# Teaching Statement

John Lalor

## Teaching Philosophy

It is important for students to have both strong fundamentals and programming skills to be successful as computer science students. There is more to studying Computer Science than simply learning how to write code. An understanding of computer science concepts, theory, and design is important so that students have the necessary fundamentals to be able to write code effectively. I am a self-taught programmer, and it wasn't until I took courses during my Masters in CS theory and software design that I was able to really improve myself as a programmer.

A principle of my teaching philosophy is that instruction should be informed by the level of the students in the course I am teaching. For introductory courses, it is important that students are taught fundamental concepts alongside practical experience in a language like Python. This way students can implement what they learn quickly and can see how certain decisions impact code output. An important consideration for me is that students starting out in CS should feel like they can succeed from the start. For freshmen or students new to CS, fostering a welcoming environment where the students can experiment, make mistakes, and learn from them is critical for students to gain confidence and improve their skills. Too often students get frustrated with their progress or think that they can't cut it in CS because of a tough class or bad assignment. Research we did shows that the introductory Computer Science course (CS1) can have a significant impact on a student's perceptions of himself or herself as a programmer [1]. The CS1 course should be a course where students are exposed to the fundamentals of programming while being able to experiment with hands-on coding examples so that they can put what they learn into practice.

As students progress through their CS courses, more is expected from them in terms of both conceptual understanding and technical competency. Students should be able to study and understand key concepts behind certain algorithms, designs, etc. as well as be able to put that understanding into practice through programming exercises. Upperclass undergraduates should have a web presence that they maintain with course projects, open source contributions, and other examples of what they have learned during their degree so that they have ample evidence to help them in their job search to begin their careers.

**Previous experience** I have had experience with students as a tutor, a TA, and an instructor. I was a tutor as an M.S. student at DePaul University, working with both undergraduates and fellow M.S. students on assignments. My focus was on helping students in classes where Python was the primary coding language (such as the Intro to Programming course) as well as students in the Web Development courses who needed help with HTML, Javascript, and CSS. The student population at DePaul had a lot of variety, and I worked with traditional undergrads and masters students as well as students who had returned to school from work to learn new skills for a career change (a group to which I belong). I also worked with students who were working on their degree remotely via Skype and Google Hangouts. My techniques as a tutor were to ask questions and guide the students to answer questions for themselves by pointing to references from their coursework, such

as notes or class slides, that would provide guidance for the student. I encouraged students to experiment and to try different approaches. If the code threw an error or didn't work as expected, the students could see that it wasn't a major problem, but an opportunity to learn and see what went wrong in order to identify the problem and work on a fix.

While working on my Ph.D. at UMass, I was a TA for the Introduction to Programming course at Amherst College. There I again noticed that many students were hesitant to try things because they were worried about their code failing. I showed these students that error messages can be very useful for identifying bugs in their code. I stressed that if code was written after thoughtful consideration of what the expected output was, error messages would help, not hurt them. However, writing code carelessly and without prior planning would lead to error messages that are not informative, because anything number of things could be wrong.

This year I taught a first year seminar on Artificial Intelligence in Healthcare as part of the UMass First Year Seminar program (FYS). The FYS program was started to give incoming students a seminar-style course with a small class size and also to assist with the transition to life at the university. I designed the curriculum myself to give the first year students a high-level overview of artificial intelligence algorithms, in particular machine learning models, through the lens of healthcare. We studied how different machine learning models have been implemented in healthcare scenarios and discussed the implications of trusting models such as these with patient care decisions. The students completed homework assignments where they were asked to reflect on the challenges of such algorithms and whether they were comfortable or not with having an AI make medical decisions about their care. The course also incorporated university-wide content for all first year students such as time management skills, availability of resources on campus, and other topics to help students transition into college life. For example, for one assignment I asked the students to sign up for and attend the office hours of one of their professors, so that they were comfortable reaching out to their professors to discuss coursework, homework assignments, or research interests. For another assignment, I had the students sign up for accounts on GitHub and complete an introductory tutorial on version control software. Version control is something that is generally expected in software development but often overlooked in teaching. With this assignment the first year students are able to get familiar with version control and with GitHub so that they can use both for future projects, and start developing a web presence of code that they store on the site.

**Teaching interests** I am comfortable teaching introductory courses for incoming CS students, either using Python or in Java as the language for class assignments, as well as introductory courses in artificial intelligence, machine learning, and natural language processing. I would also be interested in teaching advanced machine learning and natural language processing courses, as well as seminars such as a more in depth version of the AI for Healthcare course that I designed this year.

## Diversity and inclusion

I am committed to fostering diversity and inclusion in the classroom and in computer science. During my M.S. degree at DePaul I worked on a research project that studied the impact of community in underrepresented groups in CS through linked-course learning communities [2, 3]. The learning communities for men of color and women in computing had positive impacts on the students involved. Students in the learning communities felt like they belonged to a community of programmers and had more positive feelings about the support available to them. We also looked at the impact of an introductory computer science course (CS1) and found that the first course a

student takes can have a significant impact on what his or her perception of computing is moving forward [1]. In my classroom I will continue to push for inclusion and making sure that students feel like they belong. I will encourage students to make use of the available resources for them to reach out for help if they need it, both with regards to help in the classroom and help with feeling like a member of the larger computer science community.

## Advising

I look forward to building a lab and advising students in their research. I have had the opportunity to advise a number of students during my Ph.D. at UMass. I was an advisor for the Industry Mentorship Independent Study course at UMass, where groups of M.S. students were paired with industry researchers and a Ph.D. student advisor to work on a research project. I advised a group of 5 M.S. students on their research project, offering them assistance on technical details, implementation questions, and general directions for the project. I was able to work with the students, provide feedback on proposed solutions, and evaluate their progress over the course of the semester. I have also advised both undergraduates and junior graduate students on research projects here at UMass.

As an advisor for future graduate students, it is important to advise students where they are. Instead of a rigid structure applied to all students, each student should be advised based on the best way to help him or her succeed. Certain students, particularly new students, may require regular interaction as they are getting familiar with the lab's work, gaining experience with current research tools and methods, and developing the ability to ask interesting research questions and propose experiments. I should be available to these students as they go through this process to provide guidance. On the other hand, if there is a student who works well without much supervision, I will respect that and provide that student with the space needed to do high quality work, while making sure that there is progress via one-on-one meetings, group meeting presentations, etc. As students gain more experience and become more senior in the lab, I will encourage them to mentor more junior students in the lab, and to gain advising experience by taking on M.S. students and undergrads for projects as well.

I think it is important for students in my lab to be part of the larger research community. Of course, participation via publications is important, but also involvement in terms of giving talks as part of workshops or invited talks at other universities, industry internships, and participation in opportunities in the larger department. The ultimate goal of a Ph.D. student is to graduate with a mastery of his or her topic, significant training in Computer Science and research methods, and a path for future success, either in industry or academia. I will pursue these goals in my lab with my students so that they leave the group and enter their professional career prepared to succeed.

## Teaching in the wider community

As I mentioned above, being a part of the larger research community is key for my future students. It is important for me as well. Giving talks and tutorials and organizing workshops for my professional colleagues is a great way to introduce them to what my lab is working on, encourage collaboration between groups, and also to learn best practices with regards to current trends and methods. Regular participation in the larger research community is key for maintaining relationships and potential collaborations. I plan on organizing workshops at the major conferences in my research area and giving talks at other universities. I currently maintain a blog where I periodically post on my ongoing research as well as post tutorials about key concepts in natural language processing.

Informal blog posts and tutorials are a good way to disseminate my work to those who may not attend conferences or read research papers but are interested in applying my work to their problems.

## References


[1] Amber Settle, John Lalor, and Theresa Steinbach. Reconsidering the impact of cs1 on novice attitudes. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, pages 229–234. ACM, 2015.

[2] Amber Settle, John Lalor, and Theresa Steinbach. A computer science linked-courses learning community. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*, pages 123–128. ACM, 2015.

[3] Amber Settle, John Lalor, and Theresa Steinbach. Evaluating a linked-courses learning community for development majors. In *Proceedings of the 16th Annual Conference on Information Technology Education*, pages 127–132. ACM, 2015.