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

String’s indexed character cannot to be assigned a New value , as Strings are **immutable. So assigning a value to a string's indexed character does not violate Python's string immutability**  
Example:  
name = "Reinforcement"  
print(id(name)) #73472  
name[0] = "V" # Raises TypeError

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

**+=** operator is used to concatenate strings, it does not violate Python’s string immutability Property. Because doing so new creates a new association with data and variable. E.g. str\_1="a" and str\_1+="b. effect of this statements to create string ab and reassign it to variable str\_1, any string data is not actually modified.

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

Accessing Characters by Positive Index Number.

Accessing Characters by Negative Index Number.

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

**“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.** We can access elements of sequence datatypes by using slicing and indexing. Indexing is used to obtaining individual element while slicing for sequence of elements.

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

indexing returns a string — Python has no special type for a single character. It is just a string of length Indexed characters and sliced substrings have datatype **String**.

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

Object that contains sequence of character datatypes are called String.

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

**+**, **+=** and **\*** allow to combine one or more smaller strings to create a larger string. **<string>.join(<sep>)** method joins element of iterable type like list and tuple to get a combined 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?**

Checking the target string with **in** or **not** Operators before using the index method to find a substring just helps confirming availability of substring and thus avoid raising of **ValueError.**  
**Example:**  
in\_string = "ineuron"  
in\_string.index('x') # Raises ValueError  
in\_string.index('u') # 3

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

Boolean Operations are simple arithmetic of True and False values. These values can be manipulated by the use of boolean operators which include **AND, Or, and NOT**. Common boolean operations are –

* or
* and
* not
* == (equivalent)
* != (not equivalent)

The String Operators and built-in methods to Produce Simple Boolean (True/False) Results are:

* **in**
* **not**
* **<string>.isalpha()**
* **<string>.isalnum()**
* **<string>.isdecimal()**
* **<string>.isdigit()**
* **<string>.islower()**
* **<string>.isnumeric()**
* **<string>.isprintable()**
* **<string>.isspace()**
* **<string>.istitle()**