Building an Object-Oriented ATM System in Python with PIN Authentication

Introduction

This project is designed to help you understand and apply the four foundational principles of Object-Oriented Programming (OOP) in Python: encapsulation, abstraction, inheritance, and polymorphism. You will build a command-line-based ATM banking system where users can create and interact with bank accounts securely using PIN authentication.

This project does not require external libraries and can be implemented using core Python features.

Project Overview

The objective is to build an ATM system where users can:

Create a bank account (either a savings or a current account)

Log in using a secure 4-digit PIN

Perform core banking operations such as deposits, withdrawals, balance checks, and interest calculations

Exit the system when done

The project structure must reflect the principles of object-oriented programming and demonstrate how encapsulation, abstraction, inheritance, and polymorphism are used in real-world systems.

Core Concepts to Implement

1. Account Types

The system will include two account types:

SavingsAccount: Earns 5% interest on the balance

CurrentAccount: Earns 2% interest on the balance

Both account types will inherit from a common abstract base class called Account.

2. Functional Requirements

Your program should support the following features:

Create an account: The user provides a name, a 4-digit PIN, and an initial balance. The account type is selected (savings or current).

Authenticate using a PIN: The user must enter the correct PIN to access account features.

Deposit money: Users can add money to their account. Only positive numbers should be accepted.

Withdraw money: Users can withdraw money only if the amount is available in the account.

Check balance: Displays the current balance of the user's account.

Calculate interest: Each account type calculates interest differently using its own logic.

Exit: The user can exit the session.

3. Object-Oriented Programming Requirements

This project must demonstrate the following OOP concepts:

Encapsulation:

The balance and PIN must be private attributes, accessible only through getter or utility methods like get_balance() and authenticate(). This ensures sensitive data is protected and cannot be modified directly from outside the class.

Abstraction:

The base class Account should be an abstract class. It should define an abstract method calculate_interest() that must be implemented in subclasses. This hides the internal implementation details of how interest is calculated.

Inheritance:

SavingsAccount and CurrentAccount should both inherit from the base Account class. Common logic such as deposit, withdraw, and get_balance should be defined in the parent class, while specific logic for calculating interest is implemented in the subclasses.

Polymorphism:

The calculate_interest() method should behave differently depending on whether the account is a savings or current account. This demonstrates polymorphism through method overriding.

Suggested Command-Line Menu

After successful authentication, the system should present a simple text menu that allows the user to select operations:

Copy code

Welcome to the ATM System

1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Calculate Interest
5. Exit

Enter your choice:

Each menu option should trigger a method in the account class, maintaining separation of concerns and reusability of code.

Technical Constraints and Suggestions

Use the abc module for creating abstract base classes and defining abstract methods.

Use private variables (__balance, __pin) to enforce encapsulation.

Use input() and print() to interact with the user through the terminal.

Type hints are optional but recommended for clarity.