In Python, an **Abstract Base Class (ABC)** is a class that cannot be instantiated directly. Instead, it serves as a **blueprint** for other classes (child classes) by defining **abstract methods** that must be implemented in the child classes.

- **Abstract Method**: A method declared in an abstract class but has no implementation (just a `pass` or a docstring).
- **Concrete Class**: A child class that **must** provide implementations for all abstract methods of its parent ABC.

What Happens if Abstract Methods Are Not Defined in the Child Class?

If a child class **does not implement all abstract methods** from its parent ABC, Python will raise a `TypeError` when you try to create an instance of that child class.

Example with Missing Implementation

```
from abc import ABC, abstractmethod

class Vehicle(ABC):
    @abstractmethod
    def start(self):
        pass

    @abstractmethod
    def stop(self):
        pass

class Car(Vehicle):  # Only implements `start()`, missing `stop()`
    def start(self):
        print("Starting the car engine")

# Attempting to create an instance of Car
car = Car()  # ❌ TypeError: Can't instantiate abstract class Car with abstract
method stop
```

Error:

```
TypeError: Can't instantiate abstract class Car with abstract method stop
```

- Since `Car` did not implement `stop()`, Python prevents instantiation.

Why Does This Happen?

- Abstract methods enforce a **contract** (rules) that all child classes must follow.
- If a child class fails to implement any abstract method, it remains **incomplete** and cannot be used.
- This ensures **consistency** in the structure of derived classes.

How to Fix It?

The child class **must implement all abstract methods** from the parent ABC.

Correct Implementation

```
class Car(Vehicle):
    def start(self):
        print("Starting the car engine")

    def stop(self):  # Now properly implemented
        print("Stopping the car engine")

car = Car()  # ✅ Works fine
car.start()  # Output: "Starting the car engine"
car.stop()   # Output: "Stopping the car engine"
```

Key Takeaways

1. **Abstract Base Classes (ABCs)** define a structure that child classes must follow.
2. **Abstract methods** must be implemented in child classes, or Python raises a `TypeError`.
3. This mechanism ensures **better code organization** and **prevents incomplete implementations**.

This is particularly useful in **large projects** where multiple developers need to ensure that certain methods are always available in derived classes. 🚀