BIL 108E
Introduction to Scientific and Engineering Computing

ASST. PROF. DR. İPEK AKIN

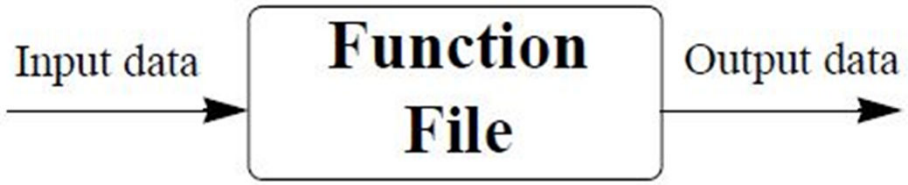
Function and Function Files

INTRODUCTION

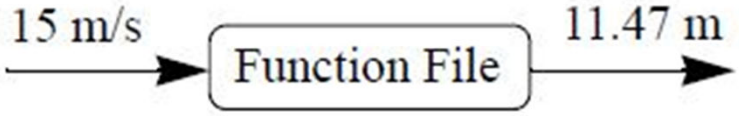
- The function can be expressed in the form of y = f(x)

- The main feature of a function file is that it has an input and an output → calculations in the function file are carried out using the input data, and the results of the calculations are transferred out of the function file by the output.

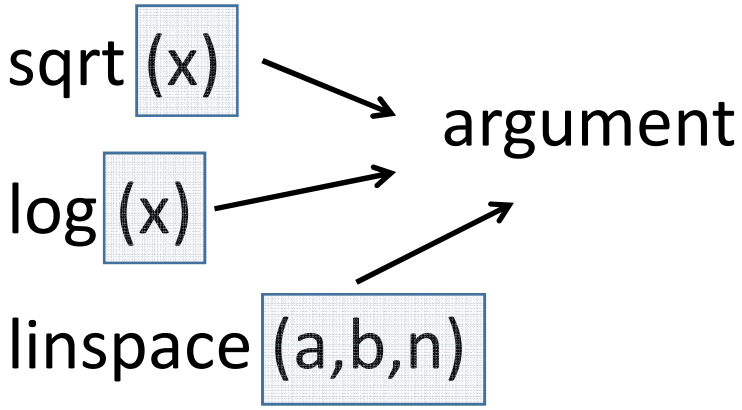
- The input and the output can be one or several variables, and each can be a scalar, vector, or an array of any size.



A very simple example of a user-defined function is a function that calculates the maximum height that a ball reaches when thrown upward with a certain velocity. For a velocity v_0, the maximum height h_{max} is given by $h_{max} = \frac{v_0^2}{2g}$, where g is the gravitational acceleration. In function form this can be written as $h_{max}(v_0) = \frac{v_0^2}{2g}$. In this case the input to the function is the velocity (a number), and the output is the maximum height (a number). For example, in SI units (g = 9.81 m/s^2) if the input is 15 m/s, the output is 11.47 m.



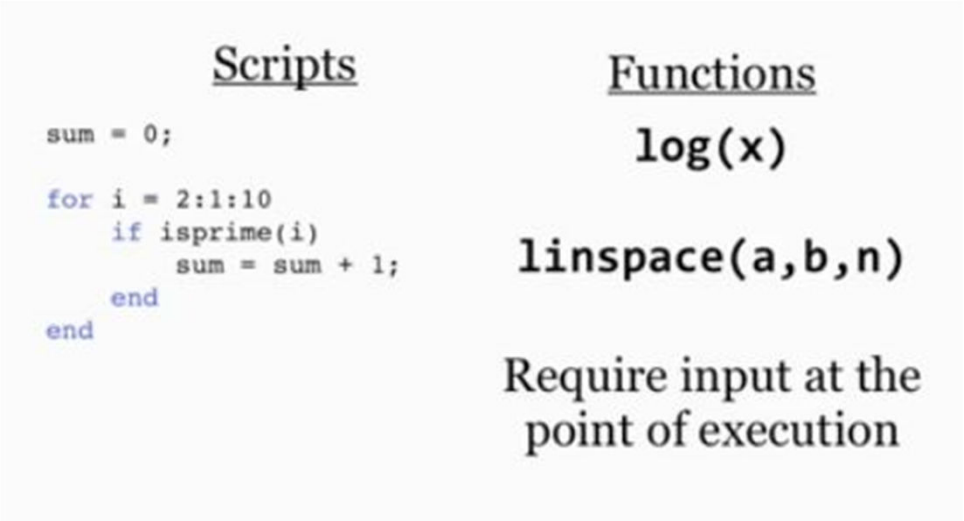
Funtions we know



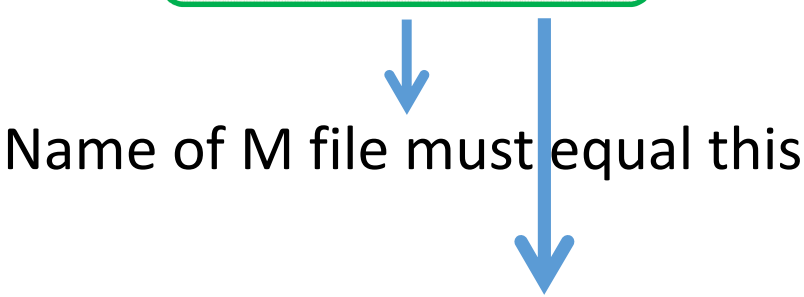
M files

1) Scripts (Week 4)

2) Functions



function [output variables] = function_name(input variables)



Name of M file must equal this

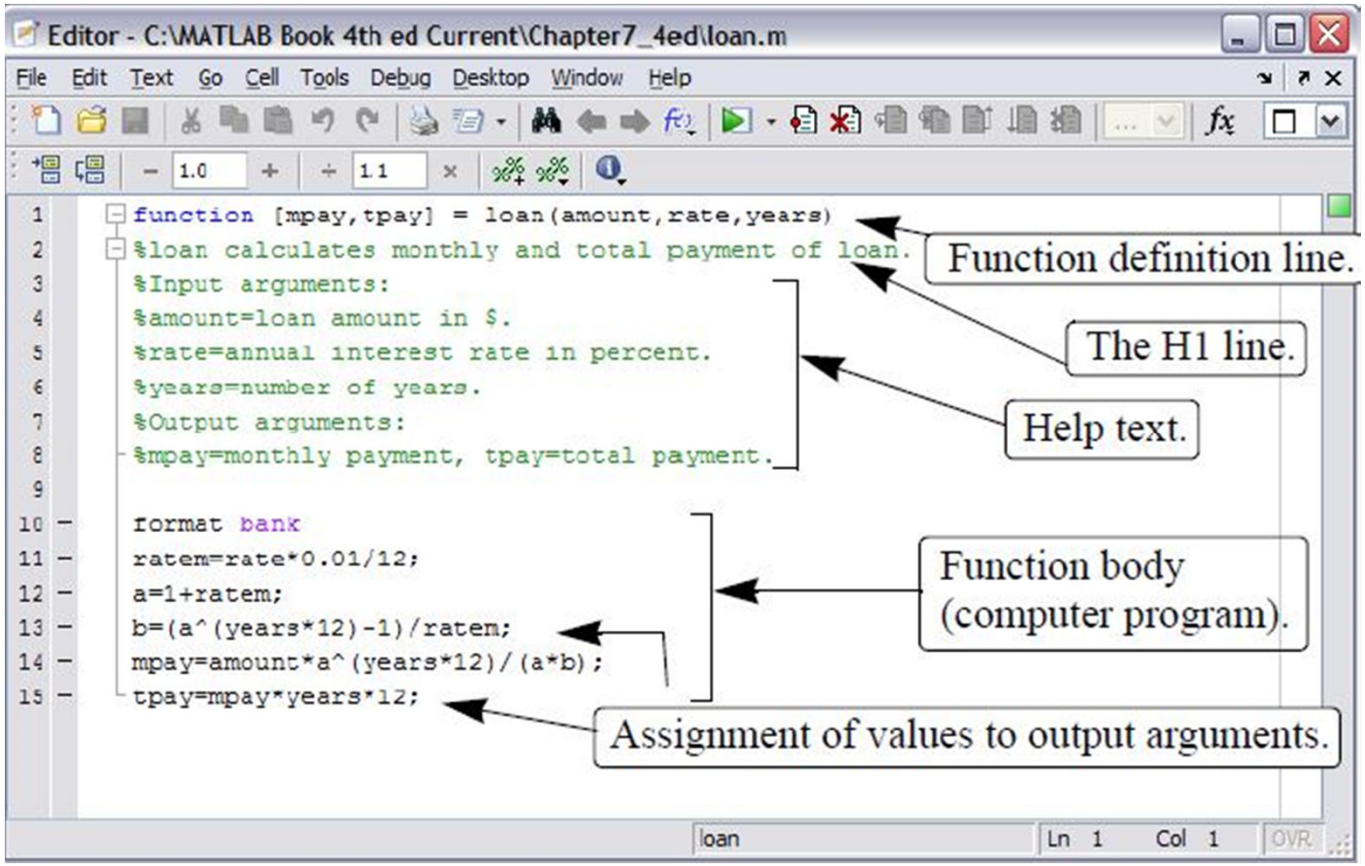
Cannot equal existing built-in function name

Example

function y = dude(x)

Structure of a Function File

- The inputs to the function are the amount of the loan, the annual interest rate, and the duration of the loan (number of years).
- This particular function calculates the monthly payment and the total payment of a loan.
- The output from the function is the monthly payment and the total payment.



Function Definition Line

- The first executable line in a function file *must* be the function definition line (Otherwise the file is considered a script file)

- The function definition line:
  - Defines the file as a function file.
  - Defines the name of the function.
  - Defines the number and order of the input and output arguments.

The form of the function definition line is:

```
function [output arguments] = function_name(input arguments)
```

The word "function" must be the first word, and must be typed in lowercase letters.

A list of output arguments typed inside brackets.

The name of the function.

A list of input arguments typed inside parentheses.

Input and Output Arguments

- Used to transfer data into and out of the function.
- The input arguments are listed inside parentheses following the function name.

Function definition line	**Comments**
function [mpay,tpay] = loan(amount,rate,years)	Three input arguments, two output arguments.
function [A] = RectArea(a,b)	Two input arguments, one output argument.
function A = RectArea(a,b)	Same as above; one output argument can be typed without the brackets.
function [V, S] = SphereVolArea(r)	One input variable, two output variables.
function trajectory(v,h,g)	Three input arguments, no output arguments.

The H1 Line and Help Text Lines

- The H1 line and help text lines are comment lines (lines that begin with the percent, %, sign) following the function definition line.

- They are optional but are frequently used to provide information about the function.

The H1 Line and Help Text Lines

- The H1 line is the first comment line and usually contains the name and a short definition of the function.

- When a user types (in the Command Window) **lookfor a_word**, MATLAB searches for **a_word** in the H1 lines of all the functions, and if a match is found, the H1 line that contains the match is displayed.

Function Body

- The function body contains the computer program (code) that actually performs the computations.

- The code can use all MATLAB programming features. This includes calculations, assignments, any built-in or user-defined functions, flow control (conditional statements and loops), comments, blank lines, and interactive input and output.

Local and Global Variables

- All the variables in a function file are local (the input and output arguments and any variables that are assigned values within the function file).

- This means that the variables are defined and recognized only inside the function file.

Local and Global Variables

- When a function file is executed, MATLAB uses an area of memory that is separate from the workspace (the memory space of the Command Window and the script files).

Local and Global Variables

- Each function file has its own local variables, which are not shared with other functions or with the workspace of the Command Window and the script files. It is possible, however, to make a variable common (recognized) in several different function files, and perhaps in the workspace too.

- This is done by declaring the variable **global** with the global command, which has the form:

```
global variable_name
```

Local and Global Variables

Several variables can be declared global by listing them, separated with spaces, in the global command

```
global GRAVITY_CONST FrictionCoefficient
```

- The variable has to be declared global in every function file that the user wants it to be recognized in. The variable is then common only to these files.
- The `global` command must appear before the variable is used. It is recommended to enter the `global` command at the top of the file.
- The `global` command has to be entered in the Command Window, or in a script file, for the variable to be recognized in the workspace.
- The variable can be assigned, or reassigned, a value in any of the locations in which it is declared common.
- The use of long descriptive names (or all capital letters) is recommended for global variables in order to distinguish them from regular variables.

Saving a Function File

Function definition line	**File name**
function [mpay,tpay] = loan(amount,rate,years)	loan.m
function [A] = RectArea(a,b)	RectArea.m
function [V, S] = SphereVolArea(r)	SphereVolArea.m
function trajectory(v,h,g)	trajectory.m

Using a User-Defined Function

- A function can be used by assigning its output to a variable (or variables), as a part of a mathematical expression, as an argument in another function, or just by typing its name in the Command Window or in a script file.

- **In all cases the user must know exactly what the input and output arguments are.**

- **An input argument can be a number, a computable expression, or a variable that has an assigned value**.

- **The arguments are assigned according to their position in the input and output argument lists in the function definition line**.

Using a User-Defined Function

```
>> [month total]=loan(25000,7.5,4)

month =
         600.72
total =
       28834.47
```

First argument is loan amount, second is interest rate, and third is number of years.

The loan function is used with two pre-assigned variables and a number as the input arguments:

```
>> a=70000;   b=6.5;
>> [x y]=loan(a,b,30)
```

Define variables a and b.

Use a, b, and the number 30 for input arguments and x (monthly pay) and y (total pay) for output arguments.

```
x =
         440.06
y =
      158423.02
```

Example (User-defined function for a math function)

Write a function file (name it chp7one) for the function $f(x) = \frac{x^4\sqrt{3x+5}}{(x^2+1)^2}$. The input to the function is x and the output is $f(x)$. Write the function such that x can be a vector. Use the function to calculate:

(*a*) $f(x)$ for $x = 6$.

(*b*) $f(x)$ for $x = 1, 3, 5, 7, 9,$ and 11.

Solution (User-defined function for a math function)

```
function y=chp7one(x)
y=(x.^4.*sqrt(3*x+5))./(x.^2+1).^2;
```

Function definition line.

Assignment to output argument.

- Note that the mathematical expression in the function file is written for element by-element calculations. In this way if *x* is a vector, *y* will also be a vector.
- The function is saved and then the search path is modified to include the directory where the file was saved. As shown below, the function is used in the Command Window.

Solution (User-defined function for a math function)

(a) Calculating the function for x=6 can be done by typing chp7one(6) in the Command Window, or by assigning the value of the function to a new variable:

```
>> chp7one(6)
ans =
    4.5401
>> F=chp7one(6)
F =
    4.5401
```

Solution (User-defined function for a math function)

(b) To calculate the function for several values of *x*, a vector with the values of *x* is created and then used for the argument of the function.

```
>> x=1:2:11
x =
     1     3     5     7     9    11
```

```
>> chp7one(x)
ans =
    0.7071    3.0307    4.1347    4.8971    5.5197    6.0638
```

Another way is to type the vector *x* directly in the argument of the function.

```
>> H=chp7one([1:2:11])
H =
    0.7071    3.0307    4.1347    4.8971    5.5197    6.0638
```

Example (Converting temperature units)

Write a user-defined function (name it **FtoC**) that converts temperature in degrees F to temperature in degrees C. Use the function to solve the following problem.

The change in the length of an object, ΔL, due to a change in the temperature, ΔT, is given by: $\Delta L = \alpha L \Delta T$, where α is the thermal expansion coefficient. Determine the change in the area of a rectangular (4.5 m by 2.25 m) Al ($\alpha = 23.10^{-6}$ 1/°C) plate if the temperature changes from 40°F to 92°F.

Solution

A user-defined function that converts degrees F to degrees C is:

```
function C=FtoC(F)
%FtoC converts degrees F to degrees C
C=5*(F-32)./9;
```

Function definition line.

Assignment to output argument.

A script file (named Chapter7Example2) that calculates the change of the area of the plate due to the temperature is:

```
a1=4.5; b1=2.25; T1=40; T2=92; alpha=23e-6;
deltaT=FtoC(T2)-FtoC(T1);

a2=a1+alpha*a1*deltaT;
b2=b1+alpha*b1*deltaT;
AreaChange=a2*b2-a1*b1;
fprintf('The change in the area is %6.5f meters
square.',AreaChange)
```

Using the `FtoC` function to calculate the temperature difference in degrees C.

Calculating the new length.

Calculating the new width.

Calculating the change in the area.

Executing the script file in the Command Window gives the solution:

```
>> Chapter7Example2
The change in the area is 0.01346 meters square.
```

Comparison Between Script Files and Function Files

- Both script and function files are saved with the extension .m (that is why they are sometimes called M-files).
- The first executable line in a function file is (must be) the function definition line.
- The variables in a function file are local. The variables in a script file are recognized in the Command Window.
- Script files can use variables that have been defined in the workspace.
- Script files contain a sequence of MATLAB commands (statements).
- Function files can accept data through input arguments and can return data through output arguments.
- When a function file is saved, the name of the file should be the same as the name of the function.

Anonymous and Inline Functions

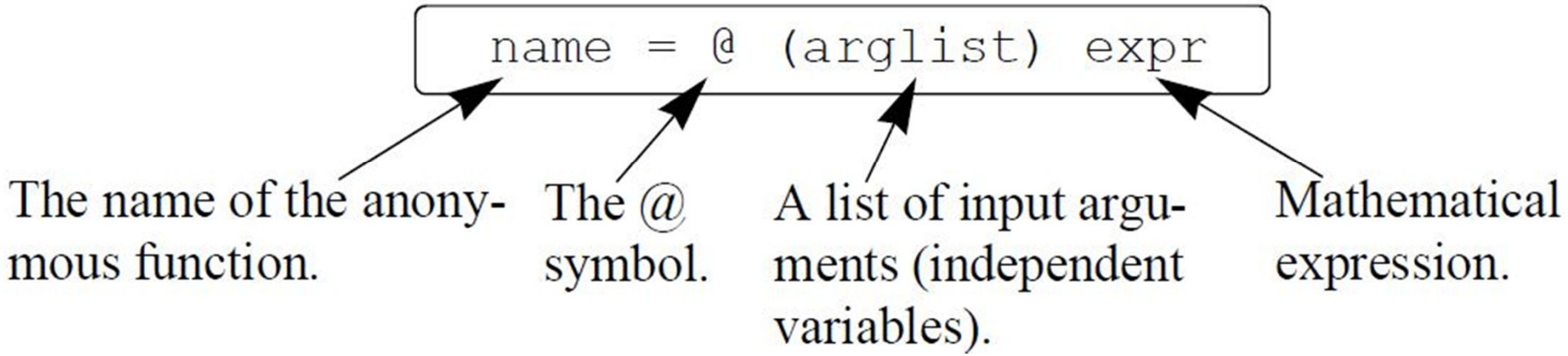
- An anonymous function is a user-defined function that is defined and written within the computer code (not in a separate function file) and is then used in the code.

- Anonymous functions can be defined in any part of MATLAB (in the Command Window, in script files, and inside regular user-defined functions).

- Anonymous functions were introduced in MATLAB 7.

Anonymous Functions

- An anonymous function is a simple (one-line) user-defined function that is defined without creating a separate function file (M-file).
- Anonymous functions can be constructed in the Command Window, within a script file, or inside a regular user-defined function.



Anonymous Functions

A simple example is: `cube = @ (x) x^3`, which calculates the cube of the input argument.

The command creates the anonymous function and assigns a handle for the function to the variable name on the left-hand side of the = sign.

- The `expr` consists of a single valid mathematical MATLAB expression.
- The mathematical expression can have one or several independent variables. The independent variable(s) is (are) entered in the `(arglist)`. Multiple independent variables are separated with commas. An example of an anonymous function that has two independent variables is: `circle = @ (x,y) 16*x^2+9*y^2`
- The mathematical expression can include any built-in or user-defined functions.
- The expression must be written according to the dimensions of the arguments (element-by-element or linear algebra calculations).

Anonymous Functions

- The expression can include variables that are already defined when the anonymous function is defined. For example, if three variables `a`, `b`, and `c` are defined (have assigned numerical values), then they can be used in the expression of the anonymous function `parabola = @ (x) a*x^2+b*x+c`.

- Once an anonymous function is defined, it can be used by typing its name and a value for the argument (or arguments) in parantheses.

Example (Anonymous Functions)

The function $f(x) = \dfrac{e^{x^2}}{\sqrt{x^2+5}}$ can be defined (in the Command Window) as an anonymous function for x as a scalar by:

```
>> FA = @ (x) exp(x^2)/sqrt(x^2+5)
FA =
    @(x)exp(x^2)/sqrt(x^2+5)
```

- If a semicolon is not typed at the end, MATLAB responds by displaying the function.
- The function can then be used for different values of *x*, as shown below.

```
>> FA(2)
ans =
   18.1994
>> z = FA(3)
z =
   2.1656e+003
```

Example (Anonymous Functions)

If x is expected to be an array, with the function calculated for each element, then the function must be modified for element-by-element calculations.

```
>> FA = @ (x) exp(x.^2)./sqrt(x.^2+5)
FA =
    @(x)exp(x.^2)./sqrt(x.^2+5)
>> FA([1 0.5 2])
ans =
    1.1097    0.5604   18.1994
```

Using a vector as input argument.

Inline Functions

Defined without creating a separate function file (M-file). As already mentioned, anonymous functions replace the inline functions used in earlier versions of MATLAB.

Inline functions are created with the **inline** command according to the following format:

```
name = inline('math expression typed as a string')
```

Inline Functions

A simple example is `cube = inline('x^3')`, which calculates the cube of the input argument.

- The mathematical expression can have one or several independent variables.
- Any letter except i and j can be used for the independent variables in the expression.
- The mathematical expression can include any built-in or user-defined functions.
- The expression must be written according to the dimension of the argument (element-by-element or linear algebra calculations).
- The expression *cannot* include pre assigned variables.
- Once the function is defined it can be used by typing its name and a value for the argument (or arguments) in parentheses (see example below).
- The `inline` function can be used as an argument in other functions.

Inline Functions

For example, the function: $f(x) = \dfrac{e^{x^2}}{\sqrt{x^2+5}}$ can be defined as an inline function for x by:

```
>> FA=inline('exp(x.^2)./sqrt(x.^2+5)')
FA =
     Inline function:
     FA(x) = exp(x.^2)./sqrt(x.^2+5)
>> FA(2)
ans =
   18.1994
>> FA([1 0.5 2])
ans =
    1.1097    0.5604   18.1994
```

Expression written with element-by-element operations.

Using a scalar as the argument.

Using a vector as the argument.

Inline Functions

An inline function that has two or more independent variables can be written by using the following format:

```
name = inline('mathematical expression','arg1',
                                    'arg2','arg3')
```

In the format shown here the order of the arguments to be used when calling the function is defined. If the independent variables are not listed in the command, MATLAB arranges the arguments in alphabetical order. For example, the function $f(x, y) = 2x^2 - 4xy + y^2$ can be defined as an inline function by:

Inline Functions

```
>> HA=inline('2*x^2-4*x*y+y^2')

HA =
     Inline function:
     HA(x,y) = 2*x^2-4*x*y+y^2
```

Once defined, the function can be used with any values of x and y. For example, HA(2,3) gives:

```
>> HA(2,3)
ans =
    -7
```