Lesson 3: Tracing algorithms

In this lesson, students will learn how to use trace tables to understand algorithms and detect errors in programs. They will be introduced to the concept of trace tables as a tool to track the values of variables at each step of the code. Through guided practice and independent exercises, students will gain hands-on experience in filling out trace tables for different code examples, including while loops, for loops, and lists. By the end of the lesson, students will have a solid understanding of how trace tables can be used to analyze code and improve their coding skills.

## **Objectives:**

- Students will be able to use a trace table to walk through code that contains a while loop, a for loop, and a list of items.

- Students will be able to use a trace table to detect and correct errors in a program.

## **Materials:**

- Computers or laptops with Python IDE installed

- Projector or smartboard

- Handouts with code examples and trace tables

- Whiteboard and markers

## **Bell-Ringer Activity (5 minutes):**

1. Display a code snippet on the board or projector screen.

2. Ask students to individually trace through the code and write down the values of variables at each step.

3. After 2 minutes, ask students to share their answers with a partner.

4. Select a few students to share their answers with the whole class and discuss any differences or errors.

## **Introduction (10 minutes):**

1. Explain to students that today's lesson will focus on using trace tables to understand algorithms.

2. Define a trace table as a tool used to walk through code and track the values of variables at each step.

3. Emphasize that trace tables are useful for locating logic errors and understanding how an algorithm works.

4. Explain that trace tables will be particularly important in the upcoming lessons on searching and sorting algorithms.

5. Briefly review the concepts of while loops, for loops, nested loops, and lists, as these will be used in the code examples.

## **Direct Instruction (20 minutes):**

1. Display a code example on the board or projector screen.

2. Walk through the code step by step, explaining the purpose of each line and the expected values of variables at each step.

3. Model how to fill out a trace table for the code example, demonstrating how to track the values of variables.

4. Emphasize the importance of paying attention to the flow of the code and the changes in variable values.

5. Repeat this process with a few more code examples, gradually increasing the complexity of the algorithms.

6. Encourage students to ask questions and participate in discussions about the code examples and trace tables.

## **Guided Practice (20 minutes):**

1. Distribute handouts with code examples and trace tables to each student.

2. Instruct students to work in pairs or small groups to complete the trace tables for the given code examples.

3. Circulate around the classroom, providing assistance and guidance as needed.

4. After students have completed the trace tables, bring the class back together for a discussion.

5. Select a few groups to share their trace tables and explain their thought process.

6. Discuss any differences or errors in the trace tables and guide students towards the correct solutions.

## **Independent Practice (20 minutes):**

1. Assign students a coding task that requires them to use a trace table to detect and correct errors in a program.

2. Provide each student with a different code snippet and a blank trace table.

3. Instruct students to individually trace through the code, filling out the trace table and identifying any logic errors.

4. Encourage students to think critically and analyze the code for potential mistakes.

5. After students have completed the task, ask them to compare their trace tables and error corrections with a partner.

6. Select a few students to share their findings with the whole class and discuss the correct solutions.

## **Exit Ticket (10 minutes):**

1. Distribute exit tickets to each student.

2. Ask students to write a brief summary of how trace tables can be used to understand algorithms and detect errors in programs.

3. Collect the exit tickets before the end of the class.

## **Closure (5 minutes):**

1. Review the main concepts covered in the lesson, emphasizing the importance of trace tables in understanding algorithms.

2. Remind students that trace tables will be particularly useful in the upcoming lessons on searching and sorting algorithms.

3. Encourage students to continue practicing using trace tables to improve their coding skills.

4. Thank the students for their participation and effort in the lesson.

**Common Core Standards:**

- CCSS.ELA-LITERACY.RST.9-10.3 - Follow precisely a complex multistep procedure when carrying out experiments, taking measurements, or performing technical tasks, attending to special cases or exceptions defined in the text.

- CCSS.ELA-LITERACY.RST.9-10.4 - Determine the meaning of symbols, key terms, and other domain-specific words and phrases as they are used in a specific scientific or technical context relevant to grades 9-10 texts and topics.