MAT 1206 – Introduction to MATLAB

CHAPTER 03: Basic programming structures

Lesson 3: Functions

Content

- Introduction to Functions
- Primary and Sub-Functions
- Global Variables

Introduction to Functions

A function is a group of statements that together perform a task. In MATLAB, functions are defined in separate files. The name of the file and of the function should be the same.

Functions can accept more than one input arguments and may return more than one output arguments

Syntax of a function statement is:

function [out1,out2, ..., outN] = myfun(in1,in2,in3, ..., inN)

Cont.

Example:

The following function named **mymax** should be written in a file named **mymax.m**. It takes five numbers as argument and returns the maximum of the numbers.

```
function maxNum = mymax(n1, n2, n3, n4, n5)
    %This function calculates the maximum of the five numbers given as input
    maxNum=max([n1, n2, n3, n4, n5]);
end
```

The first line of a function starts with the keyword **function**. It gives the name of the function and order of arguments. In our example, the **mymax** function has five input arguments and one output argument.

You can call the function as: **mymax(34, 78, 89, 23, 11)**

Cont.

Example:

A MATLAB function that calculates the area of a rectangle given its length and width.

```
function area = calculateRectangleArea(length, width)
    % Calculate the area of the rectangle
    area = length * width;
end


% Call the function and provide the length and width values
length = 5;
width = 3;
area = calculateRectangleArea(length, width);
fprintf('The area of the rectangle is: %.2f\n', area);
```

Cont.

Example: Creating a Fibonacci Sequence Function

1. Create a MATLAB function called **fibonacci_sequence** that takes an input argument n, representing the number of terms in the Fibonacci sequence to generate.
2. Inside the function, write code to generate the Fibonacci sequence up to the n^{th} term. The function should return the generated Fibonacci sequence as an output argument.
3. Write a MATLAB script that takes an integer from the user and calls the **fibonacci_sequence** function. Store the output in a variable.
4. Finally, display the generated Fibonacci sequence using the disp function or any other suitable means.

Cont.

MATLAB function fibonacci_sequence:

```
function sequence = fibonacci_sequence(n)
    F=[0 1];
    for i=3:n
        F(i)=F(i-1) + F(i-2);
    end
    sequence=F;
end
```

Main script:

```
n=input('Enter an integer ');
f_secuence=fibonacci_sequence(n);
disp(f_secuence);
```

Cont.

Example:

Write another function called **sum_square** to calculate the square summation of the Fibonacci sequence that generated in the previous example. Then update the main script to print the sum of squares.

Cont.

MATLAB function **sum_square**:

```
function s = sum_square(F)
    A=F.^2;
    s=sum(A);
end
```

Updated Main script:

```
n=input('Enter an integer ');
f_secuence=fibonacci_sequence(n);
disp(f_secuence);

sum=sum_square(f_secuence);
disp(['Sum of squares is ' num2str(sum)])
```

Primary and Sub-Functions

Any function must be defined within a file. Each function file contains a required primary function that appears first and any number of optional sub-functions that comes after the primary function and used by it.

Primary functions can be called from outside of the file that defines them, but sub-functions can not be called from other functions, outside the function file.

Sub-functions are visible only to the primary function and other sub-functions within the function file that defines them.

Cont.

Example

Write a function named **quadratic** that would calculate the roots of a quadratic equation. The function would take three inputs, the quadratic coefficient, the linear co-efficient and the constant term. It would return the roots.

Cont.

MATLAB function **quadratic**:

```
function [x1,x2] = quadratic(a,b,c)
    d = disc(a,b,c);
    x1 = (-b + d) / (2*a);
    x2 = (-b - d) / (2*a);
end
function dis = disc(a,b,c)
    %function calculates the discriminant
    dis = sqrt(b^2 - 4*a*c);
end % end of sub-function
```

Main script:

```
[x1,x2]=quadratic(2,4,-4)
```

Cont.

Example:

Write a MATLAB function named **calculate_volume** to calculate the volume of a cylinder by taking radius and its height as user inputs.

Cont.

MATLAB function calculate_volume:

```
function volume = calculate_volume(r,h)
    base_area=pi*r^2;
    volume=base_area*h;
end
```

Main Script:

```
r=input('Enter the radius ');
h=input('Enter the height ');
volume=calculate_volume(r,h);
fprintf('The volume of the cylinder is %f',volume);
```

Cont.

MATLAB function calculate_volume with sub function calculate_area:

```
function volume = calculate_volume(r,h)
    base_area=calculate_area(r);
    volume=base_area*h;
end

function area = calculate_area(r)
    area= pi*r^2;
end
```

Main Script:

```
r=input('Enter the radius ');
h=input('Enter the height ');
volume=calculate_volume(r,h);
fprintf('The volume of the cylinder is %f',volume);
```

Global Variables

Global variables can be shared by more than one function. For this, you need to declare the variable as global in all the functions.

The global declaration must occur before the variable is actually used in a function. It is a good practice to use capital letters for the names of global variables to distinguish them from other variables.

Cont.

Example

Let us create a function file named **average.m** and type the following code in it:

```
function avg = average(nums)
    global TOTAL
    avg = sum(nums)/TOTAL;
end
```

Create a script file and type the following code in it:

```
global TOTAL;
TOTAL = 10;
n = [34, 45, 25, 45, 33, 19, 40, 34, 38, 42];
av = average(n)
```

Questions/queries?