# Q1.

**Q1, Create a vehicle class with an init method having instance variables as name_of_vehicle, max_speed and average_of_vehicle.**

In [1]:
```
class Vehicle:
    def __init__(self, name_of_vehicle, max_speed, average_speed):
        self.name_of_vehicle = name_of_vehicle
        self.max_speed = max_speed
        self.average_speed = average_speed

car = Vehicle("Car", 200, 60)
bike = Vehicle("Bike", 120, 30)

print(f"{car.name_of_vehicle} - Max Speed: {car.max_speed} km/h, Avg Speed: {car.average_speed} km/h")
print(f"{bike.name_of_vehicle} - Max Speed: {bike.max_speed} km/h, Avg Speed: {bike.average_speed} km/h"
```

```
Car - Max Speed: 200 km/h, Avg Speed: 60 km/h
Bike - Max Speed: 120 km/h, Avg Speed: 30 km/h
```

# 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.**

In [3]:
```
class Vehicle:
    def __init__(self, name_of_vehicle, max_speed, average_speed):
        self.name_of_vehicle = name_of_vehicle
        self.max_speed = max_speed
        self.average_speed = average_speed

class Car(Vehicle):
    def seating_capacity(self, capacity):
        return f"{self.name_of_vehicle} - Seating Capacity: {capacity}"

car = Car("Sedan", 180, 70)
print(f"{car.name_of_vehicle} - Max Speed: {car.max_speed} km/h, Avg Speed: {car.average_speed} km/h")
print(car.seating_capacity(5))
```

```
Sedan - Max Speed: 180 km/h, Avg Speed: 70 km/h
Sedan - Seating Capacity: 5
```

# Q3.

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

***Ans = Multiple inheritance is where a class can inherit attributes and methods from more than one parent class.***

In [4]:
```python
class Animal:
    def __init__(self, species):
        self.species = species

    def speak(self):
        pass

class Mammal(Animal):
    def speak(self):
        return "Mammal sound"

class Bird(Animal):
    def speak(self):
        return "Bird sound"

class Platypus(Mammal, Bird):
    def __init__(self):
        super().__init__("Platypus")
perry = Platypus()

print(f"Species: {perry.species}")
print(f"Platypus speaks: {perry.speak()}")
```

```
Species: Platypus
Platypus speaks: Mammal sound
```

# Q4.

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

**class.**

Getter Method: A getter method is used to retrieve the value of an instance variable. It's a method that is used to get the value of a private attribute.

Setter Method: A setter method is used to set the value of an instance variable. It's a method that is used to modify the value of a private attribute and might include validation checks before assigning the value.

In [7]:
```python
class Student:
    def __init__(self, name, age):
        self._name = name
        self._age = age


    def get_name(self):
        return self._name


    def get_age(self):
        return self._age


    def set_name(self, name):
        if isinstance(name, str):
            self._name = name
        else:
            print("Invalid name format")

    def set_age(self, age):
        if isinstance(age, int) and age >= 0:
            self._age = age
        else:
            print("Invalid age format")

student = Student("Alice", 20)

print(f"Name: {student.get_name()}")
print(f"Age: {student.get_age()}")

student.set_name("Bob")
student.set_age(22)

print(f"Updated Name: {student.get_name()}")
print(f"Updated Age: {student.get_age()}")
```

```
Name: Alice
Age: 20
Updated Name: Bob
Updated Age: 22
```

# Q5.

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

Method overriding in Python is a concept where a subclass provides a specific implementation of a method that is already defined in its parent class. This allows the subclass to customize or extend the behavior of the inherited method without changing its name or signature.

In [8]:

```python
class Shape:
    def area(self):
        return 0

class Circle(Shape):
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return 3.14 * self.radius * self.radius

class Square(Shape):
    def __init__(self, side):
        self.side = side

    def area(self):
        return self.side * self.side

circle = Circle(5)
square = Square(4)

print("Area of Circle:", circle.area())
print("Area of Square:", square.area())
```

```
Area of Circle: 78.5
Area of Square: 16
```