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.

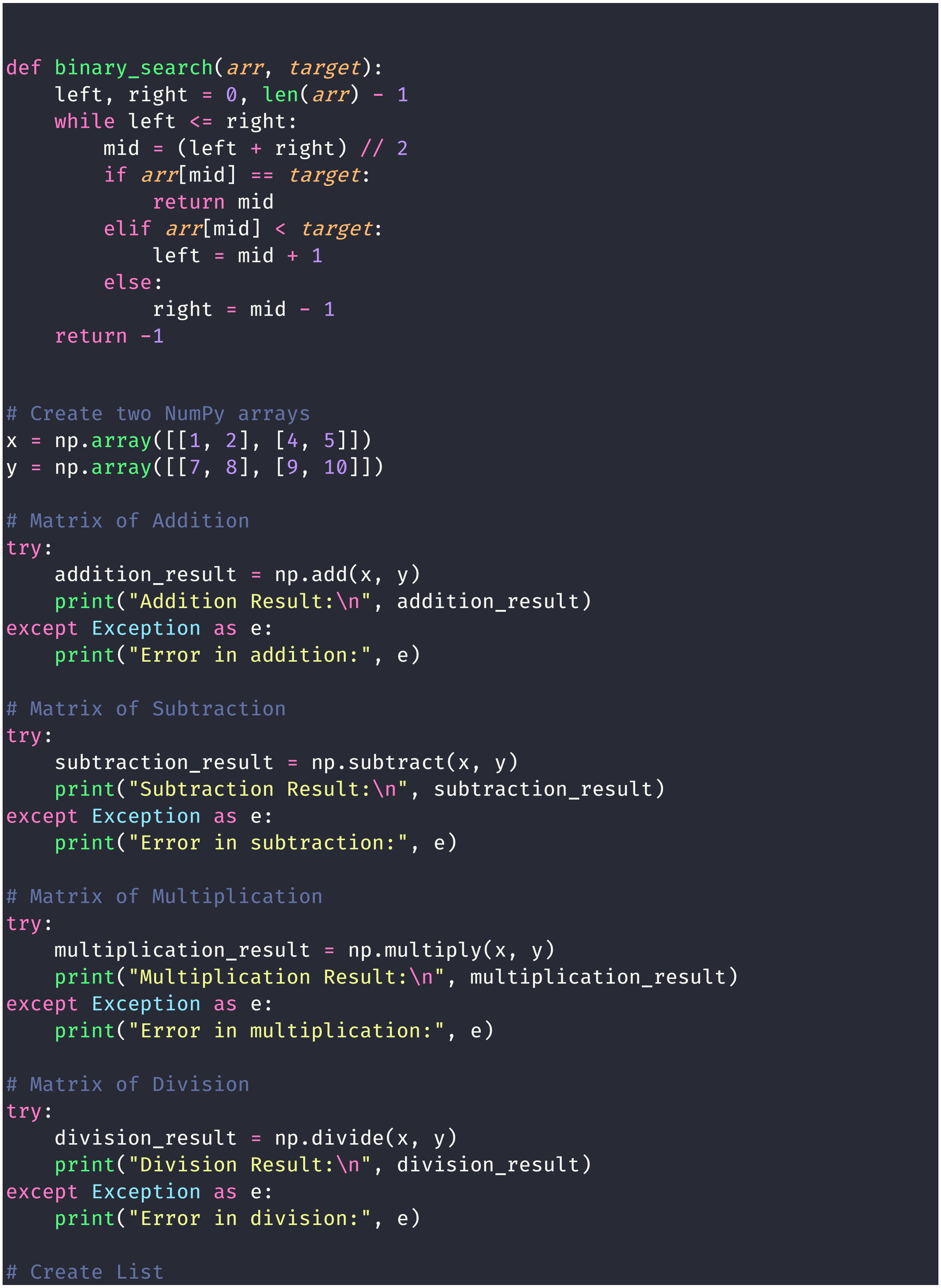
2. **Linear Search Function**:
   - Searches for a target element in a list using a linear search algorithm.
   - Returns the index if the element is found, otherwise returns `-1`.
3. **Binary Search Function**:
   - Searches for a target element in a sorted list using a binary search algorithm.
   -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

Problems and Considerations

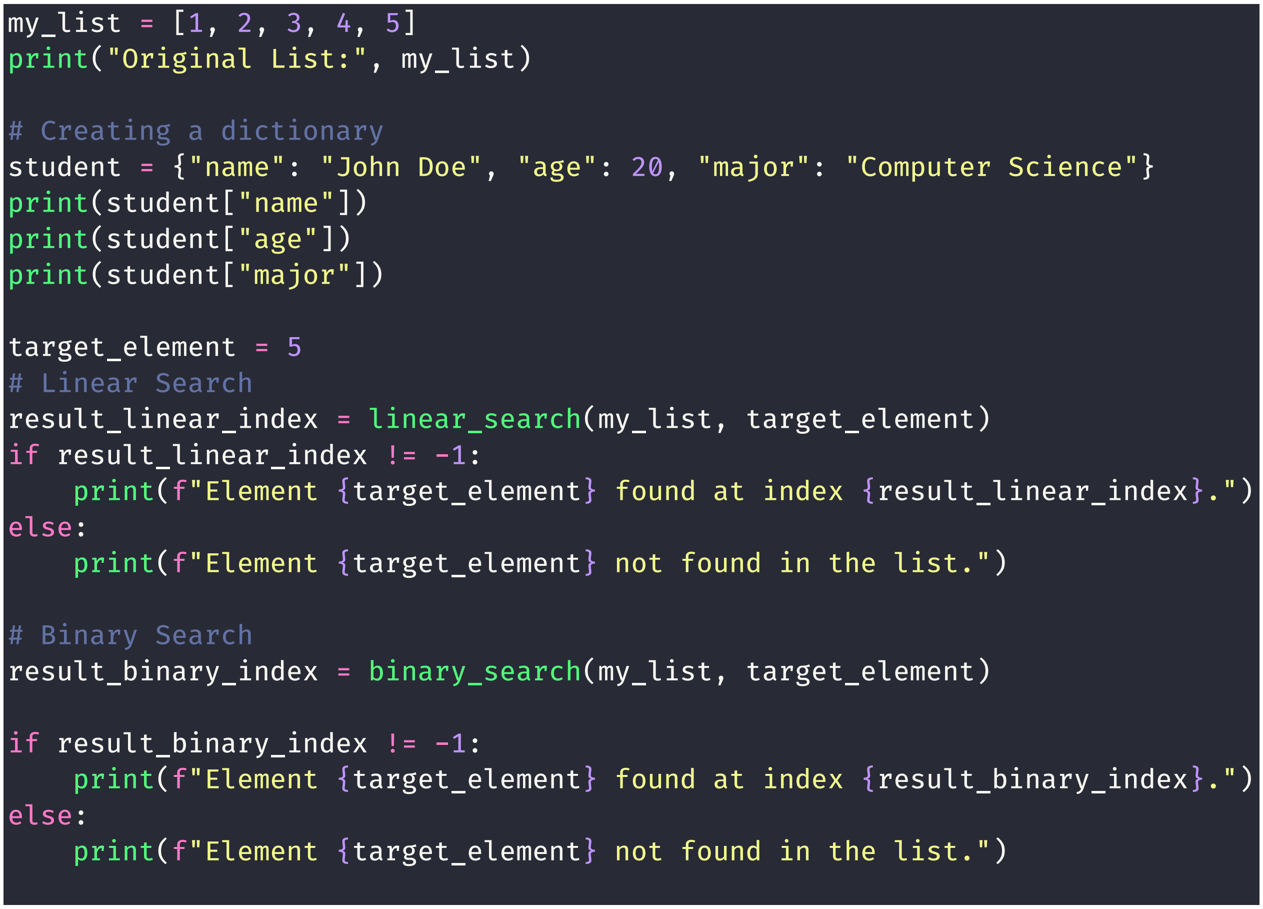
- **Matrix Operations**: Ensure matrices are compatible for element-wise operations.
- **Search Functions**: Linear search works on any list, while binary search requires the list to be sorted.
- **Error Handling**: Properly handle exceptions in matrix operations to avoid runtime errors.
- **Dictionary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

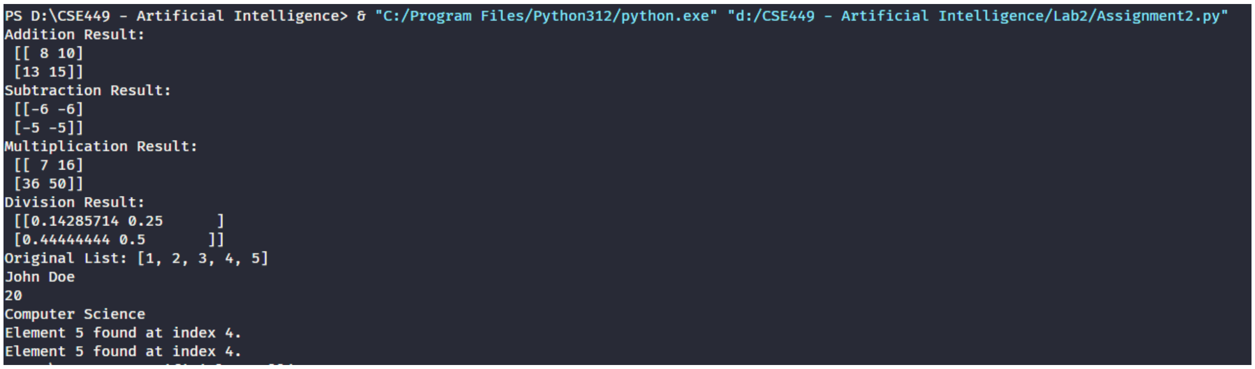


```
def binary_search(arr, target):
    left, right = 0, len(arr) - 1
    while left <= right:
        mid = (left + right) // 2
        if arr[mid] == target:
            return mid
        elif arr[mid] < target:
            left = mid + 1
        else:
            right = mid - 1
    return -1


# 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
```



```python
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:



```
PS D:\CSE449 - Artificial Intelligence> & "C:/Program Files/Python312/python.exe" "d:/CSE449 - Artificial Intelligence/Lab2/Assignment2.py"
Addition Result:
 [[ 8 10]
 [13 15]]
Subtraction Result:
 [[-6 -6]
 [-5 -5]]
Multiplication Result:
 [[ 7 16]
 [36 50]]
Division Result:
 [[0.14285714 0.25      ]
 [0.44444444 0.5       ]]
Original List: [1, 2, 3, 4, 5]
John Doe
20
Computer Science
Element 5 found at index 4.
Element 5 found at index 4.
```