**🧾 Python Tuples - Complete Notes**

**✅ What is a Tuple?**

A **tuple** is a collection of Python objects **ordered** and **immutable** (cannot be changed after creation).

**✅ Tuple Characteristics**

* Ordered (elements have a fixed position)
* Immutable (can’t add, remove, or change elements)
* Can contain mixed data types
* Allow duplicate values
* Can be nested (tuples inside tuples)
* Use **round brackets** ()

**✅ Why Use Tuples Instead of Lists?**

| **Feature** | **Tuple** | **List** |
| --- | --- | --- |
| Mutability | Immutable | Mutable |
| Performance | Faster (due to immutability) | Slower |
| Use Case | Fixed data (e.g., coordinates) | Changeable data |
| Hashable | Yes (can be used as dict keys) | No |

**Use tuple** when you want to **ensure data cannot be changed**.

**✅ Tuple Syntax & Examples**

# Creating a tuple

t1 = (1, 2, 3)

t2 = ("apple", "banana", "cherry")

t3 = (1, "hi", True, 4.5)

# Single element tuple (note the comma)

t4 = (5,)

# Tuple without parenthesis (tuple packing)

t5 = 10, 20, 30

# Nested tuple

t6 = (1, (2, 3), 4)

print(t1, t2, t3, t4, t5, t6)

**✅ Accessing Tuple Elements**

t = (10, 20, 30, 40, 50)

# Indexing

print(t[0]) # Output: 10

print(t[-1]) # Output: 50

# Slicing

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

**✅ Tuple Immutability Example**

t = (1, 2, 3)

# t[0] = 10 ❌ Error: 'tuple' object does not support item assignment

**✅ Tuple Built-in Functions with Examples**

| **Function** | **Description** | **Example** |
| --- | --- | --- |
| len() | Returns number of elements | len((1,2,3)) → 3 |
| max() | Returns max value (only same types) | max((1,2,3)) → 3 |
| min() | Returns min value | min((1,2,3)) → 1 |
| sum() | Returns sum of numbers | sum((1,2,3)) → 6 |
| any() | Returns True if at least one is True | any((0, 0, 1)) → True |
| all() | Returns True if all are True | all((1, 2, 3)) → True |
| tuple() | Converts an iterable to tuple | tuple([1, 2]) → (1, 2) |
| count(x) | Count occurrences of x | (1,2,2,3).count(2) → 2 |
| index(x) | Return index of first occurrence | (1,2,3).index(2) → 1 |

**✅ Looping Through a Tuple**

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

for item in t:

print(item)

**✅ Tuple Unpacking**

t = (1, 2, 3)

a, b, c = t

print(a, b, c) # Output: 1 2 3

**✅ When to Use Tuples**

* Data that **shouldn’t change**
* As **dictionary keys**
* Returning **multiple values** from a function
* **Faster** than lists (performance boost)
* **Safer** than lists (less prone to accidental changes)

es, the in keyword **works with tuples** in Python. It is used to **check if a value exists within the tuple**, and it returns a boolean value (True or False).

**✅ Syntax:**

value in tuple

**✅ Example:**

fruits = ('apple', 'banana', 'cherry')

print('banana' in fruits) # True

print('orange' in fruits) # False

**✅ Explanation:**

* 'banana' in fruits → ✅ returns True because 'banana' is present in the tuple.
* 'orange' in fruits → ❌ returns False because 'orange' is not present.

**✅ Why use in with tuple?**

* To **check existence** of an element before using it.
* Useful in **conditionals** and **loop filters**.

**✅ Bonus: Use in if-condition**

if 'apple' in fruits:

print("Yes, it's in the list.")