

### Teaching Philosophy

My teaching philosophy is rooted in the belief that Computer Science is not only a discipline of algorithms and abstraction, but also a way of reasoning about and shaping the world responsibly. As computing becomes increasingly embedded in everyday life, and as large language models reshape how we write, build, and reason about software, students must learn not only how to build technical systems, but also how to evaluate whether those systems are reliable, interpretable, and worthy of trust. This conviction mirrors my research on provenance systems: just as my work asks how we verify the authenticity of physical and digital artifacts, **my teaching asks how we verify the authenticity of understanding**. My goal as an educator is therefore to help students connect **foundational concepts, real-world systems, and societal impact**, equipping them to be thoughtful creators who can design and analyze computing systems with both technical rigor and responsibility.

To achieve this, I emphasize three principles. First, I make abstract concepts concrete by connecting them to intuition and real systems. Second, I foster independence through scaffolded, project-based learning that gradually shifts ownership from instructor to student. Third, I cultivate a learning environment designed for depth: one where students from diverse backgrounds feel comfortable asking questions and experimenting, and where the presence of AI tools becomes an opportunity to strengthen, rather than shortcut, genuine understanding.

### Teaching and Mentoring Approach

I believe students learn most deeply when they actively engage with ideas rather than passively receive them. In my teaching, I therefore emphasize active inquiry, structured problem solving, and practical context.

As a Graduate Tutor for Programming Methodology (CS1101S) at the National University of Singapore (NUS), I worked closely with students who often found topics such as recursion and abstraction difficult because of their apparent lack of real-world grounding. To address this, I used scaffolding to break complex ideas into intuitive intermediate steps, helping students reason conceptually before implementing solutions in code. This approach improved not only technical understanding, but also confidence in tackling unfamiliar problems.

In software engineering courses such as CS3203, I treated the classroom as an engineering environment rather than only a lecture space. I guided student teams through the software development lifecycle under realistic constraints, including evolving requirements, integration challenges, and collaborative design trade-offs. My emphasis was not merely on producing a working system, but on helping students justify design decisions and develop habits of careful engineering judgment. Across these teaching roles, my evaluations have **consistently been rated above the departmental average**.

My mentoring philosophy extends these same ideas into research. I aim to provide enough guidance for students to make progress while giving them enough ownership to develop independence and curiosity. I have mentored undergraduate and master's students on projects in mobile sensing, computer vision, and trustworthy systems, encouraging them not only to implement methods but also to ask why a system

should work, what assumptions it relies on, and how it might fail in practice. Several of my mentees have co-authored papers with me and presented at top-tier venues. As a project advisor for NUS Orbital, I also mentored student teams on project scoping, milestones, and technical decision-making, for which I received the **Best Advisor Award**.

## ■■■■■ A Learning Environment for Deep Understanding

Large language models can now generate code, summarize concepts, and solve textbook problems on demand. This reality makes it more important, not less, to build a classroom where students develop genuine understanding. My goal is to create an environment in which students learn the most, not the fastest. To achieve this, I design activities that target what LLMs cannot replace: **the ability to formulate problems precisely, reason about why a solution works, and evaluate whether an output is trustworthy**. For instance, I use assignments where students must critique, debug, or explain AI-generated code rather than produce code from scratch, shifting the focus from syntax fluency to conceptual mastery. I also incorporate process-oriented assessments such as design justifications and iterative reflections, which reward reasoning over results. At the same time, I make expectations explicit, normalize productive struggle, and use examples drawn from diverse applications, including public health, food safety, digital media, and trustworthy AI, to show that computing is relevant to a wide range of lived experiences and societal needs. Rather than banning AI tools, I teach students to engage with them critically: to recognize their limitations, verify their outputs, and understand the foundational principles that make such evaluation possible. My aim is a classroom where every student builds both the competence to work alongside AI and the confidence to think independently of it.

## ■■■■■ Teaching Interests

I am prepared to teach core undergraduate courses such as Introduction to Computer Science, Software Engineering, Database Systems, and Computer Graphics, drawing on my experience in CS1101S, CS3203, CS2102, CS3241, and CS4247. I am also well positioned to teach advanced courses related to my research, including Computer Vision, Mobile Computing and Trustworthy AI.

In the longer term, I would be excited to develop interdisciplinary courses that connect foundational computing to real-world trust and sensing. For example, I envision an advanced seminar on *Trustworthy Sensing and Provenance*, where students study how systems recover and verify evidence about the physical and digital world while gaining hands-on experience building and evaluating such systems.

Ultimately, I aim to help students develop both **technical rigor** and a strong sense of **responsibility** in the systems they build. I would be excited to bring this research-led, student-centered approach to teaching and mentorship in my future faculty role.

## ■■■■■ Selected Teaching Experience

- **Graduate Tutor, National University of Singapore (Aug 2020 – Aug 2025)**
  - **CS3203 Software Engineering Project:** AY2020–2021 S1, AY2020–2021 S2, AY2021–2022 S2
  - **CS1101S Programming Methodology:** AY2022–2023 S1, AY2023–2024 S1, AY2024–2025 S1
  - **CS2102 Database Systems:** AY2022–2023 S2, AY2023–2024 S2
- **Teaching Assistant, National University of Singapore (Aug 2019 – May 2020)**
  - **CS3241 Computer Graphics:** AY2019–2020 S1
  - **CS4247 Graphics Rendering Techniques:** AY2019–2020 S2
- **Project Advisor, NUS Orbital (May 2020 – Aug 2020)**