In Python, **shallow copies** and **deep copies** refer to different ways of copying an object and its contents. The key difference lies in how they handle nested objects (objects that contain other objects, such as lists of lists or dictionaries containing lists).

**1. Shallow Copy**

A **shallow copy** creates a new object but does **not recursively copy** the objects inside it. Instead, it copies references to the original objects. This means that changes made to **mutable nested objects** in the copied version **will also reflect** in the original.

**Example of Shallow Copy**

import copy

original\_list = [[1, 2, 3], [4, 5, 6]]

shallow\_copied\_list = copy.copy(original\_list)

# Modify a nested list

shallow\_copied\_list[0][0] = 99

print(original\_list) # [[99, 2, 3], [4, 5, 6]]

print(shallow\_copied\_list) # [[99, 2, 3], [4, 5, 6]]

**Explanation**

* A new list shallow\_copied\_list is created, but it contains references to the same inner lists as original\_list.
* Changing shallow\_copied\_list[0][0] also affects original\_list[0][0] since both point to the same inner list.

**2. Deep Copy**

A **deep copy** creates a new object and **recursively copies** all objects inside it. This ensures that changes to the copied object do **not** affect the original.

**Example of Deep Copy**

import copy

original\_list = [[1, 2, 3], [4, 5, 6]]

deep\_copied\_list = copy.deepcopy(original\_list)

# Modify a nested list

deep\_copied\_list[0][0] = 99

print(original\_list) # [[1, 2, 3], [4, 5, 6]]

print(deep\_copied\_list) # [[99, 2, 3], [4, 5, 6]]

**Explanation**

* deepcopy() creates a completely **independent copy** of the original object.
* Changes in deep\_copied\_list do **not** affect original\_list.

**Key Differences Between Shallow Copy and Deep Copy**

| **Feature** | **Shallow Copy** | **Deep Copy** |
| --- | --- | --- |
| Copies only the reference of nested objects | ✅ Yes | ❌ No |
| Fully independent copy | ❌ No | ✅ Yes |
| Modifying the nested object affects the original | ✅ Yes | ❌ No |
| Performance (Speed & Memory Usage) | ✅ Faster | ❌ Slower |

**When to Use What?**

* Use **shallow copy (copy.copy())** if you want to **preserve references** to mutable objects.
* Use **deep copy (copy.deepcopy())** if you need **full independence** between the original and copied object.