**Lab 2**

**09/07/2024**

**Brief Summary:**

The script addresses the following tasks and problems:

1. **Matrix Arithmetic Using NumPy**:
   * **Addition**: Adds two matrices x and y.
   * **Subtraction**: Subtracts matrix y from matrix x.
   * **Multiplication**: Element-wise multiplication of matrices x and y.
   * **Division**: Element-wise division of matrix x by matrix y.

Each operation is enclosed in a try-except block to handle potential errors.

1. **Linear Search Function**:
   * Searches for a target element in a list using a linear search algorithm.
   * Returns the index if the element is found, otherwise returns -1.
2. **Binary Search Function**:
   * Searches for a target element in a sorted list using a binary search algorithm.
   * Returns the index if the element is found, otherwise returns -1.
3. **List and Dictionary Operations**:
   * Creates and prints a list my\_list.
   * Creates and prints values from a dictionary student.

**Problems and Considerations**

* **Matrix Operations**: Ensure matrices are compatible for element-wise operations.
* **Search Functions**: Linear search works on any list, while binary search requires the list to be sorted.
* **Error Handling**: Properly handle exceptions in matrix operations to avoid runtime errors.
* **Dictionary Access**: Accessing dictionary values safely to avoid key errors.

**Code:**

import numpy as np

def linear\_search(*arr*, *target*):

    for index in range(len(*arr*)):

        if *arr*[index] == *target*:

            return index

    return -1

def binary\_search(*arr*, *target*):

    left, right = 0, len(*arr*) - 1

    while left <= right:

        mid = (left + right) // 2

        if *arr*[mid] == *target*:

            return mid

        elif *arr*[mid] < *target*:

            left = mid + 1

        else:

            right = mid - 1

    return -1

# Create two NumPy arrays

x = np.array([[1, 2], [4, 5]])

y = np.array([[7, 8], [9, 10]])

# Matrix of Addition

try:

    addition\_result = np.add(x, y)

    print("Addition Result:\n", addition\_result)

except Exception as e:

    print("Error in addition:", e)

# Matrix of Subtraction

try:

    subtraction\_result = np.subtract(x, y)

    print("Subtraction Result:\n", subtraction\_result)

except Exception as e:

    print("Error in subtraction:", e)

# Matrix of Multiplication

try:

    multiplication\_result = np.multiply(x, y)

    print("Multiplication Result:\n", multiplication\_result)

except Exception as e:

    print("Error in multiplication:", e)

# Matrix of Division

try:

    division\_result = np.divide(x, y)

    print("Division Result:\n", division\_result)

except Exception as e:

    print("Error in division:", e)

# Create List

my\_list = [1, 2, 3, 4, 5]

print("Original List:", my\_list)

# Creating a dictionary

student = {"name": "John Doe", "age": 20, "major": "Computer Science"}

print(student["name"])

print(student["age"])

print(student["major"])

target\_element = 5

# Linear Search

result\_linear\_index = linear\_search(my\_list, target\_element)

if result\_linear\_index != -1:

    print(f"Element {target\_element} found at index {result\_linear\_index}.")

else:

    print(f"Element {target\_element} not found in the list.")

# Binary Search

result\_binary\_index = binary\_search(my\_list, target\_element)

if result\_binary\_index != -1:

    print(f"Element {target\_element} found at index {result\_binary\_index}.")

else:

    print(f"Element {target\_element} not found in the list.")

**Output:**

