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

Assigning a value to a string's indexed character does not violate Python's string immutability because strings in Python are immutable. Immutable means you cannot change the individual characters of a string once it's created.

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

Using the += operator to concatenate strings does not violate string immutability. It creates a new string that is the concatenation of the original string and the added string. The original strings remain unchanged.

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

There is only one way to index a character in a string in Python, using square brackets and the index (e.g., string[0]).

Q4. What is the relationship between indexing and slicing?

Indexing and slicing both involve extracting characters or substrings from a string. Indexing extracts a single character, while slicing extracts a substring.

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

The indexed character's exact data type is a string (of length 1). The data form of a slicing-generated substring is also a string.

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

In Python, characters do not have a distinct data type. Characters are represented as strings of length 1.

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

Operators: + (concatenation), \* (repetition)

Method: str.join(iterable) (e.g., ' '.join(['Hello', 'World']))

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?

Using in or not in before the index method helps avoid raising a ValueError if the substring is not present in the target string.

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

Operators: ==, !=, in, not in

Methods: str.startswith(prefix), str.endswith(suffix), str.isalnum(), str.isalpha(), str.isnumeric(), etc.