Computational Thinking

Computational Thinking is the mental discipline of thinking about a problem using concepts from computer science, with the ultimate goal of solving the problem with the help of a computer. It has five foundations (ADPDA):

Abstraction

Abstraction involves ignoring details that are unnecessary for the current issue at hand, and being able to talk about and reason about a thing apart from those details. It also involves thinking about generic classes of things. Naming new abstractions is critical.

Decomposition

Decomposition is the act of breaking down a large, complex problem, concept, or system into smaller, more manageable, and under-standable parts. This can be done top-down (starting from the whole) or bottom-up (identifying necessary parts before having a complete picture of the whole). Naming the decomposed parts is critical.

Pattern Recognition

Consciously looking for patterns such as: when this happens, that should follow; this happens repeatedly, based on these constraints; if this condition is true, then that action should occur; these steps must occur in sequence. Understanding how a pattern generalizes, and specializes, is critical.

Data Representation

Everything in computers is just bits, so we must represent real-world attributes and information in bits somehow. Deciding what information is necessary to represent, and how to represent it, and how it relates to other information, is important to solving a problem.

Algorithms

‘An algorithm is an unambiguous method of solving a specific problem’ [from Wikipedia]. In computer science, it is described using basic operations, decision points, and control flow. Designing good algorithms involves thinking hard about the most general concepts, then how each instance specializes, and what parameterization is needed.