In [ ]:
```
Q1: Create a vehicle class with an init method having instance variables as name_of_vehicle, max_speed, and average_of_vehicl
```

In [1]:
```
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

# Example usage
car = Vehicle("Car", 200, 25)
print(f"Name: {car.name_of_vehicle}, Max Speed: {car.max_speed}, Average: {car.average_of_vehicle}")
```

```
Name: Car, Max Speed: 200, Average: 25
```

In [ ]:
```
a method named seating_capacity which takes capacity as an argument and returns the name of the vehicle and its seating capac
```

In [2]:
```
class Car(Vehicle):
    def seating_capacity(self, capacity):
        return f"{self.name_of_vehicle} has a seating capacity of {capacity}."

# Example usage
car_instance = Car("Car", 200, 25)
print(car_instance.seating_capacity(5))
```

```
Car has a seating capacity of 5.
```

In [ ]:
```
What is multiple inheritance? Write a python code to demonstrate multiple inheritance
```

```
Multiple Inheritance: It is a feature in object-oriented programming where a class can inherit attributes and methods from
more than one parent class.
```

In [3]:
```
class A:
    def method_a(self):
        print("Method from class A")

class B:
    def method_b(self):
        print("Method from class B")

class C(A, B):
    def method_c(self):
        print("Method from class C")

# Example usage
obj_c = C()
obj_c.method_a()
obj_c.method_b()
obj_c.method_c()
```

```
Method from class A
Method from class B
Method from class C
```

In [ ]:
```
What are getter and setter in Python? Create a class and create a getter and a setter method in this class.
```

```
Getter and Setter methods are used to get and set the values of private attributes in a class.
```

In [4]:
```
class MyClass:
    def __init__(self):
        self._my_private_variable = None

    # Getter method
    def get_my_private_variable(self):
        return self._my_private_variable

    # Setter method
    def set_my_private_variable(self, value):
        self._my_private_variable = value

# Example usage
obj = MyClass()
obj.set_my_private_variable(42)
print(obj.get_my_private_variable())
```

```
42
```

In [ ]:
```
What is method overriding in Python? Write a python code to demonstrate method overriding.
```

```
Method Overriding: It occurs when a derived class provides a specific implementation for a method that is already defined
in its base class.
```

In [5]:
```
class Animal:
    def speak(self):
        print("Animal speaks")

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

# Example usage
dog = Dog()
dog.speak()  # This will call the overridden method in the Dog class
```

```
Dog barks
```

In [ ]:
```

```