Python&Math Initiative Project

(PyMath Project)

WP2 - ALGORITHMS AND MATHEMATICS

Session: Day 2, Session 4: Nested Loops, Arrays, and LTT Problems

Presenter: Prof. Dr. Turgay Tugay BİLGİN | Bursa Technical University

Date: 17.11.2025

Contents

CONTENTS

01

Introduction

Agenda for Session 4

1. **Review:** Quick recap of Session 3 (Basic 'For' and 'While' loops).
2. **Concept 1:** Nested Loops - What happens when we put a loop inside another loop?
3. **Simple Example:** The Multiplication Table.
4. **Concept 2:** Data Structures (Arrays) - Why we need variables that can hold lists of data.
5. **LTT Project Example:** 'Algorithm that finds intersection in sets'.
6. **Practical Skill:** How to Export your flowchart.

02

Review & Objectives

Review of Session 3

Loops: We learned that loops (While, Do-While, For) are used to repeat actions.

'For' Loop: This is a 'Counter-Controlled' loop, perfect for when we know the number of repetitions (e.g., count from 1 to N).

Accumulator Pattern: We used a 'total' variable (initialized to 0) before the loop to 'accumulate' or 'sum' values inside the loop.

LTT Example: We successfully solved the 'Sum of Arithmetic Numbers' problem.

Session 4: Objectives

1. Explain the logic of a **Nested Loop**.
2. Understand what an **Array** (Data Structure) is.
3. Use a 'For' loop to **Process** and **Print** Array elements.
4. Solve the LTT goal: 'Algorithm for set intersection'.
5. Export your flowchart as a PNG or PDF.

03

Nested Loops

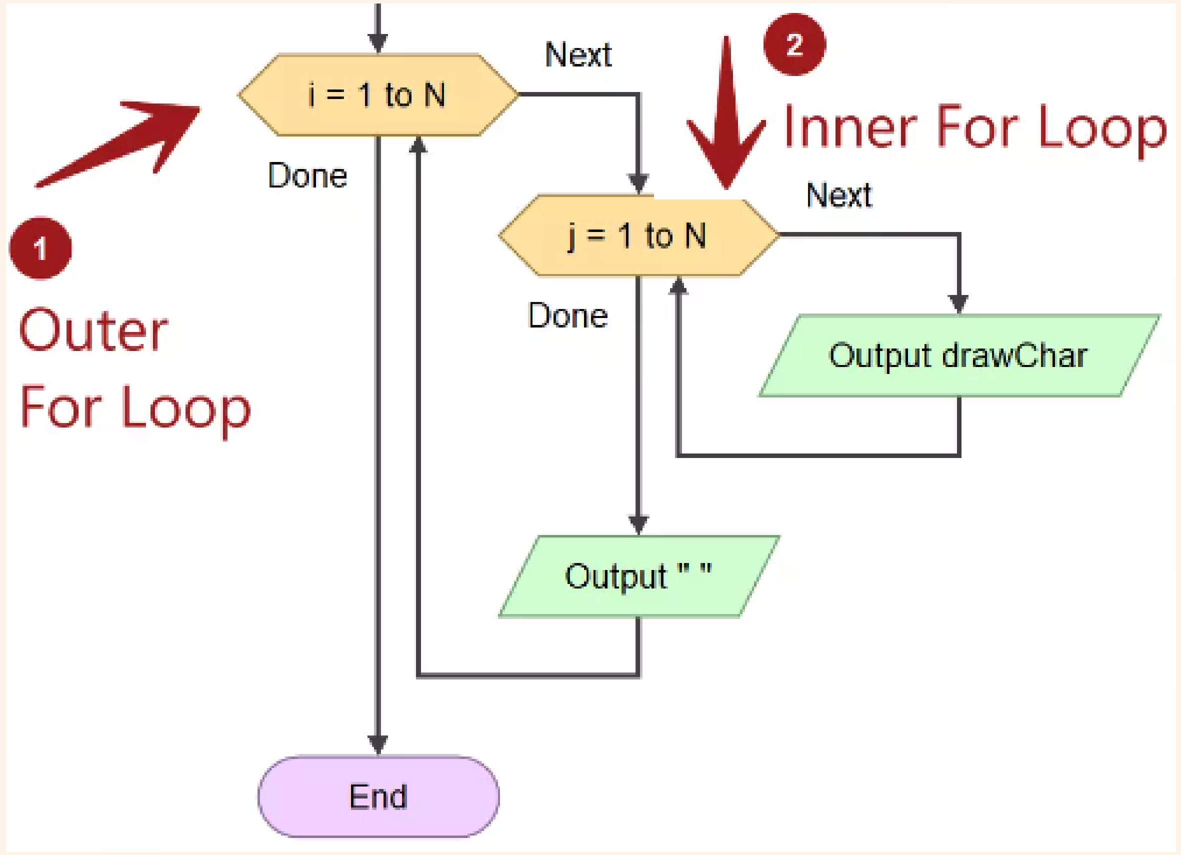
Concept 1: Nested Loops

A Nested Loop is simply a loop that is placed inside the body of another loop.

Outer Loop: The 'main' loop.

Inner Loop: The 'nested' loop.

This structure is powerful for working with two dimensions, like matrices, tables, or comparing two lists.



The Logic of Nested Loops

For every one (1) iteration of the **Outer Loop**, the Inner Loop runs completely.

Outer (i)		Inner (j)
i = 1	→	j = 1, 2, 3
i = 2	→	j = 1, 2, 3

i=1: j=1, j=2, j=3

i=2: j=1, j=2, j=3

Simple Example: The Multiplication Table

A perfect problem for nested loops!

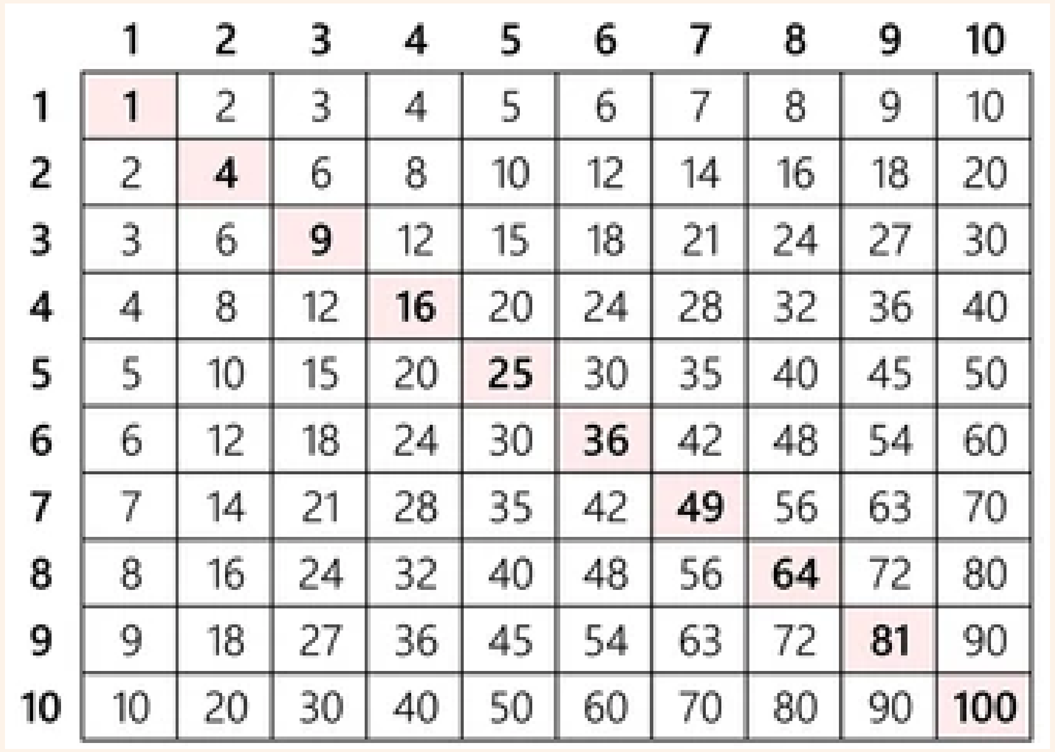
Outer Loop (i)

Manages the first number (from 1 to 10).

Inner Loop (j)

Manages the second number (from 1 to 10).

Inside the inner loop, we calculate and print `i * j`.



	1	2	3	4	5	6	7	8	9	10
1	**1**	2	3	4	5	6	7	8	9	10
2	2	**4**	6	8	10	12	14	16	18	20
3	3	6	**9**	12	15	18	21	24	27	30
4	4	8	12	**16**	20	24	28	32	36	40
5	5	10	15	20	**25**	30	35	40	45	50
6	6	12	18	24	30	**36**	42	48	54	60
7	7	14	21	28	35	42	**49**	56	63	70
8	8	16	24	32	40	48	56	**64**	72	80
9	9	18	27	36	45	54	63	72	**81**	90
10	10	20	30	40	50	60	70	80	90	**100**

Multiplication Table:

Textual Algorithm

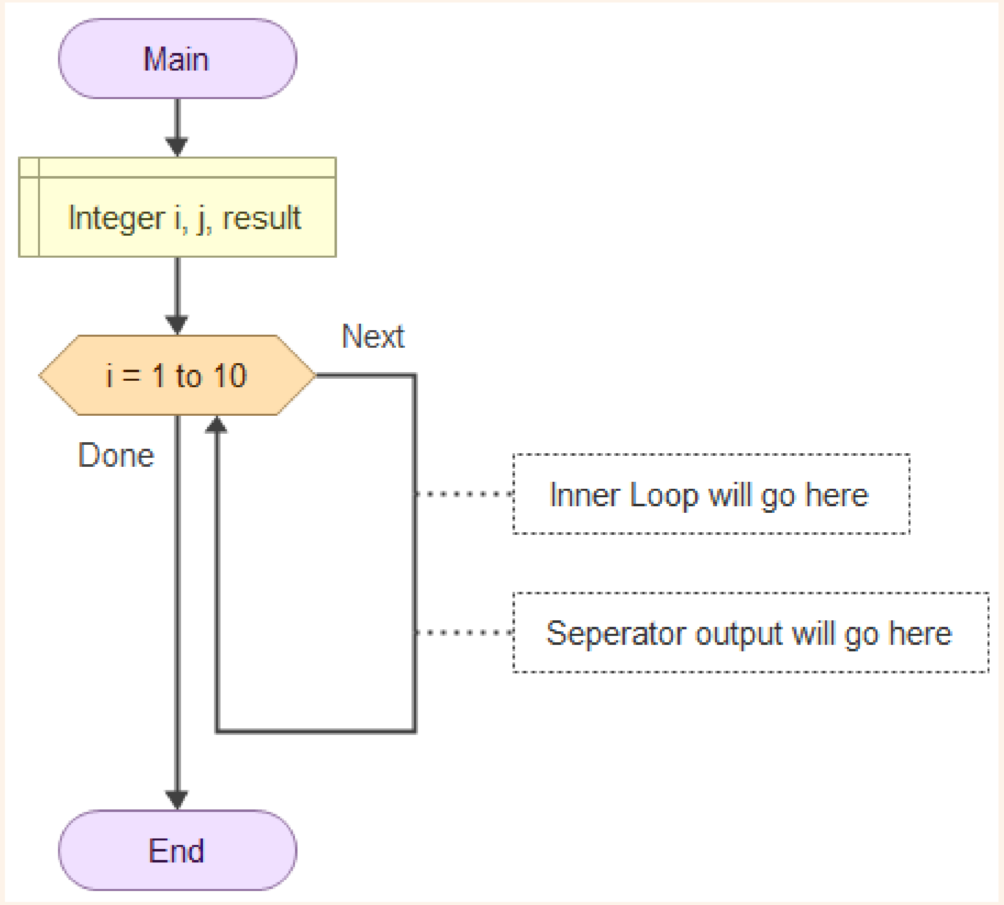
```
1  Start.
2  Declare Integers 'i', 'j', 'result'.
3  Outer FOR Loop ('i' = 1 to 10).
     4  Inner FOR Loop ('j' = 1 to 10).
          5. Process: 'result = i * j'.
          6. Output: i & ' x ' & j & ' = ' & result.
10 Stop.
```

Symbolic Math

$$T = \left(\prod_{i=1}^{10} \prod_{j=1}^{10} (i \cdot j) \right)$$

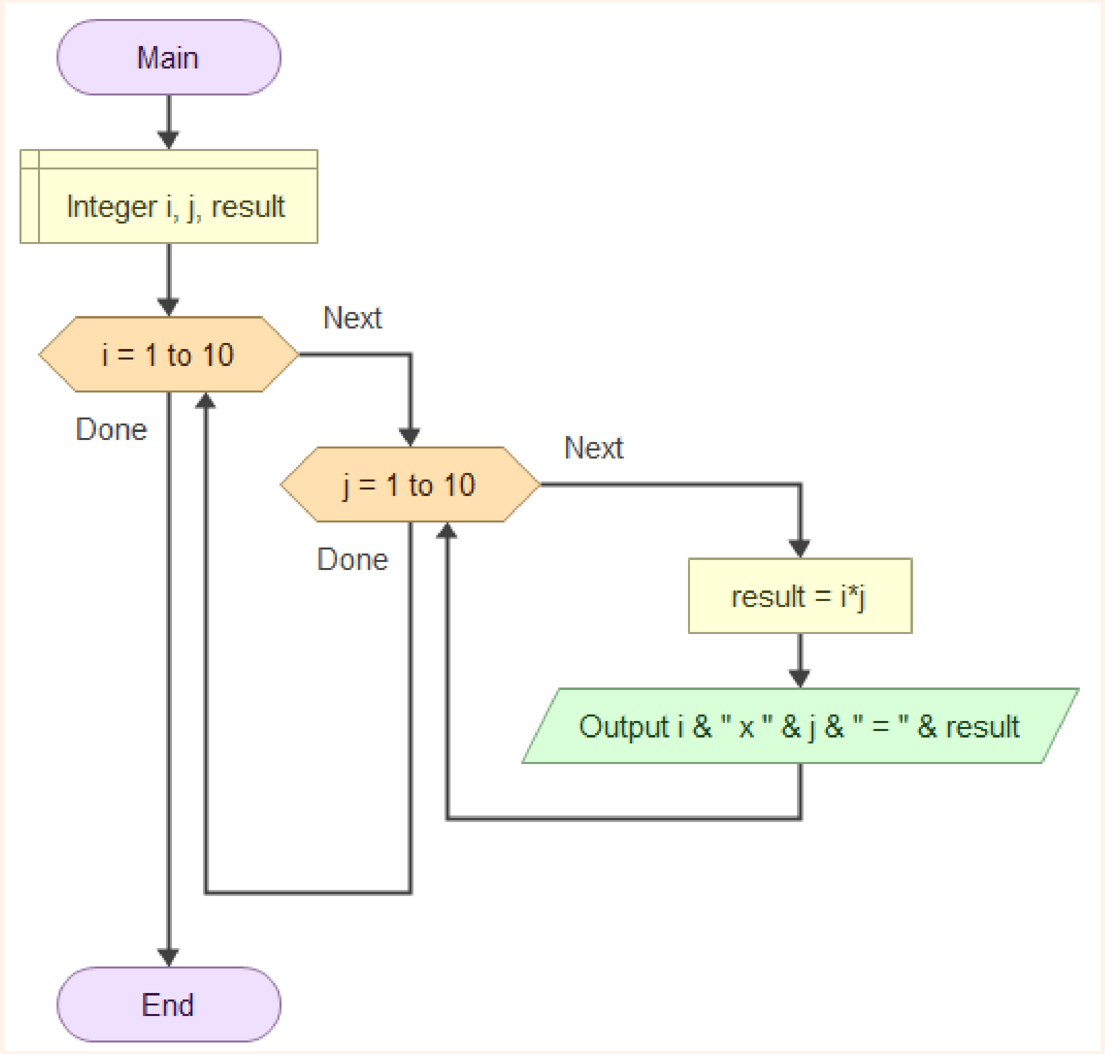
Multiplication Table: Flowgorithm (Outer Loop)

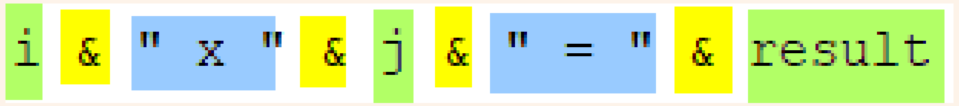
```
MAIN
Declare: Integer i, j, result
FOR i = 1 to 10, Step 1
        (The Inner Loop will go here)
        (The separator Output will go here)
    END
```



Multiplication Table: Completed Flowchart

```
MAIN
Declare: Integer i, j, result
FOR i = 1 to 10, Step 1
    FOR j = 1 to 10, Step 1
    Process: result = i * j
    Output: i & " x " & j & " = " & result
    END (Inner Loop)
Output: "----------------"
END (Outer Loop)
END
```





In Output greens **variable**, blues **string** and “**&**” character merges variables and strings

Multiplication Table: Trace

Outer Loop (i = 1)

j=1: 1x1=1

j=2: 1x2=2

...

j=10: 1x10=10

Output: '--------------'

Outer Loop (i = 2)

j=1: 2x1=2

j=2: 2x2=4

...

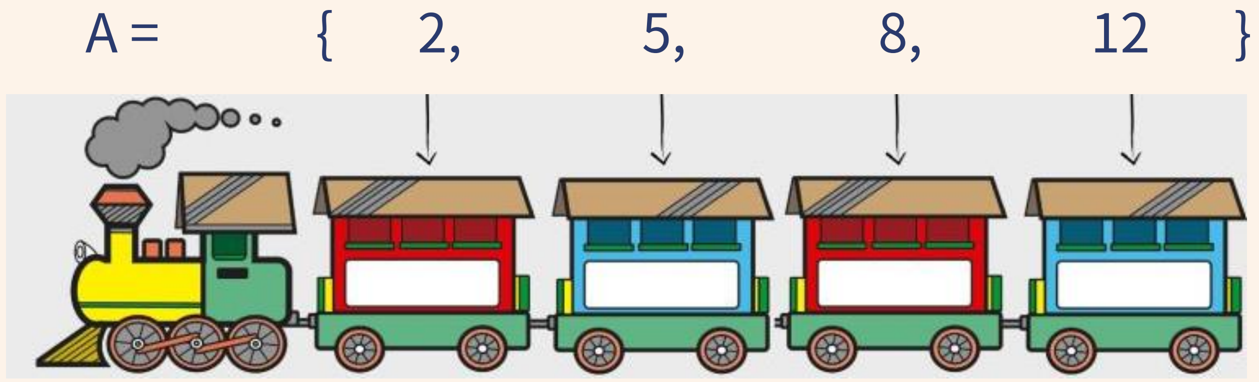
j=10: 2x10=20

Output: '--------------'

This continues until i = 10 and j = 10.

04

Arrays



Concept 2: Data Structures

Why We Need More Than Simple Variables

So far, variables ('number', 'total', 'i') can only hold one value at a time.

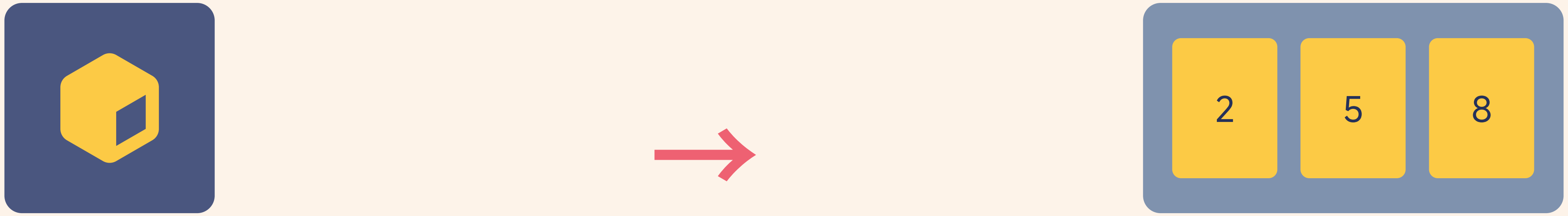
Problem:
How to store a **list** of numbers? Like a mathematical set:

$$A = \{2, 5, 8, 12\}$$

We need a **Data Structure**.

What is an Array?

The simplest data structure.



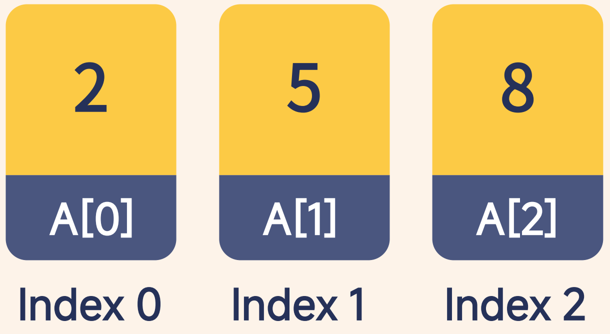
Simple Variable
(e.g., Integer)

Array
(Collection of Integers)

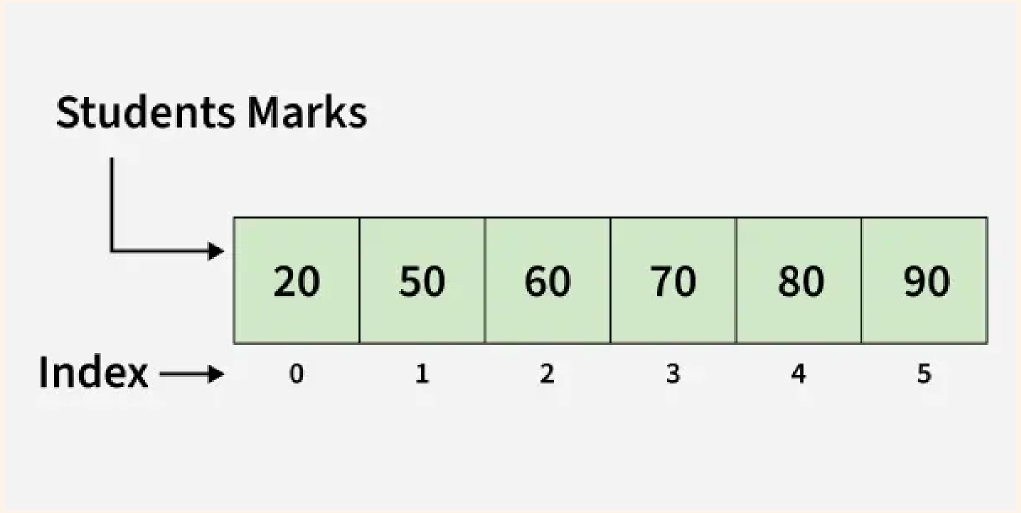
An **Array** is a collection of items (same type) stored under a **single variable name**.

The Most Important Concept: Index

How to Access Items in an Array?



IMPORTANT: Index starts at 0 (zero-based indexing).



Array Size

The total number of elements.

5

VS

Maximum Index

The last 'address'.

4

Max Index = Size - 1

How to Create (Declare) an Array

Use the same **Declare** symbol as before, but check the "Array?" box.

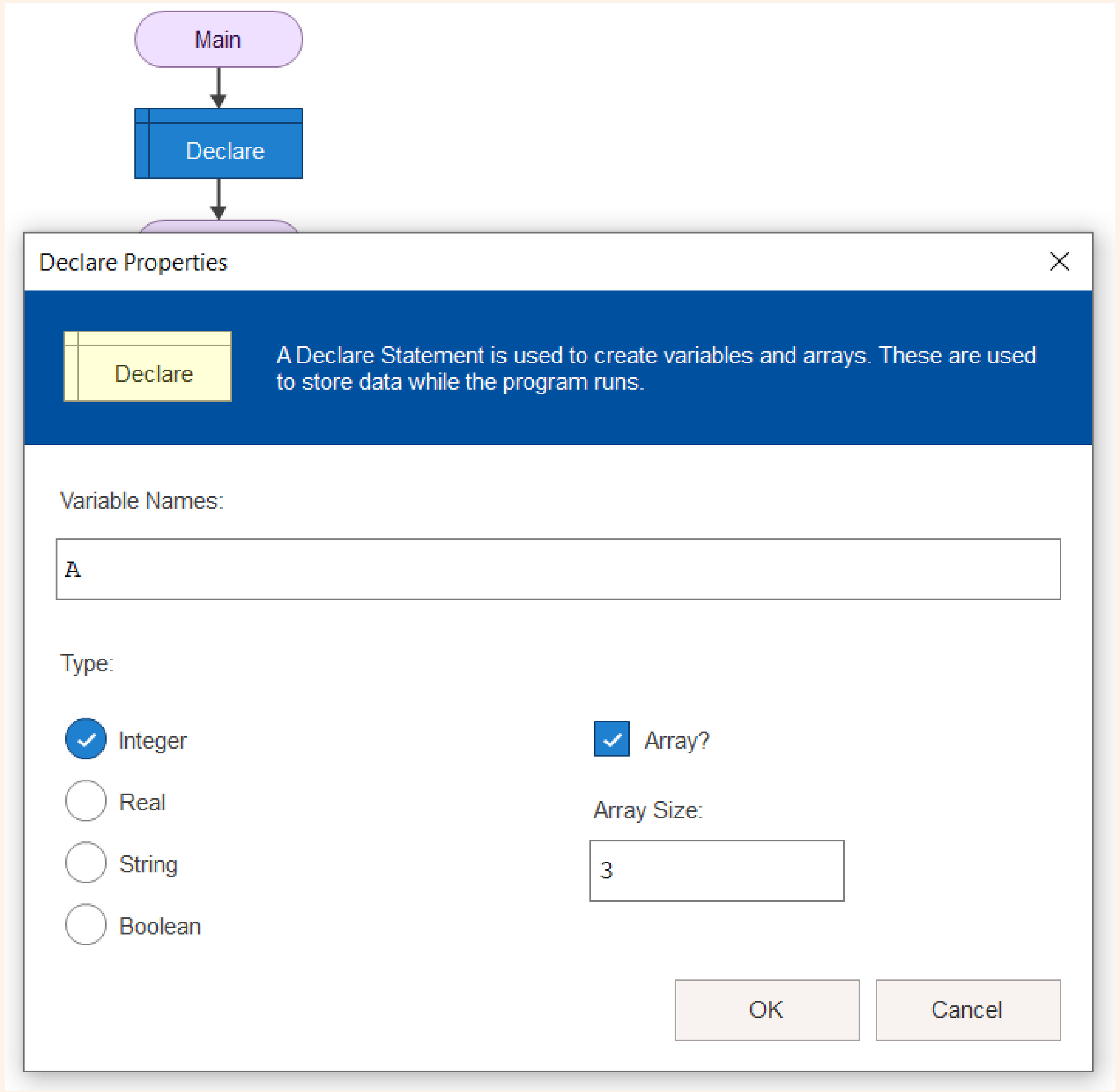
Example: Integer Array 'A' with 3 elements

Name: A

Type: Integer

Array?: (Checked)

Size: 3



Declare Properties

A Declare Statement is used to create variables and arrays. These are used to store data while the program runs.

Variable Names: A

Type:

[✓ Integer] Real String Boolean

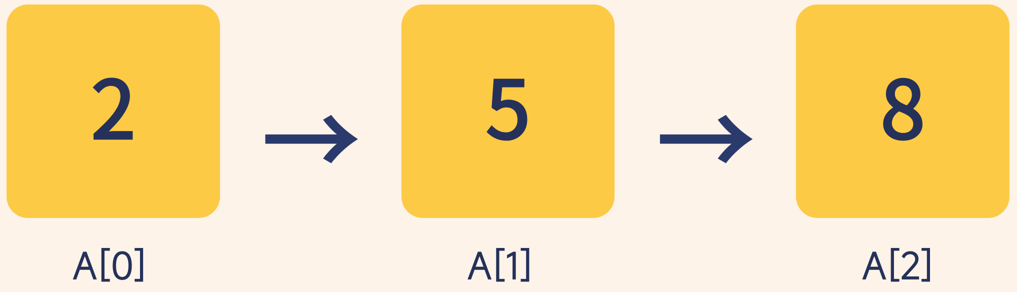
[✓ Array?]

Array Size: 3

How to Initialize Array Elements (Method 1)

Manual Method:

Use the **Process (Assign)** symbol for each index.

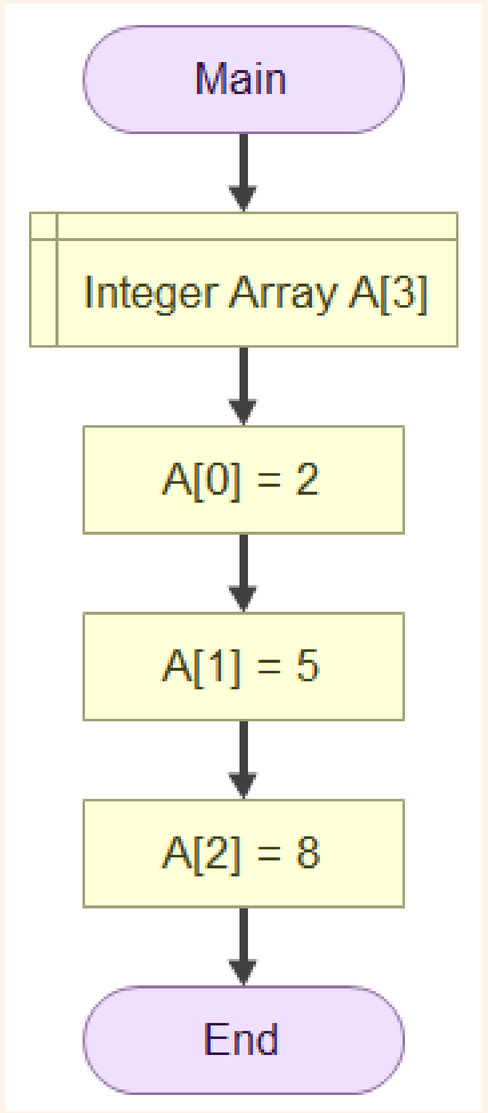


Process: A[0] = 2

Process: A[1] = 5

Process: A[2] = 8

* Okay for 3 numbers, but not efficient for 100.

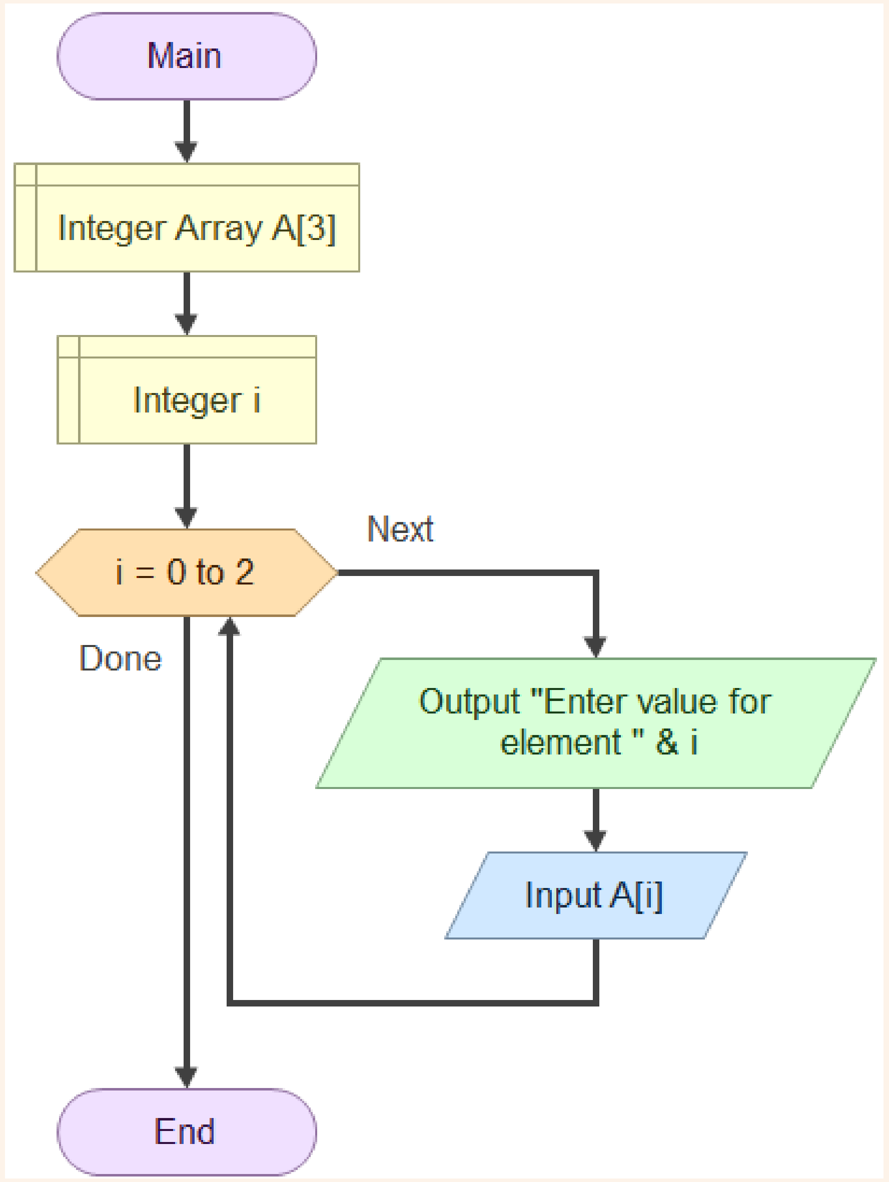


How to Initialize (Method 2)

Loop Method

A much better way: Use a **'For' loop** to ask the user to fill the array.

```
FOR i = 0 to 2, Step 1
Output: "Enter a value for element " & i
Input: A[i]
END
```

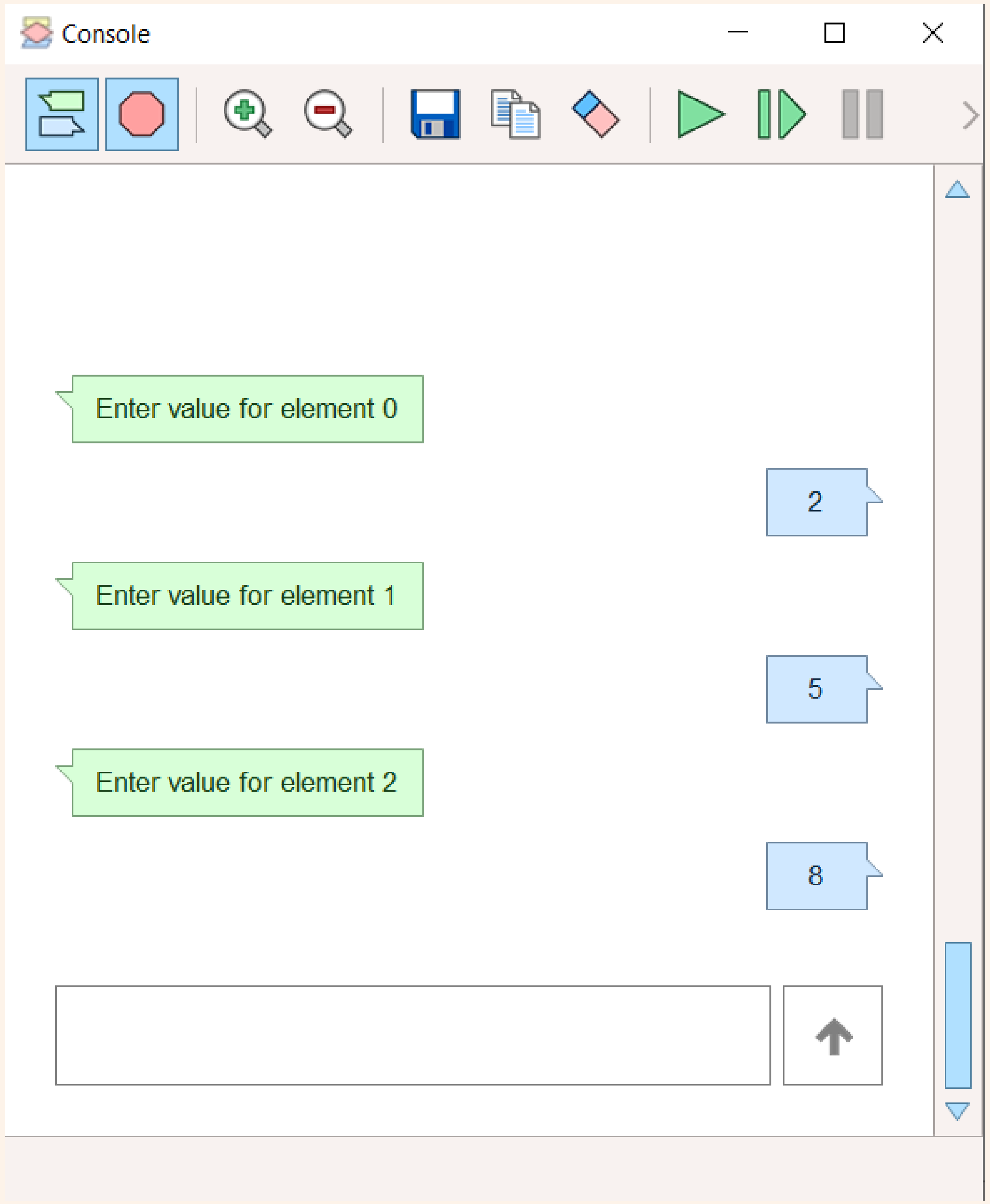


How to Initialize (Method 2)

Loop Method

A much better way: Use a **'For' loop** to ask the user to fill the array.

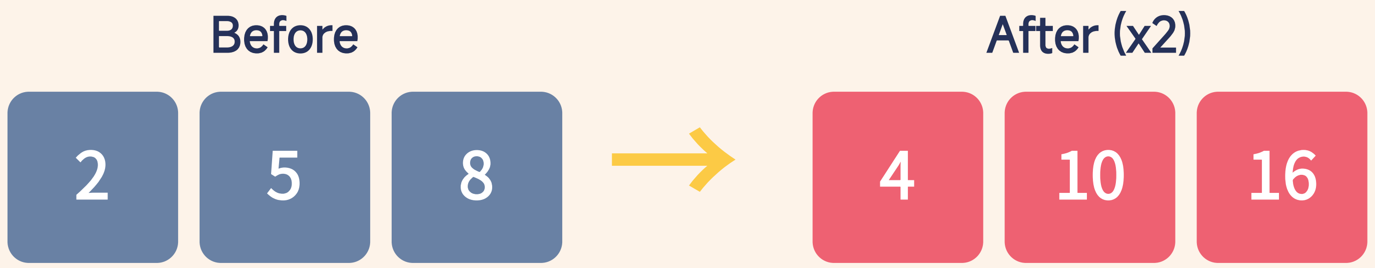
```
FOR i = 0 to 2, Step 1
Output: "Enter a value for element " & i
Input: A[i]
END
```



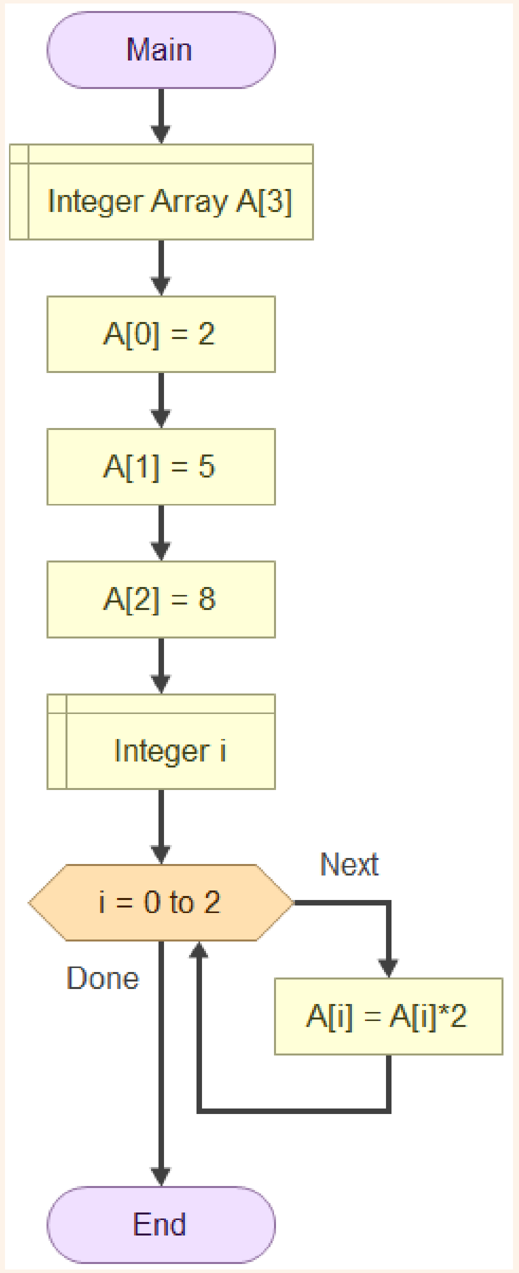
```
Enter value for element 0
2
Enter value for element 1
5
Enter value for element 2
8
```

How to Process Array Elements

"Processing" means doing something with every element. Use a **For Loop**!



FOR i = 0 to 2, Step 1 Process: A[i] = A[i] * 2



How to Print Array Elements

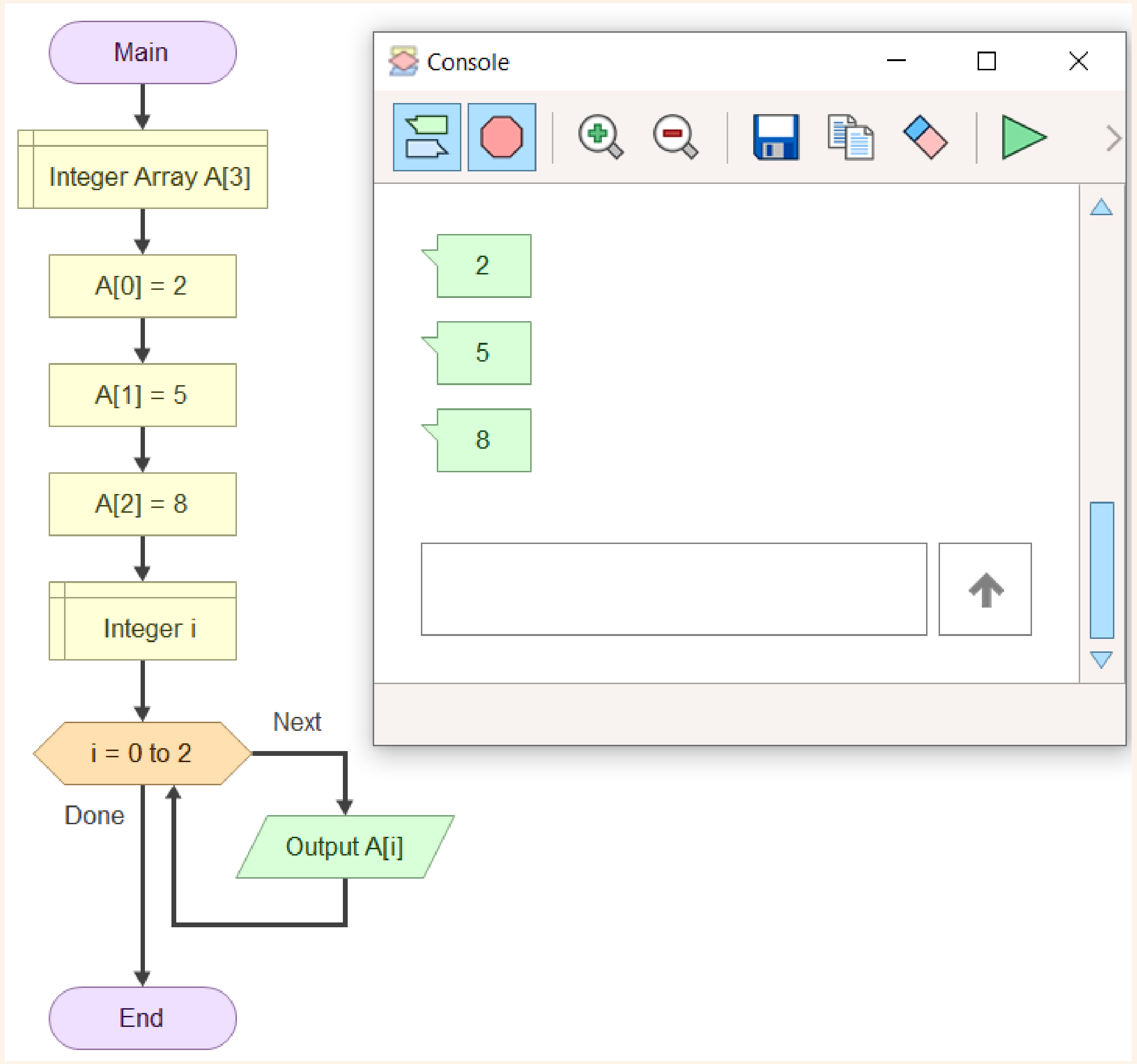
We cannot just use 'Output: A'. This will not work.

We must use a **For Loop** to print every element, one by one.

```
MAIN
Declare: Integer Array A[3]
Declare: Integer i

Process: A[0] = 2
Process: A[1] = 5
Process: A[2] = 8

FOR i = 0 to 2, Step 1
Output: A[i]
END
```



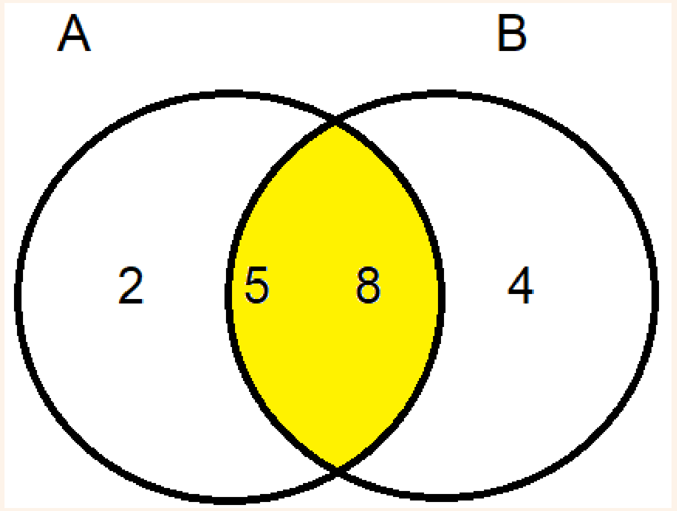
05

LTT Project Example

LTT Project Example

Intersection of Sets

Find the numbers that exist in **both** sets.



Set A

{2, 5, 8}

Set B

{8, 4, 5}

Intersection (Output)

5, 8

LTT Example: The Logic

How do we compare two sets (Arrays)?

We must compare every element from Set A against every element from Set B.

1. Arrays

To store Set A and Set B.

2. Nested Loops

Outer for A, Inner for B.

3. Decision (If)

To check if A[i] == B[j].

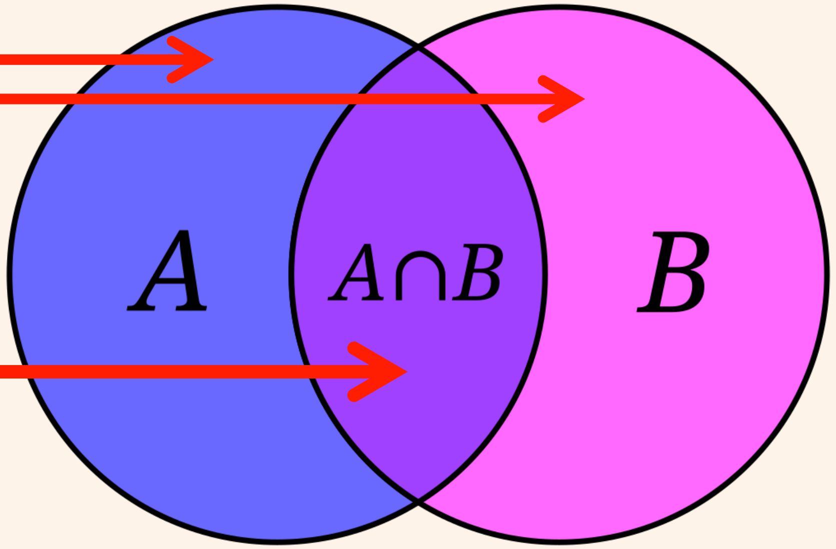
LTT Example: Logic Trace

A = {2, 5, 8}   B = {8, 4, 5}

i=0 (A[0]=2) vs j=0,1,2 → No match
i=1 (A[1]=5) vs j=2 (B[j]=5) → Match! Output: 5
i=2 (A[2]=8) vs j=0 (B[j]=8) → Match! Output: 8

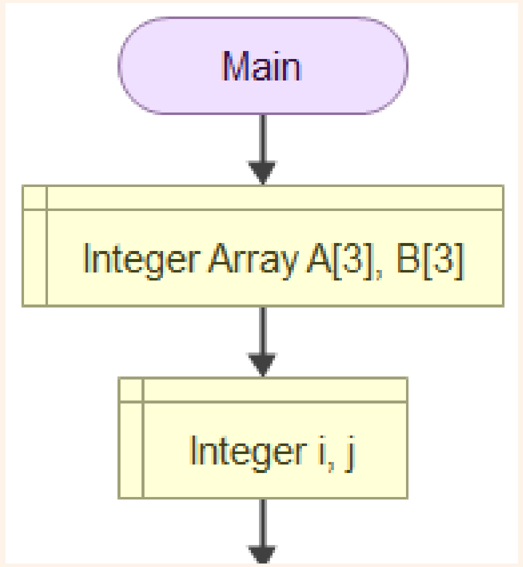
LTT Example: Textual Algorithm

```
1   Start.
2   Declare Integer Arrays A[3], B[3].
3   Declare Integer 'i', 'j'.
4-5 Initialize Arrays A and B.
6   Outer FOR Loop ('i' = 0 to 2).
      7 Inner FOR Loop ('j' = 0 to 2).
          8 IF (A[i] == B[j])
          9 Output: A[i]
12  Stop.
```



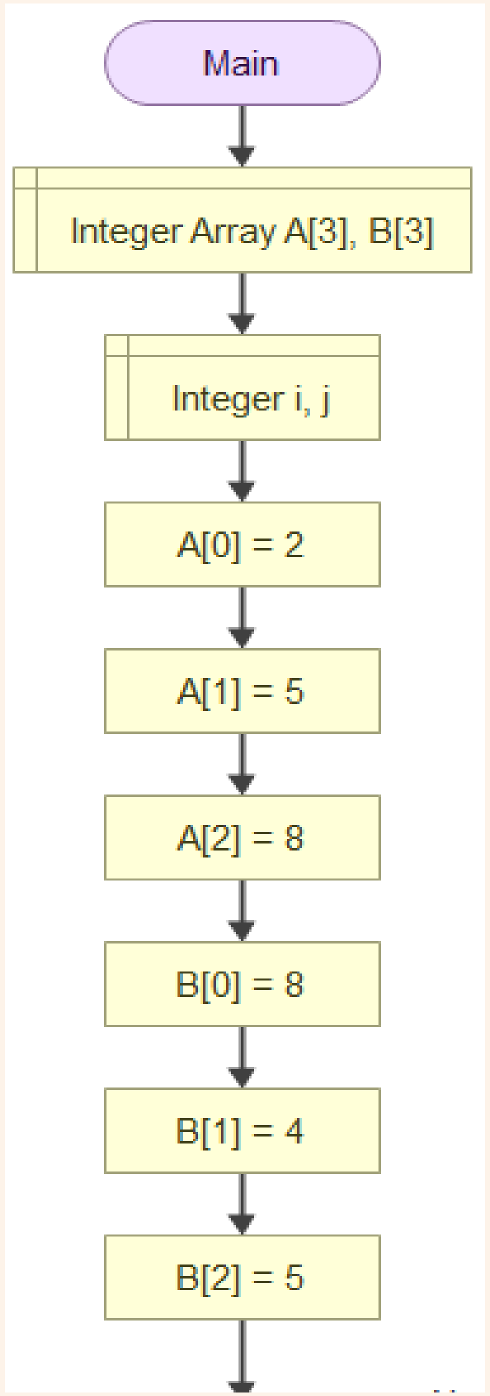
LTT Example: Flowgorithm (Step 1)

```
MAIN
Declare: Integer Array A[3]
Declare: Integer Array B[3]
Declare: Integer i
Declare: Integer j
```



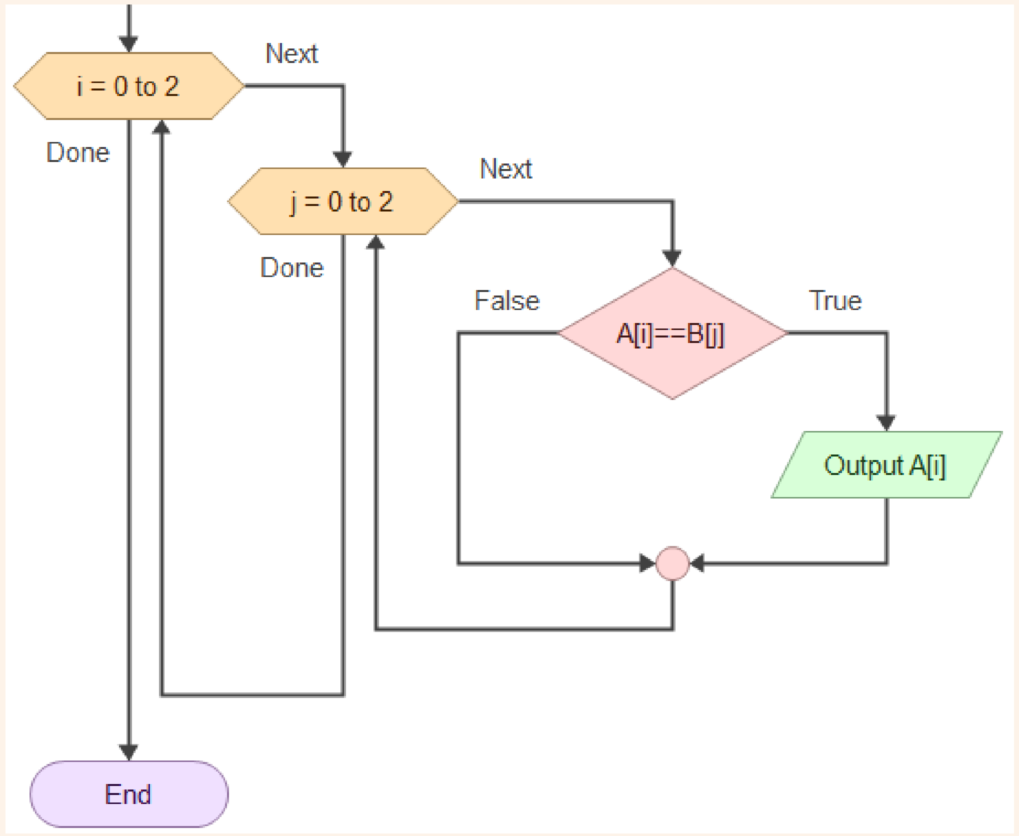
LTT Example: Flowgorithm (Step 2)

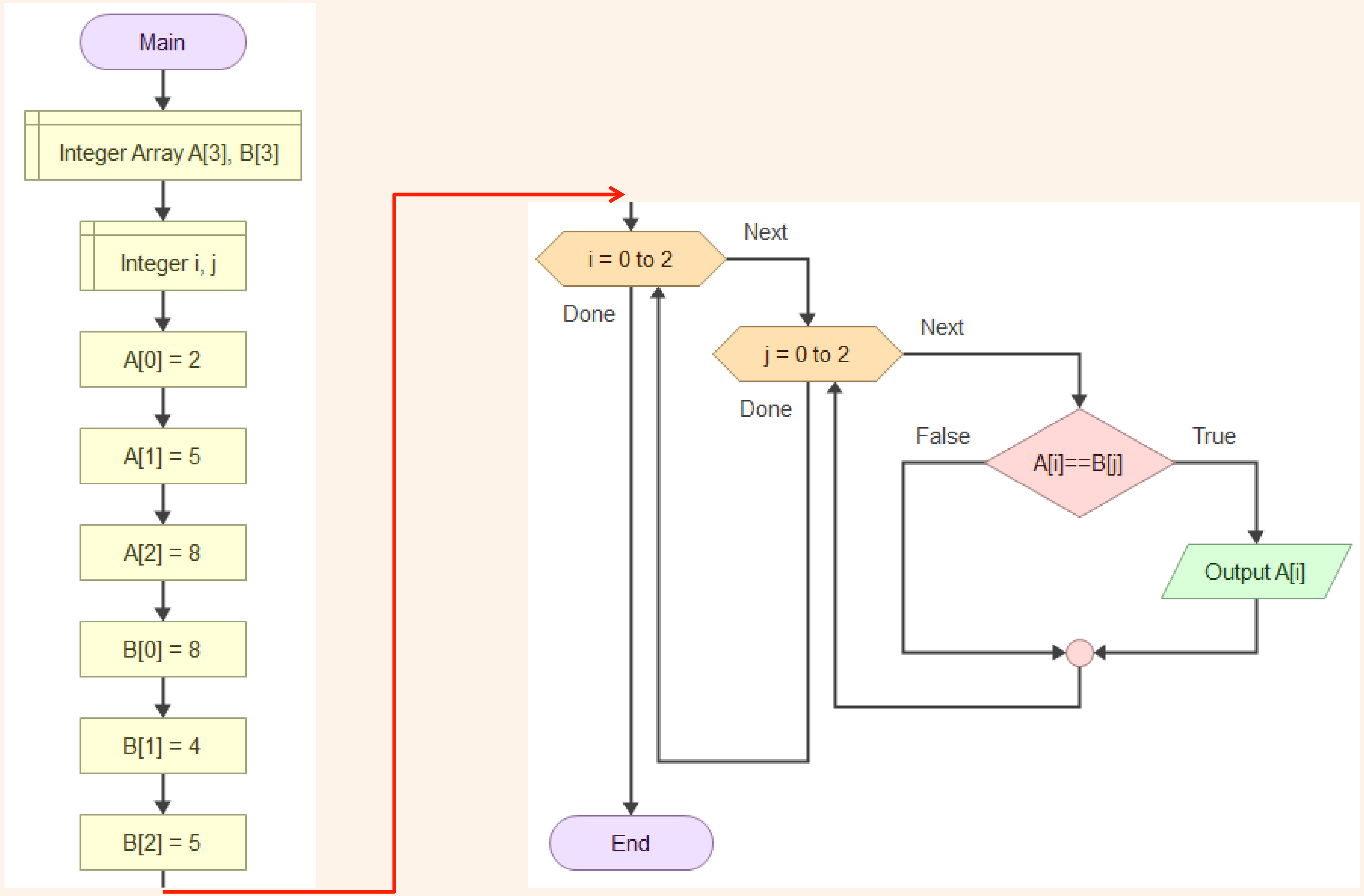
```
MAIN
... (Declarations)
Process: A[0] = 2
Process: A[1] = 5
Process: A[2] = 8
Process: B[0] = 8
Process: B[1] = 4
Process: B[2] = 5
```



LTT Example: Flowgorithm (Step 3)

```
MAIN
... (Initialization)
FOR i = 0 to 2, Step 1
FOR j = 0 to 2, Step 1
IF (A[i] == B[j])
Output: A[i]
END (Inner Loop)
END (Outer Loop)
END
```




Main
Integer Array A[3], B[3]
Integer i, j
A[0] = 2
A[1] = 5
A[2] = 8
B[0] = 8
B[1] = 4
B[2] = 5
i = 0 to 2
Next
Done
j = 0 to 2
Next
Done
False
A[i]==B[j]
True
Output A[i]
End

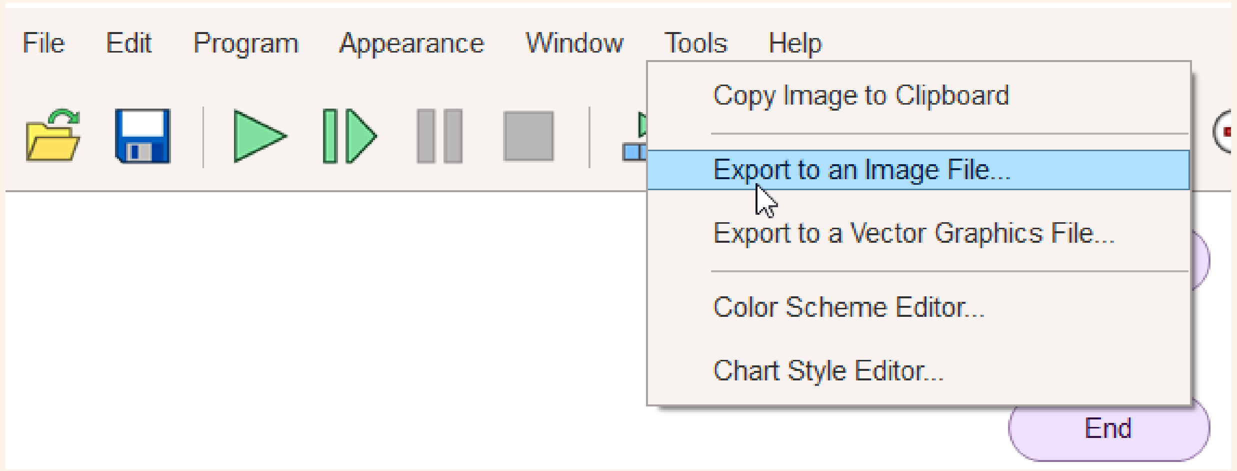
Practical Skill: Exporting

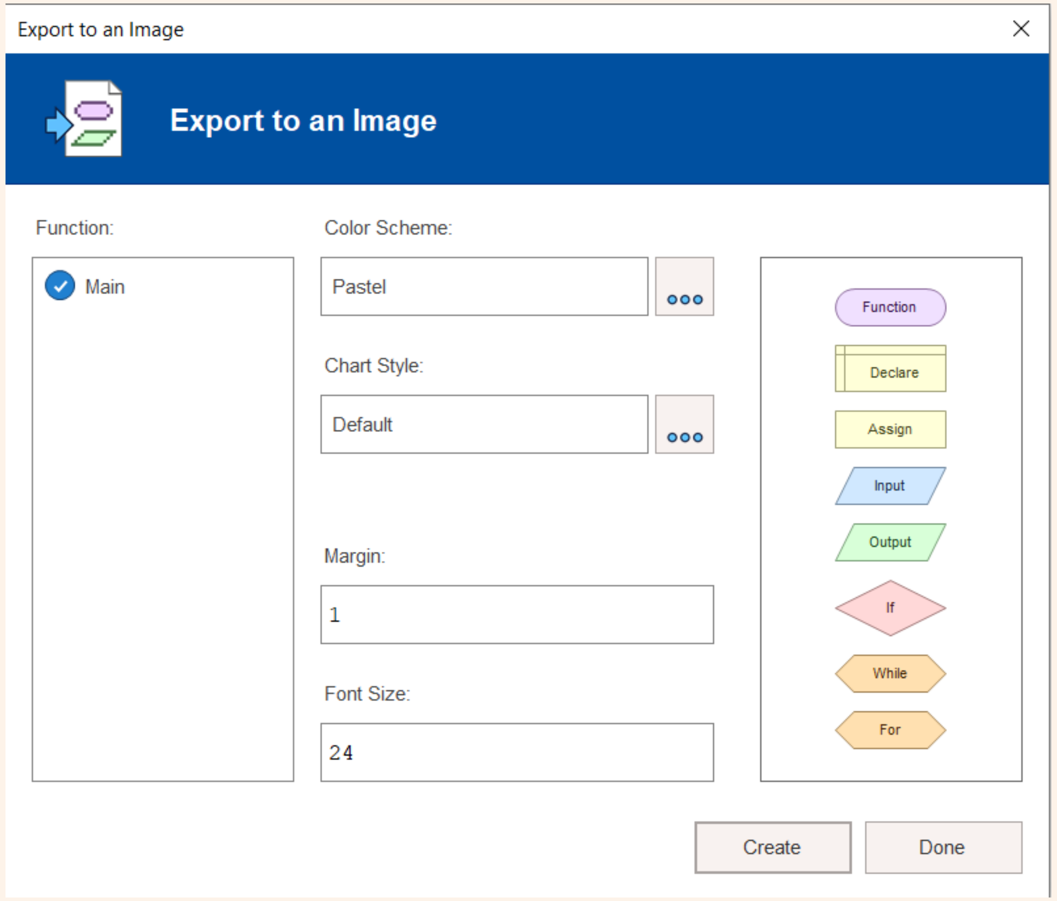
How do we save our complex algorithm for our students?

Flowgorithm allows you to export your flowchart as an image (PNG, JPEG) or a vector file (SVG, PDF).

How to Export:

1. Go to the **Tools** menu.
2. Select **"Export to an Image File..."**
3. Choose your format (e.g., PNG).
4. Save the file.





Function: ✓ Main

Color Scheme: Pastel

Chart Style: Default

Margin: 1

Font Size: 24

06

Summary

Session 4: Summary (Part 1)

Nested Loops

A loop inside another loop. The Inner loop runs completely for each step of the Outer loop.

Multiplication Table

We used nested loops to create this, which is essential for 2D data or comparing lists.

Session 4: Summary (Part 2)

Data Structures (Arrays)

Variables that hold a *list* of items (of the same type). The index starts at 0.

Processing Arrays

We *must* use a **For Loop** (from 0 to Size-1) to access every element.

LTT Example: Set Intersection

We successfully combined **Arrays, Nested Loops**, and **'If'** statements.

End of Session 4

Do you have any questions?

THANK YOU