3. What Are Tuples?

- A **Tuple** is a **data type** in Python, like strings or lists.

Syntax: defined with **parentheses** `( )`.

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

-
- **Types of brackets in programming**:
  - `( )` → Parentheses
  - `[ ]` → Square brackets (lists)
  - `{ }` → Curly braces (dict, set)

---

4. Key Property

Tuples are **immutable** → cannot be changed after creation.

```
spices = ("cardamom", "clove", "cinnamon")
# You CANNOT do: spices[0] = "ginger" ❌
```

-

---

5. Example: Masala Spices

```
masala_spices = ("cardamom", "clove", "cinnamon")
```

- These values are fixed → once defined, they can't be changed.

---

6. Tuple Unpacking

Assigning tuple values into multiple variables at once.

```
spice1, spice2, spice3 = masala_spices
print(spice1, spice2, spice3)
# Output: cardamom clove cinnamon
```

-
- Condition: the **number of variables must match the number of values**.

---

7. Tuple with Ratios

Tuples can store numeric values, like proportions:

```
ginger_ratio, cardamom_ratio = (2, 1)
print(ginger_ratio, cardamom_ratio)
# Output: 2 1
```

-
- Behind the scenes, Python is unpacking the tuple `(2, 1)`.

---

8. Swapping Variables

Python allows swapping values **without a temporary variable**.

```
ginger_ratio, cardamom_ratio = cardamom_ratio, ginger_ratio
```

-
- Very concise and **powered by tuples**.
- Example:
    - Before swap → ginger = 2, cardamom = 1
    - After swap → ginger = 1, cardamom = 2

---

9. Membership Test

Check if a value exists inside a tuple with `in`.

```
masala_spices = ("cardamom", "clove", "cinnamon")

print("ginger" in masala_spices)   # False
print("cinnamon" in masala_spices) # True
```

-

🔑 Summary

- Tuples are:
  - Created with `( )`
  - **Immutable** (cannot be changed)
  - Useful for storing fixed sets of values
  - Support **unpacking** into multiple variables
  - Enable **swapping variables easily**
  - Allow **membership checks** with `in`

👉 They are like **lists**, but *unchangeable*. Perfect for data you don't want to modify.