Abstract Base Classes (ABCs) — Validated Duck Typing

Why this concept exists

Pure duck typing defers errors until deep runtime. Large systems and frameworks need early, explicit validation of expected behavior without forcing concrete inheritance hierarchies.

Target Usage

```
gateway = UpiGateway()
gateway.pay(500)
```

Coding Problem

Design a plugin or extension system where implementations must provide specific behavior. Incorrect implementations should fail early and clearly.

Baseline Solution

```
from abc import ABC, abstractmethod

class PaymentGateway(ABC):
    @abstractmethod
    def pay(self, amount):
        pass

class UpiGateway(PaymentGateway):
    def pay(self, amount):
        print(f"Paid {amount} via UPI")
```

Runtime Validation

```
isinstance(gateway, PaymentGateway)
```

Key Insight

ABCs are still inheritance, but used only for validation, not reuse. They formalize duck typing at runtime and provide safe extension points for dynamic systems.