Data Structure and Algorithm

Lab Manual # 03

Lab Instructor: Engr. Shahryar Khan

Lab Title: **Building Proficiency in Array Operations**

Lab Overview

This lab focuses on mastering arrays and their applications in real-world problem-solving scenarios through Python. The manual covers 1-D, 2-D, and 3-D arrays implemented using classes and objects to simulate practical applications such as to-do lists, expense trackers, tic-tac-toe games, and 3D maze solvers. By the end of this lab, students will gain a deeper understanding of how arrays are used to solve real-life problems efficiently, and how they can be extended to implement features in applications like Facebook, Instagram, and Snapchat.

Lab Objectives

* Build a strong foundation in implementing and manipulating 1-D, 2-D, and 3-D arrays using Python's object-oriented programming approach.
* Learn to design solutions for real-world problems by leveraging array operations and object-oriented principles.
* Utilize arrays to develop real-life applications such as to-do list managers, library systems, and 3D maze solvers.
* Demonstrate using arrays in designing features for complex systems such as social media platforms and image processing tools.

Lab Requirements

* **Python Environment:** Ensure Python 3.10+ is installed. Use the official Python website for downloads (<https://python.org>).
* **VSCode Installation:** Download and install Visual Studio Code (<https://code.visualstudio.com>).
* **Documentation**: Use the official Python documentation for reference (<https://docs.python.org/3>).Reference Official Documentation: Use the Python official documentation (https://docs.python.org/3/) and sample programs to enhance learning.

**Note:** For the guided tasks type the code yourself.

**Guided Tasks**

**Task 1:** To-Do List Application

**Objective**: Manage tasks using a 1-D array of strings.

**Implementation**:

class ToDoList:

def \_\_init\_\_(self):

self.tasks = []

def add\_task(self, task):

self.tasks.append(task)

def remove\_task(self, task):

if task in self.tasks:

self.tasks.remove(task)

else:

print(f"Task '{task}' not found.")

def view\_tasks(self):

print("Tasks:")

for i, task in enumerate(self.tasks, 1):

print(f"{i}. {task}")

to\_do = ToDoList()

to\_do.add\_task("Buy groceries")

to\_do.add\_task("Complete homework")

to\_do.view\_tasks()

to\_do.remove\_task("Buy groceries")

to\_do.view\_tasks()

**Task 2:** Expense Tracker

**Objective:** Track daily expenses using a 1-D array.

**Implementation:**

class ExpenseTracker:

def \_\_init\_\_(self):

self.expenses = []

def add\_expense(self, amount):

self.expenses.append(amount)

def total\_expenses(self):

return sum(self.expenses)

def max\_expense(self):

return max(self.expenses) if self.expenses else 0

def min\_expense(self):

return min(self.expenses) if self.expenses else 0

expense\_tracker = ExpenseTracker()

expense\_tracker.add\_expense(20.5)

expense\_tracker.add\_expense(100.75)

print("Total Expenses:", expense\_tracker.total\_expenses())

print("Max Expense:", expense\_tracker.max\_expense())

print("Min Expense:", expense\_tracker.min\_expense())

**Task 3:** Student Grade Tracker

**Objective**: Store and analyze a student's grades using a 1-D array.

**Implementation**:

class GradeTracker:

def \_\_init\_\_(self):

self.grades = []

def add\_grade(self, grade):

self.grades.append(grade)

def average\_grade(self):

return sum(self.grades) / len(self.grades) if self.grades else 0

def highest\_grade(self):

return max(self.grades) if self.grades else 0

def lowest\_grade(self):

return min(self.grades) if self.grades else 0

grades = GradeTracker()

grades.add\_grade(85)

grades.add\_grade(90)

grades.add\_grade(78)

print("Average Grade:", grades.average\_grade())

print("Highest Grade:", grades.highest\_grade())

print("Lowest Grade:", grades.lowest\_grade())

**Task 4:** Library Management System

**Objective:** Manage book details using a 2-D array.

**Implementation**

class Library:

def \_\_init\_\_(self):

self.books = []

def add\_book(self, title, author, status):

self.books.append([title, author, status])

def search\_book(self, title):

for book in self.books:

if book[0] == title:

return book

return None

def display\_books(self):

for book in self.books:

print(book)

library = Library()

library.add\_book("Book1", "Author1", "Available")

library.add\_book("Book2", "Author2", "Issued")

library.display\_books()

**Task 5:** RGB Image Processing

**Objective:** Store and manipulate RGB values using a 3-D array.

**Implementation:**

class RGBImage:

def \_\_init\_\_(self, rows, cols):

self.image = [[[0, 0, 0] for \_ in range(cols)] for \_ in range(rows)]

def update\_pixel(self, row, col, rgb):

self.image[row][col] = rgb

def get\_pixel(self, row, col):

return self.image[row][col]

image = RGBImage(2, 2)

image.update\_pixel(0, 0, [255, 0, 0])

image.update\_pixel(0, 1, [0, 255, 0])

print("Pixel RGB Value:", image.get\_pixel(0, 1))

--------------------------------------- xxxxx ------- xxxx -------------------------------------------------

**Exercise Questions**

**Easy Problems (5 Questions)**

1. **To-Do List Enhancement**  
   Create a to-do list program that allows users to mark tasks as "completed" and filter only completed tasks to display.  
   **Hint**: Use a 1-D array to store tasks and a parallel array to store their completion status (True/False).
2. **Daily Expense Calculator**  
   Write a program to store daily expenses in an array and calculate the total expenses for the first seven days.  
   **Hint**: Use a for loop to sum up the first seven elements of the array.
3. **Student Grade Summary**  
   Develop a program to store grades of students for a single subject and display grades greater than or equal to the class average.  
   **Hint**: Calculate the average first, then use a loop to filter grades that meet the condition.
4. **Find the Maximum Element**  
   Create a program to find the maximum number in a list of positive integers entered by the user.  
   **Hint**: Use a max() function or iterate through the array with a for loop.
5. **Simple Library Search**  
   Write a program to store book names in a library and allow a user to search for a specific book by its name.  
   **Hint**: Use the in keyword to check if the book is in the array.

**Intermediate Problems (5 Questions)**

1. **Expense Breakdown by Category**  
   Develop a program to track expenses for different categories (food, travel, utilities, etc.) using a 2-D array. Calculate the total expenses for each category.  
   **Hint**: Use a nested list where each row corresponds to a category, and each column is an expense.
2. **Attendance Tracker**  
   Create a program to track attendance for 5 employees over 5 days using a 2-D array. Calculate the attendance percentage for each employee.  
   **Hint**: Use a loop to count the number of 1s in each row and divide by the total days.
3. **Matrix Addition**  
   Write a program to add two 2-D matrices (3x3) and display the resulting matrix.  
   **Hint**: Use nested loops to add corresponding elements from two matrices.
4. **Sort Grades**  
   Develop a program to store student grades for five students and sort them in descending order.  
   **Hint**: Use a sorting algorithm or Python's sorted() function with the reverse parameter set to True.
5. **2-D Tic-Tac-Toe Enhancements**  
   Extend the tic-tac-toe game to announce the winner (player "X" or "O") or declare it as a draw after all moves are completed.  
   **Hint**: Check rows, columns, and diagonals for identical values to determine the winner.

**Advanced Problems (5 Questions)**

1. **Facebook Notifications System**  
   Implement a notifications queue to manage and display sequential updates, allowing users to "clear all" or view the latest five notifications.  
   **Hint**: Use a 1-D array as a queue and maintain a size limit for the array.
2. **Instagram Image Filter**  
   Write a program to apply a grayscale filter on a 3-D array representing RGB pixel values of an image. Convert each pixel to grayscale using the formula:  
   Gray = (R + G + B) / 3.  
   **Hint**: Iterate through each pixel (row and column) and apply the formula to update the pixel values.
3. **Snapchat Streak Tracker**  
   Develop a program to store a 2-D array of streak counts between users over a week. Calculate the highest streak for each user and display the user with the longest streak.  
   **Hint**: Iterate through each row to find the maximum streak and use it to identify the user.
4. **Twitter Hashtag Tracker**  
   Create a program to count the occurrences of hashtags in a given list of tweets. Display the top three most-used hashtags.  
   **Hint**: Use a dictionary to store hashtag counts and sort the dictionary by values to find the top three.

All the best