***Oops Task Assignment***

**Q1, Create a vehicle class with an init method having instance variables as name\_of\_vehicle, max\_speed**

**and average\_of\_vehicle.**

**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. Create a child class car from the vehicle class created in Que 1, which will inherit the vehicle class.**

**Create a method named seating\_capacity which takes capacity as an argument and returns the name of**

**the vehicle and its seating capacity.**

**class Car(Vehicle):**

**def seating\_capacity(self, capacity):**

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

**car = Car("Toyota Camry", 140, 30)**

**print(car.seating\_capacity(5))**

**Output:**

**Toyota Camry has a seating capacity of 5 people.**

**Q3. What is multiple inheritance? Write a python code to demonstrate multiple inheritance.**

**Multiple inheritance is a feature in object-oriented programming languages that allows a subclass to inherit attributes and methods from multiple parent classes.**

**class Animal:**

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

**self.name = name**

**def speak(self):**

**pass**

**class Mammal(Animal):**

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

**super().\_\_init\_\_(name)**

**def speak(self):**

**return "I am a mammal."**

**class Bird(Animal):**

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

**super().\_\_init\_\_(name)**

**def speak(self):**

**return "I am a bird."**

**class Bat(Mammal, Bird):**

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

**super().\_\_init\_\_(name)**

**def speak(self):**

**return "I am a bat."**

**bat = Bat("Fruit Bat")**

**print(bat.name)**

**print(bat.speak())**

**Q4. What are getter and setter in python? Create a class and create a getter and a setter method in this**

**class.**

**In Python, getter and setter methods are used to access and modify object attributes in a controlled manner. They are commonly used to enforce data encapsulation and to prevent direct access to object attributes from outside the class.**

**A getter method is used to get the value of an object attribute, and a setter method is used to set the value of an object attribute. In Python, getter and setter methods are often implemented using property decorators.**

**class Person:**

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

**self.\_name = name**

**@property**

**def name(self):**

**return self.\_name**

**@name.setter**

**def name(self, value):**

**self.\_name = value.capitalize()**

**person = Person("john")**

**print(person.name) # Output: "John"**

**person.name = "jane"**

**print(person.name) # Output: "Jane"**

**Q5.What is method overriding in python? Write a python code to demonstrate method overriding.**

**Method overriding is a feature of object-oriented programming that allows a subclass to provide a different implementation for a method that is already defined in its parent class. When a method is overridden in the subclass, the method in the parent class is not called when the method is invoked on an object of the subclass.**

**class Animal:**

**def make\_sound(self):**

**print("generic animal sound")**

**class Dog(Animal):**

**def make\_sound(self):**

**print("bark")**

**class Cat(Animal):**

**def make\_sound(self):**

**print("meow")**

**class Bird(Animal):**

**pass**

**animals = [Dog(), Cat(), Bird()]**

**for animal in animals:**

**animal.make\_sound()**

**Output:**

**bark**

**meow**

**generic animal sound**