**Python Advance Assignment-12**

**1. Does assignment to an indexed character of a string violate Python’s immutability for strings?**

Ans.

String is an immutable. So we can't do any updation or deletion with the string items through indexing.

In [2]:

s**=**'shiavnsh'

s[2]**=**'t'

print(s)

**---------------------------------------------------------------------------**

**TypeError** Traceback (most recent call last)

**<ipython-input-2-889182652773>** in <module>

1 s**='shiavnsh'**

**----> 2** s**[2]='t'**

3 print**(**s**)**

**TypeError**: 'str' object does not support item assignment

**2. Does string concatenation, using the += operator, violate Python’s immutability for strings? Why or why not?**

Ans.

No it doesnot violate it because string class is having an inbuilt method which is add () which takes strings and concatenate them

**3. How many ways are there in Python to index a given character?**

Ans. There are various ways in pytho to get some item at some index position.

1) By simply using the indexing and slicing method inside square bracket. Eg: item[1] or item[1:4] or item[3::2], etc

2) By using list index method. **list.index(position)**. In this way one can get the item at some index position.

3) Using built in method like **getitem ()** and **index**. We can use this inside the class

**4. How, precisely, are indexing and slicing related?**

Ans.

**Indexing**: It is a technique used to get the item of that particuar index position or in cas eof mutable object we can also update the value or add the new value by just giving the index position and assignment operator.

**Slicing**: It is use when we wan to extract some range of a item from some object.

**Example**

l **=** [11,22,33,44,55,66,77,88]

indexing**=**l[5] *#getting that item at position 5*

print(f'Indexing: {indexing}')

slicing**=**l[1:5] *#getting items from position 1 till 4*

print(f'Slicing: {slicing}')

Indexing: 66

Slicing: [22, 33, 44, 55]

type(indexing)

int

type('.')

str

**5. What is the exact data type of an indexed character? What is the data type of a substring produced from slicing?**

Ans.

The exact data type for the indexed character is **CHAR** but generally it is refer to as a string. Or it could that which was earlier before the indexing.

The data type of a substring produced from slicing is also a string

**6. In Python, what is the relationship between the string and character “types”?**

Ans.

Character refers to a single letter, number, space, punctuation mark or a symbol. Its data type is actually a **CHAR** which stores character data in a fixed-length field.

While String is a sequence of characters. A group of different different character compose a string. We enclosed it by ' ' Strings are arrays of bytes representing Unicode characters. Its type is **string**

**7. Name at least two operators and one method that enable you to build a larger string out of one or more smaller strings.**

Ans.

In this case concatenation and a kind of multiple times of string comes into action. Concatenation needed **+** operator while multiple time needed '\*' operator.

In case of methods there are a method which do the same:

**add**and **mul**

print('shivansh'**+**'jayara')

print('shivansh '**\***2)

print(str**.**\_\_add\_\_('shivansh','jayara'))

print(str**.**\_\_mul\_\_('shivansh ',2))

shivanshjayara

shivansh shivansh

shivanshjayara

shivansh shivansh

**8. If you are going to use the index method to locate a substring, what is the advantage of first testing the target string by using in or not in?**

Ans.

When we first check the target string using **in** or **not in** containment operator then we could get to know whether that particular target string is present or not. If it is not present the it will give True as a result else will False. Its better to check before becaue if we are not testinng it first and the substring is not present in the string then it will result in an exception

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

Ans.

**contains**is a built in method which produces the Boolean values in result.

Others are also some methods which result the boolean values: 'isalnum','isalpha', 'isascii','isdecimal','isdigit','isidentifier', 'islower', 'isnumeric','isprintable', 'isspace', 'istitle', 'isupper',

The operator which used in this is **'in'** which is also called as containment operator