TYPE-A:TUPLES:CH-12

1) Discuss the utility and significance of tuples, briefly.

sol:
Tuples in Python are ordered, immutable collections that can store elements of different types. They are useful when you want to group data together without allowing changes, ensuring data integrity. Tuples are significant because they are faster than lists, can be used as keys in dictionaries, and are ideal for representing fixed collections of items like coordinates, dates, or configuration values.

2) If a is (1, 2, 3)

1. what is the difference (if any) between a * 3 and (a, a, a) ?
2. Is a * 3 equivalent to a + a + a ?
3. what is the meaning of a[1:1] ?
4. what is the difference between a[1:2] and a[1:1] ?

sol:
1. a * 3 repeats the elements of the tuple three times to give (1, 2, 3, 1, 2, 3, 1, 2, 3), while (a, a, a) creates a new tuple containing three references to the original tuple: ((1, 2, 3), (1, 2, 3), (1, 2, 3)).
2. Yes, a * 3 is equivalent to a + a + a; both produce (1, 2, 3, 1, 2, 3, 1, 2, 3).
3. a[1:1] returns an empty tuple () because the start and end index are the same.
4. a[1:2] returns a tuple with the element at index 1 (2,), while a[1:1] returns an empty tuple (). The difference is that one selects an element, the other selects no elements.

3) Does the slice operator always produce a new tuple ?

sol:
Yes, the slice operator on a tuple always produces a new tuple. It does not modify the original tuple, since tuples are immutable. The new tuple contains the selected elements from the original tuple based on the slice indexes.

4) The syntax for a tuple with a single item is simply the element enclosed in a pair of matching parentheses as shown below :
t = ("a")
Is the above statement true? Why? Why not ?

sol:
The statement is not true. Writing t = ("a") does not create a tuple, it creates a string because parentheses alone are not enough to define a single-element tuple. To create a tuple with one element, you must include a comma after the element: t = ("a",). The comma distinguishes it as a tuple.

5) Are the following two assignments same ? Why / why not ?
1.
```
  T1 = 3, 4, 5
  T2 = ( 3, 4 , 5)
```
2.
```
  T3 = (3, 4, 5)
  T4 = (( 3, 4, 5))
```

sol:

- T1 = 3, 4, 5 and T2 = (3, 4, 5) are the same. In Python, tuples can be created with or without parentheses, so both produce the tuple (3, 4, 5).

- T3 = (3, 4, 5) and T4 = ((3, 4, 5)) are also the same. The extra parentheses in T4 do not create a nested tuple; they are just grouping parentheses. Both result in (3, 4, 5).

- All four assignments produce identical tuples.

6) What would following statements print? Given that we have tuple= ('t', 'p', 'l')

1. print("tuple")
2. print(tuple("tuple"))
3. print(tuple)

sol:
1. tuple
2. ('t', 'u', 'p', 'l', 'e')
3. ('t', 'p', 'l')

7) How is an empty tuple created ?

sol:
An empty tuple is created using a pair of empty parentheses (). You can also use the tuple() function without any arguments to create an empty tuple.

8) How is a tuple containing just one element created ?

sol:
A tuple containing just one element is created by placing a comma after the element inside parentheses. The comma is required; otherwise, Python will treat it as a single value, not a tuple.

9) How can you add an extra element to a tuple ?

sol:
Tuples are immutable, so you cannot directly add an element to an existing tuple. To "add" an element, you create a new tuple by concatenating the original tuple with a tuple containing the new element.

10) When would you prefer tuples over lists ?

sol:
Tuples are preferred over lists when you want to store a fixed collection of items that should not be modified, ensuring data integrity. They are also useful when you need faster performance or want to use the sequence as a key in a dictionary, because tuples are immutable and hashable, unlike lists.

11) What is the difference between (30) and (30,) ?

sol:
(30) is not a tuple, it is just the number 30 enclosed in parentheses. Python treats it as an integer.

(30,) is a tuple with a single element 30. The comma is required to tell Python that it is a tuple.

12) When would sum( ) not work for tuples ?

sol:
The sum() function will not work for tuples that contain non-numeric elements or a mix of numbers and non-numbers. It only works when all elements in the tuple are numbers (integers or floats).

13) Do min( ), max( ) always work for tuples ?

sol:
No, min() and max() do not always work for tuples. They work only if all elements in the tuple are comparable (e.g., all numbers or all strings). If the tuple contains mixed or non-comparable types like numbers and strings, Python will raise a TypeError.

14) Is the working of in operator and tuple.index( ) same ?

sol:
No, the working of in and tuple.index() is not the same.

- The in operator checks whether an element exists in the tuple and returns True or False.

- The tuple.index() method returns the index of the first occurrence of the element in the tuple. If the element is not found, it raises a ValueError.

15) How are in operator and index( ) similar or different ?

sol:

Feature	**in Operator**	**index() Method**
Purpose	Checks if an element exists	Finds the index of an element
Return Value	Boolean (True / False)	Integer (position of element)
Error on Missing	No error, returns False	Raises ValueError
Usage Example	20 in t → True	t.index(20) → 1