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

Ans1- Assigning a value to a string's indexed character violates Python's string immutability because strings in Python are immutable, meaning their contents cannot be changed after creation. Attempting to modify a character in a string will result in a TypeError.

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

Ans2- Using the += operator to concatenate strings in Python does not violate string immutability. It creates a new string with the concatenated content, leaving the original strings unchanged.

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

Ans3- you can index a character in a string using positive indexing, negative indexing, slicing, extended slicing with a stride, iterating over characters, or by using methods like find() or index(). There are multiple ways to access characters in a string, depending on your specific needs.

Q4. What is the relationship between indexing and slicing?

Ans4- Indexing is used to access individual characters in a string, providing a specific character at a given position.

Slicing is used to extract a substring from a string by specifying a range of indices, creating a new string containing a portion of the original string.

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

Ans5- An indexed character in Python is of the str (string) data type. It's a single-character string.

In Python, strings are sequences of characters. Each character is itself a string (of length 1). So, a string is a sequence of string "types."

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

Ans6- relationship between characters and strings in Python allows for a consistent and unified approach when working with individual characters and entire strings, as they share the same data type (str).

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

Ans7- a) Concatenation Operator +: You can use the + operator to concatenate (join) two or more strings together to create a larger string.

b) Augmented Assignment Operator +=: The += operator can be used to concatenate and update a string in place. It adds the content of the right-hand string to the left-hand 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?

Ans8- The benefit of first checking the target string with in or not in before using the index() method to find a substring is to avoid raising a ValueError if the substring is not found in the target string.

This approach allows you to handle cases where the substring may or may not be present in a more controlled and error-free manner.

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

Ans9- a) Comparison Operators: Comparison operators like == (equal), != (not equal), < (less than), > (greater than), <= (less than or equal to), and >= (greater than or equal to) are used to compare strings and produce Boolean results.

b) startswith(): The startswith() method checks if a string starts with a specified substring and returns True or False.