**Day-22 Morning Assessment**

**A. Core Python & Data Handling**

1. Instead of calling save() manually, wrap it inside methods like add\_book(), borrow\_book(), etc.  
  
import json  
  
class Library:  
   def \_\_init\_\_(self, filename="library.json"):  
       self.books = {}  
       self.members = {}  
       self.filename = filename  
       self.load()  
  
   def save(self):  
       with open(self.filename, "w") as f:  
           json.dump({  
               "books": {isbn: b.\_\_dict\_\_ for isbn, b in self.books.items()},  
               "members": {mid: m.\_\_dict\_\_ for mid, m in self.members.items()}  
           }, f, indent=4)  
  
   def load(self):  
       try:  
           with open(self.filename, "r") as f:  
               data = json.load(f)  
               # reconstruct objects if using dataclasses (later step)  
       except FileNotFoundError:  
           pass  
  
   def add\_book(self, book):  
       self.books[book.isbn] = book  
       self.save()   # auto-save  
  
2. def search\_books(self, query):  
   query = query.lower()  
   return [b for b in self.books.values()   
           if query in b.title.lower() or query in b.author.lower()]  
  
3. def sort\_books(self, by="title"):  
   return sorted(self.books.values(), key=lambda b: getattr(b, by))  
  
4. borrowed = [b for b in self.books.values() if not b.is\_available]  
  
5. import csv  
  
def export\_csv(self, filename="library.csv"):  
   with open(filename, "w", newline="") as f:  
       writer = csv.writer(f)  
       writer.writerow(["ISBN", "Title", "Author", "Available"])  
       for b in self.books.values():  
           writer.writerow([b.isbn, b.title, b.author, b.is\_available])  
  
6. from dataclasses import dataclass  
  
@dataclass  
class Book:  
   isbn: str  
   title: str  
   author: str  
   is\_available: bool = True  
  
@dataclass  
class Member:  
   member\_id: str  
   name: str  
   borrowed\_books: list = None  
Then store in dictionaries:  
  
self.books: dict[str, Book] = {}  
self.members: dict[str, Member] = {}  
  
7. def borrowed\_report(self):  
   for m in self.members.values():  
       pairs = list(zip([m.name]\*len(m.borrowed\_books), m.borrowed\_books))  
       print(pairs)  
  
8. import re  
  
def validate\_isbn(isbn: str) -> bool:  
   pattern = r"^(97(8|9))?\d{9}(\d|X)$"  
   return bool(re.match(pattern, isbn))  
  
**B. Advanced OOP Concepts**

9. class Member:  
   def \_\_init\_\_(self, member\_id, name):  
       self.member\_id = member\_id  
       self.name = name  
       self.borrowed\_books = []  
  
class StaffMember(Member):  
   def remove\_book(self, library, isbn):  
       if isbn in library.books:  
           del library.books[isbn]  
           library.save()  
  
10. @dataclass  
class Book:  
   isbn: str  
   title: str  
   author: str  
   is\_available: bool = True  
  
   def \_\_eq\_\_(self, other):  
       return self.isbn == other.isbn  
  
   def \_\_lt\_\_(self, other):  
       return self.isbn < other.isbn