TEACHING PHILOSOPHY

Years of immersion in university studies and work have endowed me with invaluable insights into effective learning and teaching. Merging my interpersonal skills with academic experience, I've laid the foundations of my teaching philosophy. In essence, my purpose is to "guide students into critical thinking, paying attention to details, and finding innovative ways to solve problems either independently or in collaboration with academic integrity, this is done by ensuring the diversity in personalities, culture, and disability." In the realm of instructing a Python programming course tailored for Computer and Electrical Engineering students, my teaching philosophy is rooted in the belief that fostering a dynamic, hands-on, and inclusive learning environment is essential. It goes beyond merely grasping the intricacies of Data Structures, Java, and Python programming; it aims to cultivate problem-solving skills and an enduring passion for coding. My guidance for students centers on several key aspects as listed below:

**Enthusiasm.** Maintaining enthusiasm for the course is paramount, serving as the greatest motivator for active learning and ensuring student success. Defining the goals and real-world applications of the course is crucial in maintaining students' enthusiasm, promoting excellence, and making the learning and teaching experience enjoyable.

Example: First, In CIS151, where students tackle problems in code, encouraging more profound thinking can be challenging for those new to programming. We addressed this by highlighting real-life applications of the course. For instance, we illustrated how Python can be used to automate the data analytics process, allowing students to analyze large complex business data.

Second, In CI351, students are equipped with advanced programming skills and critical thinking skills, this is an intensive courses that require more time and grasping challenging computer science concepts such as object oriented programming, data structures, to encourage enthusiasm in this code, student use Java programming which is mostly used by industry in the software development, hearing all these applications students get excited to be to gain practical skills to be able to develop and implement useful software applications or products .

**Differentiation and Diversity.** Recognizing diverse cultures, academic backgrounds, knowledge levels, and learning styles is critical when teaching and interacting with students. Various teaching methods, such as demonstration, interactive, project-based, and problem-based styles, should be employed for a diverse classroom environment.

Example: During lab sessions where mostly are writing code to solve real-world examples, some students may have a programming background, leading to potential disengagement if hands-on exercises are too easy. To address this, incorporating one or two optional difficult problems increases participation, as students seek support from the teaching assistant supervising the lab.

Considering cultural differences, my experiences as an international student highlight the need for active learning environments and personal interaction to overcome barriers. When, I am leading the lab, I walk around the lab and if there are know questions directed towards me, I go to each person and ask them the progress on the task, and most of the time students who are shy to ask in the large group, they would ask for help or discussion, this way there is no pressure on the student to raise hands or speak louder to get my attention.

**Effective Communication.** Communication is the key to fostering student success. Utilizing both in-person and online tools facilitates effective communication, ensuring that important announcements are consistently disseminated and taken seriously by students.

Example: In instances of changes, such as office hours, I communicate these during the lab and post announcements on the blackboard ultra, accompanied by emails to ensure that every student is informed, and I emphasize important information such as course policies during the beginning of each lab session.

**Flexible Teaching Management**: Teaching is not about forcing students to learn but about removing obstacles and fostering positive mindsets on their independent learning paths. Providing general instructions and allowing students more freedom while offering guidance and suggestions when needed is crucial. Each student is held accountable for their learning.

Example: During CIS151 and CIS351 lab sessions, I clarify trick questions and occasionally use examples. However, I encourage students to think and devise logical solutions independently. Lab instructions are crafted to support both collaborative and independent work while homework’s and exams are for independent work, with assistance provided in the thinking process, allowing students to develop solutions on their own.

**Continuous Feedback and Iterative Improvement.** Recognizing diverse academic backgrounds, I provide continuous, constructive feedback, allowing for individualized support and acknowledging the varied starting points of each student. Encouraging consistent programming practice, sharing additional learning tools, and providing opportunities for revision embracing the uniqueness of each student's learning journey.

Example: By embracing continuous feedback, I seek to enhance teaching quality. Adopting a peer-to-peer paradigm, students respond to questions and pose queries, allowing me to gain an intuitive understanding of their proficiency and identify potential areas for improvement.

**Assessment Based on Feedback**. The process of seeking and obtaining feedback from students proves more effective in enhancing teaching quality compared to solely lecturing. In my feedback-based assessment approach, I adopt a peer-to-peer paradigm where students respond to my questions and pose queries for me to address. This method allows me to gain an intuitive understanding of students' proficiency in the subject matter and aids in identifying potential areas of improvement in my teaching.

Example: In my interactions with students, I adhere to two processes. Initially, I inquire about their implementation, its objectives, and the underlying logic. Subsequently, they enumerate any existing technical challenges and engage in discussions with me. This approach enables us to promptly pinpoint critical issues and devise appropriate solutions.

**More Hands-On Activities.** Combining theory with practice efficiently facilitates knowledge acquisition, especially for computer and electrical engineering—a practice-oriented discipline. Striking a balance between providing theoretical and practical aspects of the course is crucial.

Example: As a teaching assistant for a programming course, I read, review, and solve labs in advance. If I encounter something tricky, I provide a brief review before students start working on it during in-person sessions. Additionally, I offer extra tools, such as Python, and Java documentation links, as references, and encourage students to read and practice in their study time.

**Discipline and attention**: are essential in both the classroom and lab sessions to ensure students are engaged and learning without distractions, such as noise or inattentiveness. For example, in the CIS 351 course (Spring 2025), before starting the lab session, I ask students who have completed the lab in advance to answer questions about methods, topics, or concepts covered in the class. This not only encourages active participation but also provides an opportunity to review key material before diving into the lab. In all programming courses, I maintain discipline by walking around the classroom to monitor student progress on labs or homework. This strategy helps keep students focused and engaged, ensuring they remain alert and involved throughout the session.