Computational Thinking Skills in Early Childhood

This handout describes six computational thinking skills that young children develop and use in early childhood.

Computational thinking (CT) is a set of critical thinking and problem-solving skills, practices, and processes that everyone, including computer scientists, can learn and use in various contexts.

There are six key CT skills that are commonly highlighted as developmentally appropriate skills for young children:

1. Algorithms and Sequencing
2. Pattern Recognition
3. Decomposition and Modularity
4. Abstraction and Representation
5. Testing and Debugging
6. Conditional Logic and Loops

Computer science (CS) is the study of computers and computational systems. CT skills are used regularly within the broader field of CS to write code, design software, and troubleshoot systems. However, CT skills are also used outside the context of coding, as part of everyday life.

Algorithms and Sequencing

Algorithm: set of step-by-step instructions required to solve a problem or complete a task

Sequencing: act of arranging objects or instructions in a specific order

Examples in Everyday Life

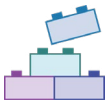
- A step-by-step morning routine **(algorithm)**
- Identifying the order of steps for brushing your teeth **(sequencing)**: (1) Squeeze toothpaste on your toothbrush. (2) Hold the toothbrush under the tap. (3) Brush your teeth for two minutes. (4) Spit the toothpaste into the sink. (5) Rinse your mouth with water.

Pattern Recognition

Identifying or creating a sequence of repeating items, such as objects, shapes, and numbers

Examples in Everyday Life

- Making a bracelet with a repeating pattern of colored beads (for example, following an AB pattern)
- Singing and clapping “Pat-a-Cake” with the educator, repeating a clapping pattern



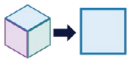
Decomposition and Modularity

Decomposition: breaking down a large problem or task into multiple smaller steps

Modularity: combining smaller steps, or modules, in different ways to solve a problem or complete a task

Examples in Everyday Life

- Breaking down all the steps needed to clean up a messy room before deciding what to tackle first **(decomposition)**: trash needs to be thrown away, toys should be picked up, floor needs to be vacuumed, and so forth
- Setting a table for six children for snack time by assembling one place setting first with a plate, cup, and spoon and then repeat this setup for the other five children **(modularity)**



Abstraction and Representation

Abstraction: extracting key information about a complex object or concept

Representation: displaying or recognizing an object or idea using a symbol

Examples in Everyday Life

- Finding a banana by searching for a yellow fruit shaped like a C **(abstraction)**
- Identifying a red octagonal sign on the street **(abstraction)** and recognizing that the sign means to stop **(representation)**

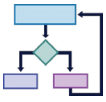
Testing and Debugging

Testing: checking if something works the way you want it to

Debugging: finding an error and using problem-solving strategies to fix the error

Examples in Everyday Life

- Checking whether a flashlight works, identifying that it does not work because it is missing batteries **(debugging)**, replacing the batteries, and testing again to make sure the flashlight works **(testing)**
- Realizing a puzzle piece is not in the right place and turning it in different orientations to figure out where the piece fits **(debugging)**



Conditional Logic and Loops

Conditional Logic: using if-then statements to make decisions about what to do next or to control the flow of an algorithm

Loop: a recurring set of steps used to determine if and how an action is repeated

Examples in Everyday Life

- Making decisions about what to wear based on the weather; for example, "IF it's sunny, THEN I will wear shorts" or "IF it's raining, THEN I will bring my umbrella." **(conditional logic)**
- Developing a dance routine consisting of three claps and a spin, and then repeating the sequence of actions two more times **(loops)**