**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 a 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.

In [1]:

str\_1 **=** 'a'

print(id(str\_1))

str\_1 **+=** 'b'

print(id(str\_1)) *# Does not Modify existing string, Creates a New String Object*

1730938128368

1731018352752

**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 letter is -1 and so on can be used to index a character

In [2]:

in\_string **=** "iNeuron Full Stack Data Science"

print(in\_string[9],in\_string[10],in\_string[2]) *# Positive Indexing*

print(in\_string[**-**1],in\_string[**-**5],in\_string[**-**2]) *# Negative Indexing*

u l e

e i c

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

**Ans:** 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.

In [3]:

in\_string **=** "iNeuron Full Stack Data Science"

print(in\_string[1],in\_string[3],in\_string[5]) *# Indexing*

print(in\_string[1:15]) *# Slicing*

N u o

Neuron Full St

**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 datatype **String**.

In [4]:

in\_string **=** "iNeuron Full Stack Data Science"

print(type(in\_string[3])) *# Indexing -> str*

print(type(in\_string[1:10])) *# Indexing -> str*

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

**Ans:** Object that contains sequence of character datatypes are called String.

**Q7. Identify at least two operators & 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. **.join()** method joins element of iterable type like list and tuple to get a combined string.

In [5]:

in\_string **=** 'iNeuron '

in\_string **+=** 'Full Stack Data Science'

print(in\_string **+** ' FSDS')

print('FSDS '**\***3)

print(" "**.**join(['I','N','E','U','R','O','N'])) *# List Iterable*

print(" "**.**join(('I','N','E','U','R','O','N'))**.**lower()) *# Tuple Iterable*

iNeuron Full Stack Data Science FSDS

FSDS FSDS FSDS

I N E U R O N

i n e u r o n

**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 ValueError  
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**
* **.isalpha()**
* **.isalnum()**
* **.isdecimal()**
* **.isdigit()**
* **.islower()**
* **.isnumeric()**
* **.isprintable()**
* **.isspace()**
* **.istitle()**