Society | DOI:10.1145/2557447 Esther Shein

Should *Everybody* Learn to Code?

Not everyone needs coding skills, but learning how to think like a programmer can be useful in many disciplines.

To gauge the ability of professional graphic designers to do basic programming, Brian Dorn, then a graduate student at the Georgia Institute of Technology (Georgia Tech), asked a group of them to read and modify a piece of program code. The idea was to see whether they could turn themselves into informal programmers and figure out how to develop automated functions in Adobe Photoshop. Unfortunately, when the designers conducted Web searches to look for information on the code they needed, they sometimes used results that pointed them in the wrong direction, which was toward Java—when they actually needed to be using JavaScript for this particular project.

One of the underlying causes could have been tied to the participants' "lack of sufficient general, abstract knowledge of the computing and/or programming structures at play," wrote Dorn in *Communications* in May 2011.

His advisor, Mark Guzdial, who relayed the story, said the findings indicate to him "that there are a lot of people who need knowledge of computer science ... who are going to use it in their lives, but because they never learned anything about computer science, they are teaching it [to] themselves and coding inefficiently, and wasting a lot of time and getting frustrated."

If someone is going to become a knowledge worker, or take on any job "that requires an undergraduate degree," they should know how to read a piece of code that is useful to them and be able to make changes to it, says Guzdial, a professor and director of Contextualized Support for Learning in the School of Interactive Computing at Georgia Tech.

People ranging from former President Bill Clinton to Facebook creator Mark Zuckerberg to physicist, cosmologist, and author Stephen Hawking have expressed the belief that basic computer programming is an essential skill in today's world. "Code has become the 4th literacy. Everyone needs to know how our digital world works, not just engineers," says Mark Surman, executive director of the Mozilla Foundation, whose comments are among those of dozens of luminaries on code.org.

Second-grade students in Kevin Jarrett's Elementary 'STEMLAB' at Northfield Community School (New Jersey) participate in the 2013 Hour of Code.

The demand for computer scientists and technical professionals in the U.S. is projected to grow 34% through 2018, according to the Bureau of Labor Statistics. Many people already engage in some level of programming; Guzdial cites a 2005 Carnegie Mellon University study indicating that in 2012 there would be 90 million workers in the U.S., more than 55 million of whom would use spreadsheets and databases, which can be deemed programming. The study also projected that more than 13 million would describe themselves as "programmers" in 2012, although only three million of them would be professional software developers.

The Carnegie Mellon study also noted that a lot of people were doing programming without realizing it, by creating macros for spreadsheets or doing database queries using SQL. "So the argument is, lots of people are going to do programming," says Guzdial, "and the data we have studying how end user programmers teach themselves and the types of mistakes they make suggest if they knew something about computer science, they might not have to struggle so much later."

Many people who avoided taking science and math courses in college are now struggling as they try to teach themselves how to program, he points out. "How many more would be doing some programming if we helped them? That is the interesting part."

PHOTOGRAPH BY KEVIN JARRETT

What You Should Learn

Everyone should learn computational thinking, maintains Jeannette Wing, corporate vice president at Microsoft Research. Computational thinking helps people learn how to think abstractly and pull apart a problem into smaller pieces. One concrete way to learn aspects of those skills is programming, Wing says.

That is not to say everyone needs to learn a specific programming language like Python or C++, even though many people identify programming with turning out code, Wing says. "First of all, that is too low-level, and it is also very narrow an interpretation of what I believe is more important."

Instead of teaching everyone to churn out code, the emphasis should be on learning problem-solving skills in computer science, much like the problem-solving skills one learns in math and engineering, says Wing, who is on leave as President's Professor of Computer Science at Carnegie Mellon. Writing a program is an explicit way of expressing a solution that a human or machine can carry out, she says. "The more fundamental skill and more critical thinking skill is what comes before you write down this piece of code, and that is computational thinking."

Guzdial agrees. "Should we learn enough so that you can write a script to do something that otherwise would have to be done by hand? I would like to see that, but I cannot make the argument that it is an absolutely necessity."

He adds that ignorance of computer science puts people at a disadvantage in today's world. "Not knowing anything about programming makes it more difficult to pick it up."

The Flip Side

The issue is far more black and white to software engineer Chase Felker, who wrote an article for *Slate* magazine entitled "Maybe Not Everybody Should Learn to Code." Felker writes, "Frankly, just the idea that you can learn to code in a year gives me the creeps: I would be terrified if someone with only a couple of classes were writing programs for me, not because he (of course, and unfortunately, most programmers are men) has learned anything wrong—but because of what he doesn't know."

Computational thinking helps people learn how to think abstractly and pull a problem apart into smaller pieces.

While noting that several of his colleagues are successful self-taught programmers, and that learning to program does not necessarily have to be done at a university, Felker says people need to know more than memorizing the technology du jour and, as Wing said, they need the critical ability to think things through.

"[I]f you aren't dreaming of becoming a programmer—and therefore planning to embark on a lengthy course of study, whether self-directed or formal—I can't endorse learning to code," Felker writes. "Yes, it is a creative endeavor. At its base, it's problem-solving, and the rewards for exposing holes in your thinking and discovering elegant solutions are awesome." He goes on to say he does not believe that most people who learn to code end up learning anything that stays with them.

Referencing a comment made by New York City Mayor Michael Bloomberg in 2012 that he would learn to code, programmer Jeff Atwood, writing in his blog "Code Horror," poses the question, "...can you explain to me how Michael Bloomberg would be better at his day-to-day job of leading the largest city in the USA if he woke up one morning as a crack Java coder?" While agreeing that programming is important, Atwood says many other skills are important, too. "I would no more urge everyone to learn programming than I would urge everyone to learn plumbing," he writes.

The so-called "everyone should learn to code" movement is wrong for several reasons, according to Atwood, including the assumption that more code in the world is an inherently desirable thing. That assumes code is the goal; it puts the method before the

ACM Member News

TECHNOLOGY, ALWAYS CHANGING, SHOULD BE EASY TO USE

The two guiding principles of Jeff Johnson's 35-year career in human-computer interaction (HCI) have been the constantly changing nature of technology, and that technology should be easy to use.

Johnson, president and principal consultant for product usability consultancy UI Wizards Inc., earned B.A. and Ph.D. degrees in psychology and computer science from Yale and Stanford universities, respectively. He has worked as a user-interface designer and implementer, engineer manager, usability tester, and researcher at Xerox, Hewlett-Packard Labs, and Sun Microsystems, and recently returned from a stint as a Visiting Erskine Fellow at the University of Canterbury, Christchurch, New Zealand. These experiences have solidified Johnson's commitment to continually refresh his work to keep pace with the latest digital advances. "There never will be a time when everyone is a digital native, because the definition of that term changes as digital technology progresses," he says.

Johnson, an ACM Distinguished Speaker, recently updated his 2009 HCI book *Designing with the Mind in Mind* with a new section on peripheral vision and chapters on decision-making and hand-eye coordination.

He also has a new corporate endeavor: Wiser Usability, a consultancy that helps companies design senior-friendly websites. "Older adults tend to have limited mobility and transportation, and they could benefit most from online shopping and online access to services," Johnson says. "No one who isn't a hardcore computer-geek, least of all seniors, wants to use technology for its own sake. Digital tools that don't help seniors accomplish their goals with a minimum of learning and bother are not worth the time and expense," he adds.

—Laura DiDio

problem, and assumes adding coders to the workforce is a net positive.

"The general populace (and its political leadership) could probably benefit most of all from a basic understanding of how computers, and the Internet, work,'' he says. "Being able to get around on the Internet is becoming a basic life skill, and we should be worried about fixing that first and most of all, before we start jumping all the way into code."

Guzdial speculates there may be pushback from programmers, because they think not everyone can be taught what they do. "I am not suggesting everyone produce thousands and thousands of lines of code. I would love if everyone could graduate from a university writing 10 lines of code that are useful to them."

The point of teaching programming in high school would be to give students some level of literacy relative to programming, including the ability to think about things in terms of code, and to understand what code can do, Guzdial adds.

The State of Computer Science in Public Education

That may not, however, occur in schools. Many in the computer science field say the U.S. is severely lagging in making even basic computer science a priority in K–12 schools. "While other countries have designed and implemented national computer science education programs in order to better prepare their students for the increasingly competitive global economy, the decentralized (state, district-wide, and even school-based) educational decision-making process in the U.S. has severely hampered efforts to standardize our computer science curriculum and create coherence in student learning," according to the 2010 report, "Addressing Core Equities in K–12 Computer Science Education."

Guzdial believes the biggest problem in teaching computer science in the U.S. is the lack of teachers who know the discipline. He estimates there are about 30,000 high schools in the country, but only 2,000 Advanced Placement computer science teachers.

Students as young as five can learn to program, Guzdial maintains, but he questions why they should. "I am not sure what we can teach them from a cognitive perspective." Although there have been studies done on children learning to program in the Scratch programming language, "in general what we find is kids that young do not do the things you naturally would expect coding to involve,'' including loops and conditionals.

Students as young as five can learn some basic concepts of programming, similar to the number and counting skills children typically are taught at that age.

He says he is concerned about cognitive development. "What we know about cognitive development is you typically develop the ability to do abstract reasoning around the age of 12," and programming is a very abstract activity. Guzdial is unsure whether young children who program develop abstract reasoning earlier, or if they are only able to learn a little bit of programming skills.

Overall, though, he says computer science should be taught in schools –but starting at age five or six, when only 12% of high schools in the U.S. offer computer science courses and far fewer middle and elementary schools, creating a great divide. Guzdial says, "They are unlikely to see it again for a dozen years, so why offer it at five or six?"

Wing also says that, while age five may be too early to teach how to code, students that young can learn some basic concepts similar to the number and counting skills children typically are taught at that age. As they get older, students should be taught other concepts, like what an algorithm is, ways to represent data, and different analysis techniques in order to understand and reason, she says.

Looking Ahead

Just as students are taught reading, writing, and the fundamentals of math and the sciences, computer science may one day become a standard part of a K–12 school curriculum. If that happens, there will be significant benefits, observers say. As the kinds of problems we will face in the future will continue to increase in complexity, the systems being built to deal with that complexity will require increasingly sophisticated computational thinking skills, such as abstraction, decomposition, and composition, says Wing.

"If I had a magic wand, we would have some programming in every science, mathematics, and arts class, maybe even in English classes, too,'' says Guzdial. "I definitely do not want to see computer science on the side ... I would have computer science in every high school available to students as one of their required science or mathematics classes." C

Further Reading

C. Simard, C. Stephenson, D. Kosaraju
"Addressing Core Equities in K–12 Computer Science Education: Identifying Barriers And Sharing Strategies," 2009. http://anitaborg.org/files/ABI-csta-full-report.pdf

C. Felker
"Maybe Not Everybody Should Learn to Code," 2013. http://www.slate.com/articles/technology/future_tense/2013/08/everybody_does_not_need_to_learn_to_code.html

Martyr2
"Why Everyone Should NOT Learn to Code," 2013. http://www.coderslexicon.com/why-everyone-should-not-learn-to-code/

D. Haggard
"Why Everyone Should Learn to Program," 2011. http://reviewsindepth.com/2011/04/why-everyone-should-learn-to-program/

P. Norvig
"Teach Yourself Programming in Ten Years," 2001. http://norvig.com/21-days.html

J. Lave
"Cognition in Practice: Mind, Mathematics, and Culture in Everyday Life," 1988, Cambridge University Press.

J.R. Hayes
"The Complete Problem Solver," 1989. Lawrence Erlbaum Associates, Inc.

Esther Shein is a freelance technology and business writer based in the Boston area.