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

In [2]:
```python
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 vehicle1(self):
        print(self.name_of_vehicle,self.max_speed,self.average_of_vehicle)
```

In [5]:
```python
ob=vehicle("toyta",180,100)
```

In [6]:
```python
ob.average_of_vehicle
```

Out[6]:
```
100
```

In [7]:
```python
ob.max_speed
```

Out[7]:
```
180
```

In [8]:
```python
ob.vehicle1()
```

```
toyta 180 100
```

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.

In [9]:
```python
class clild:
```

```
  File "C:\Users\vivek\AppData\Local\Temp\ipykernel_20264\2262586473.py", line 2

    ^
IndentationError: expected an indented block
```

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

In [10]:
```python
# ans:-Inheritance is the mechanism to achieve the re-usability of code as one class(child class) can derive the properties of another class(parent
```

In [36]:
```python
class class1:
    def pw(self):
        print("this is class 1")
class class2(class1):
    def pw(self):
        print("this is class 2")
class class3(class1):
    def pw(self):
        print("this is class 3")
class class4(class3):
    pass
```

In [37]:
```python
ob=class4()
```

In [38]:
```python
ob.pw()
```

```
this is class 3
```

In [42]:
```python
class Class1:
    def m(self):
        print("In Class1")

class Class2(Class1):
    pass

class Class3(Class1):
    def m(self):
        print("In Class3")

class Class4(Class2, Class3):
    pass
```

In [44]:
```python

obj = Class4()
obj.m()
```

```
In Class3
```

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

In [45]:
```python
# Ans:-Diffrence between getter and setter
# We use getters & setters to add validation logic around getting and setting a value.
# To avoid direct access of a class field i.e. private variables cannot be accessed directly or modified by external user.
```

In [46]:
```python
class vivek:
    def __init__(self,age=0):
        self.age=age
    def get_age(self):
        return self._age
    def set_age(self,x):
        self._age=x
```

In [51]:
```python
obj=vivek()
```

In [52]:
```python
obj.set_age(21)
```

In [53]:
```python
print(obj.get_age())
```

```
21
```

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

In [56]:
```python
# ans:-Method overriding is an ability of any object-oriented programming language that allows a subclass or child class to provide a specific imple
#When a method in a subclass has the same name, same parameters or signature and same return type(or sub-type) as a method in its super-class, then
```

In [57]:
```python
class parent:
    def __init__(self):
        self.value="inside parent"
    def show(self):
        print(self.value)
class child(parent):
    def __init__(self):
        self.value="inside child"
    def show(self):
        print(self.value)
```

In [58]:
```python
obj1=parent()
obj2=child()
```

In [59]:
```python
obj1.show()
obj2.show()
```

```
inside parent
inside child
```

In [ ]:
```python

```