The New York Times | https://nyti.ms/2nSxVfK

EDUCATION LIFE

Learning to Think Like a Computer

By LAURA PAPPANO APRIL 4, 2017

In "The Beauty and Joy of Computing," the course he helped conceive for nonmajors at the University of California, Berkeley, Daniel Garcia explains an all-important concept in computer science — abstraction — in terms of milkshakes.

"There is a reason when you go to the 'Joy of Cooking' and you want to make a strawberry milkshake, you don't look under 'strawberry milkshake,' " he said. Rather, there is a recipe for milkshakes that instructs you to add ice cream, milk and fruit of your choice. While earlier cookbooks may have had separate recipes for strawberry milkshakes, raspberry milkshakes and boysenberry milkshakes, eventually, he imagines, someone said, "Why don't we collapse that into one milkshake recipe?"

"The idea of abstraction," he said, "is to hide the details." It requires recognizing patterns and distilling complexity into a precise, clear summary. It's like the countdown to a space launch that runs through a checklist — life support, fuel, payload — in which each check represents perhaps 100 checks that have been performed.

Concealing layers of information makes it possible to get at the intersections of things, improving aspects of a complicated system without understanding and grappling with each part. Abstraction allows advances without redesigning from scratch.

It is a cool and useful idea that, along with other cool and useful computer science ideas, has people itching to know more. It's obvious that computers have become indispensable problem-solving partners, not to mention personal companions. But it's suddenly not enough to be a fluent user of software interfaces. Understanding what lies behind the computer's seeming magic now seems crucial. In particular, "computational thinking" is captivating educators, from kindergarten teachers to college professors, offering a new language and orientation to tackle problems in other areas of life.

This promise — as well as a job market hungry for coding — has fed enrollments in classes like the one at Berkeley, taken by 500 students a year. Since 2011, the number of computer science majors has more than doubled, according to the Computing Research Association. At Stanford, Princeton and Tufts, computer science is now the most popular major. More striking, though, is the appeal among nonmajors. Between 2005 and 2015, enrollment of nonmajors in introductory, mid- and upper-level computer science courses grew by 177 percent, 251 percent and 143 percent, respectively.

In the fall, the College Board introduced a new Advanced Placement course, Computer Science Principles, focused not on learning to code but on using code to solve problems. And WGBH, the PBS station in Boston, is using National Science Foundation money to help develop a program for 3- to 5-year-olds in which four cartoon monkeys get into scrapes and then "get out of the messes by applying computational thinking," said Marisa Wolsky, executive producer of children's media. "We see it as a groundbreaking curriculum that is not being done yet."

Computational thinking is not new. Seymour Papert, a pioneer in artificial intelligence and an M.I.T. professor, used the term in 1980 to envision how children could use computers to learn. But Jeannette M. Wing, in charge of basic research at Microsoft and former professor at Carnegie Mellon, gets credit for making it fashionable. In 2006, on the heels of the dot-com bust and plunging computer science enrollments, Dr. Wing wrote a trade journal piece, "Computational Thinking." It was intended as a salve for a struggling field.

"Things were so bad that some universities were thinking of closing down computer science departments," she recalled. Some now consider her article a manifesto for embracing a computing mind-set.

Like any big idea, there is disagreement about computational thinking — its broad usefulness as well as what fits in the circle. Skills typically include recognizing patterns and sequences, creating algorithms, devising tests for finding and fixing errors, reducing the general to the precise and expanding the precise to the general.

It requires reframing research, said Shriram Krishnamurthi, a computer science professor at Brown, so that "instead of formulating a question to a human being, I formulate a question to a data set." For example, instead of asking if the media is biased toward liberals, pose the question as: Are liberals identified as liberal in major newspapers more often or less often than conservatives are identified as conservative?

Dr. Krishnamurthi helped create "Introduction to Computation for the Humanities and Social Sciences" more than a decade ago because he wanted students "early in their undergrad careers to learn a new mode of thinking that they could take back to their discipline." Capped at 20 students, the course now has a waitlist of more than 100.

Just as Charles Darwin's theory of evolution is drafted to explain politics and business, Dr. Wing argued for broad use of computer ideas. And not just for work. Applying computational thinking, "we can improve the efficiencies of our daily lives," she said in an interview, "and make ourselves a little less stressed out."

Computing practices like reformulating tough problems into ones we know how to solve, seeing trade-offs between time and space, and pipelining (allowing the next action in line to begin before the first completes the sequence) have many applications, she said.

Consider the buffet line. "When you go to a lunch buffet, you see the forks and knives are the first station," she said. "I find that very annoying. They should be last. You shouldn't have to balance your plate while you have your fork and knife." Dr. Wing, who equates a child filling her backpack to caching (how computers retrieve and store information needed later), sees the buffet's inefficiency as a failure to apply logical thinking and sequencing.

Computational thinking, she said, can aid a basic task like planning a trip — breaking it into booking flights, hotels, car rental — or be used "for something as complicated as health care or policy decision-making." Identifying subproblems

and describing their relationship to the larger problem allows for targeted work. “Once you have well-defined interfaces,” she said, “you can ignore the complexity of the rest of the problem.”

Can computational thinking make us better at work and life? Dr. Krishnamurthi is sometimes seduced. “Before I go grocery shopping, I sort my list by aisles in the store,” he said. Sharing the list on the app Trello, his family can “bucket sort” items by aisle (pasta and oils, canned goods, then baking and spices), optimizing their path through Whole Foods. It limits backtracking and reduces spontaneous, “i.e., junk,” purchases, he said.

Despite his chosen field, Dr. Krishnamurthi worries about the current cultural tendency to view computer science knowledge as supreme, better than that gained in other fields. Right now, he said, “we are just overly intoxicated with computer science.”

It is certainly worth wondering if some applications of computational thinking are trivial, unnecessary or a Stepford Wife-like abdication of devilishly random judgment.

Alexander Torres, a senior majoring in English at Stanford, has noted how the campus’s proximity to Google has lured all but the rare student to computer science courses. He’s a holdout. But “I don’t see myself as having skills missing,” he said. In earning his degree he has practiced critical thinking, problem solving, analysis and making logical arguments. “When you are analyzing a Dickinson or Whitman or Melville, you have to unpack that language and synthesize it back.”

There is no reliable research showing that computing makes one more creative or more able to problem-solve. It won’t make you better at something unless that something is explicitly taught, said Mark Guzdial, a professor in the School of Interactive Computing at Georgia Tech who studies computing in education. “You can’t prove a negative,” he said, but in decades of research no one has found that skills automatically transfer.

Still, he added, for the same reasons people should understand biology, chemistry or physics, “it makes a lot of sense to understand computing in our lives.” Increasing numbers of people must program in their jobs, even if it’s just Microsoft Excel. “Solving problems with computers happens to all of us every

day," he said. How to make the skills available broadly is "an interesting challenge."

•

"It's like being a diplomat and learning Spanish; I feel like it's essential," said Greer Brigham, a Brown freshman who plans to major in political science. He's taking the course designed by Dr. Krishnamurthi, which this term is being taught by a graduate student in robotics named Stephen Brawner.

On a March morning at the Brown computer science center, Mr. Brawner projected a student's homework assignment on the screen. Did anyone notice a problem? Nary a humanities hand was raised. Finally, a young woman suggested "centimeters" and "kilograms" could be abbreviated. Fine, but not enough.

Mr. Brawner broke the silence and pointed out long lines of code reaching the far side of the screen. With a practiced flurry, he inserted backslashes and hit "return" repeatedly, which drew the symbols into a neat block. It may all be directions to a machine, but computer scientists care a great deal about visual elegance. As Mr. Brawner cut out repeated instructions, he shared that "whenever we define a constant, we want that at the top of our code." He then explained the new assignment: write a program to play "rock, paper, scissors" against a computer.

Mili Mitra, a junior majoring in public policy and economics who sat with a MacBook on her lap, would not have considered this class a year ago. But seeing group research projects always being handed off to someone with computing knowledge, she decided that she "didn't want to keep passing them along." She has learned to write basic code and fetch data sets through the internet to analyze things she's interested in, such as how geographic proximity shapes voting patterns in the United Nations General Assembly.

Despite finding interactions with a computer much like "explaining things to a toddler," Ms. Mitra credits the class for instilling the habit of "going step by step and building a solution." She admits to being an impatient learner: "I jump ahead. In C.S. you don't have a choice. If you miss a step, you mess up everything."

Just as children are drilled on the scientific method — turn observations into a hypothesis, design a control group, do an experiment to test your theory — the basics of working with computers is being cast as a teachable blueprint. One thing making this possible is that communicating with computers has become easier.

"Block" programming languages like Scratch, released by the M.I.T. Media Lab a decade ago, hide text strings that look like computer keys run amok. That makes coding look less scary. Instead of keyboard letters and symbols, you might select from a menu and drag a color-coded block that says "say ( ) for ( ) secs" or "play note ( ) for ( ) beats." The colors and shapes correspond to categories like "sound" or "motion"; the blocks can be fit together like stacked puzzle pieces to order instructions. Students use this to, say, design a game.

One need not be a digital Dr. Doolittle, fluent in hard-core programming languages like Java or Python, to code. Block languages cut out the need to memorize commands, which vary depending on the computer language, because the block "is read just the way you think about it," Dr. Garcia said. Students in his Berkeley course use the block language Snap! for assignments — he doesn't teach Python until the last two weeks, and then just so they can take higher-level courses. "We tell them, 'You already know how to program,' " he said, because the steps are the same.

Computer Science A, which teaches Java, is the fastest-growing Advanced Placement course. (The number of students taking the exam in 2016 rose 18 percent over 2015 and nearly tripled in a decade.) But professors complained that "Java was not the right way" to attract a diverse group of students, said Trevor Packer, head of the A.P. program, so a new course was developed.

The course, Computer Science Principles, is modeled on college versions for nonmajors. It lets teachers pick any coding language and has a gentler vibe. There is an exam, but students also submit projects "more similar to a studio art portfolio," Mr. Packer said. The course covers working with data and understanding the internet and cyber security, and it teaches "transferable skills," he said, like formulating precise questions. That's a departure from what the College Board found in many high schools: "They were learning how to keyboard, how to use Microsoft applications." The goal is that the new course will be offered in every high school in the country.

President Obama's "Computer Science for All" initiative, officially launched last year, resulted in educators, lawmakers and computer science advocates spreading the gospel of coding. It also nudged more states to count computer science toward high school graduation requirements. Thirty-two states and the District of Columbia now do, up from 12 in 2013, according to Code.org. It's what Dr. Wing had hoped for when she advocated in her 2006 article that, along with reading, writing and arithmetic "we should add computational thinking to every child's analytical ability."

•

In an airy kindergarten classroom at Eliot-Pearson Children's School, in the Tufts University Department of Child Study and Human Development, children program with actual blocks. Marina Umaschi Bers, a child development and computer science professor, created wooden blocks that bear bar codes with instructions such as "forward," "spin" and "shake" that are used to program robots — small, wheeled carts with built-in scanners — by sequencing the blocks, then scanning them. Each "program" starts with a green "begin" block and finishes with a red "end."

Coding for the youngest students has become the trendy pedagogy, with plentiful toys and apps like Dr. Bers's blocks. Dr. Bers, who with M.I.T. collaborators developed the block language ScratchJr, is evangelical about coding. Learning the language of machines, she said, is as basic as writing is to being proficient in a foreign language. "You are able to write a love poem, you are able to write a birthday card, you are able to use language in many expressive ways," she said. "You are not just reading; you are producing."

Peer-reviewed studies by Dr. Bers show that after programming the robots, youngsters are better at sequencing picture stories. Anecdotally, she said, when they ask children to list steps for brushing teeth, they get just a few, "but after being exposed to this work, they'll have 15 or 20 steps."

Dr. Bers embeds computing in activities familiar to young children like inventing stories, doing dances and making art. At the Tufts school on a recent morning, children puzzled over a question: How does a robot celebrate spring?

"He's going to dance, and then he will pretend that he is wet," offered Hallel Cohen-Goldberg, a kindergartner with a mane of curls.

Solina Gonzalez, coloring a brown, blue and red circle with markers, peered soberly through pink-framed glasses: “He just does a lollipop dance.” Solina’s partner, Oisin Stephens, fretted about the root beer lollipop drawing she had taped to a block. “The robot won’t be able to read this,” he said. (It’s an invalid input.)

As they lurched around the carpet on their knees, the children executed computer science concepts like breaking instructions into sequenced commands, testing and debugging. One team used “repeat” and “stop repeat” blocks, forming a programming “loop,” a sequence of instructions that is continually repeated until a certain condition is reached.

It may be a leap to think of talking to machines as child’s play. But when Dr. Bers chats with students as they sequence blocks, they hardly notice they are coding. “They say, ‘Well, I’m not programming,’ ” she said.

Consciously or not, this next generation may be effortlessly absorbing the computational thinking skills so many covet, and in the process refining what we mean by digital native.

Laura Pappano is author of “Inside School Turnarounds: Urgent Hopes, Unfolding Stories.”

A version of this article appears in print on April 9, 2017, on Page ED18 of Education Life with the headline: Thinking In Code.
