# assignment=5

July 23, 2023

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

[1]:
```
#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
```

## 2 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.

[3]:
```
#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):
        self.capacity = capacity
        return self.name_of_vehicle, self.capacity
```

[5]:
```
c1= car("marshidis", 399, 390)
c1.seating_capacity(4)
```

[5]:
```
('marshidis', 4)
```

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

Multiple inheritance is a feature of object-oriented programming (OOP) languages where a class can inherit properties and methods from more than one parent class. In other words, a class can inherit from multiple classes, thus inheriting the attributes and behaviors of all its parent classes.

Here is an example of multiple inheritance in Python:

[21]:
```
class Parent1:
    def method1(self):
        print("Parent1 method1")
class Parent2:
    def method2(self):
        print("Parent2 method2")
class Child(Parent1, Parent2):
     pass

c = Child()
c.method1()
c.method2()
```

```
Parent1 method1
Parent2 method2
```

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

In object-oriented programming, getters and setters are methods used to retrieve and update the values of object properties, respectively. The use of getters and setters is a common technique for encapsulating object data and ensuring that it can be accessed and modified in a controlled and predictable manner.

[23]:
```
class Person:
    def __init__(self, name):
        self._name = name

    def get_name(self):
        return self._name

    def set_name(self, name):
        self._name = name

person = Person("jinjala sonal j")
print(person.get_name())
person.set_name("jadavbhai jinjala k")
print(person.get_name())
```

```
jinjala sonal j
jadavbhai jinjala k
```

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

Method overriding is a feature of object-oriented programming in which a subclass can provide a new implementation of a method that is already defined in its parent class. The subclass method is said to "override" the parent class method. This allows the subclass to change or extend the behavior of the method inherited from the parent class, while still maintaining the same method signature.

[24]:
```python
class Shape:
    def area(self):
        pass

class Rectangle(Shape):
    def __init__(self, width, height):
        self.width = width
        self.height = height

    def area(self):
        return self.width * self.height

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

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

rect = Rectangle(10, 20)
print(rect.area()) # Output: 200

circle = Circle(10)
print(circle.area()) # Output: 314.0
```

```
200
314.0
```

[ ]:
```python

```