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

* No, assigning a value to a string's indexed character does not violate Python's string immutability because strings are immutable in Python. When you try to assign a value to an indexed character in a string, Python will raise an error. To modify a string, you need to create a new string with the desired changes.

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

* No, using the `+=` operator to concatenate strings does not violate Python's string immutability. While strings are immutable in Python, the `+=` operator creates a new string that is the result of concatenation and assigns it to the original variable. The original string remains unchanged, and the variable is updated to reference the new string.

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

* 1. By using positive indices: You can use positive integers to index characters from the beginning of the string, where the first character has an index of 0.
* 2. By using negative indices: You can use negative integers to index characters from the end of the string, where the last character has an index of -1.

**Q4. What is the relationship between indexing and slicing?**

* Indexing is used to access a single element at a specific position, such as `my\_list[2]`, which accesses the third element in a list. Slicing, on the other hand, allows you to select a range of elements using a start and end index, as in `my\_list[1:4]`, which selects elements from index 1 (inclusive) to 4 (exclusive).

**Q5. Which operators and built-in string methods produce simple Boolean (true/false) results?**

* 1. Comparison operators (e.g., `==`, `!=`, `<`, `<=`, `>`, `>=`) used to compare strings or string elements.
* 2. Logical operators (e.g., `and`, `or`, `not`) used in Boolean expressions involving strings.