****

MAKERERE UNIVERSITY BUSINESS SCHOOL

DIPLOMA IN COMPUTER SCIENCE

COMPUTER PROGRAMMING TWO

COURSEWORK TWO

|  |  |  |
| --- | --- | --- |
| **NAME** | **REGISTRATION NUMBER** | **STUDENT NUMBER** |
| MUNGU JAKISA MAURICE | B/23/U/D0634/PS | 2300900634 |
| KAASA ASTHME | B/23/U/D0120/PS | 2300900120 |
| HAUMBA JOSEPH WANGUDI | B/23/U/D0599/PS | 2300900599 |

**DOCUMENTATION ON STUDENT CONTRIBUTION**

We worked on the program as a group of three each giving relevant contribution and research for it.

It was all a combined effort to develop the program as we taught ourselves enough to contribute since the functions seemed quite hard to work with.

Several decisions were made for the work being done to decide what best would work with the program.

KAASA ASTHME:

* worked on implementing the first part of the code that involved age entry. This involved designing the error handling of the work using try and except.

HAUMBA JOSEPH WANGUDI:

* worked on designing the conditional statements like if statements.
* Worked on the section for the fav\_number and provided the nested if statements used while considering use of nested ifs on even numbers other than odd numbers

MUNGU JAKISA MAURICE:

* provided research elements on loops and advanced looping.
* Combined code
* Included time delays in the program to best simulate a running program.

**PROGRAM SUMMARY REPORT**

This Python program is designed to interact with users, gather personal information, display a greeting, calculate their year of birth, provide response to the favorite number of a user and showcase a mini library system with book search and sorting capabilities.

**Class Design**

Class: Book

Attributes:

title: Holds the title of the book.

author: Holds the author’s name.

year\_published: Stores the year when the book was published.

Method: description(): A method that returns a formatted string describing the book, including its title, author, and publication year. It provides a clean, readable output for each book

**Function Usage**

The program employs functions to carry out specific tasks, which contributes to modularity and readability. Each function has a well-defined purpose, enhancing reusability and ease of maintenance.

1. sort\_books\_by\_year(books)

Purpose: This function sorts a list of Book objects by their year\_published attribute in ascending order.

Parameters:

books: A list of Book instances to be sorted.

Returns: A sorted list of Book instances.

Using a lambda function as the key, it sorts books based on their publication year. This function is called once to display the book list in chronological order, simplifying the process of showing books by publication date.

1. search\_book\_by\_title(books, title)

Purpose: This function searches for a Book in the provided list based on its title.

Parameters:

books: A list of Book instances to be searched.

title: A string containing the title of the book to search for.

Iterates through the list of books, comparing each book's title attribute with the search term. If a match is found, it prints the book’s description using the description() method. If not, it prints a "Book not found" message.

Each function is designed for specific tasks, allowing for organized, efficient code and reducing the need for redundant code blocks.

**Iteration Techniques**

The program uses various iteration techniques to manage user input, book display, and book search.

1. While Loop:

Purpose: The program uses while loops to handle user input for tasks that require repeated input until valid data is received.

Examples:

Age Input: A while loop is used to continuously prompt the user for their age until they enter a valid integer.

Favorite Number: Similarly, a while loop collects a favorite number from the user, ensuring input validation.

Reading Preference: In the book section, a while loop checks if the user would like to search for books by title. This loop continues until the user inputs “exit.”

1. For Loop:

Purpose: The program uses a for loop to iterate through lists, specifically the books list, to display each book and sorted list.

Examples:

Book Display: In the book listing, a for loop with the enumerate() function iterates over the list of books and prints each book’s description. The enumerate() function also provides an index for user-friendly numbering of books.

Sorted Book Display: The program iterates over sorted\_books to display books in chronological order by year, enhancing readability for the user.

1. Lambda Function:

Purpose: Within the sort\_books\_by\_year() function, a lambda function is used as a key to determine the sorting order.

This function applies lambda book: book.year\_published to sort based on the year\_published attribute, demonstrating Python’s ability to streamline single-line operations within functions.

**Key Programming Concepts**

1. Error Handling:

try-except blocks are used to validate user input, preventing crashes due to invalid data types. This is particularly important in parts of the code where the program expects an integer (e.g., for age and favorite number).

2. User Interaction and Input Validation:

The program prompts the user to interact through input() commands, with validation checks to ensure they enter acceptable values.

3. String Formatting:

The f-string formatting provides readable, clear outputs, especially when inserting user-specific or book-specific information into printed messages.

4. Object-Oriented Design:

Using the Book class showcases fundamental object-oriented programming (OOP) principles. This design encapsulates data and provides methods directly associated with Book instances, making it easy to extend or modify if additional book attributes or methods are needed.

**Summary**

This program demonstrates effective use of class-based design, modular functions, and iterative structures to achieve a simple but functional interactive user experience. The design choices emphasize code readability, modularity, and error-handling, creating a well-structured Python program.