**Q1. What are the characteristics of the tuples? Is tuple immutable?**

**Solution:**

In Python, a tuple is an ordered collection of elements, which can be of any data type. Tuples are defined using parentheses () and separated by commas.

The key characteristics of tuples are:

1.) Tuples are immutable: Once a tuple is created, its elements cannot be modified. However, you can create a new tuple by concatenating two or more tuples together.

2.) Tuples can contain elements of different data types: You can have a tuple that contains elements of different data types, such as strings, numbers, and booleans.

3.) Tuples are ordered: The order of elements in a tuple is preserved and each element can be accessed by its index.

4.) Tuples can be nested: You can create tuples that contain other tuples as elements, making them a useful data structure for representing hierarchical data.

5.) Tuples are hashable: Tuples are hashable, which means they can be used as keys in dictionaries and elements in sets.

So, we can say that, yes, tuples are immutable in Python. Once a tuple is created, its elements cannot be modified.

**Q2. What are the two tuple methods in python? Give an example of each method. Give a reason why tuples have only two in-built methods as compared to Lists.**

**Solution:**

In Python, there are two built-in methods for tuples:

1. Count() Method 2.) Index() Method

1.) count() method: This method returns the number of times a specified element appears in a tuple.

Example: T1=(1,2,3,5,4,3,5,3,5,7,8,4,6)

T1.count(3)

**Output:** 3

2.) index() method: This method returns the index of the first occurrence of a specified element in a tuple.

Example: T2=(“Apple”,”Banana”,”Mango”,”Grapes”)

T1.count(“Grapes”)

**Output:** 3

The reason tuples have only two built-in methods compared to lists is because tuples are designed to be immutable, meaning they cannot be changed once created. As a result, there is no need for methods that modify the contents of a tuple, such as append() or remove(), which are available in lists. The two methods provided for tuples are useful for working with immutable data structures, such as finding the index of an element or counting the occurrences of an element, without changing the tuple itself.

**Q3. Which collection datatypes in python do not allow duplicate items? Write a code using a set to remove duplicates from the given list.**

List = [1, 1, 1, 2, 1, 3, 1, 4, 2, 1, 2, 2, 2, 3, 2, 4, 3, 1, 3, 2, 3, 3, 3, 4, 4, 1, 4, 2, 4, 3, 4, 4]

**Solution:**

In Python, the set collection data type does not allow duplicates. A set is an unordered collection of unique elements. When you create a set, any duplicate elements are automatically removed. This means that each element in a set must be unique.

List1=[1,1,1,2,1,3,1,4,2,1,2,2,2,3,2,4,3,1,3,2,3,3,3,4,4,1,4,2,4,3,4,4]

set(List1)

**Output:** {1,2,3,4}

**Q4. Explain the difference between the union() and update() methods for a set. Give an example of each method.**

**Solution:**

In Python, union() and update() are both methods of the set collection that are used to combine two or more sets. The key difference between the two methods is that union() creates a new set that contains all the elements from the original sets, whereas update() modifies an existing set by adding all the elements from the other sets to it.

Here are examples of each method:

**union() method:**

set1 = {1, 2, 3}

set2 = {3, 4, 5}

set3 = set1.union(set2)

print(set3)

**Output:** {1,2,3,4,5}

**update() method:**

set1 = {1, 2, 3}

set2 = {3, 4, 5}

set1.update(set2)

print(set1)

**Output:** {1,2,3,4,5}

**Q5. What is a dictionary? Give an example. Also, state whether a dictionary is ordered or unordered.**

**Solution:**

In Python, a dictionary is a collection of key-value pairs, where each key is unique and associated with a value. Dictionaries are denoted by curly braces {} and each key-value pair is separated by a colon :. The keys in a dictionary must be immutable data types, such as strings or numbers, while the values can be any type of object.

Here is an example of a dictionary:

dict1 = {'apple': 2, 'banana': 4, 'orange': 3}

Dictionaries in Python are unordered, meaning that the items in a dictionary do not have a specific order. The order in which the items are inserted into the dictionary is not guaranteed to be the same as the order in which they are retrieved.

**Q6. Can we create a nested dictionary? If so, please give an example by creating a simple one-level nested dictionary.**

**Solution:**

Yes, in Python, we can create nested dictionaries where a dictionary can have another dictionary as its value. A nested dictionary is useful when we need to store data in a hierarchical format.

Here is an example of a simple one-level nested dictionary:

dic1={"Name":{"Employee\_Name1":"Harshit", "Employee\_Name2": "Rohit","Employee\_Name3": "Ritesh"},"Experience":{"Harshit": 3, "Rohit": 6, "Ritesh": 4}}

**Q7.) Using setdefault() method, create key named topics in the given dictionary and also add the value of the key as this list ['Python', 'Machine Learning’, 'Deep Learning']**

**dict1 = {'language' : 'Python', 'course': 'Data Science Masters'}**

**Solution:**

dict1 = {'language': 'Python', 'course': 'Data Science Masters'}

topics\_list = ['Python', 'Machine Learning', 'Deep Learning']

# add a new key 'topics' with value 'topics\_list' using setdefault()

dict1.setdefault('topics', topics\_list)

print(dict1)

**# Output:** {'language': 'Python', 'course': 'Data Science Masters', 'topics': ['Python', 'Machine Learning', 'Deep Learning']}

**Q8. What are the three view objects in dictionaries? Use the three in-built methods in python to display these three view objects for the given dictionary.**

**dict1 = {'Sport': 'Cricket' , 'Teams': ['India', 'Australia', 'England', 'South Africa', 'Sri Lanka', 'New Zealand']}**

**Solution:**

In Python, dictionaries have three view objects, which are essentially dynamic views on the dictionary's keys, values, and key-value pairs. These views reflect any changes made to the dictionary, as they are updated dynamically. The three view objects are:

dict\_keys: This view object contains the keys of the dictionary.

dict\_values: This view object contains the values of the dictionary.

dict\_items: This view object contains the key-value pairs of the dictionary.

Here is an example of how to display these view objects for the given dictionary:

dict1 = {'Sport': 'Cricket', 'Teams': ['India', 'Australia', 'England', 'South Africa', 'Sri Lanka', 'New Zealand']}

# get the view object for keys

keys\_view = dict1.keys()

print(keys\_view) # output: dict\_keys(['Sport', 'Teams'])

# get the view object for values

values\_view = dict1.values()

print(values\_view) # output: dict\_values(['Cricket', ['India', 'Australia', 'England', 'South Africa', 'Sri Lanka', 'New Zealand']])

# get the view object for key-value pairs

items\_view = dict1.items()

print(items\_view) # output: dict\_items([('Sport', 'Cricket'), ('Teams', ['India', 'Australia', 'England', 'South Africa', 'Sri Lanka', 'New Zealand'])])

In this example, we use the keys(), values(), and items() methods to get the three view objects for the dict1 dictionary. We then print each view object to display its contents. Note that the dict\_keys, dict\_values, and dict\_items view objects are not lists, but they can be converted to lists using the list() function if needed.