**Day-7 Evening Assessment**

**1. Order and OnlineOrder**

class Order:  
   def \_\_init\_\_(self, order\_id):  
       self.order\_id = order\_id  
  
   def process\_payment(self):  
       print(f"Processing payment for Order #{self.order\_id}")  
  
class OnlineOrder(Order):  
   def \_\_init\_\_(self, order\_id, email):  
       super().\_\_init\_\_(order\_id)  # Calls Order's \_\_init\_\_ to set order\_id  
       self.email = email          # Adds email attribute specific to OnlineOrder  
  
   def process\_payment(self):  
       super().process\_payment()  # Calls Order's process\_payment()  
       print(f"Sending confirmation email to {self.email}")  
  
# Creating an OnlineOrder object  
order = OnlineOrder(101, "customer@example.com")  
order.process\_payment()

-Order class is a base class.

-In this \_\_init\_\_ initializes the order\_id and it prints a message that the payment is being processed for that order.

-OnlineOrder class is a subclass that inherits from order.

- The constructor calls the parent constructor to set order\_id.

- It sets the additional attribute.

-It overrides the method.Calls the method on this object.

**2.Employee and Manager**

class Employee:  
   def \_\_init\_\_(self, name, salary):  
       self.name = name  
       self.salary = salary  
  
   def display(self):  
       print(f"Name: {self.name}, Salary: ₹{self.salary}")  
  
class Manager(Employee):  
   def \_\_init\_\_(self, name, salary, department):  
       super().\_\_init\_\_(name, salary)  
       self.department = department  
  
   def display(self):  
       super().display()  
       print(f"Department: {self.department}")  
  
m = Manager("Shaik", 90000, "IT")  
m.display()

-Manager is a subclass of Employee.

-Manager extends Employee by adding a department.

-The display() method is overridden to include department info after showing base employee data.

-Creates a manager object with name, salary and department and calls the display() method.

**3.Vehicle and Car**

class Vehicle:  
   def start(self):  
       print("Vehicle started")  
  
class Car(Vehicle):  
   def start(self):  
       super().start()  
       print("Car is ready to go")  
  
c = Car()  
c.start()

-Vehicle class is a parent class and car class is a child class.

-Car inherits from vehicle and car defines its own version

-When a car starts, the basic vehicle functionality is performed first.

-Then, car behaviour is added.

**4. User Login System**

class User:  
   def \_\_init\_\_(self, username):  
       self.username = username  
  
   def login(self):  
       print(f"{self.username} logged in")  
  
class Admin(User):  
   def login(self):  
       super().login()  
       print(f"{self.username} has admin privileges")  
  
a = Admin("admin\_user")  
a.login()

-Admin is a subclass of User, so it inherits its attributes and methods.

-Since Admin doesn’t define its own \_\_init\_\_, it uses the one from User.

-Admin provides its own version of login().

-It calls the User class version of login() inside the overridden method.

**5.Shape and Circle**

class Shape:  
   def \_\_init\_\_(self):  
       print("This is a shape")  
  
   def area(self):  
       print("Area formula not defined")  
  
class Circle(Shape):  
   def \_\_init\_\_(self, radius):  
       super().\_\_init\_\_()  
       self.radius = radius  
  
   def area(self):  
       super().area()  
       print("Circle Area:", 3.14 \* self.radius \* self.radius)  
  
c = Circle(5)  
c.area()

-Circle class inherits from the Shape.

-Circle provides an actual formula to calculate area.

-Super() calls the parent class’s constructor.

-Circle overrides the area() method. Then the Circle is called.

-Calls the parent area() method and then prints the actual area.

**6.Person and Student**

class Person:  
   def \_\_init\_\_(self, name):  
       self.name = name  
  
   def show(self):  
       print(f"Name: {self.name}")  
  
class Student(Person):  
   def \_\_init\_\_(self, name, grade):  
       super().\_\_init\_\_(name)  
       self.grade = grade  
  
   def show(self):  
       super().show()  
       print(f"Grade: {self.grade}")  
  
s = Student("Ali", "A")  
s.show()

-Person class is a base class, Student inherits from the Person.

-Student is a type of Person, nut also has a grade.

-The show() method is reused from Person, and extended in Student to also show the grade. Student overrides the show() method.

-Creates a Student object with and calls the show() method.

**7. BankAccount and SavingsAccount**

class BankAccount:  
   def \_\_init\_\_(self, balance):  
       self.balance = balance  
  
   def show\_balance(self):  
       print(f"Balance: ₹{self.balance}")  
  
class SavingsAccount(BankAccount):  
   def \_\_init\_\_(self, balance, interest):  
       super().\_\_init\_\_(balance)  
       self.interest = interest  
  
   def show\_balance(self):  
       super().show\_balance()  
       print(f"Interest Rate: {self.interest}%")  
  
acc = SavingsAccount(10000, 5)  
acc.show\_balance()

-SavingsAccount inherits from BankAccount.

-A BankAccount holds balance only and a SavingsAccount is a type of bank account that also earns interest.

-The show\_balance() method in SavingsAccount shows both balance and interest rate.

-SavingsAccount overrides show\_balance() to add interest info.

-Creates a SavingsAccount object and calls show\_balance().

**8. Product and ElectronicProduct**

class Product:  
   def \_\_init\_\_(self, name):  
       self.name = name  
  
   def details(self):  
       print(f"Product: {self.name}")  
  
class ElectronicProduct(Product):  
   def \_\_init\_\_(self, name, warranty):  
       super().\_\_init\_\_(name)  
       self.warranty = warranty  
  
   def details(self):  
       super().details()  
       print(f"Warranty: {self.warranty} years")  
  
p = ElectronicProduct("Laptop", 2)  
p.details()

-ElectronicProduct inherits from Product.

-super().\_\_init\_\_ calls the parent constructor.

-ElectronicProduct overrides details() method.

-Creates an ElectronicProduct object with and calls the details() method.

**9.Animal and Dog**

class Animal:  
   def sound(self):  
       print("Animal sound")  
  
class Dog(Animal):  
   def sound(self):  
       super().sound()  
       print("Dog barks")  
  
d = Dog()  
d.sound()

-Dog inherits from Animal.

-Dog overrides the sound() method to define its own behaviour.

-Creates a Dog object and calls the sound() method.

**10.Book and EBook**

class Book:  
   def \_\_init\_\_(self, title):  
       self.title = title  
  
   def show(self):  
       print(f"Title: {self.title}")  
  
class EBook(Book):  
   def \_\_init\_\_(self, title, file\_size):  
       super().\_\_init\_\_(title)  
       self.file\_size = file\_size  
  
   def show(self):  
       super().show()  
       print(f"File Size: {self.file\_size} MB")  
  
eb = EBook("Python Guide", 5)  
eb.show()

-EBook inherits from Book.

-Uses super().\_\_init\_\_() to reuse the parents constructor.

-EBook overrides show() to add file size info.

-Creates an EBook object and calls the show() method.