Unit 6 - Pedagogy Strand

Computing vs. Coding vs. Computer Programming vs. Computer Science

On Coding vs. Computer Science

"Coding is learning to use a telescope whereas computer science is the physics behind how a telescope works" - MIT Student

On Computer Science

"Computer Science is the study of solving problems with computers" - MIT Student

On Math and Computer Programming

"I think understanding procedures and processes is important. But there's a fantastic way to do that in the modern world. It's called programming. Programming is how most procedures and processes get written down these days, and it's also a great way to engage students much more, and to check they really understand." - Conrad Wolfram, Director of Wolfram Research

Basic Definitions

Computing is the use or operation of computers.

Coding is the process of assigning code to something for the purposes of classification or identification. It deals with translation; from English into morse code, from psuedocode into java or from java into python.

Computer programming (or programming) is the action or process of writing computer programmes. It is the process that leads from an original formulation of a computer problem to executable programmes.

Computer Science is the study of the principles and use of computers.

Computational Thinking

"Computational Thinking is the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent." - Jeannette Wing, President's Professor of Computer Science, Computer Science Department, Carnegie Mellon University

"Computational Thinking (CT) involves a set of problem-solving skills and techniques that software engineers use to write programs that underlie the computer applications you use such as search, email and maps. Here are specific techniques: decomposition, pattern recognition, pattern generalisation and abstraction and algorithmic design." - Google on Exploring Computational Thinking

Since Computational Thinking is a universal skill set that is not only application to computers, it is the most favourable to teach outside of the computer science classroom. We have already touched upon how to infused computer science/computational thinking into the everyday classroom. In this unit, we will explore lesson planning for computational thinking.

Lesson Planning (for Computational Thinking)

Since incorporating computational thinking into classrooms would differ greatly depending on the subject and grade level, we encourage exploration of the following resources to find lesson planning activities and hints specific to what you need.

Google has provided a set of classroom-ready lessons, examples and programs for teachers to incorporate into their subjects' curriculum. They can be filtered by subjects, grade levels and type of resource.

Amber Settle, DePaul University, and Baker Franke and his colleagues at The University of Chicago Laboratory Schools introduce infusing computational thinking into english, history and other subjects and discuss the pros and cons in their paper, Infusing Computational Thinking into the Middle- and High-School Curriculum.

Readings & Videos
Google: Exploring Computational Thinking
Explore this site extensively. It goes into greater depth regarding computational thinking and provides links to scholarly articles on computational thinking in specific classroom subjects. Its resources are directed at teachers in all subjects, with no or some programming experience.

Reflection Questions
1. Why is computational thinking important?
2. What computational thinking abilities do student acquire from traditional methods of teaching?
3. How do students currently acquire computational thinking abilities?

Additional Issues to Consider:
1. Which of the following should high school students learn in school: computing, coding, computer programming, computer science, computational thinking. Which one should be prioritized? Why?
2. How can you infuse computational thinking into your subject's curriculum? Think about examples of specific activities. What challenges do you anticipate you might face?
3. How would increasing computational thinking benefit students in classroom? Outside the classroom? As they prepare for college? What about life after high school or college?