Lesson Plan: What is an Algorithm?

Instructional Days: 1-2

Topic Description: Problem-solving is at the heart of computer science: whether it's games or working with data, we're trying to create tools to help us solve whole categories of problems. This unit introduces the idea of an "algorithm" as a set of instructions used to solve a problem; this sets the context for our discussion of searching and sorting algorithms later in the unit.

Objectives:

The student will be able to:

- Define the word "algorithm."
- Create algorithms to solve puzzles.

Outline of the Lesson:

- Discussion: Context
    - Recap what we've learned so far:
        - What is a computer?
        - What's in a computer? Software and hardware.
        - What is computer science used for in our professional and personal lives?
        - How do we encode and decode messages? Why is code important?
        - What is binary?
        - Is Scratch software or hardware? What Scratch skills have we learned?
    - Where are we going next?
        - Computer science is problem solving. We have a task to complete and some rules to abide by while we're completing it. We're trying to get from a question and a set of rules to a solution.
        - Think about the problem-solving process as a journey from A to B:
            - More than one way to get there
            - Some ways might be better than another, depending on what "better" means: faster? Fewer steps?
            - Might need tools and assistance along the way.
            - Ask students if they can think of any other similarities

Lesson Plan: What is an Algorithm?

- Activity: Problem Solving and Mazes
    - Trying to go from A to B while following a set of rules.
    - Emphasize thinking about problem solving strategies. Ask students:
        - How did you figure this out?
        - What steps did you go through in your head?
    - Debrief:
        - Take-home message: solving a problem requires a process.
        - The process varies depending on the type of problem we're trying to solve: was it a tilt maze? Was it a no-left-turn maze?
- Introduction to Algorithms
        - Computer science as problem-solving: we want to make step-by-step instructions to solve *categories* of problems.
        - Algorithms are just step-by-step procedures for completing a task, performing a calculation, or solving a problem.
            - Refer back to maze activity:
                - Commonly-used steps in solving mazes: figure out where start is, figure out where finish is, understand the rules, try making paths from the start and the finish, trial and error. These are all pieces of the "maze-solving algorithm," or the maze-solving process.
                - We used different steps when the mazes had different rules—the algorithm for a tilt maze is different than the algorithm for a basic maze.
                - We use the same steps for mazes in the same category, even if the solutions are different.
            - Connect back to PB & J activity, Scratch mazes, and LEGO encoding activity: someone must be able to follow your algorithm to solve the problem. Algorithms should be precise and should break down the problem-solving process into small steps.
- Video: What is an Algorithm?
    - Watch video
    - Discuss:
        - What is an algorithm?

        - What everyday algorithms do you use to solve problems or complete tasks?
- Activity: Solving Puzzles
    - Pass out the student handout
    - Have the students go through the handout individually, in groups, or as a class. It might be helpful to debrief, discuss, and check-in after each of the three puzzles so that the class can be on the same page and understand what an algorithm is supposed to look like. Some parts of the number maze and handshake puzzles are a little tricky, so the instructor might have to provide some hints.
        - As the students create their sets of instructions, they can use statements they saw in their Scratch programming like "if" and "repeat until."
    - Debrief:
        - What is an algorithm?
            - Your algorithms should be able to help someone solve *any* word search, number maze, or handshake puzzle.
        - Which puzzle was the hardest to solve? What algorithm was the hardest to write? Why?
        - How did you create an algorithm for each puzzle? What steps did you take?
            - Optional: what problem-solving techniques did you use?
            - Optional: could you write an algorithm for making algorithms?