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

ANS:

In python, the string data types are immutable. Which means a string value cannot be updated. We can verify this by trying to update a part of the string which will led us to an error.

# Can not reassign

t= "Tutorialspoint"

print type(t)

t[0] = "M"

When we run the above program, we get the following output −

t[0] = "M"

TypeError: 'str' object does not support item assignment

We can further verify this by checking the memory location address of the position of the letters of the string.

.

x = 'banana'

for idx in range (0,5):

print x[idx], "=", id(x[idx])

When we run the above program we get the following output. As you can see above a and a point to same location. Also N and N also point to the same location.

b = 91909376

a = 91836864

n = 91259888

a = 91836864

n = 91259888

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

ANS:

Normally, += on immutable objects is supposed to create a new object before clearing the reference to the old object, so the new and old objects should have overlapping lifetimes, forbidding equal id values. With the optimization in place, the string after the += has the same ID as the string before the += .

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

ANS:

The Python string data type is a sequence made up of one or more individual characters that could consist of letters, numbers, whitespace characters, or symbols. As the string is a sequence, it can be accessed in the same ways that other sequence-based data types are, through indexing and slicing.

## Indexing

Indexing means referring to an element of an iterable by its position within the iterable. Each of a string’s characters corresponds to an index number and each character can be accessed using its index number. We can access characters in a String in Two ways :

1. Accessing Characters by Positive Index Number
2. 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:

 Indexing is used to obtain individual elements. Slicing: Slicing is used to obtain a sequence of elements. Indexing and Slicing can be be done in Python Sequences types like list, string, tuple, range objects.

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

ANS:

Strings are Arrays

Like many other popular programming languages, 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:

The ampersand symbol is the recommended concatenation operator. It is used to bind a number of string variables together, creating one string from two or more individual strings.

There are two string operators. The first is the concatenation operator ('.'), which returns the concatenation of its right and left arguments. The second is the concatenating assignment operator (' .= '), which appends the argument on the right side to the argument on the left side.

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:

The index() method returns the index of the first occurence of a substring in the given string. It is same as the [find()](https://www.tutorialsteacher.com/python/string-find) method except that if a substring is not found, then it raises an exception.

### Syntax:

str.index(substr, start, end)

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

ANS:

There are sixteen possible two-input Boolean operators. Except for and and or , they are rarely needed in practice. Because of this, True , False , not , and , and or are the only built-in Python Boolean operators.