DAY-22 MORNING ASSESSMENT

# A CORE PYTHON AND DATA HANDLING

1. Auto-save on every operation  
Currently, you call self.\_save() manually. Instead, just call \_save() inside insert\_book(), discard\_book(), enroll\_member(), borrow(), and receive().  
  
2.   
def search\_books(self, query: str):  
   query = query.lower()  
   results = [b for b in self.books.values() if query in b.title.lower() or query in b.author.lower()]  
   if results:  
       for b in results:  
           print(f"[{b.book\_id}] {b.title} by {b.author}")  
   else:  
       print("No books found matching query.")  
  
3.   
def sort\_books(self, key: str = "title"):  
   if key == "title":  
       sorted\_books = sorted(self.books.values(), key=lambda b: b.title)  
   elif key == "author":  
       sorted\_books = sorted(self.books.values(), key=lambda b: b.author)  
   elif key == "availability":  
       sorted\_books = sorted(self.books.values(), key=lambda b: b.available, reverse=True)  
   else:  
       print("Invalid sort key")  
       return  
   for b in sorted\_books:  
       print(f"[{b.book\_id}] {b.title} by {b.author} — {'Available' if b.available else 'Issued'}")  
  
4. def borrowed\_books(self):  
   return [b for b in self.books.values() if not b.available]  
  
5. import csv  
def export\_csv(self, filename="library\_export.csv"):  
   with open(filename, "w", newline="", encoding="utf-8") as f:  
       writer = csv.writer(f)  
       writer.writerow(["Book ID", "Title", "Author", "ISBN", "Available"])  
       for b in self.books.values():  
           writer.writerow([b.book\_id, b.title, b.author, b.isbn, b.available])  
   print(f"Exported data to {filename}")  
  
6. Use dataclasses  
from dataclasses import dataclass  
  
@dataclass  
class Book:  
   book\_id: str  
   title: str  
   author: str  
   isbn: str  
   available: bool = True  
Replace old Book class with dataclass.  
Similarly, members can also be dataclasses.  
  
7. zip() for custom member-book report  
  
def member\_book\_report(self):  
   pairs = [(m.name, loan["book\_id"]) for m in self.members.values() for loan in m.borrowed]  
   for name, book in pairs:  
       print(f"{name} → {book}")  
  
8. import re  
def validate\_isbn(isbn: str) -> bool:  
   pattern = r"^(97(8|9))?\d{9}(\d|X)$"   # ISBN-10 or ISBN-13  
   return bool(re.match(pattern, isbn))

Advanced OOP Concepts  
9. class StaffMember(BaseMember):  
   def display\_info(self) -> str:  
       return f"Staff {self.member\_id} - {self.name} (Full permissions)"  
  
   def remove\_book(self, portal, book\_id: str):  
       if book\_id in portal.books:  
           del portal.books[book\_id]  
           portal.\_save()  
           print("Book removed by staff.")  
       else:  
           print("Book not found.")  
Normal members cannot discard books.  
Only StaffMember has permission.  
  
10.   
@dataclass  
class Book:  
   book\_id: str  
   title: str  
   author: str  
   isbn: str  
   available: bool = True  
  
   def \_\_eq\_\_(self, other):  
       if isinstance(other, Book):  
           return self.isbn == other.isbn  
       return False  
  
   def \_\_lt\_\_(self, other):  
       if isinstance(other, Book):  
           return self.isbn < other.isbn  
       return NotImplemented  
book1 == book2 → compares ISBN.  
book1 < book2 → compares ISBN numerically.