Inheritance Questions in Python

1. Single Inheritance:

Create a base class Person with a method display\_name(). Inherit it in a class Student and call the method.

class Person:

   def display\_name(self):

       print("Name is:Abhi")

class Student(Person):

   def display\_name(self):

       print("Abhi is a student")

s=Student()

s.display\_name()

o/p:Abhi is a student

2. Multilevel Inheritance:

Design 3 classes: Animal → Mammal → Dog, where each class has its own method and Dog inherits all behaviors.

class Animal:

  def speak(self):

      print("animal speaks in the generic way")

class Mammal(Animal):

  def eat(self):

      super().speak()

      print("Mammal eats specific food")

class Dog(Mammal):

  def bark(self):

      super().eat()

      print("Dog barks")

d=Dog()

d.speak()

d.eat()

d.bark()

o/p:

animal speaks in the generic way

Mammal eats specific food

Dog barks

3. Multiple Inheritance:

Create two classes Flyable and Swimmable, each with a method. Derive a class Duck from both and call both methods.

class Flyable:

   def fly(self):

       print("Birds fly")

class Swimmable:

   def swim(self):

       print("Fish swim")

class Duck(Flyable, Swimmable):

   def show(self):

       self.fly()

       self.swim()

       print("Duck can both fly and swim")

d = Duck()

d.show()

o/p: Bird fly

Fish swim

Duck can both fly and swim

4. Hierarchical Inheritance:

Define a parent class Vehicle, and create two child classes Car and Bike. Show how each inherits from Vehicle.

class Vehicle:

   def engine(self):

       print("Every Vehicle has Engine")

class Car(Vehicle):

   def drive(self):

       super().engine()

       print("Drive a car")

class Bike(Vehicle):

   def ride(self):

       super().engine()

       print("Ride a bike")

c = Car()

c.drive()

b = Bike()

b.ride()

o/p:Every Vehicle has Engine

Drive a car

Every Vehicle has Engine

Ride a bike

5. Use super() in a derived class to call a parent class's method. What happens if both classes have the same method name?

class A:

   def person(self):

       print("Person A is ABHI")

class B(A):

   def person(self):

       super().person()

       print("Person B is ANU")

b=B()

b.person()

o/p:

Person A is ABHI

Person B is ANU

6. What is Method Resolution Order (MRO) in multiple inheritance? Demonstrate using a diamond problem structure.

7. Define a constructor in the base class. In the derived class, call it using super().\_\_init\_\_() and add new attributes.

class Person:

   def \_\_init\_\_(self, name):

       self.name = name

class Student(Person):

   def \_\_init\_\_(self, name, age):

       super().\_\_init\_\_(name)

       self.age = age

   def show(self):

       print("Name is:", self.name)

       print("Age is:", self.age)

s = Student("Abhi", 21)

s.show()

o/p:

Name is: Abhi

Age is: 21

8. Can you override a method in Python? Write a base class Shape with a method area() and override it in Circle.

Yes.

class Shape:

   def area(self):

       print("Area")

class Circle(Shape):

   def area(self):

       print("Area of circle = πr²")

c = Circle()

c.area()

o/p:

Area of circle = πr²

Polymorphism Questions in Python

9. Method Overriding:

Write a base class Animal with method speak(). Create subclasses Dog, Cat that override speak().

class Animal:

    def speak(self):

        print("Animal sound")

class Dog(Animal):

    def speak(self):

        print("Dog barks")

class Cat(Animal):

    def speak(self):

        print("Cat meows")

d = Dog()

d.speak()

c = Cat()

c.speak()

o/p:

Dog barks

Cat meows

10. Polymorphic Behavior:

Create a list of objects of Dog, Cat, Cow, each inheriting from Animal. Iterate and call speak() method.

class Animal:

   def speak(self):

       print("Animal makes sound")

class Dog(Animal):

   def speak(self):

       print("Dog barks")

class Cat(Animal):

   def speak(self):

       print("Cat meows")

class Cow(Animal):

   def speak(self):

       print("Cow scream")

animals = [Dog(), Cat(), Cow()]

for a in animals:

   a.speak()

o/p:Dog barks

Cat meows

Cow scream

11. Simulated Method Overloading:

Python doesn’t support method overloading directly. Show how you can use default or \*args to mimic it.

class age:

   def show(self, \*args):

       print("Age is:", args)

d = age()

d.show(18)

d.show(30, 20)

o/p:

Age is: (18,)

Age is: (30, 20)

12. Write a class Calculator with a method add() that supports 2 and 3 arguments using default parameters or \*args.

class Calculator:

   def add(self,a,b,c=0):

       print("Sum is:", a+b+c)

cal=Calculator()

cal.add(5,5)

cal.add(4,2,8)

o/p:

Sum is: 10

Sum is: 14

13. Can you override the \_\_str\_\_() method in Python? Create a class Book that returns a custom string when printed.

class Book:

   def \_\_init\_\_(self, title, author\_name):

       self.title = title

       self.author\_name = author\_name

   def \_\_str\_\_(self):

       return f"'{self.title}' by {self.author\_name}"

b = Book("English dictionary", "Padma")

print(b)

o/p:'English dictionary' by Padma

14. Demonstrate polymorphism using duck typing. Write a function start\_engine(vehicle) that takes any object with a method start().

class Car:

   def start(self):

       print("Start the car")

class Bike:

   def start(self):

       print("Start the bike")

def start\_engine(vehicle):

   vehicle.start()

start\_engine(Car())

start\_engine(Bike())

o/p:

Start the car

Start the bike

15. How does polymorphism help in writing more generic functions in Python? Provide a small real-world code snippet.

class YouTube:

   def open(self):

       print("Loading YouTube---")

class Amazon:

   def open(self):

       print("Loading Amazon---")

class Instagram:

   def open(self):

       print("Loading Instagram---")

def open\_website(site):

   site.open()

yt = YouTube()

amz = Amazon()

insta = Instagram()

open\_website(yt)

open\_website(amz)

open\_website(insta)

o/p:

Loading YouTube---

Loading Amazon---

Loading Instagram---