Python Concepts Report

Shallow vs Deep Copy and Multiple Inheritance

August 15, 2025

1 Shallow vs Deep Copy

1.1 Definition

Shallow Copy creates a new top-level object, but nested objects are still references to the originals. **Deep Copy** creates a fully independent copy, recursively copying all nested objects.

1.2 Examples

```
a = [[1, 2], [3, 4]]

# Shallow copy
b = a
b[0][0] = 99
print(a)  # [[99, 2], [3, 4]] -- inner list is shared

# Deep copy
c = a.copy()
c[0][0] = 42
print(a)  # [[99, 2], [3, 4]] -- unaffected
```

1.3 Comparison Table

Feature	Shallow Copy	Deep Copy
Copies nested objects	No	Yes
Speed	Faster	Slower
Memory usage	Less	More
Risk of shared mutation	High	None

2 Multiple Inheritance in Python

2.1 Definition

Multiple inheritance occurs when a class inherits from more than one parent class:

```
class Child(Parent1, Parent2):
    pass
```

2.2 Method Resolution Order (MRO)

Python determines which method to call using the **C3 linearization algorithm**. The order can be seen with:

```
print(Child.mro())
```

2.3 Child and Parent Have the Same Method

If a child class defines a method with the same name as its parent, the child's method overrides it.

```
class Parent:
    def greet(self):
        print("Hello from Parent")

class Child(Parent):
    def greet(self):
        print("Hello from Child")

Child().greet()  # "Hello from Child"
```

2.4 Two Parents Have the Same Parent (Diamond Problem)

```
class A:
    def show(self): print("A")

class B(A):
    def show(self): print("B")

class C(A):
    def show(self): print("C")

class D(B, C):
    pass

print(D.mro())
D().show()  # "B" because B comes first in MRO
```

2.5 Using `super()` in Multiple Inheritance

showing how it works

```
class A:
    def __init__(self): print("A init"); super().__init__()

class B(A):
    def __init__(self): print("B init"); super().__init__()

class C(A):
    def __init__(self): print("C init"); super().__init__()

class D(B, C):
    def __init__(self): print("D init"); super().__init__()
```

```
D()
# Output:
# D init
# B init
# C init
# A init
```