Lecture 3

Variables and Logical Operators in MATLAB

Al-Mustaqbal University / College of Technical Engineering
Department (Medical Device Engineering)
Class (First)
Computer Programming and Applications I / Code (UOMU0204026)
Lecturer (Msc. Heba Hussien)

2nd term – Lecture 3 Variables and Logical Operators

in MATLAB

Lecture Objectives

By the end of this lecture, students will be able to:

1. Understand the concept of variables
2. Use assignment statements
3. Apply logical operators
4. Write simple MATLAB programs

1. Variables in MATLAB

A variable is **a name used to store a value in memory**. Variables are fundamental building blocks in programming that allow us to store, retrieve, and manipulate data throughout our programs. When you create a variable in MATLAB, the system allocates a specific portion of memory to hold the value you assign to it, and the variable name serves as a reference to that memory location.

Example

```
x = 5;
y = 10;
z = x + y;
```

Explanation

- x stores the value 5
- y stores the value 10
- z stores the result of x + y, which equals 15

Rules for Naming Variables

When naming variables in MATLAB, you must follow specific rules to ensure your code runs correctly. Variable names are case-sensitive, meaning 'myVar' and 'myvar' would be considered two different variables. Additionally, MATLAB has reserved keywords that cannot be used as variable names because they have special meaning in the programming language. Following these naming conventions helps make your code more readable and maintainable for yourself and others who may read your code.

- ✓ Must start with a letter
- ✓ Can contain numbers (after the first character)
- ✓ No spaces allowed (use underscores instead)
- ✓ Cannot use reserved words (e.g., if, while, for, end)

Email (heba.hussein.abdalabbas@uomus.edu.iq)

Valid

Examples

```
a = 10;
num1 = 20;
total_sum = 30;
```

Invalid Examples

```
1num = 5;        % Error: starts with a number
total sum = 10; % Error: contains a space
```

2. Assignment Statement

An assignment statement assigns a value to a variable using the equals sign (=). This is one of the most fundamental operations in programming. It's important to understand that the assignment operator is not the same as mathematical equality. In programming, the assignment operator means 'store the value on the right side in the variable on the left side.' This distinction is crucial for understanding how programs manipulate data and update variable values throughout execution.

```
x = 7;
```

Important Note

MATLAB evaluates expressions from right to left when performing assignments. This means that the expression on the right side of the equals sign is computed first, and then the result is stored in the variable on the left. This behavior is particularly important when a variable appears on both sides of an assignment, as shown in the example below.

```
x = 5;
x = x + 3;
% Final value of x is 8
```

3. Arithmetic Operations

MATLAB supports all standard arithmetic operations that you would expect in a mathematical computing environment. These operations follow the standard order of operations (PEMDAS: Parentheses, Exponents, Multiplication/Division, Addition/Subtraction), and you can use parentheses to explicitly control the order of evaluation in complex expressions. Understanding these operations is essential for performing calculations and manipulating numerical data in your programs.

Operation	Symbol
Addition	+
Subtraction	-
Multiplication	*
Division	/
Power	^

Table 1: Arithmetic Operators in MATLAB

Example

```
a = 10;
b = 2;
c = a^b;  % c = 100 (10 raised to power 2)
```

4. Logical Operators

Logical operators are essential for making decisions in your programs. They allow you to compare values and create conditions that determine which parts of your code should execute. In MATLAB, logical operations return either 1 (true) or 0 (false), which can be used in conditional statements and loops. Understanding how these operators work is fundamental to writing programs that can make decisions based on data and user input.

Comparison Operators

Comparison operators (also called relational operators) allow you to compare two values and determine their relationship. The result of a comparison is always a logical value: 1 (true) if the comparison is valid, or 0 (false) if it is not. These operators work with numeric values, characters, and other data types, making them versatile tools for building conditional logic in your programs.

Operation	Symbol
Equal	==
Not equal	~=
Greater than	>
Less than	<
Greater or equal	>=
Less or equal	<=

Table 2: Comparison Operators in MATLAB

Example

```
x = 5;
y = 10;

x < y   % result = 1 (true)
x > y   % result = 0 (false)
```

Logical Operations

Logical operations allow you to combine multiple conditions into more complex expressions. The AND operator (&&) returns true only when both conditions are true. The OR operator (||) returns true when at least one condition is true. The NOT operator (~) inverts a logical value, changing true to false and vice versa. These operators are fundamental for building sophisticated conditional logic that can handle complex decision-making scenarios.

Operation	Symbol
AND	&&

Operation	Symbol
OR	| |
NOT	~

Table 3: Logical Operators in MATLAB

Example

```
x = 5;
y = 10;

(x < y) && (y > 0)  % result = 1 (both conditions true)
```

5. Complete Program Example

The following program demonstrates the practical application of variables, arithmetic operations, and logical operators. It shows how these concepts work together in a real program: variables store values, arithmetic operations perform calculations, and logical operators make comparisons that result in true/false values.

```
a = 8;
b = 3;

sum = a + b;
isGreater = a > b;

disp(sum);       % Displays: 11
disp(isGreater); % Displays: 1 (true)
```

6. In-Class Activities

Practice is essential for mastering programming concepts. The following activities are designed to reinforce the concepts learned in this lecture. Work through each activity step by step, and don't hesitate to experiment with different values and approaches. Remember that making mistakes and debugging your code is an important part of the learning process.

Activity 1

Write a program that stores two numbers, calculates their sum, and displays the result. This activity will help you practice variable declaration, arithmetic operations, and output display.

Activity 2

Write a program that checks if a number is greater than 10. Use a comparison operator to make this determination and display the result.

Activity 3

Write a program that checks if a number is between 1 and 20 using logical operators. This activity requires combining multiple comparison operators with a logical AND operation.

7. Quiz Questions

Test your understanding of the concepts covered in this lecture by answering the following questions. Take your time to think through each answer before checking the explanation. These questions are designed to identify areas where you might need additional practice or review.

Q1

What is the output of the following code?

```
x = 4;
x = x + 6;
```

Answer: x = 10

Q2

What is the difference between = and ==?

Answer: = is the assignment operator (stores a value), while == is the comparison operator (checks if two values are equal).

Q3

What is the result of: 5 > 3?

Answer: 1 (true)

Q4

True or False: What is the result of x = (5 < 3)?

Answer: False. The expression (5 < 3) evaluates to 0 (false), so x = 0.

Q5

What is the result of: (5 > 3) && (2 < 1)?

Answer: 0 (false).
The first
condition is true, but the second is false, so the AND result is false.

8. Assignment

Complete the following programming exercises to reinforce your understanding of variables, operators, and basic program structure. These assignments progressively increase in difficulty, starting with basic calculations and moving toward more complex logical operations. Submit your completed programs along with comments explaining your approach.

Question 1

Write a program that stores two grades, calculates the average, and displays the result. This exercise practices variable declaration, arithmetic operations, and the use of the disp() function for output.

Question 2

Write a program that takes input from the user using the input() function and checks whether the number is positive or negative. Display an appropriate message based on the result of your logical comparison.

Question 3

Write a program that checks whether a number is even or odd. Use the mod() function to determine the remainder when dividing by 2. A number is even if mod(x,2) equals 0, and odd otherwise.

Hint: Use mod(x,2) to check if a number is divisible by 2.

Question 4 (Challenge)

Write a program that takes 3 numbers and finds the largest number using logical operators. This is a more challenging exercise that requires you to combine multiple comparison operations with logical operators to determine the maximum value among three inputs.

9. Summary

This lecture introduced the fundamental concepts of variables and logical operators in MATLAB programming. These concepts form the foundation for all subsequent programming topics and are essential for writing any meaningful program. Understanding how to store and manipulate data using variables, and how to make decisions using logical operators, will enable you to create increasingly sophisticated programs as you progress through the course.

- Variables store data in memory and can be manipulated throughout your program

Al-Mustaqbal University / College of Technical Engineering
Department (Medical Device Engineering)
Class (First)
Computer Programming and Applications I / Code (UOMU0204026)
Lecturer (Msc. Heba Hussien)

2nd term – Lecture 3 Variables and Logical Operators in MATLAB

- Assignment statements (=) give values to variables, evaluating from right to left
- Arithmetic operators (+, -, *, /, ^) perform mathematical calculations
- Logical operators (&&, ||, ~) help in decision making and creating conditions

Email (heba.hussein.abdalabbas@uomus.edu.iq)