Department of Social Statistics

Faculty of Humanities and Social Sciences

University of Sri Jayewardenepura

MIT/ITE 1213 – Fundamentals of Programming

Tutorial – 07

Arrays and NumPy

Single-Dimensional Arrays

Python has several ways to work with collections of data. For numerical work, arrays are efficient compared to lists. Python provides the **"array"** module for simple 1D arrays.

```
import array

# Create an integer array
numbers = array.array('i', [1, 2, 3, 4, 5])
print("Initial array:", numbers)

# Access an element
print("Third element:", numbers[2])  # 3

# Update an element
numbers[2] = 99
print("After updating 3rd element:", numbers)

# Append a new element
numbers.append(6)
print("After appending 6:", numbers)

# Remove an element
numbers.remove(1)
print("After removing 1:", numbers)
```

Iterating Through Arrays

```
for num in numbers:
    print(num)
```

Multidimensional Arrays

For multi-dimensional arrays, Python's built-in array isn't enough. Instead, we use **NumPy**, which is designed for numerical computing.

NumPy Basics

```
import numpy as np

# Create a 2D array (matrix)
matrix = np.array([[1, 2, 3],
                   [4, 5, 6],
                   [7, 8, 9]])

print(matrix)
print("Shape:", matrix.shape)  # (3, 3)
```

Accessing Elements

```
print("Element at row 0, col 1:", matrix[0][1])   # 2

print("Element at row 2, col 2:", matrix[2, 2])   # 9
```

Slicing

```
print("Submatrix (first 2 rows, last 2 cols):")
print(matrix[0:2, 1:3])
```

Matrix Operations

```
a = np.array([[1, 2],
           [3, 4]])

b = np.array([[5, 6],
           [7, 8]])

print("Addition:\n", a + b)
print("Element-wise Multiplication:\n", a * b)
print("Matrix Product:\n", np.dot(a, b))  # Or a @ b
```

Exercises

1. Create an integer array with numbers from 1 to 10. Remove all even numbers.

2. Create an array [5, 10, 15, 20, 25]. Insert 100 at index 2.

3. Write a program to check if a given number exists in the array.

4. Given an array [2, 4, 6, 8, 10], compute the maximum and minimum manually (without using max() or min()).
5. Create a 4×4 NumPy array filled with zeros, and set the diagonal elements to 1.

6. Create a 3×3 matrix, and calculate the sum of each row and sum of each column separately.

7. Create two 3×3 arrays and check if they are equal element-wise.
8. Write a function that takes a list [2, 3, 5, 7] and returns a new NumPy array with each element cubed.