**🐍 Python Objects, Mutability & Immutability**

**Python Data Types — Mutability Table**

| **Data Type** | **Example** | **Mutable?** | **Explanation** |
| --- | --- | --- | --- |
| int | x = 5 | ❌ Immutable | You cannot change the value in-place. Assigning a new value creates a new object. |
| float | pi = 3.14 | ❌ Immutable | Same behavior as int. |
| bool | flag = True | ❌ Immutable | Values like True or False cannot be changed. |
| str (string) | "hello" | ❌ Immutable | Cannot change characters of a string after creation. |
| tuple | (1, 2, 3) | ❌ Immutable | Contents cannot be changed, though if it contains mutable elements (like lists), they can be changed. |
| list | [1, 2, 3] | ✅ Mutable | You can add, remove, or modify elements. |
| dict (dictionary) | {"name": "Ali", "age": 25} | ✅ Mutable | Keys and values can be changed or added. |
| set | {1, 2, 3} | ✅ Mutable | Elements can be added or removed (but no duplicates). |
| frozenset | frozenset([1, 2, 3]) | ❌ Immutable | Like a set, but cannot be changed after creation. |
| complex | z = 2 + 3j | ❌ Immutable | Complex numbers are immutable like integers. |
| bytes | b'hello' | ❌ Immutable | Binary data — fixed once created. |
| bytearray | bytearray(b'hello') | ✅ Mutable | A mutable version of bytes. |
| range | range(5) | ❌ Immutable | Produces immutable sequence of numbers. |
| NoneType | None | ❌ Immutable | Represents a null value. Can't be changed. |

**✅ Notes on Dictionary (dict)**

* A **dictionary** is a key-value pair collection.
* **Mutable:** You can add, remove, or update keys/values.
* Example:

python

CopyEdit

person = {"name": "Talha", "age": 21}

person["age"] = 22 # updating value

person["city"] = "Lahore" # adding new key

**✅ Quick Summary:**

* ✅ **Mutable**: list, dict, set, bytearray
* ❌ **Immutable**: int, float, bool, str, tuple, frozenset, bytes, complex, range, NoneType

**✅ Everything in Python is an Object**

In Python, **everything is an object**: numbers, strings, lists, functions, even classes themselves.

Every object has **3 core properties**:

1. **Identity**
   * A unique address in memory.
   * Checked using id() function.
   * Identity **never changes** for an object during its lifetime.
2. **Type**
   * The type of the object (like int, str, list).
   * Checked using type() function.
   * Type **never changes**.
3. **Value**
   * The data stored in the object.
   * Can be **mutable** (changeable) or **immutable** (unchangeable).

Even if an object has an **empty value** (like None, empty list [], or empty string ""), it’s still an object with a unique identity and type.

**🔥 Mutable vs Immutable Objects**

**📝 Mutable**

* **Definition**: Objects whose value **can be changed** after creation **without changing their identity**.
* ✅ Example types: list, dict, set, bytearray
* Example:

my\_list = [1, 2, 3]

print(id(my\_list)) # Identity before change

my\_list.append(4) # Change value

print(id(my\_list)) # Identity remains the same

✔ The **value changed**, but the **identity didn’t**.

**🚫 Immutable**

* **Definition**: Objects whose value **cannot be changed** after creation. Any “change” results in **a new object with a new identity**.
* ✅ Example types: int, str, tuple, frozenset, bytes
* Example:

my\_str = "hello"

print(id(my\_str)) # Identity before change

my\_str += " world" # Creates a new string object

print(id(my\_str)) # Identity changed

❗ A “change” creates a **new object**.

**⚡ Identity Determines Mutability**

* You **never check type or value** to see if something is mutable.
* Instead, watch the **identity**:
  + If identity stays the same → ✅ mutable
  + If identity changes → 🚫 immutable

**🏁 Summary Table: Mutable vs Immutable**

| **Feature** | **Mutable** | **Immutable** |
| --- | --- | --- |
| Value Changeable | ✅ Yes | 🚫 No |
| Identity Changes? | ❌ No | ✅ Yes (on change) |
| Examples | list, dict | str, tuple |

**🧠 Pro Tip: How to check identity**

x = [1, 2, 3]

print(id(x))

x.append(4)

print(id(x)) # Same ID → mutable

y = "hello"

print(id(y))

y += " world"

print(id(y)) # New ID → immutable