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

ANS) A tuple is an ordered, immutable, and heterogeneous collection of elements, often used to store related values. In Python, tuples are denoted by parentheses ( ) and their elements can be of any type, including other tuples. Some characteristics of tuples are:

· Ordered: The elements of a tuple are ordered, meaning that they have a defined position and the position determines their value.

· Immutable: Once created, the elements of a tuple cannot be changed, added or removed, making tuples immutable.

· Heterogeneous: Tuples can contain elements of different types, such as integers, strings, lists, and other tuples.

· Indexed: Tuples elements can be accessed using their indices, just like in lists.

· Hashable: Tuples can be used as keys in dictionaries, because they are hashable, unlike lists which are mutable and therefore cannot be used as keys.

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.

ANS)

There are two built-in methods in Python for tuples:

->count(): The count() method returns the number of times a specific element appears in a tuple.

Example:

>>> my_tuple = (1, 2, 3, 2, 1)

>>> my_tuple.count(2) 2

->index(): The index() method returns the index of the first occurrence of a specific element in a tuple.

Example:

>>> my_tuple = (1, 2, 3, 2, 1)

>>> my_tuple.index(2) 1

Tuples have only two built-in methods compared to lists because tuples are intended to be simple, immutable data structures, while lists are intended to be more flexible and mutable. Tuples are designed to be efficient in terms of memory usage and performance, and providing a large number of methods would make them less efficient and less useful for their intended purpose. The two methods that are provided, count() and index(), are the most commonly used operations on tuples and are sufficient for most use cases.

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]

ANS)

In Python, the two collection data types that do not allow duplicate items are:

1. Sets: Sets are unordered collections of unique elements, represented using curly braces { }. Sets automatically remove duplicates and do not allow you to insert duplicates.

2. **Dictionaries: Dictionaries are collections of key-value**

pairs, where each key must be unique. In Python, dictionaries are represented using curly braces { } and each key-value pair is separated by a colon :

```
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]



new_list=[]

for i in List:

    if i not in new_list:

        new_list.append(i)

print(new_list)
```

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

ANS)

The union() and update() methods are used to combine two sets in Python.

union(): The union() method returns a set that contains all the unique elements from both sets, without adding any duplicates.

Example:

>>> set1 = {1, 2, 3}

>>> set2 = {2, 3, 4}

>>> set1.union(set2)

{1, 2, 3, 4}

update(): The update() method updates the set with the union of itself and another set, effectively adding all the elements from the other set to the set.

Example:

>>> set1 = {1, 2, 3}

>>> set2 = {2, 3, 4}

>>> set1.update(set2)

>>> print(set1)

a{1, 2, 3, 4}

So, the main difference between union() and update() is that union() returns a new set containing the elements from both sets, while update() modifies the original set to include the elements from the other set.

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

ANS)

A dictionary is a collection of key-value pairs in Python. In a dictionary, keys are used to access values, and each key must be unique. Dictionaries are represented using curly braces { } and each key-value pair is separated by a colon :.

Example:

>>> my_dict = {'a': 1, 'b': 2, 'c': 3}

>>> print(my_dict['b']) 2

Dictionaries are unordered collections, meaning that they do not maintain the order of the key-value pairs. This means that when you access items in a dictionary, you cannot rely on the order of the items.

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

ANS)

Yes, you can create a nested dictionary in Python. A nested dictionary is a dictionary that contains one or more dictionaries as values. Here is an example of a simple one-level nested dictionary:

pythonCopy code

```
>>> nested_dict = {'dict1': {'a': 1, 'b': 2}, 'dict2': {'c': 3, 'd': 4}}

 >>> print(nested_dict) {'dict1': {'a': 1, 'b': 2}, 'dict2': {'c': 3, 'd': 4}}

 >>> print(nested_dict['dict1'])

{'a': 1, 'b': 2}
```

In this example, nested_dict is a dictionary that contains two dictionaries as values, dict1 and dict2. You can access the values of the nested dictionaries using the keys of the outer dictionary, as shown in the example.

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'}

ANS)

The setdefault() method is used to set the default value of a key in a dictionary, if it does not already exist. Here is how you can use the setdefault() method to create the key 'topics' in the dictionary dict1 and add the value as the list ['Python', 'Machine Learning', 'Deep Learning']:

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

```
 >>> dict1.setdefault('topics', ['Python', 'Machine Learning', 'Deep Learning'])

>>> print(dict1)

{'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']}

ANS)

In dictionaries, there are three view objects: keys(), values(), and items().

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

>>> keys = dict1.keys()

>>> values = dict1.values()

>>> items = dict1.items()

>>>

>>> print("Keys:", keys)

>>> print("Values:", values)

>>> print("Items:", items)
```

Output:

```
Keys: dict_keys(['Sport', 'Teams'])

Values: dict_values(['Cricket', ['India', 'Australia', 'England', 'South Africa',
```

'Sri Lanka', 'New Zealand']])

Items: dict_items([('Sport', 'Cricket'), ('Teams', ['India', 'Australia', 'England', 'South Africa', 'Sri Lanka', 'New Zealand'])])