**Data Abstraction in Python**

Used to hide irrelevant details from the user and show the details that are relevant to the users.

**Implementation of Data Abstraction in Python**

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

**class** **Car**(ABC):

**def** \_\_init\_\_(self, brand, model, year):

self.brand = brand

self.model = model

self.year = year

*# Create abstract method*

@abstractmethod

**def** printDetails(self):

**pass**

*# Create concrete method*

**def** accelerate(self):

print("Speed up ...")

**def** break\_applied(self):

print("Car stopped")

**class** **Hatchback**(Car):

**def** printDetails(self):

print("Brand:", self.brand)

print("Model:", self.model)

print("Year:", self.year)

**def** sunroof(self):

print("Not having this feature")

**class** **Suv**(Car):

**def** printDetails(self):

print("Brand:", self.brand)

print("Model:", self.model)

print("Year:", self.year)

**def** sunroof(self):

print("Available")

*# Create an instance of the Hatchback class*

car1 = Hatchback("Maruti", "Alto", "2022")

*# Call methods*

car1.printDetails()

car1.accelerate()

car1.sunroof()

**Abstract Classes in Python**

In Python, an abstract class is a class that cannot be instantiated on its own and is designed to be a blueprint for other classes.

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

*# Define an abstract class*

**class** **Animal**(ABC):

@abstractmethod

**def** sound(self):

**pass** *# This is an abstract method, no implementation here.*

*# Concrete subclass of Animal*

**class** **Dog**(Animal):

**def** sound(self):

**return** "Bark" *# Providing the implementation of the abstract method*

*# Create an instance of Dog*

dog = Dog()

print(dog.sound()) *# Output: Bark*

**Abstract Methods**

Abstract methods are methods that are defined in an abstract class but do not have an implementation.   They serve as a blueprint for the subclasses.

**Concrete Methods**

Concrete methods are methods that have full implementations in an abstract class. These methods can be inherited by subclasses and used directly without needing to be redefined.

**Abstract Properties**

**Abstract properties** work like abstract methods but are used for properties.

**Python-interface module**

In object-oriented languages like Python, the interface is a collection of method signatures that should be provided by the implementing class.

**Difference between abstract class and interface in Python**

