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

🡪String’s indexed character cannot to be assigned a new value, as Strings are immutable.

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

🡪+= operator is used to concatenate strings, it does not violate Python’s string immutability Property. Because doing so new creates a new association with data and variable. E.g. str\_1="P" and str\_1+="Y”. effect of this statements to create string PY and reassign it to variable str\_1, any string data is not actually modified.

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

🡪 A Character in string can be indexed using string name followed by index number of character in square bracket. Positive Indexing i.e. first index is 0 an so on, or Negative Indexing i.e. last letter is -1 and so on can be used to index a character.

Q4. What is the relationship between indexing and slicing?

🡪We can access elements of sequence datatypes by using slicing and indexing. Indexing is used to obtaining individual element while slicing for sequence of elements.

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

🡪 Indexed characters and sliced substrings have datatype String.

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

🡪 Object that contains sequence of character datatypes are called String

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

🡪+, += and \* allow to combine one or more smaller strings to create a larger string. <string>.join(<sep>) method joins element, like list and tuple to get a combined 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?

🡪Checking the target string with in or not Operators before using the index method to find a substring just helps confirming availability of substring and thus avoid raising of Value Error.

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

🡪 Several operators and built-in string methods in programming languages can produce simple Boolean (true/false) results. Here are some examples:

1. Comparison Operators:

- Equality: `==` (equal to)

- Inequality: `!=` (not equal to)

- Greater than: `>` (greater than)

- Less than: `<` (less than)

- Greater than or equal to: `>=` (greater than or equal to)

- Less than or equal to: `<=` (less than or equal to)

2. Logical Operators:

- Logical AND: `&&` (and)

- Logical OR: `||` (or)

- Logical NOT: `!` (not)

3. String Methods:

- `startsWith()`: Returns true if a string starts with a specified prefix.

- `endsWith()`: Returns true if a string ends with a specified suffix.

- `contains()`: Returns true if a string contains a specified substring.

- `isEmpty()`: Returns true if a string is empty.

- `equals()`: Returns true if two strings are equal.

- `equalsIgnoreCase()`: Returns true if two strings are equal, ignoring case.

These operators and methods can be used to perform various comparisons and evaluations in programming, and their results are simple Boolean values (true or false) indicating the truth or falsity of the condition being checked.