Python Fundamentals Lab: Data Structures (Lists & Dictionaries)

Warm-Up: Basic Operations

1. **Lists:**

- Create a list of integers from 1 to 10. Write a script to:

- Print the first and last elements.

- Reverse the list.

- Find the sum of all elements.

2. **Dictionaries:**

- Create a dictionary where keys are the first 5 natural numbers and values are their squares. Write a script to:

- Print all keys and their respective values.

- Add a new key-value pair: `6: 36`.

- Check if a specific key (e.g., 4) exists in the dictionary.

Problem-Solving: Real-World Scenarios

1. **Student Grades:**

- Create a dictionary to store student names as keys and a list of their grades as values. Write a script to:

- Calculate the average grade for each student.

- Find the student with the highest average.

2. **Shopping Cart:**

- Implement a shopping cart using a dictionary where keys are product names and values are their prices. Write a script to:

- Add items to the cart.
- Calculate the total price of items in the cart.

3. **Employee Database:**
- Create a list of dictionaries where each dictionary represents an employee with attributes: `name`, `age`, and `department`. Write a script to:
- Filter employees by a specific department.
- Find the average age of employees.

Challenge Section

1. **Word Frequency Counter:**
- Given a string, count the frequency of each word using a dictionary.

2. **Inventory Management:**
- Create an inventory system for a store using a dictionary where keys are item names and values are dictionaries containing `price` and `quantity`. Write a script to:
- Update stock after a sale.
- Restock items.

3. **Nested Data Manipulation:**
- Create a nested dictionary to represent a library where keys are genres and values are lists of books. Write a script to:
- Add a new book to a genre.
- List all books of a specific genre.

4. **Anagram Checker:**
- Write a function to check if two given strings are anagrams of each other. Two strings are

anagrams if they use the same characters in the same frequency, regardless of order.

- Example:

  ```python
  Input: "listen", "silent"
  Output: True
  ```

5. **Two-Sum Problem:**

- Write a function that takes a list of integers and a target number and returns the indices of two numbers in the list that add up to the target.

- Example:

  ```python
  Input: nums = [2, 7, 11, 15], target = 9
  Output: [0, 1]
  ```