PYTHON ABSTRACTION OR DATA HIDING
Abstraction is used to hide internal details & show only functionalities. It is blue print for other classes.
OR
Abstraction is used to hide the internal functionality of the function from the users.
OR
Hiding all implemenations, show only essential parts. Here, using inheritance concept.

RealTimeUseCase:
Suppose you are going to an ATM to withdraw money. You simply insert your card and click some buttons and get the money. You don’t know what is happening internally on press of these buttons. Basically you are hiding unnecessary information from user.

Abstrac Class
A class that consists of one or more abstract methods is called the abstract class. An abstract class can be useful when we are designing large functions. An abstract class is also helpful to provide the standard interface for different implementations of components.

Remember Points:
1 We are unable to create instance for abstract class.
2. We want to create instance, then we must convert into concrete class.
3. Concrete class means no abstract methods.
4. Abstract methods we can override through inheritance.
5. Abstract class is base class, concrete class is child class, we can override methods

Syntax:
from abc import ABC
abc :Abstract Base Class
abc : Module Name, ABC ==>Class Name

Abstract Method:
Only function call with empty definition.Abstract class can be inherited by the subclass and abstract method gets its definition in the subclass.

Syntax:
@abstractmethod

Use of Abstraction:
Abstraction classes are meant to be the blueprint of the other class. An abstract class can be useful when we are designing large functions. An abstract class is also helpful to provide the standard interface for different implementations of components.

Example:
```
from abc import ABC,abstractmethod
class Banking(ABC):#Abstract Class
    @abstractmethod
    def MonthInterest(self):#Abstract Method
```

```
        pass
    def QuaterlyInterest(self):#Abstract Method
        pass
AA=Banking()

Example:
from abc import ABC,abstractmethod
class Banking(ABC):#Abstract Class
    @abstractmethod
    def MonthInterest(self):#Abstract Method
        pass
    @abstractmethod
    def QuaterlyInterest(self):#Abstract Method
        pass

class SBI(Banking):#Concrete Class
    def MonthInterest(self):
        print("Montly Interest is: 4.5% ")
    def QuaterlyInterest(self):
        print("Quarterly Interest is: 5.5% ")

SS=SBI()
SS.MonthInterest()
SS.QuaterlyInterest()

Example:
from abc import ABC,abstractmethod
class Banking(ABC):#Abstract Class
    @abstractmethod
    def MonthInterest(self):#Abstract Method
        pass
    def QuaterlyInterest(self):#Abstract Method
        pass

class SBI(Banking):#Concrete Class
    def MonthInterest(self):
        print("Montly Interest is: 4.5% ")
    def QuaterlyInterest(self):
        print("Quarterly Interest is: 5.5% ")

class HDFC(Banking):#Concrete Class
    def MonthInterest(self):
        print("Montly Interest is: 5.5% ")
    def QuaterlyInterest(self):
        print("Quarterly Interest is: 7.5% ")

SS=SBI()
SS.MonthInterest()
SS.QuaterlyInterest()

HH=HDFC()
HH.MonthInterest()
HH.QuaterlyInterest()

Example:
from abc import ABC,abstractmethod
class Banking(ABC):#Abstract Class
```

```
    @abstractmethod
    def MonthInterest(self):#Abstract Method
        pass
    @abstractmethod
    def QuaterlyInterest(self):#Abstract Method
        pass

class SBI(Banking):#Abstract Class
    def MonthInterest(self):
        print("Montly Interest is: 4.5% ")

SS=SBI()
SS.MonthInterest()
```

NOTE:
1 TypeError: Can't instantiate abstract class SBI with abstract methods QuaterlyInterest
2 Because still it is abstract class, We must implement all abstract class definitions (Override) in Concrete class.

Example:

```
from abc import ABC,abstractmethod
class Banking(ABC):#Abstract Class
    @abstractmethod
    def MonthInterest(self):#Abstract Method
        pass
    @abstractmethod
    def QuaterlyInterest(self):#Abstract Method
        pass

class SBI(Banking):#Abstract Class
    def MonthInterest(self):
        print("Montly Interest is: 5.5% ")

class UBI(Banking):#Concrete Class
    def MonthInterest(self):
        print("Montly Interest is: 4.5% ")
    def QuaterlyInterest(self):
        print("Montly Interest is: 6.5% ")

UU=UBI()
UU.MonthInterest()
UU.QuaterlyInterest()
```

Concrete Methods in Abstract Base Classes :
Concrete classes contain only concrete (normal)methods whereas abstract class contains both concrete methods as well as abstract methods. Concrete class provide an implementation of abstract methods, the abstract base class can also provide an implementation by invoking the methods via super().

Example:

```
from abc import ABC
class BaseClass(ABC): #Abstract Base Class
    def BaseMethod(self):
        print("AbstractBaseClass")
```

```
class Concrete(BaseClass): #Concrete Class
    def ConcreteMethod(self):
        super().BaseMethod()
        print("Subclass With ConcreteMethod")

#Driver code
CC=Concrete()
CC.ConcreteMethod()
```