

**FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY**

page1image9370432

(**PROG211)**

**page1image9370432**

Title : individual assignment

Issue Date : Week 2

Due Date : Week 4

Lecturer/Examiner : Mr.ELLIJAH FULLAH

Name of Student/ :Alhaji Amadu Wurie Bah

Student ID No. : 905005720

Class : D.I.T 1202F

Semester/Year : 3/2

Academic Honesty Policy Statement

page1image9370432

I/We, hereby attest those contents of this attachment are my own work. Referenced works, articles, art, programs, papers or parts thereof are acknowledged at the end of this paper. This includes data excerpted from CD-ROMs, the Internet, other private networks, and other people’s disk of the computer system.

Student’s Signature: Date: october 2025

page1image9370432

LECTURER’S COMMMENTS/GRADE:

for office use only upon receive

Remark

DATE :  
 TIME :

RECEIVER’S NAME :

Table content

Introduction……………………………………………………………………..

Why I use dictions for book catlog……………………………………………………………

Why I use list for borrowing history………………………………………………………...

Why I use tuple for library configuration…………………………………………………..

How they work together…………………………………………………………………………..

Conclusion………………………………………………………………………………………………

Refrences……………………………………………………………………………………………….

**Introduction**

This document provides the design rationale for my Library Management System implementation, focusing on why specific Python data structures were selected for different components of the system. The choices were driven by performance requirements, data integrity needs, and real-world library operations.

The Advanced Library Management System is designed to handle complex library operations including catalog management, member services, borrowing transactions, and administrative reporting. By leveraging Python's built-in data structures strategically, the system achieves both efficiency and maintainability while providing a solid foundation for future enhancements.

**Why I Used Dictionary for Book Catalog**

I implemented the book catalog as a dictionary where each key is a Book ID and the value is a detailed book record.

1. Direct Access Efficiency - Library staff frequently need to access specific books by their unique identifiers during check-in/check-out operations.

2. Optimized for High-Frequency Operations - The most common operations (borrowing, returning, searching by ID) benefit from dictionary's O(1) lookup time.

3. Natural Key-Value Relationship - Each book naturally has a unique identifier that maps perfectly to its detailed information.

4. Space Optimization - While storing some overhead, the performance benefits outweigh the memory cost for the core catalog.

5. Update Simplicity - Updating book status (available/borrowed) or editing details is straightforward with direct key access.

Example: During peak hours, when multiple students are returning books, the system can instantly update each book's status without performance degradation.

**Why I Used List for Borrowing History**

I used a list to maintain complete borrowing history for analytical purposes.

1. Chronological Integrity - Borrowing records must be maintained in the order they occur for accurate reporting and analytics.

2. Append-Only Nature - Borrowing transactions are inherently sequential and new records are always added to the end.

3. Iteration Efficiency - Generating reports requires processing all transactions, which lists handle efficiently.

4. Memory-Linear Growth - The memory usage grows predictably with the number of transactions.

5. Simple Archive Management - Old records can be easily batched and archived while maintaining order.

Example: The librarian can generate monthly reports by iterating through the borrowing history list and filtering transactions by date.

**Why I Used Tuple for Library Configuration**

I used tuples for all system configuration data that should remain constant during operation.

The LIBRARY\_CONFIG tuple contains immutable settings: ('Max Books Per Member', 'Loan Period Days', 'Fine Per Day', 'Renewal Limit', 'Reservation Period').

1. Runtime Safety - Critical configuration parameters cannot be accidentally modified during program execution.

2. Intent Communication - Using tuples clearly indicates that these values represent system constants.

3. Memory Efficiency - Tuples consume less memory than dictionaries or classes for static configuration data.

4. Error Prevention - Any attempt to modify configuration during runtime raises an immediate error.

5. Consistency Enforcement - All parts of the system reference the same immutable configuration values.

Example: If the system needs to check if a member has exceeded their book limit, it references LIBRARY\_CONFIG[0] knowing this value cannot change unexpectedly.

**How They Work Together**

The integration of these data structures creates a robust system:

1. Book Borrowing Process:

a. Validate member eligibility using LIBRARY\_CONFIG limits

b. Access book details instantly from catalog dictionary

c. Append transaction record to borrowing history list

2. System Reporting:

a. Iterate through borrowing history list for transaction analysis

b. Access book details from catalog dictionary as needed

c. Reference LIBRARY\_CONFIG for policy enforcement validation

3. Inventory Management:

a. Use dictionary efficiency for frequent status updates

b. Maintain chronological audit trail in history list

c. Enforce business rules through configuration tuple

Alternatives I Considered

Why Not Use Database Instead of Dictionary for Catalog?

· For this academic project, simplicity and Python focus were priorities

· Dictionary provides adequate performance for small to medium libraries

· Avoids external dependencies and complex setup

Why Not Use Dictionary for Borrowing History?

· Dictionaries don't maintain insertion order (though OrderedDict could work)

· Historical reporting requires chronological sequence

· List more naturally represents sequential events

Why Not Use Class or Dictionary for Configuration?

· Classes add unnecessary complexity for simple configuration

· Dictionaries are mutable and could be accidentally modified

· Tuple perfectly represents fixed, constant values

**Conclusion**

My data structure selection was guided by specific operational requirements:

. Dictionary for book catalog to support high-frequency access operations

· List for borrowing history to maintain chronological transaction records

· Tuple for system configuration to ensure runtime integrity of critical parameters

This architecture provides an optimal balance of performance, data integrity, and maintainability for a library management context.

**References**

1. Python Software Foundation. (2025). Python 3.12 Documentation. Retrieved from https://docs.python.org/3/

2. Beazley, D. M. (2023). Python Distilled. Addison-Wesley Professional.

3. Matthes, E. (2023). Python Crash Course (3rd ed.). No Starch Press.

4. Programiz. (2025). Python Data Structures and Algorithms. Retrieved from https://www.programiz.com/

[File Content End]