SUPERIOR UNIVERSITY

NAME	ALI MUSTAFA SHAH
REGISTRATION NUMBER	SU92-BSDSM-S24-005
SECTION	2A
SEMESTER	2ND
TASK	9
SUBJECT	OOP(LAB)
SUBMITED TO	SIR RASIKH

LAB-9 TASK

Task 1; Program for managing different types of documents:

1. Define a parent class called Document with attributes title and author. Include a method display_info to display the title and author of the document.

2. Create a child class Book inheriting from Document. The Book class should have additional attributes like genre and pages. Implement function overriding for the display_info method to include the genre and pages information.

3. Create another child class Article inheriting from Document. The Article class should have additional attributes like journal and DOI (Digital Object Identifier). Implement function overriding for the display_info method to include the journal and DOI information.
4. Implement function overloading in the Book class to handle different ways of initializing

a Book object. Allow initialization with just title and author, or title, author, genre, and pages.

5. Implement function overloading in the Article class to handle different ways of initializing Article object. Allow initialization with just title and author, or title, author, journal, and DOI.

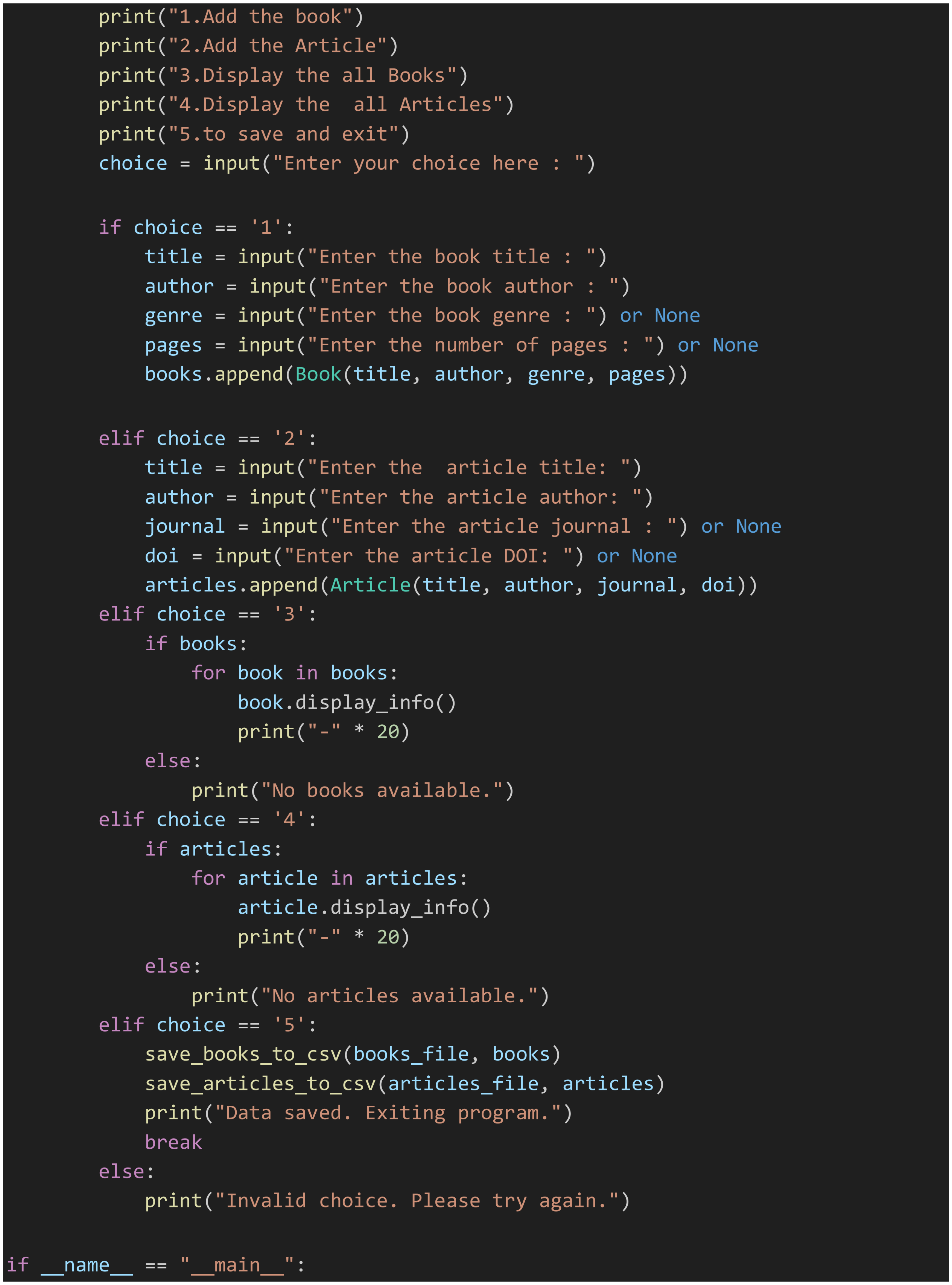
6. Implement file handling to store and retrieve information about books and articles. Use text files to store the information in a structured format.

(Bonus marks if you create CSV instead of a simple TXT file)

SOLUTION

```python
import csv

class Document:
    def __init__(self, title, author):
        self.title = title
        self.author = author



    def display_info(self):
        print(f"Title: {self.title}")
        print(f"Author: {self.author}")



class Book(Document):
    def __init__(self, title, author, genre=None, pages=None):
        super().__init__(title, author)
        self.genre = genre
        self.pages = pages



    def display_info(self):
        super().display_info()
        if self.genre:
            print(f"Genre: {self.genre}")

        if self.pages:
            print(f"Pages: {self.pages}")



class Article(Document):
    def __init__(self, title, author, journal=None, doi=None):
        super().__init__(title, author)
        self.journal = journal
        self.doi = doi

    def display_info(self):
        super().display_info()
        if self.journal:
            print(f"Journal: {self.journal}")

        if self.doi:
            print(f"DOI: {self.doi}")



def save_books_to_csv(file_name, books):
    with open(file_name, 'w', newline='') as file:

        writer = csv.writer(file)
        writer.writerow(["Title", "Author", "Genre", "Pages"])

        for book in books:
            writer.writerow([book.title, book.author, book.genre, book.pages])

def load_books_from_csv(file_name):
    books = []

    try:
        with open(file_name, 'r') as file:
            reader = csv.DictReader(file)
            for row in reader:
                books.append(Book(row['Title'], row['Author'], row['Genre'],
row['Pages']))

    except FileNotFoundError:
        print(f"File '{file_name}' not found. Starting with an empty book list.")

    return books
```

```python
def save_articles_to_csv(file_name, articles):
    with open(file_name, 'w', newline='') as file:
        writer = csv.writer(file)
        writer.writerow(["Title", "Author", "Journal", "DOI"])

        for article in articles:
            writer.writerow([article.title, article.author, article.journal,
article.doi])



def load_articles_from_csv(file_name):
    articles = []

    try:
        with open(file_name, 'r') as file:
            reader = csv.DictReader(file)
            for row in reader:
                articles.append(Article(row['Title'], row['Author'],
row['Journal'], row['DOI']))

    except FileNotFoundError:
        print(f"File '{file_name}' not found. Starting with an empty article
list.")

    return articles



def main():
    books_file = "books.csv"

    articles_file = "articles.csv"

    books = load_books_from_csv(books_file)

    articles = load_articles_from_csv(articles_file)

    while True:
        print("\nDocument Management System")
```



```python
        print("1.Add the book")
        print("2.Add the Article")
        print("3.Display the all Books")
        print("4.Display the  all Articles")
        print("5.to save and exit")
        choice = input("Enter your choice here : ")

        if choice == '1':
            title = input("Enter the book title : ")
            author = input("Enter the book author : ")
            genre = input("Enter the book genre : ") or None
            pages = input("Enter the number of pages : ") or None
            books.append(Book(title, author, genre, pages))

        elif choice == '2':
            title = input("Enter the  article title: ")
            author = input("Enter the article author: ")
            journal = input("Enter the article journal : ") or None
            doi = input("Enter the article DOI: ") or None
            articles.append(Article(title, author, journal, doi))
        elif choice == '3':
            if books:
                for book in books:
                    book.display_info()
                    print("-" * 20)
            else:
                print("No books available.")
        elif choice == '4':
            if articles:
                for article in articles:
                    article.display_info()
                    print("-" * 20)
            else:
                print("No articles available.")
        elif choice == '5':
            save_books_to_csv(books_file, books)
            save_articles_to_csv(articles_file, articles)
            print("Data saved. Exiting program.")
            break
        else:
            print("Invalid choice. Please try again.")

if __name__ == "__main__":
    main()
```