4) Tuple

- **Understanding how generators work in Python.**

A tuple is an ordered, immutable sequence data type in Python. It is similar to a list but cannot be modified after creation.

Characteristics of Tuples:

- Ordered: Elements maintain their position (index-based access).
- Immutable: Once created, elements cannot be added, removed, or changed.
- Heterogeneous: Can store elements of different data types.
- Hashable (if all elements are hashable): Can be used as keys in dictionaries.
- Memory Efficient: Typically consumes less memory than lists due to immutability.

Syntax :

```
t = (1, 2, 3)
t = (42,)
t = tuple([1, 2, 3])
```

Immutability in Tuples :-

What Does Immutability Mean?

- Once a tuple is created, its structure and elements cannot be altered.
- No methods exist to modify the tuple (unlike lists, which have append(), remove(), etc.).
- Attempting to modify a tuple raises a TypeError.

- ## Creating and accessing elements in a tuple.

Syntax for Creating Tuples

Tuples are defined using parentheses ( ) (optional in some cases) with elements separated by commas.

Methods of Creating Tuples:

1. **Standard Tuple**
   my_tuple = (1, 2, 3, "hello")

Single-Element Tuple (Requires a Trailing Comma)

single_tuple = (4,) # Without comma, Python treats it as an integer.

Empty Tuple

empty_tuple = ()

2. **Accessing Tuple Elements**

A. Indexing (Zero-Based)

- Tuples support positive (left-to-right) and negative (right-to-left) indexing.
- Syntax: tuple[index]

- **Basic operations with tuples: concatenation, repetition, membership.**

- **concatenation (+)**

  - the + operator merges two tuples into a new tuple; originals stay unchanged
  - both operands must be tuples (trying to concatenate a list or other type raises typeerror)
  - you can also use +=, which under the hood creates a new tuple and reassigns it .

```
tuple1 = (1, 2, 3)

tuple2 = (4, 5)


result = tuple1 + tuple2

print(result)
```

- **repetition (*)**
  - the * operator repeats the entire tuple's contents n times, returning a new tuple
  - repeating a single-element tuple works too, e.g. (10,) * 5 → (10, 10, 10, 10, 10) .
  - underlying tuples are immutable so repetition just builds a new sequence.

```
colors = ("red", "green")

result = colors * 3

print(result)

# Output: ('red', 'green', 'red', 'green', 'red', 'green')
```

- **membership (in, not in)**
  - x in tup returns true if x exists in the tuple; otherwise false. similarly, not in checks the inverse .
  - internal check uses a linear search, so it's O(n).

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


print("banana" in fruits)      # Output: True

print("grape" not in fruits)   # Output: True
```