

As a teacher of computer science, I strive to teach the fundamental concepts pertaining to the subject at hand to all, help develop a deeper understanding of the subject in most of the students, and to encourage a number of students to think creatively. My teaching experience is broad in two ways: (i) I have taught a variety of subjects as a teaching assistant in the Computer Science Department at the University of Iowa ranging from the first Java programming class to Automata Theory, and (ii) I have been involved in teaching/mentoring responsibilities over a broad range of students from freshman undergraduates to graduate students over the course of five years. Often, my interaction with research colleagues involves communicating ideas during group meetings and seminar settings similar to that of a classroom. Over the last few years, I have developed a teaching philosophy containing the following key ingredients: friendly and approachable demeanor, enthusiasm during presentation, and communicating abstract ideas preceded by intuitive and easy examples. In the following, I expound on these elements.

Perhaps the most challenging and important aspect in a student-teacher dynamic is the degree of approachability and openness of the teacher. I believe that the most serious impediment to learning is the intimidation factor that students are susceptible to. In such a state, it is difficult for the teacher to establish whether students are adequately absorbing concepts being taught. On the contrary, an open environment where students feel that it is easy to ask questions gives the teacher a chance to adapt based on continuous student feedback. Hence, I especially strive to create a friendly environment to ensure that students find me approachable during the first few days of class.

The presence of a friendly environment is often inadequate for learning. Without a sufficient level of interaction and engagement, the students might miss the importance of a seemingly banal concept. Therefore, it is necessary for teachers to integrate discussion and interaction into their presentation. I try to quickly learn the names of students to help me interact with them personally. I often turn to the class during presentation of material and ask them questions about the subject to create this kind of interaction. Also, I have found it useful to give students a chance to guide me towards a solution based upon a collective sense of intuition. This creates enthusiasm and energy in the classroom and students tend to take more interest in the material.

Algorithmic and programming techniques are a diverse tool-set. It is important to have developed a sharp intuition in order to skillfully apply the tools. Practicing their use is an extremely important way of developing this intuition. I, therefore, emphasize the importance of practice throughout the semester. Working through easy practice problems first helps build confidence and provides necessary preparation to tackle harder problems later. This requires that the problems be carefully selected. So, I spend a lot of my preparation time toward creating appropriate problem sets. I also believe that homeworks and quizzes are not just for the purpose of evaluation but also provide a chance to communicate with students.

I am a strong proponent of having examples precede details and concepts¹. Different levels of abstraction of a concept is fundamental in computer science. Typically, any piece of working computer code, say in Java, will have a high level pseudo-code that describes it. This high level description will have a description of the algorithm in some natural language layered above the pseudo-code, and so on. In my experience, students benefit enormously if any algorithm in its lowest level of detail is preceded by an example that helps highlight and develop some intuition. For example, when teaching students about how to implement the “bubble sort” algorithm to sort a given array, I first present the principle pictorially. Next, I point out to the students the invariant that once a lightest element has “bubbled” to the top, this partitions the array into a sorted part and an unsorted part and now all that remains to be done is to sort the unsorted part. Following this step, I encode the natural language description into pseudo-code. By this time, most students who have some basic knowledge of Java can implement the “bubble sort” algorithm.

I feel that teaching is an exciting and challenging endeavor. Each class or a group of students brings with it a new set of challenges. Overcoming these challenges is what makes teaching exciting. Observing my students develop intellectually gives me a great sense of accomplishment and pride.

¹<http://gowers.wordpress.com/2007/10/19/my-favourite-pedagogical-principle-examples-first/>