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

Ans. The string itself is immutable but the label can change. Assigning a new value to an existing variable is perfectly valid.

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

Ans. += 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="a" and str\_1+="b. effect of this statements to create string ab 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?

Ans. We can access characters in a String in Two ways : Accessing Characters by Positive Index Number. Accessing Characters by Negative Index Number.

Q4. What is the relationship between indexing and slicing?

Ans. “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?

Ans. Indexed characters and sliced substrings have datatype String.

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

Ans. Strings in Python are arrays of bytes representing unicode characters. However, Python does not have a character data type, a single character is simply a string with a length of 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.

Ans. When you merge or combine two or more strings in Python, it is called string concatenation. The simplest and most common method of concatenating a string is using the plus symbol (“+”).

METHOD:

a = “Python”

b = “is”

c = “cool”

print(a + b + c)

The asterisk (\*) operator is used when you want to concatenate the same string repeatedly.

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. 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 ValueError.

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

Ans. The String Operators and built-in methods to Produce Simple Boolean (True/False) Results are:

in

not

<string>.isalpha()

<string>.isalnum()

<string>.isdecimal()

<string>.isdigit()

<string>.islower()

<string>.isnumeric()

<string>.isprintable()

<string>.isspace()

<string>.istitle()