ASSIGNMENT - 12

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

Ans: Assigning a value to a string's indexed character violates Python's string immutability. Strings in Python are immutable, meaning once created, their content cannot be changed. Attempting to directly change a character at a specific index will result in an error.

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

Ans: Using the += operator to concatenate strings does not violate string immutability. The += operator creates a new string by concatenating the existing string with the new one. It doesn't modify the original string; instead, it creates a new string object containing the concatenated result.

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

Ans: In Python, there is one primary way to index a character in a string:

By using square brackets [] with the index position (starting from 0) of the character within the string.

Q4. What is the relationship between indexing and slicing?

Ans: The relationship between indexing and slicing is that slicing is an extension of indexing. Indexing retrieves a single character at a specific position, while slicing retrieves a substring by specifying a range of indices (start:end).

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

Ans: An indexed character in Python is of type str, representing a single character. The data form of a substring generated by slicing is also a str type, representing a sequence of characters.

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

Ans: In Python, strings are sequences of characters, and character "types" are not explicitly defined. Characters themselves are represented as strings of length 1. There isn't a distinct "character" type separate from the str type.

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: Operators and a method used to combine smaller strings into a larger string:

Operators:

+: Concatenates two strings to create a larger string.

\*: Replicates a string by multiplying it with an integer to create a larger string.

Method:

join(): Joins multiple strings together using a specified delimiter to create a larger 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: Checking the presence of a substring with in or not in before using the index method to find a substring helps avoid potential errors. If the substring doesn't exist within the target string, using index will raise a ValueError. Checking with in or not in allows handling the case when the substring is not present more gracefully.

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

Ans: Operators (<, <=, >, >=, ==, !=) and certain built-in string methods (startswith(), endswith(), isalnum(), isalpha(), isnumeric(), islower(), isupper(), isspace(), isdigit(), isdecimal(), isidentifier(), isprintable()) produce simple Boolean (true/false) results when applied to strings.