Lesson 6: Coding Sorting Algorithms

In this lesson, learners will analyze and evaluate code for bubble sort and insertion sort in Python, as well as compare different implementations of the bubble sort algorithm. There are demo versions of bubble sort and insertion sort that output the steps of each pass which are linked in the “You may also need” section below, where you will also find the commented code used in the slides and worksheets and the 3 versions of bubble sort.

Learners will first be presented with different statements referring to bubble sort and insertion sort where they have to work out whether they are true or false. The remaining slides demonstrate and explain a step-by-step Python implementation for each sorting algorithm, as well as two improvements to the bubble sort code. The slides for both the bubble sort and insertion sort in Python build up from the inside out, focussing first on how one pass is performed before adding the functionality to repeat this process.

Each of the Python activities culminates with a worksheet that gets learners to further explore and understand the demoed code with questions and tracing the algorithm with a given set of data. Furthermore, a more efficient version of bubble sort is presented in a worksheet so learners can compare the efficiency of these two implementations.

## **Objectives:**

- Students will be able to interpret and analyze the code for bubble sort and insertion sort algorithms.

- Students will be able to trace the code for both sorting algorithms with input data.

- Students will be able to identify factors that could influence the efficiency of a bubble sort implementation.

## **Materials:**

- Slides or presentation software

- Worksheets

- Computers with Python installed

- Demo versions of bubble sort and insertion sort code

- Commented code used in the slides and worksheets

- Three versions of bubble sort code

## **Bell-Ringer Activity:**

- Display several statements about bubble sort and insertion sort on the board or projector.

- Ask students to work individually or in pairs to determine whether each statement is true or false.

- After a few minutes, discuss the answers as a class and clarify any misconceptions.

## **Introduction:**

- Explain to students that in this lesson, they will be analyzing and evaluating two sorting algorithms: bubble sort and insertion sort.

- Emphasize the importance of understanding how these algorithms work and their efficiency in sorting data.

- Explain that they will be working with Python code examples and worksheets to practice tracing and analyzing the algorithms.

## **Direct Instruction:**

- Present the slides or use the presentation software to guide the direct instruction.

- Start by explaining the bubble sort algorithm, step-by-step, using the demo version of the code.

- Show the code on the slides and explain each line, highlighting how the algorithm compares and swaps elements.

- Demonstrate the bubble sort algorithm with a sample input and show the steps of each pass.

- Discuss the time complexity of bubble sort and its limitations.

- Next, introduce the insertion sort algorithm in a similar manner, explaining each step and demonstrating with a sample input.

- Discuss the time complexity of insertion sort and compare it to bubble sort.

- Show the different implementations of bubble sort code and explain the improvements made in each version.

- Discuss how these improvements can make the algorithm more efficient.

## **Guided Practice:**

- Provide the students with worksheets that include questions and exercises related to the code examples and explanations.

- Guide the students through the worksheets, answering any questions and providing assistance as needed.

- Encourage students to work in pairs or small groups to discuss and solve the problems together.

## **Independent Practice:**

- Assign additional exercises or coding challenges related to bubble sort and insertion sort.

- Provide students with a set of data and ask them to trace and analyze the sorting algorithms on their own.

- Encourage students to experiment with different inputs and compare the efficiency of bubble sort implementations.

## **Exit Ticket:**

- Give students a short exit ticket to assess their understanding of the lesson objectives.

- The exit ticket could include questions such as:

- Explain the steps of the bubble sort algorithm.

- Compare the time complexity of bubble sort and insertion sort.

- Identify one improvement made in the different versions of bubble sort code.

## **Closure:**

- Review the key concepts covered in the lesson, emphasizing the importance of understanding and analyzing sorting algorithms.

- Encourage students to continue exploring and experimenting with different sorting algorithms and their implementations.

- Provide additional resources or references for further study if available.

## **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.