**SUMMARY REPORT**

This report provides an overview of a Python program designed to facilitate user interaction through a series of personalized prompts. The program requests the user's name, age, and favorite number, implementing features such as error handling for non-numeric inputs and conditional logic to assess the favorite number's properties. Additionally, it includes a `Book` class, representing book objects with attributes like title, author, and publication year, along with a function to sort these books.

**1. Class Design**

* Class: “Book”

Purpose. The “Book”class is designed to represent a book with its essential attributes.

**Attributes:**

title: A string that represents the title of the book.

author: A string that represents the author of the book.

year\_published: An integer that indicates the year the book was published.

**Method:**

* description\_of\_the\_book: This method constructs and returns a string that contains the title, author, and year of publication into a single, descriptive string.

**Instantiation of Book Objects:**

Four instances of the “Book “ class are created:

**2. Function Usage**

* sort\_books\_by\_year(books)

Purpose: This function is responsible for sorting a list of Book objects based on the year\_published attribute.

Parameters: It takes one argument, “books”, which is expected to be a list of Book objects.

Return Value: The function returns a sorted list of books.

* Error Handling:

The function checks if the input list is empty. If it is, it returns an empty list, avoiding any potential errors that could occur from attempting to sort an empty list.

Sorting Mechanism: The function uses “sorted()” with a lambda function as its key. The lambda function extracts the `year\_published` attribute from each Book object for sorting.

**3. Iteration Techniques**

The program demonstrates iteration with both for loops and while loops:

* Using a For Loop:

After sorting the books, a for loop iterates over each Book object in the sorted list and prints its description using the previously defined description method.

* Using a While Loop:

A while loop is implemented to iterate through the sorted book list . The loop uses an index variable to access each book's description, demonstrating a different iteration technique that is often used when the number of iterations is not predetermined.

**4. Advanced Looping**

* Searching for a Book:

The program allows the user to search for a book by its title.

A while loop prompts the user for input continually until they enter "exit" to stop the search.

Inside this loop, another for loop searches through the sorted list of books to check if the title entered matches any in the list. This nested loop structure effectively implements both iteration and conditional checks.

If a match is found, it prints the book's details; if not, it informs the user that the book wasn't found.

**5. Error Handling**

* User Input Validation:

When the program prompts the user for their age and favorite number, it wraps this functionality in a try-except block to handle potential ValueError. This occurs specifically when the user inputs non-numeric values. If an error is caught, the program prints a message informing the user to enter a valid number, which enhances the user experience by preventing the program from crashing.

**Conclusion**

In summary, the Python program successfully combines user engagement, error management, and object-oriented programming principles to create an interactive application. The use of functions and classes enhances code organization and promotes reuse.