Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @dataengineeringtamil

🏠 Python OOP - Abstraction Guide

✅ What is Abstraction?

Abstraction means **hiding the internal details** and showing only the **essential features** to the outside world.

It helps in:

- Hiding complex logic
- Exposing only required behavior
- Keeping your code clean and secure

🔑 Real-Life Analogy

Think of a **TV remote**:

- You press the "Power" or "Volume" button
- You don't know (or care) how the circuit inside works

You just use **what is exposed**, and the **complexity is hidden**.
That's **abstraction**.

🔧 Abstraction in Python

Python supports abstraction using the **abc module** (Abstract Base Classes).

- Use `from abc import ABC, abstractmethod`

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @dataengineeringtamil

- Create an abstract class by inheriting from `ABC`
- Use `@abstractmethod` to define methods without implementation
- You **cannot create objects** of an abstract class directly

---

📘 Basic Syntax

```
from abc import ABC, abstractmethod

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

class Car(Vehicle):
    def start_engine(self):
        print("Car engine started")

c = Car()
c.start_engine()
```

❌ This would raise an error:

```
v = Vehicle()  # ❌ TypeError: Can't instantiate abstract class
```

💼 Real-Life Project Analogy

```
from abc import ABC, abstractmethod

# Architect defines the plan
class FeaturePlan(ABC):
    @abstractmethod
    def login(self):
        pass
    @abstractmethod
    def logout(self):
        pass

# Developer implements it
class WebApp(FeaturePlan):
    def login(self):
        self.__encrypt()   # internal logic hidden
        print("WebApp Login Done ✅")

    def logout(self):
        print("WebApp Logout Done ✅")

    def __encrypt(self):
        print("Encrypting user data...")  # hidden from outside

# Usage
app = WebApp()
app.login()
app.logout()
```

🧠 How Is This Secure?

- The abstract class **exposes only required methods** (`login`, `logout`)
- The actual logic like `__encrypt()` is **hidden inside the implementation**
- You enforce **structure + security + modularity**

✨ Abstract Method with Implementation

Yes, abstract methods **can** have a default implementation, but:

> Subclasses **must still override them**.

```
from abc import ABC, abstractmethod

class Demo(ABC):
    @abstractmethod
    def greet(self):
        print("Hello from abstract method!")

class SubClass(Demo):
    def greet(self):
        super().greet()
        print("Hello from subclass")

SubClass().greet()
```

🏆 Benefits of Abstraction

1. **Security** – Hide sensitive data and internal logic
2. **Simplicity** – Only expose what is necessary
3. **Flexibility** – Easy to update internal code
4. **Enforces structure** – Forces child classes to implement required methods

---

❓ Common Interview Questions

1. What is abstraction?
2. How is abstraction achieved in Python?
3. What is the difference between abstraction and encapsulation?
4. Can you instantiate an abstract class?
5. What is the role of `@abstractmethod`?
6. Can an abstract class have normal methods?

7. Can abstract methods have code inside?

🧳 Resume Tip

Project Line Example:

> "Applied OOP abstraction by designing an abstract base `Payment` interface and implementing concrete classes for `CreditCard`, `UPI`, and `Wallet` payments in an e-commerce system."

About the Author

Gowtham SB is a **Data Engineering expert, educator, and content creator** with a passion for **big data technologies, as well as cloud and Gen AI** . With years of experience in the field, he has worked extensively with **cloud platforms, distributed systems, and data pipelines**, helping professionals and aspiring engineers master the art of data engineering.

Beyond his technical expertise, Gowtham is a **renowned mentor and speaker**, sharing his insights through engaging content on **YouTube and LinkedIn**. He has built one of the **largest Tamil Data Engineering communities**, guiding thousands of learners to excel in their careers.

Through his deep industry knowledge and hands-on approach, Gowtham continues to **bridge the gap between learning and real-world implementation**, empowering individuals to build **scalable, high-performance data solutions**.

Socials

🎥**YouTube** - https://www.youtube.com/@dataengineeringvideos

📷**Instagram** - https://instagram.com/dataengineeringtamil

📷**Instagram** - https://instagram.com/thedatatech.in

🤝**Connect for 1:1** - https://topmate.io/dataengineering/

💼**LinkedIn** - https://www.linkedin.com/in/sbgowtham/

🌐**Website** - https://codewithgowtham.blogspot.com

💻**GitHub** - http://github.com/Gowthamdataengineer

💬**Whats App** - https://lnkd.in/g5JrHw8q

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @dataengineeringtamil

📧**Email** - atozknowledge.com@gmail.com

📱 **All My Socials** - https://lnkd.in/gf8k3aCH