**Abstraction**

Abstraction in python is defined as **a process of handling complexity by hiding unnecessary information from the user.**

We have abstract class and abstract methods in python

**Abstract class**

1. Python provides the **abc** module to use the abstraction in the Python program.
2. A class which contains one or more abstract methods is called an abstract class.
3. An abstract method is a method that has a declaration but does not have an implementation
4. Abstract class can be inherited by the subclass.
5. Abstract method gets its definition in the subclass
6. While we are designing large functional units we use an abstract class.
7. When we want to provide a common interface for different implementations of a component, we use an abstract class.

Python provides the **abc** module to use the abstraction in the Python program.

from abc **import** ABC

**class** ClassName(ABC):

We import the ABC class from the **abc** module.

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 and then registering concrete classes as implementations of the abstract base. A method becomes abstract when decorated with the keyword @abstractmethod.

# Python program showing

# abstract base class work

**from** abc **import** ABC, abstractmethod

**class** Polygon(ABC):

    @abstractmethod

**def** noofsides(self):

**pass**

**class** Triangle(Polygon):

    # overriding abstract method

**def** noofsides(self):

**print**("I have 3 sides")

**class** Pentagon(Polygon):

    # overriding abstract method

**def** noofsides(self):

**print**("I have 5 sides")

**class** Hexagon(Polygon):

    # overriding abstract method

**def** noofsides(self):

        print("I have 6 sides")

**class** Quadrilateral(Polygon):

    # overriding abstract method

**def** noofsides(self):

**print**("I have 4 sides")

# Driver code

R **=** Triangle()

R.noofsides()

K **=** Quadrilateral()

K.noofsides()

R **=** Pentagon()

R.noofsides()

K **=** Hexagon()

K.noofsides()

**Concrete Methods in Abstract Base Classes:**

Concrete classes contain only concrete (normal)methods whereas abstract classes may contain both concrete methods and abstract methods. The concrete class provides an implementation of abstract methods, the abstract base class can also provide an implementation by invoking the methods via super().

# Python program invoking a

# method using super()

**import** abc

**from** abc **import** ABC, abstractmethod

**class** R(ABC):

**def** rk(self):

        print("Abstract Base Class")

**class** K(R):

**def** rk(self):

        super().rk()

        print("subclass ")

# Driver code

r **=** K()

r.rk()

In the above program, we can invoke the methods in abstract classes by using super()

**Points to Remember**

Below are the points which we should remember about the abstract base class in Python.

* An Abstract class can contain the both method normal and abstract method.
* An Abstract cannot be instantiated; we cannot create objects for the abstract class.

Abstraction is essential to hide the core functionality from the users. We have covered the all the basic concepts of Abstraction in Python.