planned to study technology. This created a cycle of wealthy tech intellectuals who would then coach their children, making it nearly impossible for someone outside the industry to feel included as a first-year CS student.

To tackle this phenomenon, research shows that we should be reaching out to kids before they solidify their concept of what they can or can't be when they grow up. Unfortunately, this happens extraordinarily early and, according to Camp and Gürer,[4] access to computers and training in computer science should begin at preschool levels to give students from historically underrepresented groups the greatest chance against developing insecurities around their capabilities.

Even still, the school system is slow to change, and with budgets that are dwindling and teachers who are overworked, the chances of incorporating dedicated CS classes in K–5 across the world will remain low well into the next decade.

That's where you come in.

Together, you and I can work to integrate computer science and coding into classes that students are already taking so we can help them develop the reasoning, persistence, and holistic mindset that it takes to be truly resilient innovators.

Computer science also promotes problem-solving. When faced with difficulties, students educated in computer science can come prepared with a toolbox overflowing with logic, creativity, and critical thinking skills. With these practical gizmos, they are more likely to be able to dig themselves out of issues that initially appear to be over their head.

The power of teamwork combined with the boundless possibilities of technology can create wonders that no single individual could accomplish alone.

But wait, there's more! Integrating computer science into other classes also unleashes the art of collaboration. Your classroom can be full of students huddled together brainstorming ideas, debugging code, and making digital discoveries. The power of teamwork combined with the boundless possibilities of technology can create wonders that no single individual could accomplish alone.

As if that weren't enough, computer science also has a huge PR issue, and I believe CS integration can fix this.

Imagine how different it is to have your first introduction to computer science happen through the creative cajoling of an art or music teacher versus the mandatory requirements of a pedantic professor. Up until this point, most people who have studied computer science found the latter, but the former has so much more potential when it comes to reaching people who wouldn't generally self-select into the subject the way it is now.

Coders tend to be drawn to code and artists tend to be drawn to art. What if we could create more artists who liked to code and coders who liked to art? Imagine the effect that would have on the future when it comes to the people represented within the tech industry … and therefore the way that technological products represent the communities that they serve. This is a more responsible future, and a far less dangerous one.

Let me elaborate.