# Assignment 12

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

**Answer:**

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 ?

**Answer:**

**+=** 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.

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

2664305477256  
2664345250032

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

**Answer:**

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\_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?

**Answer:**

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\_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?

**Answer:**

Indexed characters and sliced substrings have datatype **String**.

in\_string = "iNeuron Full Stack Data Science"  
print(type(in\_string[3])) *# Indexing -> str*  
print(type(in\_string[1:10])) *# Indexing -> str*

<class 'str'>  
<class 'str'>

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

**Answer:**

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 ?

**Answer:**

**+**, **+=** 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.

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 ?

**Answer:**

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?

**Answer:**

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()**