Q1. Does assigning a value to a string's indexed character violate Python's string immutability?

Q2. Does using the += operator to concatenate strings violate Python's string immutability? Why or why not?

Q3. In Python, how many different ways are there to index a character?

Q4. What is the relationship between indexing and slicing?

Q5. What is an indexed character's exact data type? What is the data form of a slicing-generated substring?

Q6. What is the relationship between string and character "types" in Python?

Q7. Identify at least two operators and one method that allow you to combine one or more smaller strings to create a larger string.

Q8. What is the benefit of first checking the target string with in or not in before using the index method to find a substring?

Q9. Which operators and built-in string methods produce simple Boolean (true/false) results?

Answer:

Q1. No, assigning a value to a string's indexed character does not violate Python's string immutability. The string object itself is immutable, but the assignment simply creates a new string object in the same variable name.

Q2. Yes, using the += operator to concatenate strings does violate Python's string immutability. It creates a new string object and rebinds the variable name to the new object.

Q3. In Python, there is only one way to index a character in a string, using square brackets [] and the index of the character.

Q4. Slicing is a way to extract a substring from a larger string by specifying a range of indices. It uses the same square bracket notation as indexing, but with a colon between two indices to indicate the range.

Q5. An indexed character in a string is of type string, while a slicing-generated substring is also of type string.

Q6. In Python, a string is a sequence of characters, and each character in the string is of type string.

Q7. The + operator and the join() method allow you to combine one or more smaller strings to create a larger string. For example, "hello" + "world" or "".join(["hello", "world"]).

Q8. Checking the target string with in or not in before using the index method to find a substring can prevent a ValueError if the substring is not found. The in or not in operator returns a Boolean value indicating whether the substring is present in the string, while the index method raises a ValueError if the substring is not found.

Q9. The in and not in operators and the startswith() and endswith() methods produce simple Boolean (true/false) results.