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

Ans: - Yes, assigning a value to a string’s indexed character does violate Python’s string immutability. Strings in Python are immutable, which means they cannot be changed after they are created. If you try to assign a value to a string’s indexed character, Python will raise a TypeError

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

Ans: - No, using the += operator to concatenate strings does not violate Python’s string immutability. When you use += to concatenate strings, Python creates a new string that is the result of the concatenation and assigns it to the variable. The original strings remain unchanged.

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

Ans: - . In Python, you can index a character in a string in two ways: using positive indexing (from the beginning of the string) and using negative indexing (from the end of the string)

Q4. What is the relationship between indexing and slicing?

Ans: - Indexing and slicing in Python are related in that they both allow access to specific elements of a sequence such as a string. Indexing retrieves a single character at a specific position, while slicing retrieves a range of characters (a substring) from the string

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

Ans: - An indexed character in Python is a string of length one61112. The data form of a slicing-generated substring is also a string

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

Ans: - In Python, there is no separate character type; a character is considered a string of length one. Therefore, the relationship between string and character types in Python is that they are essentially the same

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

Ans: - In Python, you can combine one or more smaller strings to create a larger string using the + operator, the join() method, and the % operator

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?

Ans: -. Before using the index() method to find a substring, it’s beneficial to first check the target string with in or not in because the index() method raises an exception if the substring is not found. By checking with in or not in first, you can avoid this exception

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

Ans: - In Python, the operators that produce Boolean results include comparison operators (==, !=, <, <=, >, >=) and membership operators (in, not in). Built-in string methods that produce Boolean results include startswith(), endswith(), islower(), isupper(), istitle(), isalnum(), isalpha(), and isspace()