

Abstraction

Wednesday, July 23, 2025 7:02 PM

Abstraction is used to hide the implementation from the user.
User can see it, access it but can't see what is happening behind or in backend.

Eg: Whatsapp Message

It is divided into 3 sub topics:

- i. Abstract Method
- ii. Abstract Class
- iii. Concrete Class

1. Abstract Method: It is a method which have only function declaration but not the function definition.

Class Cname:

```
Def Mname(args):  
    Pass
```

2. Abstract Class: If we have at least 1 abstract method in the class, it is an abstract class.
 - Here object creation is not possible.
3. Concrete Class: If a class don't have any abstract method then it is called as concrete class.
 - We can create the object for concrete class.
 - All the classes where object is created is concrete class only.

- Abstraction is mainly used in project designing.
- For using abstraction 'abc' module is used.
- From this 'abc' module, we have to use ABC(Abstract Base Class)

Syntax:

From abc import ABC, abstractmethod

Class Cname:

```
@abstractmethod  
Def Mname(args):  
    Pass
```

→ abstract method } → abstract class

Class Cname1(Cname):

```
Def Mname(args):  
    S B
```

Obj=Cname1()

} → concrete class

Eg:

```
from abc import ABC, abstractmethod  
class Cal(ABC):
```

```

@abstractmethod
def add(a,b):
    pass
@abstractmethod
def sub(a,b):
    pass
@abstractmethod
def mul(a,b):
    pass
@abstractmethod
def div(a,b):
    pass

class Cal_up(Cal):
    def add(a,b):
        print(a+b)
    def sub(a,b):
        print(a-b)
    def mul(a,b):
        print(a*b)
    def div(a,b):
        print(a/b)

ob1=Cal_up()
Cal_up.add(10,2)
Cal_up.sub(10,2)
Cal_up.mul(10,2)
Cal_up.div(10,2)

```

Assignment:

Q. Write a Python program to demonstrate the concept of **abstraction** using the abc module. Create an abstract class named Vehicle with an abstract method start_engine(). Then, create two subclasses Car and Bike that inherit from Vehicle and implement the start_engine() method in each subclass. Finally, create objects of both subclasses and call the start_engine() method to show how abstraction works.

