Івано-Франківський національний технічний університет нафти і газу

Кафедра комп’ютерних систем і мереж

**ЛАБОРАТОРНА РОБОТА №7**

з дисципліни Практикум Python

Тема: Розрообка програм з використанням класів в Python

Виконав:

Ст. гр. КІ-23-2 (Боднар Р.В.)

Перевірив: (Кропивницький Д. Р.)

м. Івано-Франківськ

2025

**Мета роботи:** Ознайомитись з принципами реалізації об’єктноорієнтованого програмування у мові Python та навчитись використовувати його для розроблення програмного забезпечення.

**Варіант 1**

**Завдання:**

1. Розробити клас "домашня бібліотека". Додати конструктор, який приймає ціле число – порядковий номер книги та словник, що містить інформацію про книгу наступному форматі: 1) автор; 2) назва; 3) видавництво; 4) жанр; 5) рік видання. Реалізувати можливість роботи з довільним числом книг, пошуку по книгах за декількома параметрами (за автором, за роком видання, за жанром тощо), додавання книг у бібліотеку, видалення книг з неї, доступу до книги за номером. Написати програму, що буде демонструвати всі розроблені елементи класу.

**Хід роботи**

**Код розроблених класів та їх використання:**

from dataclasses import dataclass, asdict

from typing import Dict, List, Optional, Tuple, Any

@dataclass

class Book:

    """Represents a single book."""

    author: str

    title: str

    publisher: str

    genre: str

    year: int

    def \_\_str\_\_(self) -> str:

        return f"{self.title} by {self.author} ({self.year})"

    def to\_dict(self) -> Dict[str, Any]:

        return asdict(self)

class HomeLibrary:

    """

    HomeLibrary holds Book instances indexed by integer IDs.

    Search supports:

      - substring match for string fields (case-insensitive),

      - exact match for non-strings,

      - range for numeric fields if you pass a 2-tuple (min, max),

      - membership if you pass a list/tuple/set of allowed values.

    """

    def \_\_init\_\_(self) -> None:

        self.\_books: Dict[int, Book] = {}

        self.\_next\_id: int = 1

    def add\_book(self, book: Book, book\_id: Optional[int] = None) -> int:

        """Add a book. If book\_id is None, it is auto-assigned. Returns the id used."""

        if book\_id is None:

            book\_id = self.\_next\_id

            self.\_next\_id += 1

        else:

            if book\_id in self.\_books:

                raise ValueError(f"Book id {book\_id} already exists.")

            if book\_id >= self.\_next\_id:

                self.\_next\_id = book\_id + 1

        self.\_books[book\_id] = book

        return book\_id

    def remove\_book(self, book\_id: int) -> Optional[Book]:

        """Remove and return the book by id, or None if not found."""

        return self.\_books.pop(book\_id, None)

    def get\_book(self, book\_id: int) -> Optional[Book]:

        """Return Book or None if not found."""

        return self.\_books.get(book\_id)

    def update\_book(self, book\_id: int, \*\*fields) -> bool:

        """

        Update fields of a book (author, title, publisher, genre, year).

        Returns True if updated, False if book\_id not found.

        """

        book = self.\_books.get(book\_id)

        if not book:

            return False

        for k, v in fields.items():

            if hasattr(book, k):

                setattr(book, k, v)

        return True

    def search\_books(self, \*\*criteria) -> List[Tuple[int, Book]]:

        """

        Search books by named criteria.

        Examples:

          search\_books(author="Tolkien")            # substring match (case-insensitive)

          search\_books(year=(1930,1950))           # year in range inclusive

          search\_books(genre=["Fantasy","Sci-Fi"]) # genre in list

          search\_books(year=1997)                  # exact numeric match

        Returns list of (id, Book).

        """

        results: List[Tuple[int, Book]] = []

        for bid, book in self.\_books.items():

            ok = True

            for key, value in criteria.items():

                attr = getattr(book, key, None)

                if attr is None:

                    ok = False

                    break

                if isinstance(value, (list, tuple)) and len(value) == 2 and \

                   all(isinstance(x, (int, float)) for x in value):

                    if not (value[0] <= attr <= value[1]):

                        ok = False

                        break

                elif isinstance(value, (list, tuple, set)):

                    if attr not in value:

                        ok = False

                        break

                elif isinstance(value, str) and isinstance(attr, str):

                    if value.lower() not in attr.lower():

                        ok = False

                        break

                else:

                    if attr != value:

                        ok = False

                        break

            if ok:

                results.append((bid, book))

        return results

    def list\_books(self) -> List[Tuple[int, Book]]:

        """Return list of (id, Book)."""

        return list(self.\_books.items())

    def \_\_len\_\_(self) -> int:

        return len(self.\_books)

    def \_\_iter\_\_(self):

        return iter(self.\_books.items())

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

    lib = HomeLibrary()

    b1 = Book("J.R.R. Tolkien", "The Hobbit", "Allen & Unwin", "Fantasy", 1937)

    b2 = Book("George Orwell", "1984", "Secker & Warburg", "Dystopian", 1949)

    id1 = lib.add\_book(b1)

    id2 = lib.add\_book(b2)

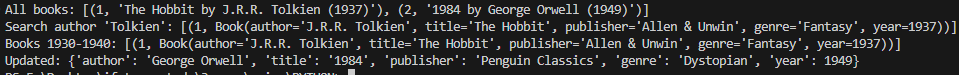
    print("All books:", [(i, str(b)) for i,b in lib.list\_books()])

    print("Search author 'Tolkien':", lib.search\_books(author="Tolkien"))

    print("Books 1930-1940:", lib.search\_books(year=(1930,1940)))

    lib.update\_book(id2, publisher="Penguin Classics")

    print("Updated:", lib.get\_book(id2).to\_dict())

****

**Рисунок 1 - Результат виконання завдання**

**Висновок:** У ході виконання роботи було розроблено класи Book та HomeLibrary, що дозволили ознайомитись з принципами об’єктно-орієнтованого програмування в Python. Реалізовано додавання, видалення, пошук та оновлення інформації про книги, а також роботу з усією бібліотекою.