Q1. Does assigning a value to a string's indexed character violate Python's string immutability?
Answer: We cannot assign a value to an indexed character in a string because it violates string immutability. But we can split the string into indexes and concatenate a string at the desired index as follows :-

hlp = ‘123678’

In the example ‘456’ can be inserted at position 2, so

hlp = hlp[:2] + ‘456’ + hlp[3:]

print(hlp) >>>> ‘12345678

Q2. Does using the += operator to concatenate strings violate Python's string immutability? Why or why not?
Answer: The string itself is immutable but the label can change. Assigning a new value to an existing variable is perfectly valid. Python does not have constants. This is independent from data type mutability

Q3. In Python, how many different ways are there to index a character?
Answer: There are two ways to access a character by index in python. Accessing Characters by Positive Index Number. Accessing Characters by Negative Index Number

Q4. What is the relationship between indexing and slicing?
Answer: “Indexing” means referring to an element of an iterable by its position within the iterable. “Slicing” means getting a subset of elements from an iterable based on their indices

Q5. What is an indexed character's exact data type? What is the data form of a slicing-generated substring?
Answer: An indexed character data type is str (string) .
The data form of a slicing generated substring is str (string)

Q6. What is the relationship between string and character "types" in Python?
Answer: Strings are ordered sequences of character data, and thus can be indexed in this way.

Q7. Identify at least two operators and one method that allows you to combine one or more smaller strings to create a larger string.
Answer: Here are the operators and methods

- Concatenate multiple strings: + , += operator.
- Concatenate strings and numbers: + , += operator, str() , format() , f-string.
- Concatenate a list of strings into one string: join()
- Concatenate a list of numbers into one string: join() , str()

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?
Answer: The “in” and ”not in” conditions checks if a substring is present inside a string. It loops through the iterable using indexes and reduces code size and possibility of error.

Q9. Which operators and built-in string methods produce simple Boolean (true/false) results?
Answer: The logical operators and, or and not are also referred to as boolean operators. While and as well as or operator needs two operands, which may evaluate to true or false, not operator needs one operand evaluating to true or false. Boolean and operator returns true if both operands return true