Lesson 5: String Slicing

In this lesson, students will learn three important techniques for handling strings in Python: slicing a string to extract a portion of it, checking for a substring within a string using the `in` operator, and performing ASCII conversions using `chr()` and `ord()`. These techniques are essential for manipulating and analyzing text data in programming. The lesson will include a bell-ringer activity, direct instruction, guided practice, independent practice, and an exit ticket to assess student understanding.

## **Objectives:**

By the end of this lesson, students will be able to:

- Use a substring in a program.

- Use the `in` operator to check for a substring.

- Use `chr()` and `ord()` to perform ASCII conversions.

## **Materials:**

- Computers with Python IDE installed

- Projector and screen

- Handouts with coding exercises

- Whiteboard and markers

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

1. Display the following code on the screen:

```python

string = "Hello, World!"

print(string[0:5])

```

2. Ask the students to predict the output of the code.

3. Allow a few students to share their answers.

4. Run the code and discuss the output with the students.

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

1. Explain to the students that in this lesson, they will learn three new techniques for handling strings in Python.

2. Write the three techniques on the whiteboard:

- Slicing a string and storing it in a variable.

- Checking for a substring within a string using the `in` operator.

- Performing ASCII conversions using `chr()` and `ord()`.

3. Discuss the importance of these techniques in programming and how they can be used to manipulate and analyze text data.

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

1. **Slicing a string and storing it in a variable:**

- Explain that slicing allows us to extract a portion of a string by specifying the start and end indices.

- Demonstrate the following code on the screen:

```python

string = "Hello, World!"

substring = string[0:5]

print(substring)

```

- Discuss the output and explain how the start index is inclusive and the end index is exclusive.

- Provide additional examples and encourage students to try different slicing techniques.

2. **Checking for a substring within a string using the `in` operator:**

- Explain that the `in` operator can be used to check if a substring exists within a string.

- Demonstrate the following code on the screen:

```python

string = "Hello, World!"

if "Hello" in string:

print("Substring found!")

else:

print("Substring not found!")

```

- Discuss the output and explain how the `in` operator returns a boolean value.

- Provide additional examples and encourage students to experiment with different substrings.

3. **Performing ASCII conversions using `chr()` and `ord()`:**

- Explain that ASCII is a character encoding standard that represents characters as numeric codes.

- Demonstrate the following code on the screen:

```python

character = "A"

ascii\_code = ord(character)

print(ascii\_code)

ascii\_code = 65

character = chr(ascii\_code)

print(character)

```

- Discuss the output and explain how `ord()` returns the ASCII code of a character, and `chr()` returns the character corresponding to an ASCII code.

- Provide additional examples and encourage students to explore different characters and ASCII codes.

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

1. Divide the students into pairs.

2. Distribute handouts with coding exercises related to the three techniques discussed.

3. Instruct the students to work together to solve the exercises, applying the concepts learned in the direct instruction.

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

5. Encourage students to discuss their approaches and solutions with their partners.

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

1. Assign a coding challenge to the students:

- Challenge: Create a message decoder that converts a string of ASCII codes into the corresponding characters.

- Provide a sample input string of ASCII codes for the students to decode.

- Instruct the students to use the `chr()` function to convert each ASCII code into a character and concatenate them to form the decoded message.

- Encourage students to test their decoder with different input strings.

2. Allow the students to work individually on the coding challenge.

3. Monitor their progress and provide assistance as needed.

4. Encourage students to think critically and problem-solve independently.

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

1. Distribute exit tickets to the students.

2. Ask them to write a brief summary of the three string handling techniques covered in the lesson.

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

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

1. Review the key concepts and techniques covered in the lesson.

2. Ask students if they have any questions or if there are any topics they would like to review in the next lesson.

3. Remind students to practice the string handling techniques and explore additional coding challenges on their own.

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.