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

**Ans**: String’s indexed character cannot to be assigned a new value as strings are **immutable.**

Example:

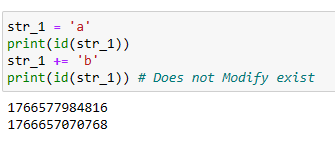
Name = “Reinforcement”

Print (id(name)) #73472

Name [0] = “v” #Raises TypeError

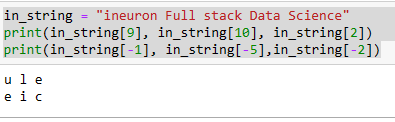
**Q2. Does using the += operator to concatenate strings violate Python's string immutability? Why or why not?**

**Ans: +=** operator is used to concatenate strings. It does not violate python’s string immutability property because doing so creates new association with data and variable E.G**., str\_1= “a”** and **str\_1=”b”.** Effect of this statement is to create string ab and reassign it to variable **str\_1**. Any string data is not actually modified.



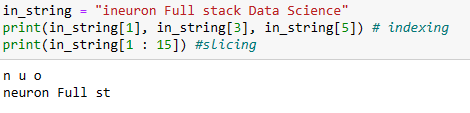
**Q3. In Python, how many different ways are there to index a character?**

**Ans:** A character in string can be indexed using string name followed by index number of character in square bracket. Positive indexing i.e. first index is 0 an so on, or negative indexing i.e. last latter is -1 and so on can be used to index a character

****

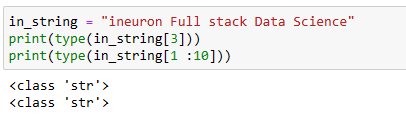
**Q4. What is the relationship between indexing and slicing?**

Ans: we can access elements of sequence data types 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?**

Ans: Indexed characters and sliced substrings have data type string

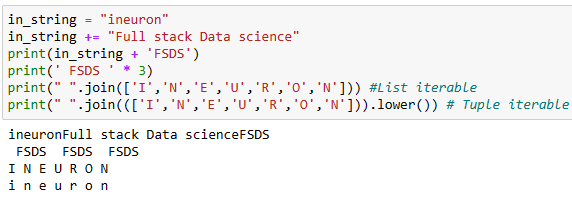
****

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

**Ans:** object that contains sequence of characters 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.**

**Ans: + , +=** and \* allow to combine one or more smaller strings to create a larger string. **<string> . join(<sep>)** method joins elements of iterable type 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?**

**Ans: 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 value error**

**In\_ string. Index(‘u’) #3**

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

**Ans:** 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()**