MATLAB for Beginners: Functions

Connect with me: Youtube | LinkedIn | WhatsApp Channel | Web | Facebook | Twitter

- Download PDF
- To access the updated lecture notes, please click on the following link: https://yasirbhutta.github.io/matlab/docs/graphics.html

Functions

A function is a self-contained block of code that performs a specific task and can be called upon multiple times within your program. In MATLAB, functions can be defined using the function keyword followed by the function name, input arguments (optional), output arguments (optional), and the function body. Here's the basic structure:

```
function [output_arguments] = function_name(input_arguments)
  % Function body
  % Your code here...
  % ...
  % Perform calculations, data manipulation, etc.
  % ...
end
```

Description:

- **function keyword:** This tells MATLAB that you are defining a function.
- **output_arguments:** This is an optional comma-separated list of variables that will store the results of the function's calculations. These variables will be accessible outside the function.
- **function_name:** This is the name of your function. Choose a descriptive name that reflects what the function does.
- **input_arguments:** This is an optional comma-separated list of variables that will be passed to the function when it is called. These variables will be accessible inside the function.
- **% Function body:** This is the main part of your function, where you write the code that performs the desired task. You can use any MATLAB commands and functions within the function body.
- **end:** This marks the end of your function definition.

> **Important:**
>
> - This declaration statement must be the first executable line of the function. Valid function names begin with an alphabetic character, and can contain letters, numbers, or underscores.
> - While it's not strictly required, matching the function name and file name in MATLAB is a commonly followed and recommended practice for function files (m-files containing only function definitions).

Example #1: Addition Function

```
% This function adds two numbers together and returns the sum.
function sumResult = addNumbers(a, b)
    % Define the input arguments:
    % - a: The first number to be added (numerical value)
    % - b: The second number to be added (numerical value)

    % Perform addition and store the result in a variable.
    sumResult = a + b;
end
```

This function takes two numbers as input and returns their sum. You can call the function like this:

```
y = addNumbers(4,9);
disp(y);
```

Example #2: Simple function to square a number

```
function result = square(x)
  result = x * x;
end
```

This function takes a number as input and returns its square. You can call the function like this:

```
y = square(5);
disp(y);
```

The variable `y` will now contain the value 25.

Example #3: Function to calculate the area of a rectangle

```
function area = rectangleArea(width, height)
  area = width * height;
end
```

This function takes the width and height of a rectangle as inputs and returns its area. You can call the function like this:

```
area = rectangleArea(3, 4);
```

The variable `area` will now contain the value 12.

Example #4: Area of a Triangle

```
function area = calculateTriangleArea(base, height)
    % Calculates the area of a triangle given its base and height

    area = 0.5 * base * height;
end
```

Example #5: Calculates the area of a circle

```
function area = circle_area(radius)
  % Calculate the area of a circle
  area = pi * radius^2;
end
```

You can call the function like this:

```
% Example usage
radius = 5;
my_area = circle_area(radius);
dispg(['The area of the circle is:', num2str(my_area)]);
```

Example #6: Area and Perimeter of a Rectangle

```
function [area, perimeter] = rectangleProperties(length, width)
    % Calculates the area and perimeter of a rectangle

    % Area
    area = length * width;

    % Perimeter
    perimeter = 2 * (length + width);

    % Display results
    fprintf('Area: %f\n', area);
    fprintf('Perimeter: %f\n', perimeter);
end
```

Example #7: Power Function

```
function result = powerFunction(base, exponent)
    % Calculates the power of a number
    result = base ^ exponent;
end
```

Example #8: Function to convert Celsius to Fahrenheit

```
function fahrenheit = celsiusToFahrenheit(celsius)
  fahrenheit = (celsius * 9/5) + 32;
end
```

This function takes a temperature in Celsius as input and returns the equivalent temperature in Fahrenheit. You can call the function like this:

```
fahrenheit = celsiusToFahrenheit(20);
```

The variable `fahrenheit` will now contain the value 68.

Example #9: positive or negative number

function takes a number as input and returns a string indicating whether it is positive, negative, or zero.

```
function output = positive_or_negative(x)
  % This function determines whether a number is positive, negative, or zero.
  %
  % Args:
  %   x: A number (integer or float).
  %
  % Returns:
  %   A string indicating whether the number is positive, negative, or zero.

  if x > 0
    output = 'positive';
  elseif x < 0
    output = 'negative';
  else
    output = 'zero';
  end
end
```

Code Explanation:

1. Takes a single input argument x, which can be any numeric type.
2. Uses an if statement to check the value of x:
   - If x is greater than 0, the function returns the string "positive".
   - If x is less than 0, the function returns the string "negative".
   - Otherwise, the function returns the string "zero".

Example Usage:

```
number = 5;
result = positive_or_negative(number);
disp(result); % This will print "positive"
```

Example #10: Function to find the minimum value in a list:**

```
function minValue = findMin(data)
  minValue = data(1);
  for i = 2:length(data)
    if data(i) < minValue
      minValue = data(i);
    end
  end
end
```

This function takes a list of numbers as input and returns the smallest number in the list. You can call the function like this:

```
data = [5, 1, 8, 3];
minVal = findMin(data);
```

The variable `minVal` will now contain the value 1.

Intermediate

Example #11: Factorial Calculator

```
function fact = factorialCalc(n)
    % Calculates the factorial of a non-negative integer

    if n == 0
        fact = 1;
    else
        fact = 1;
        for i = 1:n
            fact = fact * i;
        end
    end
```

Example #12: Fibonacci Sequence Generator

```
function fibSeq = fibonacci(n)
    % Generates the first n terms of the Fibonacci sequence

    fibSeq = zeros(1, n);
    fibSeq(1) = 0;
    fibSeq(2) = 1;

    for i = 3:n
        fibSeq(i) = fibSeq(i-1) + fibSeq(i-2);
    end
end
```

Example #13: Least common multiple (LCM)

This function takes two positive integers as input and returns their LCM using the Euclidean algorithm.

```
function lcm = leastCommonMultiple(a, b)
  if a == 0 || b == 0
    lcm = 0;
  else
    while b ~= 0
      tmp = b;
      b = mod(a, b);
      a = tmp;
    end
    lcm = a;
  end
end
```

Example #14: Function for Grade Conversion

```
function grade = grade_converter(score)

if score < 0 || score > 100
    error('Score must be between 0 and 100');
end

if score >= 90
    grade = 'A';
elseif score >= 80
    grade = 'B';
elseif score >= 70
    grade = 'C';
elseif score >= 60
    grade = 'D';
else
    grade = 'F';
end
```

```
end
```

Using the '@' Operator

Anonymous function: An anonymous function is a function that is not stored in a program file, but is associated with a variable whose data type is function_handle. Anonymous functions can accept multiple inputs and return one output. They can contain only a single executable statement. [^1]

Example Usage:

```
score = 85;
grade = grade_converter(score);

fprintf('Score: %d, Grade: %s', score,grade)
```

Syntax:

```
function_handle = @(input_arguments) expression
```

Example #: Basic Anonymous function

```
square = @(x) x^2;  % Function to square a number

result = square(5);  % Call the function, result will be 25
```

Example #: Anonymous Function with Multiple Inputs

```
g = @(x, y) x + y;

sumResult = g(3, 5);  % Evaluating the function with x=3 and y=5
```

Example #: Using Built-in Functions with Anonymous Functions

```
h = @(a, b) sqrt(a^2 + b^2);

hypotenuse = h(3, 4);  % Evaluating the hypotenuse of a right triangle
```

True/False (Mark T for True and F for False)

- Functions can only return one output value. **True or False**

- The order of arguments matters when calling a function. **True or False**
- Variables defined within a function are local to that function. **True or False**
- MATLAB functions can only be called from other functions, not from script files. **True or False**
- Anonymous functions can accept multiple input arguments, but they can only directly return a single output value in MATLAB. **True or False**
-

Multiple Choice (Select the best answer)

> What is the purpose of the "function" keyword in MATLAB? a. Declares a variable b. Defines a loop c. Marks the beginning of a function d. Specifies a data type

> How can you return multiple values from a MATLAB function? a. Use a semicolon between values b. Separate values with commas c. Enclose values in square brackets d. Use square brackets and commas

> What is the purpose of input arguments in a MATLAB function? a. They define the function's name b. They determine the output of the function c. They provide values for computation within the function d. They are optional and not necessary

> Which command is used to call a function in MATLAB? a. call b. run c. execute d. The function name itself

> What is the primary difference between a function and a script in MATLAB? a) Functions can return values, while scripts cannot. b) Functions have their own workspace, while scripts share the base workspace. c) Functions can be called multiple times, while scripts can only run once. d) All of the above.

> What happens if you try to call a function with fewer arguments than it expects? a) The function calls the input function to prompt for missing values. b) The function uses default values for missing arguments. c) The function generates an error. d) The function uses random values for missing arguments.

> How do you define a function in MATLAB? a. for b. function c. if d. while

> How is a function file named in MATLAB typically saved?

1. [ ] With a ".m" extension
2. [ ] With a ".mat" extension
3. [ ] With a ".func" extension
4. [ ] Without any file extension

> In MATLAB, what is the role of the "end" keyword inside a function?

1. [ ] Marks the end of the function definition
2. [ ] Specifies the end of a loop or conditional statement
3. [ ] Ends the execution of the entire script
4. [ ] Indicates the end of a variable assignment

> How can you call a MATLAB function that is saved in a separate file?

1. [ ] Use the "call" command

2. ☐ Execute the function file directly
3. ☐ Import the function into the script
4. ☐ Use the function name in the script

> What is the output of the following MATLAB code snippet?

```
function result = multiply_numbers(a, b)
    result = a * b;
end

x = 3;
y = 4;
z = multiply_numbers(x, y);
disp(z);
```

1. ☐ 7
2. ☐ 12
3. ☐ 1
4. ☐ 0

> Which of the following correctly defines an anonymous function in MATLAB?

1. ☐ function_name = @(input_arguments) expression
2. ☐ function function_name(input_arguments) expression
3. ☐ anonymous_func = function(input_arguments) expression
4. ☐ function(input_arguments) expression

> What is the output of the following code?

```
square = @(x) x^2;
result = square(4);
disp(result);
```

1. ☐ 2
2. ☐ 4
3. ☐ 8
4. ☐ 16

> Which of the following is a valid way to create an anonymous function with two input arguments?

1. ☐ myfun = @(x, y) x * y;
2. ☐ myfun = function(x, y) x * y;
3. ☐ myfun(x, y) = x * y;
4. ☐ myfun = inline('x * y', 'x', 'y');

Exercises

- Write a function `sum3(num1,num3,num3)` that takes three numbers as input and returns the sum.
- Write a function that calculates the area of a circle, given its radius.
- Write a function `SumNum(num1)` that takes a number as input and returns the sum of numbers from 1 to that number (num1).
- Write a function `sumSquares(x)` that takes a vector of numbers as input and returns the sum of their squares.
- Write a function `isEven(x)` that takes a number as input and returns true if it is even, and false otherwise.
- Write a program with three functions:
  1. **`isEven(n)`:** This function takes an integer `n` as input and returns `True` if `n` is even and `False` otherwise. You can use the modulo operator (`%`) to check for evenness.
  2. **`printTable(n)`:** This function takes an integer `n` as input and prints its multiplication table. The table should show the product of `n` with each number from 1 to 10, formatted like `n * i = n * i`, where `i` is the current number in the loop.
  3. **`main`:** The main program should:
     - Prompt the user to enter an integer.
     - Use the `isEven(n)` function to check if the entered number is even.
     - If the number is even, call the `printTable(n)` function to print its multiplication table.
     - If the number is odd, print a message indicating the number is odd and not eligible for printing a table.

Example output:

```
Enter an integer: 4
4 is even! Here's its multiplication table:
4 * 1 = 4
4 * 2 = 8
4 * 3 = 12
...
4 * 10 = 40
```

- Write a function `findMax(data)` that takes a list of numbers as input and returns the maximum value in the list.
- Write a function `findSum(data)` that takes a list of numbers as input and returns the sum of allthe numbers in the list.
- Write a function `findProduct(data)` that takes a list of numbers as input and returns the product of all the numbers in the list.
- Write a function `avgPositive(data)` that takes a list of numbers as input and returns the average of all positive numbers in the list.

Review Questions

- What is a function in MATLAB?

- How do you define a function in MATLAB? Provide an example.
- How do you define a function in MATLAB? What are the essential components of a function definition?
- How do functions return output values?
- What is the difference between a function and a script in MATLAB?
- Describe the process of calling a function, including argument passing and return value handling.
- What happens if you try to call a function with the wrong number of arguments?
- How can you create plots within a function and display them to the user?
- How can you pass plotting options as arguments to functions to customize visualizations?
- Demonstrate how to pass multiple input arguments and return multiple output values.

References and Bibliography

- Declare function name, inputs, and outputs - MathWorks Help Center

- [^1]: Anonymous Functions - Mathworks