**Que 1 : Single, Multilevel, Multiple, Hierarchical, and Hybrid inheritance in Python.**

**1. Single Inheritance**

One parent → one child.

class Parent:

def display\_parent(self):

print("This is the Parent class.")

class Child(Parent): # Single inheritance

def display\_child(self):

print("This is the Child class.")

obj = Child()

obj.display\_parent()

obj.display\_child()

**2. Multilevel Inheritance**

One parent → child → grandchild.

class Grandparent:

def display\_gp(self):

print("This is the Grandparent class.")

class Parent(Grandparent):

def display\_parent(self):

print("This is the Parent class.")

class Child(Parent): # Multilevel inheritance

def display\_child(self):

print("This is the Child class.")

obj = Child()

obj.display\_gp()

obj.display\_parent()

obj.display\_child()

**3. Multiple Inheritance**

One child → inherits from multiple parents.

class Father:

def skill\_father(self):

print("Father knows driving.")

class Mother:

def skill\_mother(self):

print("Mother knows cooking.")

class Child(Father, Mother): # Multiple inheritance

def skill\_child(self):

print("Child knows painting.")

obj = Child()

obj.skill\_father()

obj.skill\_mother()

obj.skill\_child()

**4. Hierarchical Inheritance**

One parent → multiple children.

class Parent:

def display\_parent(self):

print("This is the Parent class.")

class Child1(Parent):

def display\_child1(self):

print("This is Child 1.")

class Child2(Parent):

def display\_child2(self):

print("This is Child 2.")

obj1 = Child1()

obj1.display\_parent()

obj1.display\_child1()

obj2 = Child2()

obj2.display\_parent()

obj2.display\_child2()

**5. Hybrid Inheritance**

Combination of two or more types. (Example: Multiple + Multilevel)

class Base:

def show\_base(self):

print("This is the Base class.")

class A(Base):

def show\_a(self):

print("This is Class A.")

class B(Base):

def show\_b(self):

print("This is Class B.")

class C(A, B): # Hybrid: Multiple + Hierarchical

def show\_c(self):

print("This is Class C.")

obj = C()

obj.show\_base()

obj.show\_a()

obj.show\_b()

obj.show\_c()

**Que 2 : Using the super() function to access properties of the parent class.**

class Parent:

def \_\_init\_\_(self, name, age):

self.name = name # Parent property

self.age = age # Parent property

def display\_parent(self):

print(f"Parent Name: {self.name}")

print(f"Parent Age: {self.age}")

class Child(Parent):

def \_\_init\_\_(self, name, age, grade):

# Call Parent's constructor using super()

super().\_\_init\_\_(name, age)

self.grade = grade # Child property

def display\_child(self):

# Accessing parent's method

super().display\_parent()

print(f"Child Grade: {self.grade}")

# Create object of Child class

obj = Child("Krishna", 20, "A")

# Call method

obj.display\_child()

**How it works**

1. **super().\_\_init\_\_(name, age)** → Calls the **parent class constructor** so we can reuse its initialization logic instead of rewriting it.
2. **super().display\_parent()** → Calls the parent’s method inside the child’s method.
3. **Parent properties** (self.name, self.age) are **accessible in the child** because super() sets them up.