**Assignment - 12**

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

Ans: Yes, assigning a value to a string's indexed character violates Python's string immutability. Strings in Python are immutable, meaning their contents cannot be changed after they are created. When you try to assign a value to a string's indexed character, Python will raise an error because it cannot modify the existing string object.

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

Ans: No, using the += operator to concatenate strings does not violate Python's string immutability. While strings themselves are immutable, concatenation using the += operator creates a new string object that contains the combined contents of the original strings. Therefore, the original string objects remain unchanged, and a new string object is created to hold the concatenated result.

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

Ans: There is only one way to index a character in Python, which is by using the index (position) of the character within the string. Indexing starts from 0 for the first character, 1 for the second character, and so on.

Q4. What is the relationship between indexing and slicing?

Ans: Indexing and slicing are both techniques used to access specific parts of a string in Python. Indexing refers to accessing individual characters of a string by their position (index), while slicing refers to extracting a portion of the string by specifying a range of indices.

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

Ans: The exact data type of an indexed character in Python is a string (str). When indexing a string, the result is always a single-character string. The data form of a substring generated by slicing is also a string (str), which may contain multiple characters.

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

Ans: In Python, strings are sequences of characters. Each character in a string is essentially a substring of length 1. Therefore, the relationship between strings and characters is that a string is composed of one or more characters.

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: Two operators that allow you to combine smaller strings to create a larger string are the addition operator (+) and the multiplication operator (\*). Additionally, the join() method is a string method that allows you to concatenate multiple strings by joining them with a specified separator 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?

Ans: The benefit of first checking the target string with in or not in before using the index method to find a substring is to avoid raising a ValueError if the substring is not present in the target string. Checking with in or not in allows you to verify whether the substring exists in the string before attempting to find its index.

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

Ans: Operators like ==, !=, in, and not in, along with methods like startswith(), endswith(), isalnum(), isalpha(), isdigit(), islower(), isupper(), isspace(), and istitle() produce simple Boolean (true/false) results when applied to strings. These operations and methods return True if the condition is met and False otherwise.