SMART LIBRARY CODE

import os

from typing import Dict, List, Optional

BOOKS\_FILE = "books.txt"

MEMBERS\_FILE = "members.txt"

DELIM = "|"

class Book:

def \_\_init\_\_(self, book\_id: str, title: str, author: str, isbn: str, available: bool = True):

self.book\_id = book\_id

self.title = title

self.author = author

self.isbn = isbn

self.available = available

def to\_line(self) -> str:

return DELIM.join([self.book\_id, self.title, self.author, self.isbn, "1" if self.available else "0"]) + "\n"

@staticmethod

def from\_line(line: str) -> "Book":

parts = line.strip().split(DELIM)

if len(parts) != 5:

raise ValueError("Corrupt book line: " + line)

return Book(parts[0], parts[1], parts[2], parts[3], parts[4] == "1")

def \_\_str\_\_(self):

status = "Available" if self.available else "Issued"

return f"[{self.book\_id}] {self.title} by {self.author} (ISBN: {self.isbn}) — {status}"

class Member:

def \_\_init\_\_(self, member\_id: str, name: str, borrowed: Optional[List[str]] = None):

self.member\_id = member\_id

self.name = name

self.borrowed = borrowed or [] # store book\_ids

def to\_line(self) -> str:

borrowed\_str = ",".join(self.borrowed)

return DELIM.join([self.member\_id, self.name, borrowed\_str]) + "\n"

@staticmethod

def from\_line(line: str) -> "Member":

parts = line.strip().split(DELIM)

if len(parts) != 3:

raise ValueError("Corrupt member line: " + line)

borrowed = parts[2].split(",") if parts[2] else []

return Member(parts[0], parts[1], borrowed)

def \_\_str\_\_(self):

return f"[{self.member\_id}] {self.name} — Borrowed: {self.borrowed if self.borrowed else 'None'}"

class Library:

def \_\_init\_\_(self):

self.books: Dict[str, Book] = {}

self.members: Dict[str, Member] = {}

self.load()

def load(self):

if os.path.exists(BOOKS\_FILE):

with open(BOOKS\_FILE, "r", encoding="utf-8") as f:

for line in f:

if line.strip():

b = Book.from\_line(line)

self.books[b.book\_id] = b

if os.path.exists(MEMBERS\_FILE):

with open(MEMBERS\_FILE, "r", encoding="utf-8") as f:

for line in f:

if line.strip():

m = Member.from\_line(line)

self.members[m.member\_id] = m

def save(self):

with open(BOOKS\_FILE, "w", encoding="utf-8") as f:

for b in self.books.values():

f.write(b.to\_line())

with open(MEMBERS\_FILE, "w", encoding="utf-8") as f:

for m in self.members.values():

f.write(m.to\_line())

def add\_book(self, book\_id: str, title: str, author: str, isbn: str):

if book\_id in self.books:

print("Book ID already exists.")

return

self.books[book\_id] = Book(book\_id, title, author, isbn)

self.save()

print("Book added.")

def remove\_book(self, book\_id: str):

b = self.books.get(book\_id)

if not b:

print("Book not found.")

return

if not b.available:

print("Cannot remove: Book currently issued.")

return

del self.books[book\_id]

self.save()

print("Book removed.")

def register\_member(self, member\_id: str, name: str):

if member\_id in self.members:

print("Member ID already exists.")

return

self.members[member\_id] = Member(member\_id, name)

self.save()

print("Member registered.")

def borrow\_book(self, member\_id: str, book\_id: str):

m = self.members.get(member\_id)

b = self.books.get(book\_id)

if not m:

print("Member not found.")

return

if not b:

print("Book not found.")

return

if not b.available:

print("Book already issued.")

return

b.available = False

m.borrowed.append(book\_id)

self.save()

print(f"{m.name} borrowed '{b.title}'.")

def return\_book(self, member\_id: str, book\_id: str):

m = self.members.get(member\_id)

b = self.books.get(book\_id)

if not m or not b:

print("Invalid member or book ID.")

return

if book\_id not in m.borrowed:

print("This member didn't borrow that book.")

return

b.available = True

m.borrowed.remove(book\_id)

self.save()

print(f"'{b.title}' returned by {m.name}.")

def list\_books(self):

if not self.books:

print("No books in library.")

return

for b in self.books.values():

print(b)

def list\_members(self):

if not self.members:

print("No members registered.")

return

for m in self.members.values():

print(m)

def run\_cli():

lib = Library()

MENU = """

Smart Library Manager

1) Add Book

2) Remove Book

3) Register Member

4) Borrow Book

5) Return Book

6) List Books

7) List Members

0) Exit

Choice: """

while True:

choice = input(MENU).strip()

if choice == "1":

lib.add\_book(input("Book ID: "), input("Title: "), input("Author: "), input("ISBN: "))

elif choice == "2":

lib.remove\_book(input("Book ID: "))

elif choice == "3":

lib.register\_member(input("Member ID: "), input("Name: "))

elif choice == "4":

lib.borrow\_book(input("Member ID: "), input("Book ID: "))

elif choice == "5":

lib.return\_book(input("Member ID: "), input("Book ID: "))

elif choice == "6":

lib.list\_books()

elif choice == "7":

lib.list\_members()

elif choice == "0":

break

else:

print("Invalid choice.")

if \_\_name\_\_ == "\_\_main\_\_":

run\_cli()