1) A **data type** defines the type of value a variable can hold. It tells the interpreter what kind of data is being stored and how it can be used.

**2) What is Mutable in Python?**

In Python, **mutable** means that the **object can be changed after it is created** — you can modify, add, or remove elements without creating a new object.

3) **What is Ordered in Python?**

In Python, **ordered** means that **the items in a collection are stored in a specific sequence**, and that sequence **is preserved** — you can access items by their **position (index)**.

|  |  |
| --- | --- |
| **Ordered** | Items are stored and retrieved in the same order as inserted – list, tuple |
| **Unordered** | No guarantee of order; items may appear in different positions- set |

| **Category** | **Data Type** | **Definition** |
| --- | --- | --- |
| **Numeric** | int | Integer numbers, both positive and negative, without decimals. Example: 10 |
|  | float | Real numbers with decimal points. Example: 3.14 |
|  | complex | Numbers with real and imaginary parts. Example: 3+5j |
| **Text** | str | Sequence of characters (text). Example: 'Hello' |
| **Sequence** | list | Ordered, mutable collection. Example: [1, 2, 3] |
|  | tuple | Ordered, immutable collection. Example: (1, 2, 3) |
|  | range | Represents a sequence of numbers. Example: range(1, 5) |
| **Set** | set | Unordered collection of unique items. Example: {1, 2, 3} |
|  | frozenset | Immutable version of set. |
| **Mapping** | dict | Collection of key-value pairs. Example: {'a': 1, 'b': 2} |
| **Boolean** | bool | Logical values: True or False. Used for conditional statements. |
| **None Type** | NoneType | Represents the absence of a value. Only one object: None. |

| **Data Type** | **Immutable?** | **Example** |
| --- | --- | --- |
| int | Yes | You can’t do a = 5; a[0] = 3 |
| float | Yes |  |
| bool | Yes |  |
| str | Yes | s = "hi" → s[0] = "H" ❌ |
| tuple | Yes |  |
| frozenset | Yes |  |

| **Data Type** | **Can You Change It After Creation?** | **Example** |
| --- | --- | --- |
| list | Yes | a = [1, 2]; a.append(3) |
| dict | Yes | d = {'a': 1}; d['b'] = 2 |
| set | Yes | s = {1, 2}; s.add(3) |

Ordered or not ordered:

| **Data Type** | **Ordered?** | **Example** |
| --- | --- | --- |
| list | Yes | a = [10, 20, 30] → a[0] = 10 |
| tuple | Yes | t = (1, 2, 3) → t[1] = 2 |
| str | Yes | s = "hello" → s[0] = 'h' |
| range | Yes | r = range(1, 4) → (1, 2, 3) |
| dict | Yes | Preserves insertion order: {'a': 1, 'b': 2} |
| set | no | {3, 1, 2} might display as {1, 2, 3} |

**List & Array**

**An array is a data structure used to store multiple values in a single variable, all of the same data type, and placed in contiguous memory locations.**

* ✅ **List** = Flexible, can hold **any type**, easier to change, but slower.
* ✅ **Array** = Faster, memory-efficient, but **strict in type** and less flexible.

| **Property** | **List (Python)** | **Array (array/Numpy)** |
| --- | --- | --- |
| **Size** | ✅ Dynamic (auto-resizes) | ❌ Fixed (usually pre-defined in low-level arrays) |
| **Data type** | ✅ Heterogeneous | ❌ Homogeneous (all elements same type) |
| **Insertion & Deletion** | ✅ Easier (more flexible) | ❌ Complex (especially in middle of array) |
| **Memory Efficiency** | ❌ Less efficient | ✅ More efficient (due to uniform types) |
| **Performance for math ops** | ❌ Slower (no vectorization) | ✅ Faster (especially with NumPy arrays) |
| **Flexibility** | ✅ Very high (mix types, grow/shrink) | ❌ Low (fixed type, fixed size unless re-created) |