Lesson: Encapsulation

Description: Understand how encapsulation protects data and ensures security.

Encapsulation is a fundamental OOP principle that focuses on bundling the data (attributes) and the methods (functions) that operate on the data into a single unit, typically a class. It also restricts direct access to certain parts of an object's data, enforcing controlled access through getters and setters.

Key Benefits of Encapsulation:
1. Protects sensitive data from unauthorized access.
2. Enhances code modularity by grouping related variables and methods.
3. Makes the code easier to understand and maintain.

Access Modifiers:
- `Public`: Accessible from anywhere.
- `Private`: Accessible only within the class.
- `Protected`: Accessible within the class and its subclasses.

Example:

```
class BankAccount:
    def __init__(self, balance):
        self.__balance = balance  # Private attribute

    def deposit(self, amount):
        if amount > 0:
            self.__balance += amount
        else:
            print('Invalid deposit amount')

    def get_balance(self):
        return self.__balance

account = BankAccount(1000)
account.deposit(500)
print(account.get_balance())
```

Real-World Applications:

- Encapsulating database credentials in web applications.
- Safeguarding user information in mobile apps.
- Securing financial data in banking software.

Exercises:
1. Create a class for a product with private attributes for name and price. Implement getters and setters.
2. Modify the example above to include a withdrawal method with proper validation.