A Computational Thinking Process for Problem Solving

Start by identifying a computational problem. Think about:
- Could it have multiple solutions?
- Is it a problem that includes collecting data or using a data set?
- Is there an opportunity to create a procedure (algorithm)?

Decompose the computational problem you identified to:
- Help you better understand the problem.
- Create sub-parts.
- Reveal assumptions or missing information.
- Identify where you can use CT elements to address sub-parts.
- Help organize your next steps.

Use pattern recognition to address your computational problem by:
- Collecting data or using an available data set.
- Analyzing the data.
- Representing the data (table, charts, graphs).
- Identify patterns.

Use abstraction to simplify complexity and generalize findings
- Abstractions relate to your computational problem.
- Pattern recognition and abstraction go hand-in-hand.

Design an algorithm to address your computational problem. Your design can be a flow chart, decision tree, pseudo code or other approach.
- First, establish a set of procedures.
- Then, have others follow your procedures.
- Finally, others should arrive at your expected results consistently. (If others get unexpected results, you will need to modify your design or procedures.)

Create your computational artifact. It, much like an assignment’s final report, showcases how you addressed and solved your computational problem.

Definitions Key

- **Computational problem**
  Computational problems are open-ended and may be real-world, but they must include data and an algorithm.

- **Abstraction**
  Reducing complexity by filtering out non-relevant information. This can simplify problem solving and help create a general idea of the computational problem.

- **Problem decomposition**
  Breaking down (unpacking) your computational problem into more manageable parts.

- **Pattern recognition**
  Collecting data or identifying a data set (numerical, text, audio, video, images or symbols) and analyzing it to find similarities, differences or trends.

- **Algorithm design**
  Developing a procedure (algorithm) that can be replicated by humans or computer, includes testing and redesign if the outcome is not what is expected.

- **Computational artifact**
  This can be a program, image, recording, video, presentation, webpage or anything else you can make using a computer.