Difference Between Shallow Copy and Deep Copy in Python

A **shallow copy** creates a new object but does not create copies of nested objects. Instead, it copies references to the original objects.

A **deep copy**, on the other hand, creates a completely independent copy, including nested objects.

Feature	Shallow Copy (`copy.copy()`)	DeepCopy(`copy.deepcopy()`)
Copy Method	copy.copy(obj)	copy.deepcopy(obj)
New Object Created?	✔ Yes	✔ Yes
Nested Objects Copied?	✘ No (only references copied)	✔ Yes (full copy)
Independent Modification?	✘ No (nested objects are shared)	✔ Yes (fully independent)

Shallow Copy Example

```
import copy

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

shallow_copied = copy.copy(original)

shallow_copied[0][0] = 99

print("Original:", original)

print("Shallow Copy:", shallow_copied)
```

Shallow Copy Output (unexpected change in original)

Original: [[99, 2, 3], [4, 5, 6]]

Shallow Copy: [[99, 2, 3], [4, 5, 6]]

Why? Because the nested lists are not copied, just referenced.

Deep Copy Example

```
import copy

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

deep_copied = copy.deepcopy(original)

deep_copied[0][0] = 99

print("Original:", original)

print("Deep Copy:", deep_copied)
```

Deep Copy Output (original remains unchanged)

Original: [[1, 2, 3], [4, 5, 6]]

Deep Copy: [[99, 2, 3], [4, 5, 6]]

Why? Because deepcopy() creates a new copy of everything, including nested lists.

When to Use Each?

✅ **Shallow Copy** when your object does not contain nested mutable structures or when you want changes to affect both copies.

✅ **Deep Copy** when you need a completely independent copy.