{ SPECIAL REPORT ON EDUCATION }

<2016>

THE CODING REVOLUTION

From the White House to Silicon Valley, the call for all students to learn computer programming is growing louder. Yet some critics of this vision point to logistical hurdles, and others worry about industry dictating the curriculum. Is coding for all a realistic goal—and is it one that American schools should be pursuing?

By Annie Murphy Paul

Illustration by Patric Sandri

"*Yes!* We got here first!"

Annie Murphy Paul is a frequent contributor to the *New York Times*, *Time* magazine and Slate. Paul is author of *The Cult of Personality Testing* and *Origins*, which was included in the *New York Times'* list of 100 Notable Books of 2010. Her next book, forthcoming from Crown, is entitled *Brilliant: The Science of How We Get Smarter.*

Allie and Lauren, two sixth graders at Loyola Elementary School in Los Altos, Calif., bound into Sheena Vaidyanathan's classroom a few minutes before the start of fifth period. The girls have a long-running competition with a couple of their classmates, William and Blake, over who can arrive the earliest at their computer programming class.

COMPUTER SCIENCE CLASSES: Nancy Se teaches at Augustus F. Hawkins High School in South Los Angeles, where 99 percent of the student body is African-American or Latino and 75 percent is economically disadvantaged.

The two girls take their seats at the row of new Apple desktop computers and immediately start on the assignment Vaidyanathan has provided: finding and fixing errors in a computer program. It is the kind of activity, exacting but important, that occupies much of the time of professional programmers.

A bell sounds, and more students traipse in. Vaidyanathan, a petite woman with a warm and friendly manner, welcomes them with a smile.

"What are we doing today, Mrs. V?" asks one boy as he enters.

"Debugging a program," she responds.

"Sweet!" he exclaims and darts over to a computer.

Proponents of universal coding instruction would like to see such scenes repeated all over the country. These champions include business leaders eager to employ the next generation of engineers and programmers, as well as government officials looking to ensure the U.S.'s competitiveness in the global economy. "Every student deserves a chance to learn this essential 21st century skill," tweeted Microsoft co-founder Bill Gates earlier this year. Gates and others maintain that coding is a new literacy, as important as knowing how to read or to do math. In a 2008 essay, Marc Prensky, an author and speaker who coined the term "digital native," wrote, "I believe the single skill that will, above all others, distinguish a literate person is programming literacy."

These advocates have pointed to the high demand for computer programmers and the opportunities for advancement in the field—opportunities that, they say, will continue to expand in coming years.

Obama administration officials sketch an ambitious vision for computer science education in the U.S. in which our students keep pace with those in other nations—the U.K., they note, began requiring every student to learn to program in 2014—and persistent achievement gaps in our own country between white and affluent students on the one hand and minority and disadvantaged students on the other are closed with the help of coding for all.

Achieving this vision will not be easy, however. The extension of coding instruction to all U.S. students faces steep logistical chal-

IN BRIEF

Champions of universal coding believe computer science instruction in public schools can close achievement gaps among socioeconomic groups and help U.S. students keep pace with those in other countries.
But coding for all faces steep logistical challenges, including a shortage of teachers, the absence of an agreed-on curriculum and disparities in students' access to computers.
Some critics argue that coding represents a too narrow technical focus and that it is being pushed on schools by business leaders concerned about their bottom lines.
A potential middle ground involves teaching "computational thinking": habits of mind that include breaking down a problem, designing systems, and running small experiments to see which approaches fail and which succeed. The White House's Computer Science for All Initiative embraces coding as well as computational thinking.

lenges, from a shortage of qualified teachers to a lack of curricular materials. The U.S.'s radically decentralized school system makes it difficult to enact nationwide reform—just ask the architects of the Common Core State Standards—and most schools have yet to update their rules and standards for the information age (computer science courses often do not count toward graduation requirements in math or science, for example). Meanwhile students of different ethnic and socioeconomic backgrounds come to school with enormous disparities in their knowledge of computers; research suggests that schools often perpetuate these gaps by offering deep, substantive teaching to affluent white students and more superficial instruction to minority and low-income pupils.

Moreover, there is a growing recognition that programming skills can form only part of American students' preparation for living and working in the 21st century. Coding is the *practice* of computer science, but students also need the *theory:* an understanding of the underlying principles on which computers operate. Making such "computational thinking" universal will require a massive mobilization both outside school—in the form of extracurricular programs and resources—as well as inside school—by training teachers, revamping courses, integrating computer science into other academic subjects and changing graduation requirements.

TEACHING TO THE CODE

VAIDYANATHAN, who has a master's in computer science and has taught the subject for seven years, was an early proponent of the notion that every student should learn how to code. "And I mean *every* student—girls, boys, special education students, students who don't think they are math or science 'people,'" Vaidyanathan says. But unlike some of her fellow advocates, she tempers her enthusiasm with realism. In an interview between classes, she points out some of the hurdles that stand in the way of achieving this appealing vision. "To begin with, there are not nearly enough computer science teachers to meet what would be an enormous demand," she notes.

According to Advanced Placement (AP) publisher the College Board, only 4,310 U.S. high schools offered an AP exam in computer science in 2015—about 12 percent. Moreover, a 2014 analysis of the organization's data for 2013 found that in three states, there were no female students who took the test; in eight states, zero Hispanic test takers; and in 11 states, not one African-American test taker. (In many of these states, few or even no students took the test at all.) Further, a trained and ready corps of teachers simply is not there—and human teachers, Vaidyanathan insists, are what students need. "If students are learning material that is meaningful and challenging, they are going to get stuck at some point," she says. "They're going to need a teacher who knows them, who can explain the problem in a way they understand, and who can draw on the student-teacher relationship as a source of motivation and support. A Khan Academy video can't do all that."

Not only are there too few computer science teachers, there is also the absence of an agreed-on computer science curriculum. The CSTA has developed a set of standards for kindergarten through grade 12, but these are general in the extreme (stu-

Photographs by Timothy Archibald

dents in grades six through nine should "begin to appreciate the ubiquity of computing and the ways in which computer science facilitates communication and collaboration"), meaning that teachers such as Vaidyanathan are on their own in deciding what and how to teach their students. And in an era of high-stakes math and reading tests, computer science often gets pushed aside altogether. "There is less time in schools' schedules, and less money in schools' budgets, for subjects that are not viewed as essential," Vaidyanathan observes.

Before becoming a computer science teacher and a science, technology, engineering and mathematics (STEM) program specialist in the Los Altos School District, Vaidyanathan worked as a software engineer at information technology giant Unisys and at two Silicon Valley start-ups. Her experience in the tech industry has made her keenly aware of another hitch in the plan to turn every student into a hire-ready programmer: "There's a big difference between learning the basics of coding and knowing it well enough to do it professionally." The wide gap between novice and expert in this field is the reason she pushed her district to begin coding instruction in kindergarten. "Learning coding is like learning a foreign language," Vaidyanathan explains. "We wouldn't expect students to be fluent in French or Spanish because they took a couple of semesters of instruction in high school."

Particularly challenging is the project of extending coding instruction beyond the confines of wealthy, mostly white and Asian communities such as Los Altos. Vaidyanathan's students start out with almost numberless advantages. They enter her classroom equipped with knowledge of computers and how they work; some of them have already begun to learn about coding at home or in extracurricular enrichment programs. The coding class at their public school is supported by a foundation that collects and distributes donations from the families of students enrolled in the district. The district's schools—themselves fully furnished with desktop and tablet computers—are located in the heart of Silicon Valley, and many of their students' mothers and fathers work at the nearby headquarters of Google, Apple or Facebook. These students have been gently but firmly guided down a path by the adults in their lives—a path that leads to abundant opportunity. And yet the presence of such privileged students in computing classes—and later on, in the tech industry and in the field of computer science—is often presented as the product of a "natural" sorting process, based on individuals' innate preferences and abilities.

Jane Margolis is familiar with this presumption and its corollary: that poor students and students of color must not *want* to learn how to code. Margolis, a senior researcher at the University of California, Los Angeles, Graduate School of Education & Information Studies, undertook an intensive study on how computer science is taught in three diverse public high schools in Los Angeles. The result was the 2008 book *Stuck in the Shallow End: Education, Race, and Computing,* a troubling account of what Margolis and her collaborators call "virtual segregation." Students who go to school in affluent communities, they found, are much more likely to have access to a wide range of computer science offerings—courses that are academically challenging, creative and collaborative. Poor students and students of color, in contrast, are often offered only the most basic cut-and-paste computing instruction, often using inadequate equipment.

Even when the schools serving these students possess adequate facilities, they may be "technology rich, but curriculum poor," in the words of Margolis and her colleagues, lacking the elements of a full-fledged education in computer science: "powerful learning experiences in a sequence of classes," along with "years of thoughtful, informed guidance and support." Curricular offerings cannot be one-size-fits-all, she and her co-authors maintain, but rather must take into account racial and ethnic groups' disparate histories of access to technology and training. These "have been so different, the playing fields so uneven, the chasm so deep and wide, that people are living in two different worlds," Margolis and her co-authors observe. The easy familiarity with computers that is taken for granted among affluent stu-

Teach the Prof How to Teach

Few young faculty members arrive on campus ready for the classroom. New programs are changing that

By Jennifer Frederick

As young scientists work their way through graduate school and postdoctoral fellowships, the pressures of writing grant proposals and establishing a research laboratory take priority over teaching. When these students and postdocs eventually get faculty positions, however, few of them have experience in designing and teaching their own courses, and yet they are expected to teach right away. This would not pass muster at a high school. Why should it be acceptable at a university?

Teaching graduate students how to teach offers the chance to instill in future faculty the science of what works—and does not work—in the classroom. For example, many college science professors continue to rely on lectures as the main mode of instruction, even though evidence shows that undergraduates learn best when they actively participate in classroom discussions and activities.

The best teaching strategies also draw on scientific research skills. An instructor hypothesizes that a particular exercise will help students learn a concept and integrates assessment to check the extent to which learning takes place. Just like research, the results of each "experiment" inform the design of subsequent lessons. In contrast, traditional lecture-based courses with only a few high-stakes examinations can leave both instructors and students in the dark about who is excelling or struggling.

Evidence-based engagement strategies are associated with increased student achievement, and active learning can close the achievement gap for students from underrepresented groups as well as first-generation college students. A 2014 analysis by Scott Freeman and his collaborators at the University of Washington looked at hundreds of studies and found that students in active-learning classrooms scored 6 percent higher on exams and were 1.5 times less likely to fail than students in traditional lecture courses. Other studies have found that active learning can shrink achievement gaps by as much as 45 percent. The mechanism explaining why this method of learning works especially well for some students remains elusive. Benefits may result from increased emphasis on

consistently spaced class preparation and frequent feedback that helps students recognize what they know and do not know. These findings show that properly structured college science courses can encourage underrepresented students to pursue science.

Institutions are increasingly recognizing the importance of providing teaching development opportunities for both graduate students and postdocs. Examples of national efforts include the NIH Institutional Research and Academic Career Development Awards fellowships, which are designed to provide postdocs with valuable training as well as teaching experiences in minority-serving institutions. The Center for the Integration of Research, Teaching, and Learning Network targets graduate students and postdocs as future faculty and engages them in campus and virtual-learning communities with opportunities to gain skills for academic careers.

Formal teaching postdoc programs are not common, but the Center for Teaching and Learning at Yale University, which I direct, recently established a teaching program with a unique twist. After a two-year period of training and mentored classroom teaching, the third year is devoted to teaching and related responsibilities at a regional partner institution. Strategic partnerships with a community college and a private institution serving underrepresented and first-generation students ensure that the postdocs gain practical skills by teaching in diverse classroom settings.

The science education literature has yet to report strong connections between professional development programs for faculty and undergraduate student learning outcomes. But a recent study shows that graduate students who engage in more than 55 hours of teaching development feel more confident in the classroom and have greater success securing a faculty position. Follow-up work will demonstrate the impact of teaching development for postdocs and, ultimately, the students in their classrooms.

Postdocs trained in evidence-based teaching are equipped to teach effectively and inclusively. Promoting meaningful engagement in learning benefits everyone, and the particular advantage conferred to underrepresented and first-generation college students is a critical factor for encouraging participation in science.

Jennifer Frederick *is executive director of the Center for Teaching and Learning at Yale University, where she oversees initiatives that promote teaching excellence and support student learning campus-wide. A chemist by training, Frederick also leads the Summer Institutes on Scientific Teaching, a national training program for college science faculty.*

dents may be entirely absent among their less privileged peers. Truly equal access to computer science education, Margolis tells me, is "a civil rights issue for the 21st century."

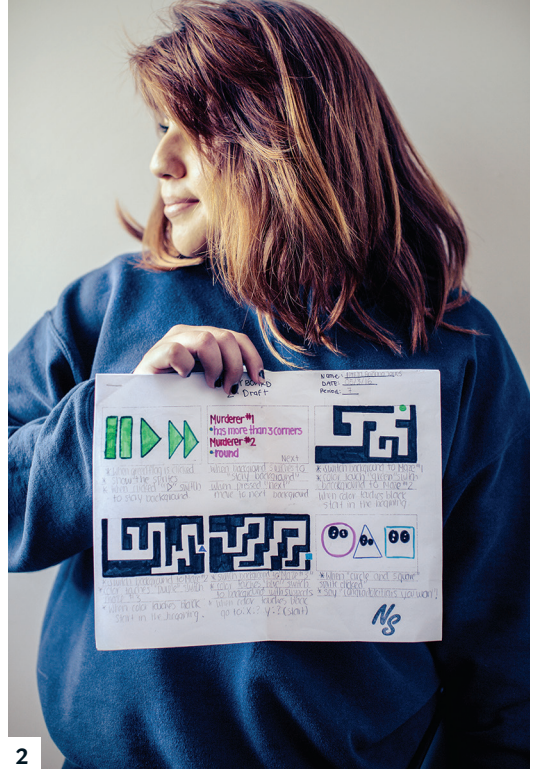
Having documented the very unequal situation on the ground, she says, "we felt we had to do something about it." She and her colleagues thus created Exploring Computer Science (ECS), a university/K–12 partnership program that has developed an introductory curriculum for high school computer science students. ECS also offers professional development to teachers in the Los Angeles Unified School District—and now to nearly two dozen other districts nationwide. Participating teachers learn how to provide hands-on computer science instruction that is rigorous yet engaging, and they are joined with other teachers in an active and growing network. Whereas other organizations, among them Black Girls Code and Hack the Hood, focus their efforts outside of school time, such as during weekend workshops and summer boot camps, Margolis and her team are committed to serving underprivileged students in their schools, aware that financial and logistic barriers often prevent such students from taking advantage of extracurricular opportunities.

Nancy Se shares this commitment. Trained by Exploring Computer Science in 2013, she has since become an ECS "teacher leader," helping other teachers learn how to engage their students in computer science. Se teaches at Augustus F. Hawkins High School in South Los Angeles, where, in the 2013–2014 school year, 99 percent of the student body was African-American or Latino and 75 percent qualified as "economically disadvantaged," according to *U.S. News & World Report*. Se sees the challenges faced by the coding-for-all movement up close every day. "Many of my students don't have computers in their homes," she says. "Their only access to the Internet is through their phones, which they use for games and texting." Given this limited acquaintance with technology, her students have a lot of ground to cover—learning not only about the manifold functions of computers but also the behind-the-curtain reality that *people* design and program these machines. Most daunting of all, Se's students must overcome stereotypes about whom computer science is "for." "Working with computers, knowing how to code—this is foreign to the identities they've developed, even as students still in high school," Se notes. "In my classes, the students and I aren't just exploring an academic subject. We're reinventing their sense of themselves in the face of very powerful cultural messages."

Paging through essays written by her 12th graders, Se points to one in which a student expresses hope for a steady income for her and her baby. "Before taking a class in computer science, it never crossed my mind that I could be involved in or even major in that field," this student wrote. "As a single mother who wants to create the best future possible for my child, I also think it would be great to have a job doing what I love and [one that] provides financial stability." Writes another student: "Economic struggles have been around my whole life. Nonetheless, I refuse to let limited resources bring me down. On the contrary, they motivate me to utilize education as a path and guide me to life-long success. I yearn to one day possess the title of [a computer] animator with an average salary of $50,281 per year. I aspire to bring my current financial struggles to an end and take care of my family on my own."

Indeed, one of the rationales often cited for expanding coding instruction, especially in poor communities, is that a lack of programming knowledge will shut young people out of a lucrative industry. But many critics argue that a narrow focus on a technical skill such as coding is not a sustainable approach. Larry Cuban, a professor emeritus of education at Stanford University, points to the nation's experiment with the early computer language Logo in the 1970s and 1980s. Led by Massachusetts Institute of Technology professor Seymour Papert, the effort to teach students Logo flared and failed, Cuban says, because it did not teach the complex skills and rich knowledge that they need. He predicts that the same fate will befall the coding-for-all movement, which he believes is being urged on schools by business leaders with their own bottom lines in

1



2

STUDENTS demonstrate an animation (*1*), storyboards (*2–3*) and artwork (*4*) for projects they are developing. Se says her students are doing more than programming: they are reinventing their sense of themselves.

3

mind: it is a "tissue paper reform" that, "after one or two uses, shreds and is tossed away."

If instruction in coding is not the answer—or at least not all of it—then what is? Jeannette M. Wing believes she knows: computational thinking. Wing is a consulting professor of computer science at Carnegie Mellon University and a corporate vice president of Microsoft Research. In 2006 she published an article in an obscure journal that quickly became a classic. "Computational thinking is a fundamental skill for everyone, not just for computer scientists," she boldly declared. This mode of thinking, she went on to explain, "involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science."

"Computational thinking includes the ability to abstract, to engage in logical and symbolic reasoning, to take a big problem and decompose it into many smaller problems," Wing says. "These are skills that everyone can use, whether they're using a computer or not."

In her article, she wrote, "To reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability." One of the educators who heard Wing's call was Aileen Owens. Owens used Wing's ideas to make computational thinking a key part of the curriculum at South Fayette Township School District, outside of Pittsburgh, where she is director of technology and innovation. In her view, instruction in computational thinking should begin early and grow progressively deeper and more complex through a series of graduated and interrelated projects.

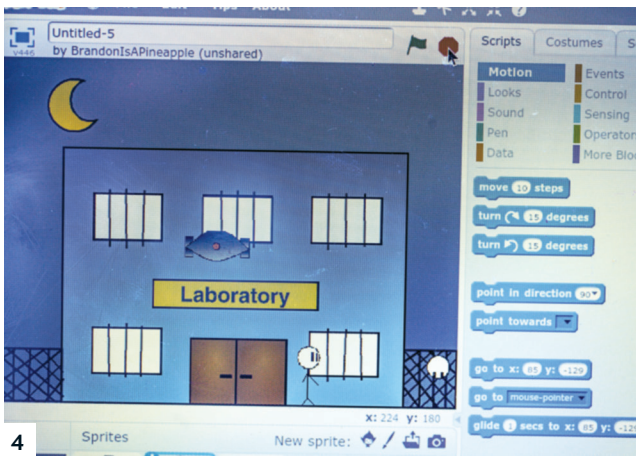
In South Fayette, students in kindergarten through second grade start learning the concepts behind programming by using what is known as a block-based coding program. In the program, called Scratch, they drag and drop blocks containing discrete commands: "Move 10 steps," "Wait 5 secs," "Turn left 90 degrees." By arranging these commands in a precise order, the students make things happen on their screens (a cartoon character advances, pauses, turns left) and begin to understand more generally how to give instructions to computers—instructions that become increasingly complex as they grow older. In grades three through five, students program motors and sensors and build Lego robots that move under their control. Computer-aided design (CAD) is taught in grades six through eight; students use CAD software to design their own inventions, and they prototype them by using a 3-D printer. By seventh grade, students have transitioned from block-based coding to text-based coding: writing code in the more complex but more flexible language employed by professional programmers.

"At every stage, the goal is to scaffold computational think-

ing so that each new level of understanding builds on the one that came before," Owens explains. "This is about much more than coding. This is about teaching habits of mind that can be used to solve problems in any realm—habits like breaking down a problem into its component parts, running small experiments to see which approaches fail and which succeed, and working together with other people to find and apply the best ideas." Using these strategies in a variety of settings shows students that computational thinking is useful well beyond the world of computers.

Even the very youngest children can learn such habits of mind. "We're teaching our kids to be problem solvers, to think logically, to engage in abstract thinking, to find patterns, to identify alternatives," says Melissa Unger, the science, technology, engineering, arts and mathematics (STEAM) teacher for kindergarten through second grade at South Fayette. "We start with questions like 'What are instructions? How do you give instructions so that a computer knows what you want it to do?' We have students 'program' their classmates, guiding them through a maze by holding up cards with arrows on them."

Like Los Altos, South Fayette is an affluent, highly educated community, home to professors from the area's many universities, as well as professionals who work in Pittsburgh's burgeoning tech sector. But computational thinking can be taught to all kinds of kids, as Excel Public Charter School in Kent, Wash., demonstrates. Excel enrolls a student body that is 37 percent African-American and 19 percent Latino; more than half its students qualify for free or reduced-price lunch. Although the school is located less than 20 miles from the Seattle area, home to tech giants Amazon and Microsoft, the students at Excel often feel cut off from the world of computers.



4

Eli Sheldon, the school's computational thinking program manager, is aiming to change that. "I work with teachers to incorporate computational thinking into the subjects they teach, whether it's English, or math, or biology," Sheldon says. "Because students are encountering the same mental tools across many different courses, they come to see how universally applicable they are, even outside of school." Sheldon has collaborated with a math teacher to create a class unit on analyzing motion data from a professional basketball game and with a humanities teacher to develop interactive case studies drawn from the criminal justice system.

Proponents of the teaching of computational thinking believe it represents all the qualities that mere coding instruction lacks: a rich and deep intellectual discipline; a flexible set of mental tools that can be used in many and varied situations; and a body of knowledge and skills of genuine and lasting usefulness—in school, in the workplace and beyond.

Before arriving at Excel, Sheldon worked for four years as a program manager at Microsoft, where he saw computational thinking put to real-world use. "Over and over, I saw engineers take an incredibly complex problem and solve it using computational thinking," Sheldon recounts. "They were really good at breaking down the problem, putting the parts in logical order, testing one part at a time to see how that one small change affected the outcome. I watched them, and I thought, 'Everyone should know how to do that.'"

The nation's approach toward computer science education is still taking shape. As the assistant director for learning and innovation at the White House's Office of Science and Technology Policy, Kumar Garg is one of those steering the effort. He can point to some successes: "When President Obama took office, only 11 states allowed computer science courses to count toward graduation," Garg observes. "Since then, there's been a sea change: [28 states and Washington, D.C.] now allow computer science courses to be applied to math or science requirements." Garg also praises a "reboot" of the AP computer science course now under way at the College Board. The new course and exam will focus on both coding *and* computational thinking, an approach Garg endorses. "Students do need to learn the fundamentals of computer science, but coding is a way of helping students see what it's all for," he explains.

Under its Computer Science for All initiative, Garg notes, the Obama administration has asked school districts to submit five-year plans for expanding access to computer science instruction; districts with well-designed proposals will be granted funding for implementation. Of course, some schools are not waiting on the slow-grinding gears of the federal government to get started on innovating in computing education, whereas others have yet to even begin tackling the subject—a situation that calls to mind an aphorism attributed to science-fiction writer William Gibson, whose work explores the interactions between humans and technology. "The future is already here," Gibson is said to have observed. "It's just not evenly distributed yet." SA

MORE TO EXPLORE

Computational Thinking. Jeannette M. Wing in *Communications of the Association for Computing Machinery,* Vol. 49, No. 3, pages 33–35; March 2006.

Stuck in the Shallow End: Education, Race and Computing. Jane Margolis. MIT Press, 2008.

Learn about President Barack Obama's Computer Science for All initiative to empower a generation of American students with the computer science skills they need to thrive in a digital economy: www.whitehouse.gov/blog/2016/01/30/computer-science-all

FROM OUR ARCHIVES

Machine Learning. Seth Fletcher; August 2013.

scientificamerican.com/magazine/sa