Abstraction in Python:

Abstraction means hiding the internal implementations of a process or method from the user. The user knows what he is doing but not how the work is being done.

Abstract class:

A class which contains one or more abstract methods is called an abstract class. An abstract method is a method that has a declaration but does not have an implementation.

How Abstract Base classes work :
By default, Python does not provide abstract classes. Python comes with a module that provides the base for defining Abstract Base classes(ABC) and that module name is ABC.

ABC works by decorating methods of the base class as abstract
A method becomes abstract when decorated with the keyword @abstractmethod

Abstract Class Instantiation : (object creation)

Abstract classes have methods that have no body. If python allows creating an object for abstract classes then using that object if anyone calls the abstract method, but there is no actual implementation to invoke. Due to the fact, an abstract class is not a concrete class, it cannot be instantiated. When we create an object for the abstract class it raises an *error*.

So we use an abstract class as a template and according to the need, we extend the class to use its members

Rules:

PVM can not create object of an abstract class

Abstract class can have both abstract(no body) and concrete class(with body)

A class having abstract method must be as an abstract class

Abstract methods must be defined within its child classes

If the child class is an abstract class then it may avoid defining the abstract methods of its child class

```python
from abc import ABC, abstractmethod

class Demo(ABC):
    @abstractmethod
    def show(self):
        pass
    def display(self):
        print("display method invoked")
class Test(Demo):
    def show(self):
        print("show method invoked")
    def greet(self):
        print("greet method invoked")

t=Test()
t.greet()
t.show()
t.display()
```