Q1. Create a `Vehicle` class with an `__init__` method

```
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` that inherits from `Vehicle`

The `Car` class inherits all properties of the `Vehicle` class and adds a method `seating_capacity`.

```
class Car(Vehicle):
    def seating_capacity(self, capacity):
        return f"{self.name_of_vehicle} has a seating capacity of
{capacity}"
```

Example Usage

```
car = Car("Toyota Innova", 180, 15)
print(car.seating_capacity(7))
```

Q3. What is multiple inheritance? Demonstrate with Python code

Multiple Inheritance

Multiple inheritance occurs when a **child class inherits from more than one parent class**.

Example

```
class Engine:
    def engine_type(self):
```

```
        return "Petrol Engine"

class Wheels:
    def wheel_count(self):
        return "4 Wheels"

class Car(Engine, Wheels):
    pass

c = Car()
print(c.engine_type())
print(c.wheel_count())
```

Q4. What are getter and setter in Python? Create a class with getter and setter methods

Getter

A getter method is used to **access (read)** the value of a private variable.

Setter

A setter method is used to **modify (update)** the value of a private variable.

Example

```
class Student:
    def __init__(self, name):
        self.__name = name   # private variable

    def get_name(self):
        return self.__name

    def set_name(self, new_name):
        self.__name = new_name

s = Student("Rahul")
print(s.get_name())
```

```
s.set_name("Amit")
print(s.get_name())
```

Q5. What is method overriding? Demonstrate with Python code

Method Overriding

Method overriding occurs when a **child class provides its own implementation** of a method that is already defined in the parent class with the **same method name and parameters**.

Example

```
class Animal:
    def sound(self):
        print("Animal makes a sound")

class Dog(Animal):
    def sound(self):
        print("Dog barks")

d = Dog()
d.sound()
```