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

In Python, a variable is nothing more than a name, and it may be reused, over and over. That’s why in below statements might seem to violate immutability of strings but in fact do not.

str1 = 'india'

str1 = 'India'

No existing string is altered in this above example; rather, two different strings are created and the name str1 is reused.

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

The concatenation operator (+) for strings may be familiar, because it is supported in many languages that have some kind of string class. Concatenation does not automatically add a space between two words. You have to do that yourself. But all strings, including literal strings such as ' ', have the same type, str, so Python has no problem carrying out the following:

str1 = 'India'

str2 = 'Gate'

str\_name = str1 + ' ' + str2

print(str\_name)

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

Each of a string’s characters corresponds to an index number and each character can be accessed using their 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?

Indexing uses a number to refer to an individual character,

according to its place within the string.

Slicing is an ability more unique to Python. It enables you to refer

to an entire substring of characters by using a compact syntax.

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

Individual characters in a string can be accessed by specifying the string name followed by a number in square brackets ([]).

find method. A simple call to this method finds the first occurrence of the substring argument and returns the nonnegative index of that instance; it returns –1 if the substring isn’t found.

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

In Python, strings are ordered sequences of character data, and thus can be indexed in this way. Individual characters in a string can be accessed by specifying the string name followed by a number in square brackets ([]).

String indexing in Python is zero-based: the first character in the string has index 0, the next has index 1, and so on. The index of the last character will be the length of the string minus one.

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

Slicing is a special ability shared by Python strings, lists, and tuples. Below list summarizes the syntax supported for slicing of strings, which produces substrings. Remember that you can’t assign into a slice of a string.

string[hotel:end]- All characters starting with hotel, up to but not including end.

string[:end]- All characters from the beginning of the string up to but not including end.

string[hotel:] - All elements from beg forward to the end of the string.

string[:] - All characters in the string; this operation copies the entire string.

string[hotel:end:step] - All characters starting with hotel, up to but not including end, moving through the string step items at a time.

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?

we can also use in and not in to test substrings that contain more than one character. In that case, the entire substring must be found to produce True. Notice that the in operator, if tested, always responds as if all strings include the empty string, '', which differs from the way

lists work. Python does not return True if you ask whether a list contains the empty list.

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

Str.isalnum()

str.isalpha()

str.isdecimal()

str.isdigit()

str.isidentifier()

str.islower()

str.isprintable()

str.isspace()

str.istitle()

str.isupper()