

LEARNING TO THINK LIKE A COMPUTER

“Confusion is OK!” reads a sign in Matt Brenner’s computer science classroom. “*Respect it.*”

Brenner has a healthy respect for confusion, whether it’s the “cognitive overload” some students experience during the early weeks of his software development courses, or the more advanced forms common among expert programmers. “When you’re designing a program,” Brenner explains, “it almost never works the first time—or the second or the third. So you go back and debug it.” Programming, he adds, is “really all about debugging, about respecting confusion as a necessary part of the process.”

There’s one area of computer science, however, where Brenner brooks no confusion: its relevance to the lives of his students. “Computers are now so small and inexpensive that they pervade—even invade—nearly every aspect of our lives,” he says. To become “productive and responsible citizens,” he adds, “students must not only know how to use computers, they must also understand the nature of computers and the software that controls them.”

In 2005, as part of its Curriculum Review process, Exeter introduced a one-term computer science diploma requirement for all four-year students, based on a proposal written by Brenner and strongly endorsed by the science department. To fulfill that requirement, students have their choice of two introductory software development courses, “Programming and Public Policy” or “Essential Programming.” In both classes, says Brenner, students “learn to think about problems in the way that computers solve them: *algorithmically*,” and how to express those solutions using Java, the computer programming language.

So learning to program is like learning another language?

“No,” says Brenner firmly. “It’s like learning an entirely *different* language.” While language students may be beginners when it comes to Spanish or Chinese, they are, Brenner points out, already completely fluent in their own language. “But 80 to 90 percent of the kids who come into computer science class have never written a computer program,” he notes. “It’s pretty unusual for high school kids to have no exposure whatsoever to a subject.”

Andrew Weinstein ’09, a day student from Hampton, NH, was one of those kids. He had no experience and little interest in programming before enrolling in “Essential Programming,” which he did “mostly just to fill the requirement.” But after just a few classes, he was hooked.

“Computers are very fast, powerful machines,” he says. “They can perform a lot of calculations very fast, with no errors, if they are programmed correctly—but they cannot think. Every computer program has to be written by humans. And I like making new programs—the actual process of writing the code and testing it to see if it works.” Both “Essential Programming” and “Programming and Public Policy” begin largely offline, using a text Brenner has developed that introduces students to fundamental programming concepts and notation.

Next students embark on a half-dozen programming projects in which they learn to create output and input, and to write a series of simple computer games. “Some kids’ brains are just wired to pick this up quickly,” says Brenner. For others, the first few weeks are spent stumbling around in the dark, before the light bulb goes off—which is just fine by Brenner. “It’s great to come to class with a working program,” he says. “The next best thing is to come with questions.”

Weinstein recalls one such class, when he was struggling to debug a particular program. “Mr. Brenner always told us to try to be ‘as dumb as a computer’ when we were programming,” he says. “He said that once we started thinking like humans, we would make mistakes. I forced myself to do exactly what a computer would do and nothing more, and I almost immediately realized the mistake I had made.”

These are just the kinds of lessons Brenner hopes his students carry away with them. “In the process of learning to create computer software, students learn how to use the breathtaking speed of computers to amplify their own abilities and creativity to solve practical problems that cannot be solved by humans acting without them. They also develop a sense of what kinds of problems do *not* yield to the brute force of a computer performing billions of instructions each second.”

—Beth Brosnan



RACHEL BLEUSTEIN '09

As part of Exeter’s new one-term computer science requirement, students in Matt Brenner’s software development courses “learn to think about problems in the way that computers solve them,” says Brenner (at rear of table, third from left). They also learn what kinds of problems computers can and cannot solve.