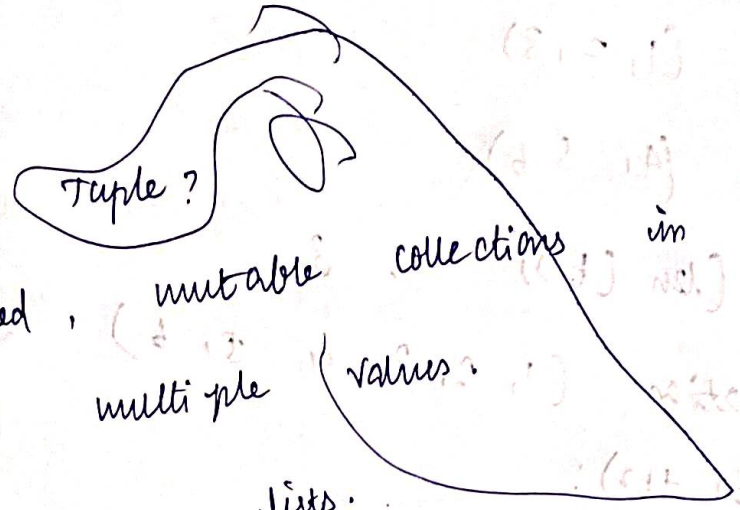
14/07/2025

Tuple

Why Tuples faster than lists ?

1. immutability
2.



What is Tuple ?

- Ordered, immutable collections in python that can hold multiple values.
- similar to lists.
- but cannot be changed.
- allow storing different data types,
- faster than lists.

Syntax:-

my_tuple = (1, 2, 3, "Hello", 4.5).

1- creating a tuple ?

- parantheses ()
- store elements of different data types.

empty tuple = ().

Single element tuple = (5,)

multi_tuple = (10, "python", 3.14)

Even though tuples use paranthesis, they can be created without them.

2. Python tuple operations:

- tuples support indexing, slicing, concatenation. just like lists:

t1 = (1, 2, 3)

t2 = (4, 5, 6)

print (len (t1)) : 3

Concatenation (1, 2, 3, 4, 5, 6)

Print (t1 + t2):

Print (t1 * 2) : (1, 2, 3, 1, 2, 3)

Repetition.

3. Accessing elements in tuples:

- use zero based indexing to access elements.

t = ("apple", "banana", "cherry")

Print (t[0]) : apple.

print (t[2]) : cherry.

4. Concatenation:

Tuples can be joined using the + operator

Ex. t1 = (1, 2, 3)

t2 = (4, 5, 6)

t3 = t1 + t2

print (t3) : (1, 2, 3, 4, 5, 6)

5. Slicing of Tuples:

tuple [start : end : step]

ex. t = (10, 20, 30, 40, 50, 60)

print (t[1:4]) = (20, 30, 40)

print (t[:3]) = (10, 20, 30)

print (t[::2]) = (10, 30, 50)

6. Deleting a tuple:

You can delete individual elements, since tuple is immutable. but you can delete entire tuple using del.

t = (1, 2, 3)

del t

print (t) This will throw error, because it is deleted.

7. Tuple built in methods:

Tuples have limited methods since they are immutable.

Common tuple methods:

t = (1, 2, 3, 4, 1, 2, 1)

print (t.count(1)) : 3.

Counts occurence of 1. print (t.index(3))

output : 2.

8. Tuple Built in Function:

t = (10, 5, 20, 8)

print (len(t)) : output : 4.

print (max(t)) : 20.

print (min(t)) : 5

print (sum(t)) : 43.

feature	Tuple (tuple)	List (list)
Mutability	immutable	Mutable
Speed	faster	slower
memory use	less memory	more memory
methods	limited	more methods.

Summary:

- Tuple is an immutable collection of items
- Elements can be accessed using indexing & slicing.
- Concatenation & repetition
- count() and index() are available.
- use less memory, and faster than list