Lesson 7: Good Habits of a Programmer

In this lesson, students will learn about the good habits of a programmer and explore alternative approaches to programming solutions. The lesson will begin with a discussion on the importance of good habits in programming, such as writing clean and organized code, using meaningful variable names, and commenting code for clarity. Students will then have the opportunity to hear from an industry programmer who will share their experiences and good practices. Following this, students will work in small groups to analyze different programming examples and discuss the advantages and disadvantages of each approach. In the independent practice portion of the lesson, students will learn how to append data to a CSV file and will be given a challenge to apply their skills and the alternative approaches they learned. The lesson will conclude with a review of the key points discussed and an emphasis on the importance of continuous learning and improvement in programming.

# **Objectives**

* Determine the good habits of a programmer
* Explore alternative approaches to programming solutions
* Append to a CSV file

## **Materials:**

- Whiteboard or projector

- Computer with internet access

- Handouts with programming examples

- Pen/pencil and paper for each student

- CSV file for the final challenge

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

- Display the following question on the board: "What do you think are the good habits of a programmer?"

- Give students 2 minutes to write down their answers individually.

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

- Select a few students to share their answers with the whole class.

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

- Explain to students that today's lesson will focus on the good habits of a programmer and alternative approaches to programming solutions.

- Discuss the importance of good habits in programming, such as writing clean and organized code, using meaningful variable names, and commenting code for clarity.

- Explain that hearing from industry programmers about their own good practices can provide valuable insights and tips for becoming better programmers.

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

- Show students a video or invite a guest speaker (if available) who is an industry programmer to share their experiences and good habits.

- After the video or speaker, facilitate a class discussion about the good habits mentioned and encourage students to ask questions.

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

- Provide handouts with programming examples that demonstrate different approaches to solving a problem.

- Divide students into small groups and assign each group a different programming example.

- In their groups, students should analyze the different approaches and discuss the advantages and disadvantages of each.

- After the discussion, each group should present their findings to the class.

## **Independent Practice (25 minutes):**

- Explain to students that they will now learn how to append data to a CSV file.

- Demonstrate the process of appending data to a CSV file using a programming language of your choice.

- Provide step-by-step instructions for students to follow along on their computers.

- Give students a challenge to complete using their CSV skills and the alternative approaches they learned in today's lesson.

- Circulate the classroom to provide assistance and answer any questions.

## **Exit Ticket (5 minutes):**

- Distribute exit tickets to each student.

- On the exit ticket, ask students to write down one good habit of a programmer they learned today and one alternative approach to programming they found interesting.

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

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

- Review the key points discussed in the lesson, including the good habits of a programmer, alternative approaches to programming solutions, and appending data to a CSV file.

- Emphasize the importance of continuous learning and improvement in programming.

- Encourage students to practice the good habits they learned and explore different approaches to problem-solving in their future programming projects.

## **Common Core Standards:**

- CCSS.ELA-LITERACY.RST.9-10.2: Determine the central ideas or conclusions of a text; trace the text's explanation or depiction of a complex process, phenomenon, or concept; provide an accurate summary of 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.

- CCSS.ELA-LITERACY.RST.9-10.5: Analyze the structure of the relationships among concepts in a text, including relationships among key terms (e.g., force, friction, reaction force, energy).