WeTeach_CS Curriculum Instructional Goals and Techniques

For either **WeTeach_CS** curriculum package, **CS For HS** or **APCSA**, the instructional goals are obvious: expose students to all learning elements and activities giving them the full opportunity to master the computer science concepts, practices and coding skills necessary to provide a successful learning experience with these important courses.

The instructional techniques recommended for either course are to follow the lesson plans set out for each unit, first providing students with access to the learning elements, including E-Lessons, PDF documents, videos, practice exercises in printed form, Google forms, or interactive experiences as provided in the E-Lessons and optional Codio learning environment.

Both courses have been designed to work effectively in a flipped classroom situation, giving students the opportunity to learn independently if necessary. However, it is highly recommended that a teacher be actively involved as both courses are quite challenging and students will need help over some of the more difficult aspects of the course.

Once students have experienced the lessons and watched the videos, they need to work through practice exercises provided in both courses, working out the new learning elements and practicing the knowledge and skills. Then they need to work through the labs provided for a logical and progressive sequence of coding skills acquired and mastered in a comprehensive way, each unit building on the previous one in a very organized and sequential process.

Summative quizzes, tests and lab tests are provided for each unit, helping to measure the concepts and skills presented in each learning module, giving ample opportunities for assessment and relearning.

Although many of the quizzes and labs are autograded, it is highly recommended, especially in the early stages, that manual grading by the teacher take place, especially when learning proper coding style, definition and use of proper identifiers and data elements, following proper output and presentation format, not always auto-gradable by automated systems, but no less important.

For example, in Java, source code indentation and alignment cannot be auto-graded but are both extremely important so that students learn proper coding style, aligning with the accepted programming conventions of the industry, better preparing them for possible opportunities further down the line for internships and possible employment.

Lab starter codes are provided for many situations, and lab solutions are given especially for the teachers still climbing the learning curve, often just ahead of the students by a month, week, or even a day or two.

Given that these courses are often electives, a liberal and grace-filled approach needs to be adopted by the instructor, providing students with opportunities for quiz or lab retakes when necessary.

However, equally important is extreme vigilance on the part of the teacher for attempts by students to take shortcuts for particularly tough assignments, discouraging the seeking or giving of inappropriate assistance to peers. The teacher must address this temptation and behavior early on in the learning process, rewarding and celebrating individual initiative coding elegance, but discouraging these attempted shortcuts through grading penalties, or even disciplinary steps as necessary.

Bottom line, the teacher needs to be actively engaged every step of the way as much as possible to ensure the best learning sequence for all students as they progress through these critical courses for the computational thinking needs of the present day.