Q1. Explain Class and Object with respect to Object-Oriented Programming. Give a suitable example.

Ans - In object- oriented programming (oop) a class is a blueprint or template for creating object In defines as a set of attributes and method that create object will have .An object is an instance of a class . It is a self contained component which consist of method and properties to make a particular type of data usefull Class A class can be thought of a blueprint for an object . It doesn't contain any data itself but defines how the data is structured and what operations can be performed on the data. A class cntain attributes (varables) and method (function) that defines the behavior of the object created from the class Object An object is an instance of a class.When a class is defined . no memory is allocated until an object of that class is created .An object is a specific implementation of the class with actual values for the attributes defined by the class

In [15]:
```
# Example
#Car and an object of this class
#Define the car class
class car :
    def __init__(self,make,model,year):
        self.make = make
        self.model = model
        self.year = year
    def display_info(self) :
        return ( f"{self.year} {self.make} {self.model}")

#create an object of the car class
my_car = car("Toyoto" , "corolla" , 2020)

#Acces the object attributes and method
print(my_car.display_info())
```

```
2020 Toyoto corolla
```

Q2. Name the four pillars of OOPs.

Ans : The four pillars of oops are 1)Encapsulation 2)Inheritance 3)polymorphism 4)Abstraction

Q3. Explain why the **init**() function is used. Give a suitable example.

Ans : The **init**() function also known as the conductor is a special method in pythod classes.It is a automatically called when a new instance of the classes is created. The primary purpose of the **init**() function is to initialize the object atteibutes with initial values.

```
why __init__() is used
1)Initialization : It sets the initial state of the object by
assigning values to the object properties
2)configuration : It allows the passing of parameters to the
```

object
3)setup : It can perform any setup or preparation before the object

In [2]:
```
#example
#person class that uses the __init__() method to initialize the name and age of
class person :
    def __init__(self , name , age):
        self.name = name
        self.age = age

    def display_info(self):
        print(f"Name : {self.name}, Age : {self.age}")

#creating an instance of the person class
person1 = person("sujen",24)

#Accessing the objects attributes and method
person1.display_info()
```

```
Name : sujen, Age : 24
```

Q4. Why self is used in OOPs?

Ans : In object oriented programming (oop) in python self is a reference to the instance of the class It is used to accces variable that belong to the class. Self is used because 1)Distibguish instance variables from local variable : - Using self help deffrentiate between instance variable within methods. Instance variables are accessed through self wherear local variables are just the vareiables names

2)Acces instance variables and modify : self allows you to access the attributes and methods of the class in python It enables each instance of class to keep of its own data

3)Modify instance variables :By using self we can modify the state of the object by changing its instance variables

4)Consistency : Self is a convention in python and makes the code more realiable and understanddable as it clearly shows that the variables belong to the instance of the class

Q5 What is inheritance ? Give an example for each type of inheritance

Ans : Inheritance is a funcdamental concept in object oriented programming (oop) that allows a class to inherit attributes and methods from another class .Their promotes code reusobility and establishes a relationship b/w the parents calss(superclass) and the child class (subclass) There are several type of inheritance including single , multiple , multilevel , hierarchical and hybrid inheritance Type of inheritance : 1)single inheritance : - A subclass inherits from one superclass 2)multiple inheritance : - A subclass inherits from more than one superclass 3)Hirarchical inheritance : - A Multiple subclass inherit from a single superclass 4)multilevel inheritance : - A subclass inherits from a superclass which itslef ia a subclass of another superclass 5)Hybrid inheritance : - A combination two or more type of inheritance

In [1]:
```
#Example 1) Single inheritance
class Animal:
    def speak(self):
        print("Animal speaks")

class Dog :
     def bark(self):
            print("Dog Barks")

#creating an instance of dog
dog = Dog()
Animal = Animal()

Animal.speak()
dog.bark()
```

```
Animal speaks
Dog Barks
```

In [4]:
```
# Example 2) multiple inheritance
class father :
    def Father_info(self):
        print("father's info")
class mother :
    def Mother_info(self):
        print("mother's info")
class child(father , mother):
    def child_info(self):
        print("child info")
#Creating an instance of child
child = child()
child.Father_info()
child.Mother_info()
child.child_info()
```

```
father's info
mother's info
child info
```

In [7]:
```
#Example 3) multilevel inheritance
class vehicle :
    def start(self):
        print("vehicle starts")

class car(vehicle):
    def drive(self):
              print("car drives")

class sports_car(car):
    def accelerate(self):
              print("sportscra accelerates")

#Creating an instance of sports car
sports_car = sports_car()
sports_car.start()
sports_car.drive()
sports_car.accelerate()
```

```
vehicle starts
car drives
sportscra accelerates
```

In [8]:
```python
# Example 4) Hierarachical inheritance
class shape :
    def draw(self):
        print("drawing shape")

class circle(shape):
    def draw_circle(self):
        print("Drawing circle")

class square(shape):
    def draw_square(self):
        print("drawing square")

#creating and instance of circle and aquare
circle = circle()
square = square()

circle.draw()
circle.draw_circle()
square.draw()
square.draw_square()
```

```
drawing shape
Drawing circle
drawing shape
drawing square
```

In [15]:
```python
# Exampele 5) Hybrid inheritance : Hybrid inheritance is a combination of two or
                                   # multilevel and multiple inheritance

class A:
    def method_a(self):
        print("Method A")

class B(A):
    def method_b(self):
         print("method B")

class C(A):
    def method_c(self):
        print("method c")

class D(B,C):
     def method_d(self):
        print("method D")

#creating an instance of D
d = D()
d.method_a()
d.method_b()
d.method_c()
d.method_d()
```

```
Method A
method B
method c
method D
```