Chapter 10 - 5Tuples

Theory:

Tuples are immutable sequences, similar to lists but cannot be changed after creation.

Code Example:

```
# ----------------------------------------
# ■ TUPLES IN PYTHON
# ----------------------------------------
# ■ A tuple is a **collection of ordered values**
# ■ Tuples can store **any data type**
# ■ Tuples are **immutable** — cannot be changed after creation
# ■ Use parentheses `()` to define tuples
# ----------------------------------------
# ■ CREATING TUPLES
# ----------------------------------------
# ■ 1. Regular tuple
t = (1, 2, 3)
print(type(t)) #
# ■ 2. Single-element tuple → must include a comma
t = (5,)
print(type(t)) #
# ■ Without comma → it's just an int
not_tuple = (5)
print(type(not_tuple)) #
# ■ 3. Tuple with mixed data types
info = (1, 45, 342, False, "rohan", "milan")
print(info)
# ----------------------------------------
# ■ IMMUTABILITY
# ----------------------------------------
# ■ Cannot change values in a tuple
# info[0] = 999 → TypeError
# ----------------------------------------
# ■ COMMON TUPLE METHODS
# ----------------------------------------
# ■ .count(value) → Count occurrences of a value
print(info.count(45)) # Output: 1
# ■ .index(value) → First index of a value
print(info.index("milan")) # Output: 5
# ■ len() → Total number of elements
print(len(info)) # Output: 6
# ■ max() and min() → Only for tuples with same data type (e.g., all numbers)
num_tuple = (10, 5, 99, 1)
print(max(num_tuple)) # Output: 99
print(min(num_tuple)) # Output: 1
# ■ sum() → Sum of elements (only numeric)
print(sum(num_tuple)) # Output: 115
# ■ sorted() → Returns a sorted list (does NOT modify original tuple)
print(sorted(num_tuple)) # Output: [1, 5, 10, 99]
print(num_tuple) # Original remains unchanged
# ■ tuple() → Convert other iterables to a tuple
lst = [1, 2, 3]
converted = tuple(lst)
```

```
print(converted) # Output: (1, 2, 3)
# ■ 'in' keyword → Check if element exists
print("milan" in info) # Output: True
# ■ Loop through tuple
for item in info:
print(item)
# ----------------------------------------
# ■ TUPLE CONCATENATION IN PYTHON
# ----------------------------------------
# ■ You can use the `+` operator to concatenate (join) tuples
# ■ Since tuples are immutable, it returns a **new tuple**
# ■ Original tuples remain unchanged
# Example:
t1 = (1, 2, 3)
t2 = (4, 5, 6)
# Concatenate
result = t1 + t2
print("Concatenated Tuple:", result) # Output: (1, 2, 3, 4, 5, 6)
# Original tuples remain unchanged
print("Original t1:", t1) # Output: (1, 2, 3)
print("Original t2:", t2) # Output: (4, 5, 6)
# ■ This will throw an error:
# print(t1 + [7, 8])
# ■ Convert to tuple first:
print(t1 + tuple([7, 8])) # Output: (1, 2, 3, 7, 8)
# ----------------------------------------
# ><■ TUPLE SLICING IN PYTHON
# ----------------------------------------
# ■ You can access parts of a tuple using slicing
# ■ Syntax: tuple[start : end : step]
# - Includes start index
# - Excludes end index
# - Step (optional) tells how many elements to skip
# Example tuple
t = ("milan", "rohan", "apple", "banana", "mango", "grapes", 42)
# ----------------------------------------
# ■ Basic Slicing
# ----------------------------------------
print(t[0:3]) # Output: ('milan', 'rohan', 'apple')
print(t[2:5]) # Output: ('apple', 'banana', 'mango')
# ----------------------------------------
# ■ Slicing from beginning or to end
# ----------------------------------------
print(t[:4]) # Output: ('milan', 'rohan', 'apple', 'banana')
print(t[3:]) # Output: ('banana', 'mango', 'grapes', 42)
# ----------------------------------------
# ■ Negative Indexing
# ----------------------------------------
# Negative index means count from end
# t[-1] = 42, t[-2] = 'grapes', etc.
print(t[-3:]) # Output: ('mango', 'grapes', 42)
print(t[-5:-2]) # Output: ('apple', 'banana', 'mango')
# ----------------------------------------
```

```
# ■ Slicing with Step
# ----------------------------------------
print(t[1:6:2]) # Output: ('rohan', 'banana', 'grapes')
print(t[::2]) # Output: ('milan', 'apple', 'mango', 42)
# Reverse the tuple using slicing:
print(t[::-1]) # Output: (42, 'grapes', 'mango', 'banana', 'apple', 'rohan', 'milan')
# ----------------------------------------
# ■ ADVANTAGES OF TUPLES
# ----------------------------------------
# ■ Safe → Data cannot be modified by accident
# ■ Faster than lists
# ■ Used for fixed data like coordinates, months, etc.
# ■ Can be used as keys in dictionaries (lists cannot)
# ----------------------------------------
# ■ TUPLE VS LIST - Quick Recap
# List:
# ■ Mutable
# ■ Can't be used as dictionary key
# ■ Slower and uses more memory
# Tuple:
# ■ Immutable
# ■ Can be used as dictionary key
# ■ Faster and more memory-efficient
```