**Q1: Vehicle Class**

class Vehicle:

def \_\_init\_\_(self, name\_of\_vehicle, max\_speed, average\_of\_vehicle):

self.name\_of\_vehicle = name\_of\_vehicle

self.max\_speed = max\_speed

self.average\_of\_vehicle = average\_of\_vehicle

**Q2: Child Class Car Inheriting Vehicle**

class Car(Vehicle):

def seating\_capacity(self, capacity):

return f"The {self.name\_of\_vehicle} has a seating capacity of {capacity}."

# Example usage

my\_car = Car("Sedan", 200, 15)

print(my\_car.seating\_capacity(5))

**Q3: Multiple Inheritance**

**Definition**:  
Multiple inheritance is a feature in object-oriented programming where a class can inherit attributes and methods from more than one parent class.

**Example Code**:

class Parent1:

def feature1(self):

return "Feature 1 from Parent1"

class Parent2:

def feature2(self):

return "Feature 2 from Parent2"

class Child(Parent1, Parent2):

def feature3(self):

return "Feature 3 from Child"

# Example usage

child\_obj = Child()

print(child\_obj.feature1())

print(child\_obj.feature2())

print(child\_obj.feature3())

**Q4: Getters and Setters**

**Definition**:  
Getters and setters in Python are methods used to get and set the values of private or protected attributes.

**Example Code**:

class Student:

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

self.\_name = name # Protected attribute

self.\_age = age # Protected attribute

# Getter for name

def get\_name(self):

return self.\_name

# Setter for name

def set\_name(self, name):

self.\_name = name

# Getter for age

def get\_age(self):

return self.\_age

# Setter for age

def set\_age(self, age):

if age > 0:

self.\_age = age

else:

raise ValueError("Age must be a positive number.")

# Example usage

student = Student("John", 20)

print(student.get\_name()) # Output: John

student.set\_name("Doe")

print(student.get\_name()) # Output: Doe

student.set\_age(25)

print(student.get\_age()) # Output: 25

**Q5: Method Overriding**

**Definition**:  
Method overriding allows a child class to provide a specific implementation of a method that is already defined in its parent class.

**Example Code**:

class Animal:

def sound(self):

return "Animals make sound."

class Dog(Animal):

def sound(self):

return "Dogs bark."

# Example usage

animal = Animal()

dog = Dog()

print(animal.sound()) # Output: Animals make sound.

print(dog.sound()) # Output: Dogs bark.