Training Report Day-4

10 June 2024

❖ TUPLES IN PYTHON:-

In Python, a tuple is an ordered, immutable collection of elements. Tuples are similar to lists, but the key difference lies in their immutability. Once a tuple is created, its elements cannot be changed or modified. This makes tuples suitable for situations where data should remain constant throughout the program.

✓ CREATING TUPLES:-

Tuples can be created using parentheses () and separating elements with commas. Even a single element should be followed by a comma to distinguish it as a tuple.

Example:-

```
# Creating an empty tuple
empty_tuple = ()
# Creating a tuple with elements
fruits = ('apple', 'orange', 'banana')
# Single element tuple single_element_
tuple = ("42",) type(single_element_tuple)
```

✓ The tuple() Constructor

It is also possible to use the tuple() constructor to make a tuple.

Example:-

```
#Using the tuple() method to make a tuple:
 mytuple = tuple(("apple", "banana", "cherry"))
# note the double round-brackets
 print(mytuple)
type(mytuple)
```

✓ Accessing Elements:-

Accessing elements in a tuple is similar to lists, using indexing. Indexing starts from 0 for the first element.

Example:-

```
fruits = ('apple', 'orange', 'banana')
```

fruits[1]

- ✓ **Index() Method:-**

The Index () method returns the first occurrence of the given element from the tuple.

Syntax: tuple. Index (element, start, and end)

Parameters:

Element: The element to be searched.

Start (Optional): The starting index from where the searching is started

End (Optional): The ending index till where the searching is done

Note: This method raises a Value Error if the element is not found in the tuple.

Example:-

```
# Creating tuples
Tuple = (0, 1, 2, 3, 2, 3, 1, 3, 2)
# getting the index of 3
res = Tuple.index(3)
print('First occurrence of 3 is', res)
# getting the index of 3 after 4th
# index res = Tuple.index(3, 4, 6)
print('First occurrence of 3 after 4th index is:', res)
```

- ✓ **Check if Item Exists:-**

To determine if a specified item is present in a tuple use the "in" keyword:

Example:-

```
#Check if "Ludhiana" is present in the tuple:
City = ("DELHI", "MUMBAI", "KOTA", "LUDHIANA")
if "LUDHIANA" in City:
print("Yes, 'LUDHIANA' is in the city tuple")
else:
print("No, 'Ludhiana' is not in city tuple")
```

- ✓ **Advantages of Tuples:**

Immutability: Tuples provide data integrity by ensuring that the data remains constant.

Performance: Tuples are generally faster than lists for certain operations due to their immutability.

Valid Dictionary Key: Tuples can be used as keys in dictionaries, unlike lists.