Abstraction

**Activity A: Library management system**

Create a program that manages a small library of books. Users should be able to:

1. Add new books.
2. Search for a book by title.
3. Remove a book.
4. Display all books.

Task 1: Decomposition

Using the grid provided, break the main problem into smaller, manageable parts. Each box represents a task and one has been done as an example.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Book Storage – How will books be stored? (e.g., a list or database) | Add Book | Search Book | Remove Book | Display Books |

Task 2: Abstraction

What information about the books can be ignored and what information about the books is important and required to solve the problem?

|  |  |
| --- | --- |
| **Ignored/Removed** | **Important/Kept** |
| Book cover images  Publication date  Author biography  Fancy user interface | Title  Author |

Task 3: Solve each part systematically

The following code can be used to store the books.

library = [] # A simple list to store books

Write an algorithm using a subroutine for each one of the sub-tasks covered in Task 1.

|  |  |
| --- | --- |
| **Add book** | def add\_book(title, author):  book = {'title': title, 'author': author}  library.append(book) |
| **Search book** | def search\_book(title):  for book in library:  if book['title'] == title:  return book  return None |
| **Remove book** | def remove\_book(title):  for book in library:  if book['title'] == title:  library.remove(book)  return True  return False |
| **Display book** | def display\_books():  for book in library:  print(f"{book['title']} by {book['author']}") |

Task 4: Combine the solutions

Use the subroutines created to add **1984 by George Orwell** and display the results.

|  |
| --- |
| add\_book("1984", "George Orwell")  display\_books()  **Output:**  1984 by George Orwell |

\*Task 3 and 4 is synoptic and requires students to have an understanding on how to write algorithms that use subroutines and arrays. You may decide to provide partial solutions as a form of scaffolding.

Key Points in this activity:

* Decomposition: Each part of the library system was treated separately.
* Abstraction: Only the essential information (title and author) was considered.
* Systematic approach: Each function is logically designed, tested, and combined.

**Activity B: Calculator program**

A program for a simple calculator that can perform addition, subtraction, multiplication, and division.

Task 1: Abstraction

Decompose the program into **four** modular parts.

Division

Multiplication

Subtraction

Addition

Task 2: Structure program into modular parts

1. Write an algorithm that uses a subroutine called Division. It must take two values as parameters and either return the answer or returns an error if the second number is 0.

|  |
| --- |
| # Module 4: Division  def divide(a, b):  if b == 0:  return "Error: Division by zero"  return a / b |

1. Write an algorithm that will use a subroutine called Calculator which will:

* Take two numbers are parameters
* Allow the user to enter the arithmetic operator
* Perform the operation against the two numbers by calling the appropriate module (e.g. Addition, Subtraction etc..) and returning the result.

|  |
| --- |
| def calculator():  """Interface for the calculator program."""  print("Simple Calculator")  x = float(input("Enter first number: "))  y = float(input("Enter second number: "))  print("Select operation: +, -, \*, /")  op = input("Enter operation: ")    if op == "+":  print(add(x, y))  elif op == "-":  print(subtract(x, y))  elif op == "\*":  print(multiply(x, y))  elif op == "/":  print(divide(x, y))  else:  print("Invalid operation")  # Run the calculator  calculator() |