OOPS

```
OOPS: stands for Object-Oriented Programming System.
It’s a programming approach where you structure your code using
objects and classes, making programs easier to design, understand, and
maintain.
```

Class

A class is a blueprint for creating objects.

In [1]:
```
class Student:
    def __init__(self, name, marks):
        self.name = name
        self.marks = marks
```

Object

An object is an instance of a class.

In [2]:
```
s1 = Student("Anu", 85)
print(s1.name)
```

```
Anu
```

Abstraction

Abstraction means hiding the internal implementation details
and showing only the essential features to the user.

In [3]:
```
from abc import ABC, abstractmethod

class Vehicle(ABC):                    # Abstract class
    @abstractmethod
    def start_engine(self):
        pass

    @abstractmethod
    def stop_engine(self):
        pass


class Car(Vehicle):                    # Child class
    def start_engine(self):
        print("Car engine started")

    def stop_engine(self):
        print("Car engine stopped")


c = Car()
c.start_engine()
c.stop_engine()
```

```
Car engine started
Car engine stopped
```

Encapsulation

Encapsulation means wrapping data and methods together and protecting data from direct access.

In [4]:
```python
class Bank:
    def __init__(self, balance):
        self.__balance = balance      # private variable

    def deposit(self, amount):
        self.__balance += amount
        print("Deposited:", amount)

    def withdraw(self, amount):
        if amount <= self.__balance:
            self.__balance -= amount
            print("Withdrawn:", amount)
        else:
            print("Insufficient balance")

    def show_balance(self):
        print("Balance:", self.__balance)


b = Bank(1000)
b.deposit(500)
b.withdraw(300)
b.show_balance()
```

```
Deposited: 500
Withdrawn: 300
Balance: 1200
```

Inheritance

Inheritance means a child class uses properties and methods of a parent class.

In [5]:
```python
class Parent:
    def show(self):
        print("This is Parent class")

class Child(Parent):
    def display(self):
        print("This is Child class")

c = Child()
c.show()
c.display()
```

```
This is Parent class
This is Child class
```

polymorphism

Polymorphism means one method name with different behavior.

In [6]:
```
class Dog:
    def sound(self):
        print("Dog barks")

class Cat:
    def sound(self):
        print("Cat meows")

d = Dog()
c = Cat()

d.sound()
c.sound()
```

```
Dog barks
Cat meows
```

In [ ]: