Lesson: Abstraction

Description: Learn how abstraction simplifies complex systems by hiding unnecessary details.

Abstraction is a key concept in Object-Oriented Programming (OOP) that focuses on hiding the implementation details of an object and only exposing the essential features to the user. This simplifies the process of programming by allowing developers to focus on the functionality rather than the implementation.

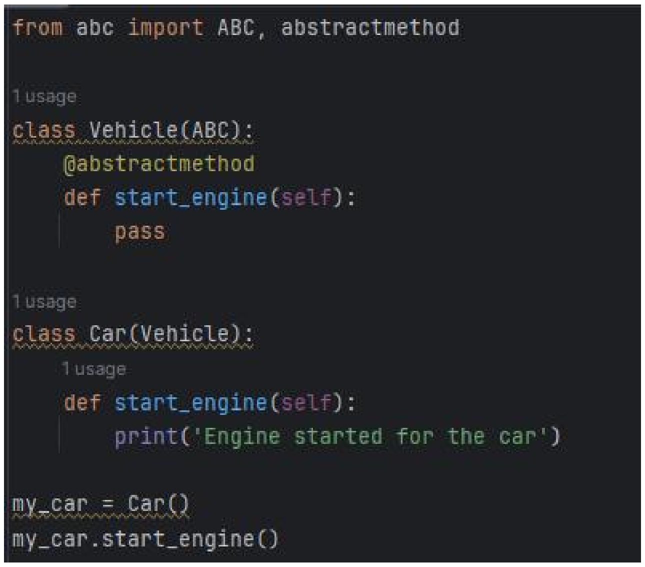
For example, when you use a smartphone, you are only concerned with its interface, such as calling, texting, or browsing the internet. You don't need to know how the hardware and software interact to make these functions work.

Key Features of Abstraction:
1. Simplifies the design of software.
2. Helps in reducing the complexity of the system.
3. Allows for better code maintenance and scalability.

Abstract Classes and Interfaces:
In OOP, abstraction is achieved using abstract classes and interfaces. An abstract class is a blueprint for other classes and cannot be instantiated. It may include abstract methods (methods without implementation) that must be implemented by derived classes.

An interface, on the other hand, is a contract that specifies a set of methods that a class must implement.

Example:



```
from abc import ABC, abstractmethod

1 usage
class Vehicle(ABC):
    @abstractmethod
    def start_engine(self):
        pass

1 usage
class Car(Vehicle):
    1 usage
    def start_engine(self):
        print('Engine started for the car')

my_car = Car()
my_car.start_engine()
```

Practical Applications:
- Abstracting database operations.
- Abstracting user interactions in GUI applications.
- Abstracting API layers in web development.

Exercises:
1. Create an abstract class for a shape with methods to calculate area and perimeter.
2. Implement specific shapes like Circle and Rectangle that inherit from the abstract class.