Day-22 Evening Assessment

11. class Library:

     def \_\_init\_\_(self):

         self.books = {}

@property

     def available\_books\_count(self):

         return sum(1 for book in self.books.values() if book.is\_available)

12. from abc import ABC, abstractmethod

  class Person(ABC):

     def \_\_init\_\_(self, name):

[self.name](http://self.name) = name

     @abstractmethod

     def get\_role(self):

         pass

class Member(Person):

     def get\_role(self):

         return "Member"

class Librarian(Person):

      def get\_role(self):

         return "Librarian"

13. class StudentMember(Member):

     def get\_role(self):

         return "Student Member"

class FacultyMember(Member)

def get\_role(self):

         return "Faculty Member"

class ResearchScholar(StudentMember, FacultyMember):

     def get\_role(self):

         return "Research Scholar"

14. class Book:

     def \_\_init\_\_(self, isbn, title, author):

         self.isbn = isbn

         self.title = title

[self.author](http://self.author) = author

     def \_\_str\_\_(self):

return f"{self.title} by {[self.author](http://self.author)}"

     def \_\_repr\_\_(self):

return f"Book(isbn={self.isbn}, title='{self.title}', author='{[self.author](http://self.author)}')"

15. class Library:

     \_instance = None

     def \_\_new\_\_(cls, \*args, \*\*kwargs):

         if cls.\_instance is None:

             cls.\_instance = super(Library, cls).\_\_new\_\_(cls)

         return cls.\_instance

16. class MemberFactory:

     @staticmethod

     def create\_member(member\_type, name):

         if member\_type == "student":

             return StudentMember(name)

         elif member\_type == "faculty":

             return FacultyMember(name)

         else:

             return Member(name)

17. class Library:

     def add\_book(self, book):

         self.books[book.isbn] = book

         return self

     def register\_member(self, member):

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

         return self

library.add\_book(book1).register\_member(member1)

18. import json

  class JsonMixin:

     def to\_json(self):

         return json.dumps(self.\_\_dict\_\_)

     @classmethod

     def from\_json(cls, data):

         return cls(\*\*json.loads(data))

class Book(JsonMixin):

     def \_\_init\_\_(self, isbn, title, author):

         self.isbn = isbn

         self.title = title

[self.author](http://self.author) = author

Error Handling & Robustness

19. class BookNotAvailableError(Exception):

     pass

class MemberNotFoundError(Exception):

     pass

20. import logging

  logging.basicConfig(level=logging.ERROR)

  try:

     with open("library\_data.txt", "r") as f:

data = [f.read](http://f.read)()

except FileNotFoundError as e:

     logging.error(f"File not found: {e}")

else:

     print("File read successfully!")

finally:

     print("File operation attempted.")