~\Downloads\assignment.md

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

Code:

```
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

    def __str__(self):
        return f"Vehicle Name: {self.name_of_vehicle}, Max Speed: {self.max_speed} km/h, Average: {self.

# Example of creating an instance of the Vehicle class
car = Vehicle("Toyota Camry", 240, 15)
print(car)
```

```
Vehicle Name: Toyota Camry, Max Speed: 240 km/h, Average: 15 km/l
```

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 ofthe vehicle and its seating capacity.

code :

```
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

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

    # Create an instance of Vehicle
vehicle = Vehicle("Generic Vehicle", 120, 15)

# Create an instance of Car
car = Car("Toyota Camry", 150, 20)

# Get seating capacity
print(car.seating_capacity(5))
```

```
Toyota Camry has a seating capacity of 5.
```

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

Ans - Multiple inheritance is a feature in object-oriented programming where a class an inherit attributes and methods from more than one parent calss. This allows a subclass to combine the functionality classes

code:

```
class Engine:
    def start_engine(self):
        return "Engine started."

class Wheels:
    def rotate_wheels(self):
        return "Wheels are rotating."

class Car(Engine, Wheels):
    def drive(self):
        return "Car is driving."

# Create an instance of Car
my_car = Car()

# Call methods from both parent classes
print(my_car.start_engine())
print(my_car.rotate_wheels())
print(my_car.drive())
```

```
Engine started.
Wheels are rotating.
Car is driving.
```

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

Ans : In python getters and setters are methods used to access (get) and modify (set) the values of private instance variables. This approach provides a way to control access to these variables, allowing you to enforce constraints or additional logic when getting or setting values.

Code:

```
class Person:
    def __init__(self, name, age):
        self.__name = name
        self.__age = age

    # 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:  # Enforce a constraint
            self.__age = age
        else:
            print("Age cannot be negative.")
```

```
# Example usage
person = Person("Alice", 30)

# Using getters
print(person.get_name())
print(person.get_age())

# Using setters
person.set_name("Bob")
person.set_age(25)

print(person.get_name())
print(person.get_age())

# Trying to set a negative age
person.set_age(-5)
```

```
Alice
30
Bob
25
Age cannot be negative.
```

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

Ans: Method overriding in python occurs when a subclass provides a spcific implementation of a method that is already defined in its parent class. This allows the subclass to modify or extend the behavior of the inherited method.

```
class Animal:
    def sound(self):
        return "some sound"

class Dog(Animal):
    def sound(self):
        return "Bark"

class cat(Animal):
    def sound(self):
        return "Meow"


animal = Animal()
dog = Dog()
cat = cat()

print(animal.sound())
print(dog.sound())
print(cat.sound())
```

```
some sound
Bark
Meow
```

----------thank you----------