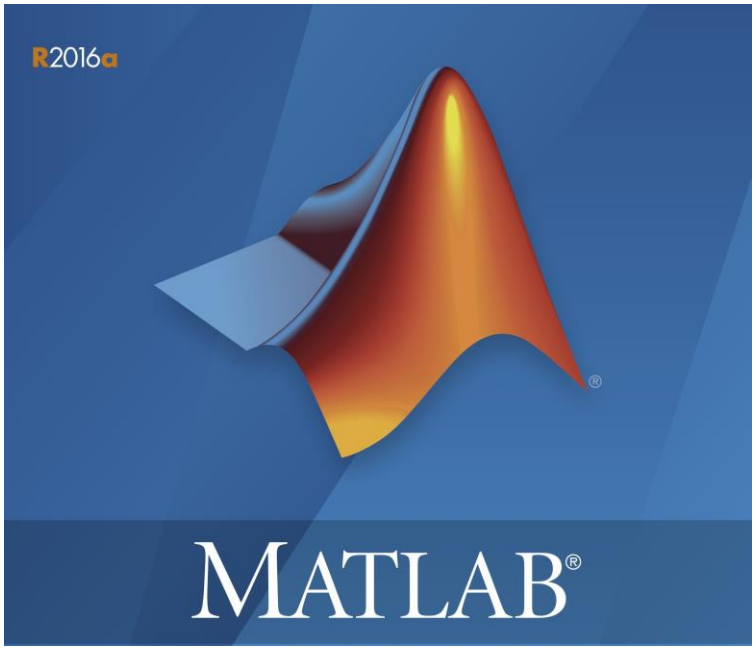
MATLAB

a short primer



Ovidiu Ivanov
'Gheorghe Asachi' Technical University,
Electrical Engineering Faculty
Iasi, Romania
www.tuiasi.ro

Part Three

Writing your first program

scripts and functions
if, for, while, switch-case

How to write a program in MATLAB

- You can write programs in the Command Window, but it will be very difficult to
  - fix writing mistakes
  - analyze large portions of code
  - debug your program

- The Editor is the preferred choice for writing
  - scripts
  - functions

Scripts and functions

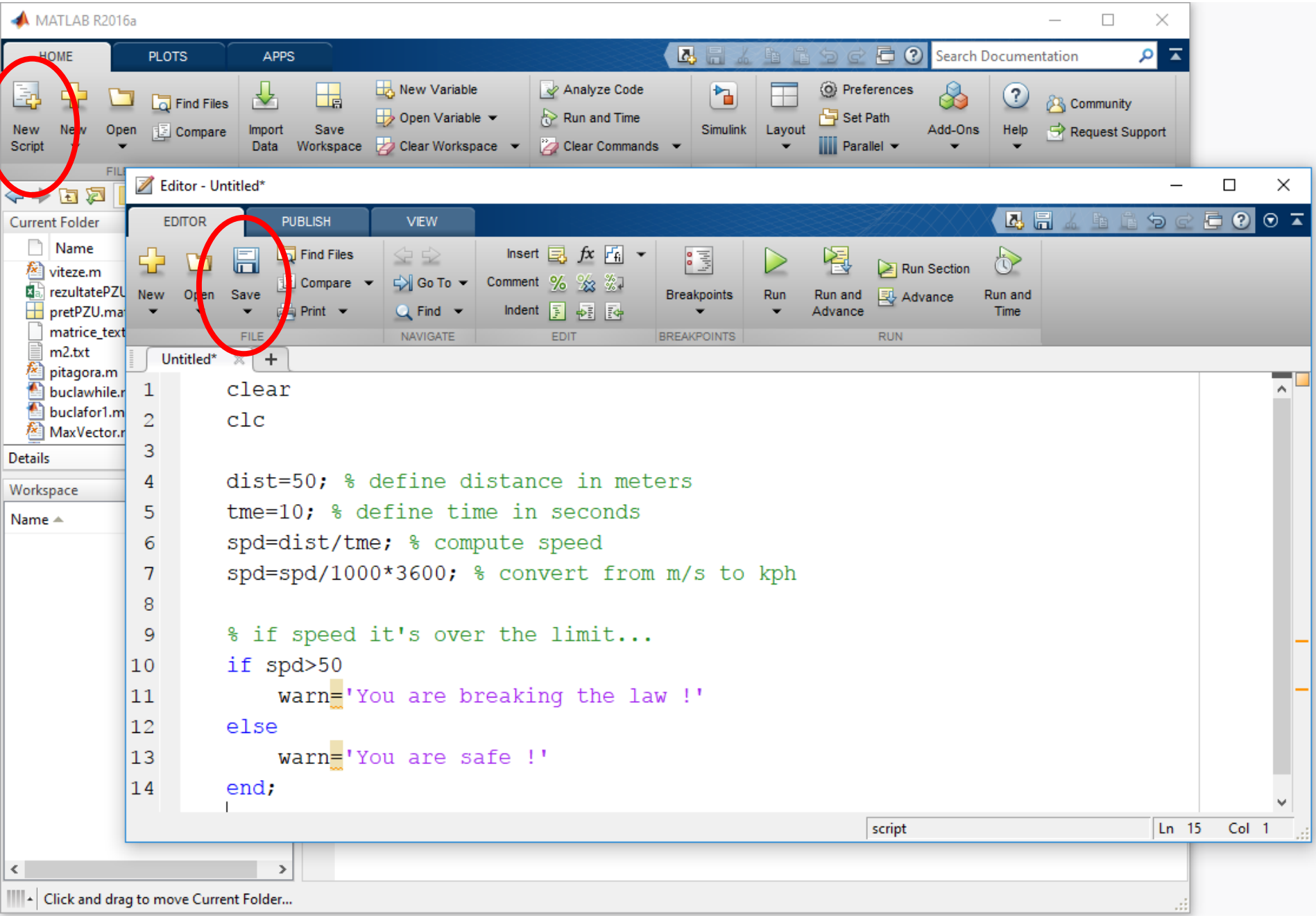
- Scripts are standalone pieces of code
- Functions are subroutines, they have input and output variables and can be used to make your program modular, to break large code into smaller pieces, for easier debugging and multiple calls
- You will typically use a script for your main program and call from it one or several functions
- Functions can call other functions

Writing a script

We will:

- Copy or write the small program we wrote in Part One into an Editor window

- Use the New Script button to open a new editor Window
- Undock the Editor window
- Write the code, watching for any syntax errors
  - Observe the automatic alignment of code
- Save the result with the Save button


MATLAB R2016a
HOME
PLOTS
APPS
Search Documentation
New Script
New
Open
Find Files
Compare
Import Data
Save Workspace
New Variable
Open Variable
Clear Workspace
Analyze Code
Run and Time
Clear Commands
Simulink
Layout
Preferences
Set Path
Parallel
Add-Ons
Help
Community
Request Support
Current Folder
Name
viteze.m
rezultatePZU
pretPZU.ma
matrice_text
m2.txt
pitagora.m
buclawhile.r
buclafor1.m
MaxVector.r
Details
Workspace
Name
Click and drag to move Current Folder...
Editor - Untitled*
EDITOR
PUBLISH
VIEW
New
Open
Save
Find Files
Compare
Print
Go To
Find
Insert
Comment
Indent
Breakpoints
Run
Run and Advance
Run Section
Advance
Run and Time
FILE
NAVIGATE
EDIT
BREAKPOINTS
RUN
Untitled*
1 clear
2 clc
3
4 dist=50; % define distance in meters
5 tme=10; % define time in seconds
6 spd=dist/tme; % compute speed
7 spd=spd/1000*3600; % convert from m/s to kph
8
9 % if speed it's over the limit...
10 if spd>50
11 warn='You are breaking the law !'
12 else
13 warn='You are safe !'
14 end;
script
Ln 15 Col 1

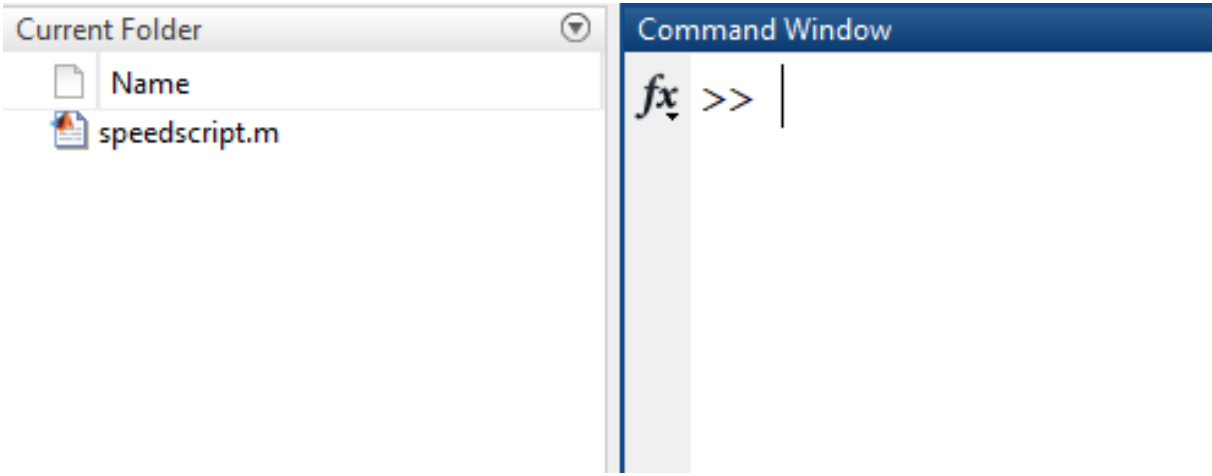
The script code

```
clear
clc

dist=50; % define distance in meters
tme=10; % define time in seconds
spd=dist/tme; % compute speed
% convert from m/s to kph
spd=spd/1000*3600;

% if speed it's over the limit...
if spd>50
    warn='You are breaking the law !'
else
    warn='You are safe !'
end;
```

Saving scripts

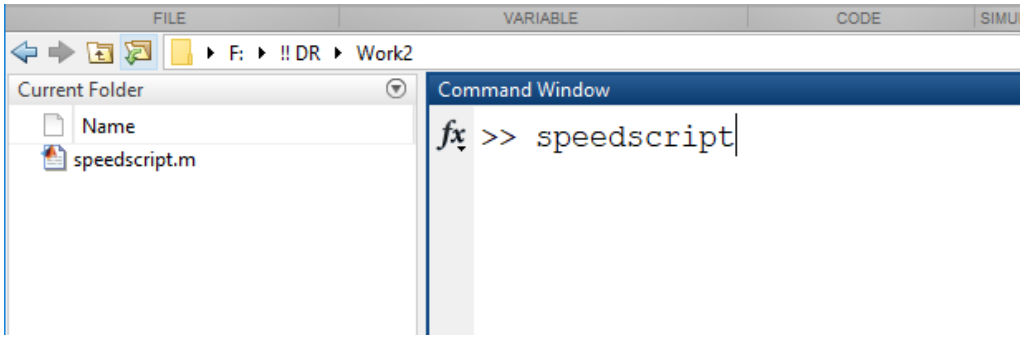
- The result of the save will be a file with extension .m.
- m-files are text files with changed extension, you can create and edit them in any text editor

- m-files can be run only from the current directory or from a directory found in the MATLAB search path.

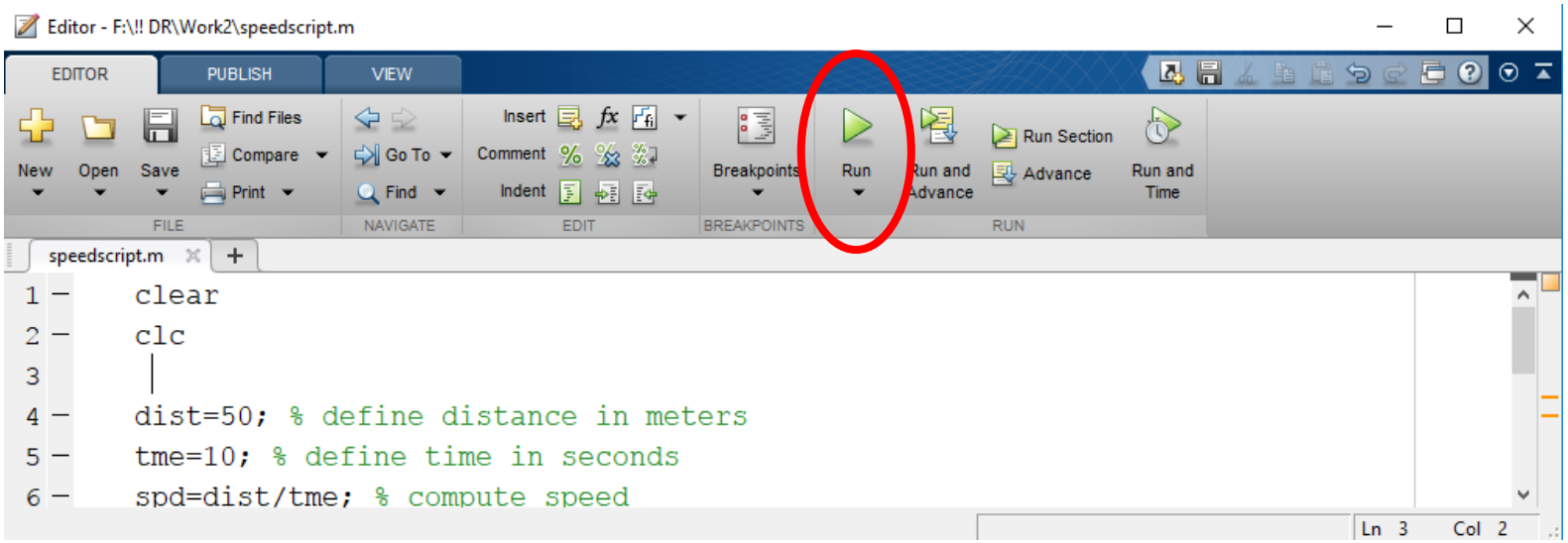


Running a script

- **Method 1:** write its name in the Command Window and press Enter

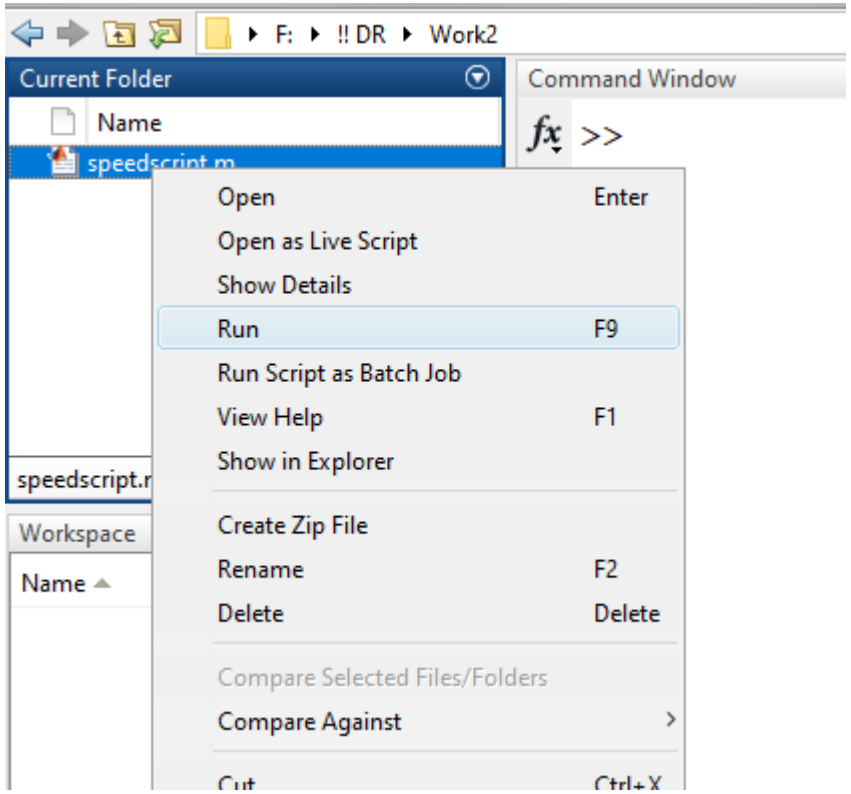


- **Method 2:** Pres the Run button in the Editor Window



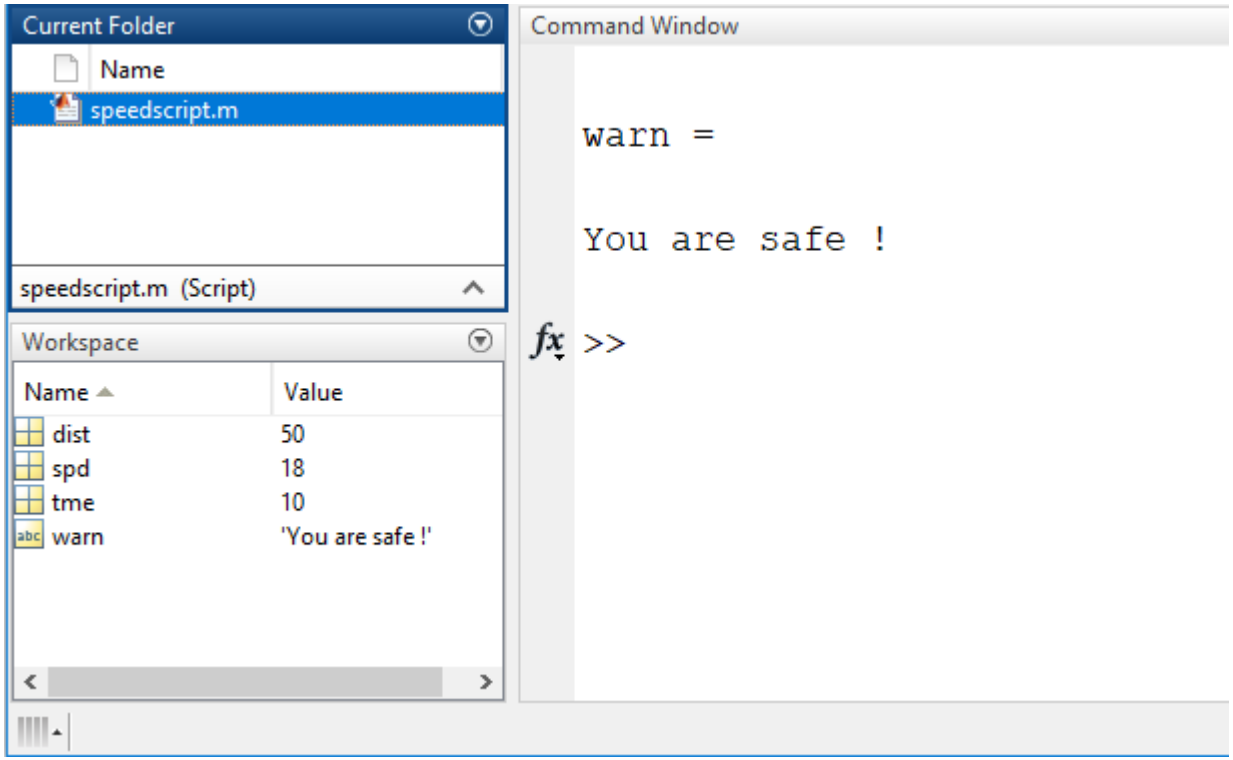
Running a script

- **Method 3:** Select the script with the mouse in the Current Directory Window. Right-click and choose the **Run** option or press **F9**.



Results

- **Whatever the method used, you should get the same result, a message that you are safe, because the speed is below the allowed limit.**



All the variables are still available in the Workspace.

Speed is 18 kph

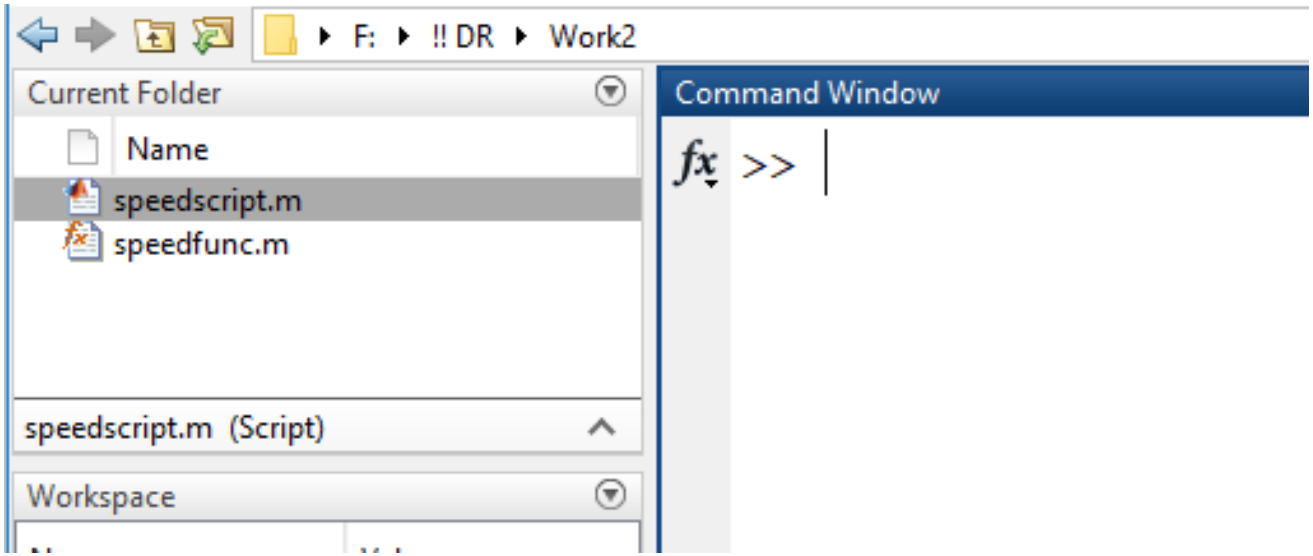
Writing a function

- Using the New Script button, create a new blank file and copy the script content in it
- Then, alter it to look this way
- Then save it.

```
function [spd,warn]=speedfunc(dist,tme)

clc;
spd=dist/tme; % compute speed
spd=spd/1000*3600; % convert m/s to kph

% if speed it's over the limit...
if spd>50
    warn='You are breaking the law !';
else
    warn='You are safe !';
end;
```

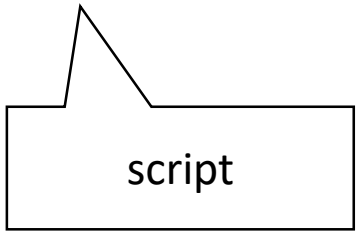
Writing a function

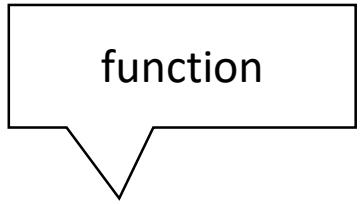
- You will get a second file, with the same extension .m, but a different graphic icon
- Functions are text files too.



The differences

```
clear
clc

dist=50; % define distance in meters
tme=10; % define time in seconds
spd=dist/tme; % compute speed
% convert from m/s to kph
spd=spd/1000*3600;

% if speed it's over the limit...
if spd>50
    warn='You are breaking the law !'
else
    warn='You are safe !'
end;
```






```
function [spd,warn]=speedfunc(dist,tme)

clc
spd=dist/tme; % compute speed
spd=spd/1000*3600; % convert m/s to kph

% if speed it's over the limit...
if spd>50
    warn='You are breaking the law !';
else
    warn='You are safe !';
end;
```

Function header

```
function [spd,warn]=speedfunc(dist,tme)
function [output variables]=function_name(input variables)
```

- A function must have a header
- The .m file name must be the same with the one used in the function header
- Multiple input and output variables are separated by commas

Running a function

- You can call a function in the Command Window, in a script or in another function
- Input and output variables of a function are formal, you will often use other names, or even directly numbers, for the input and output variables
- You can call a function several times in a program, with different input variables

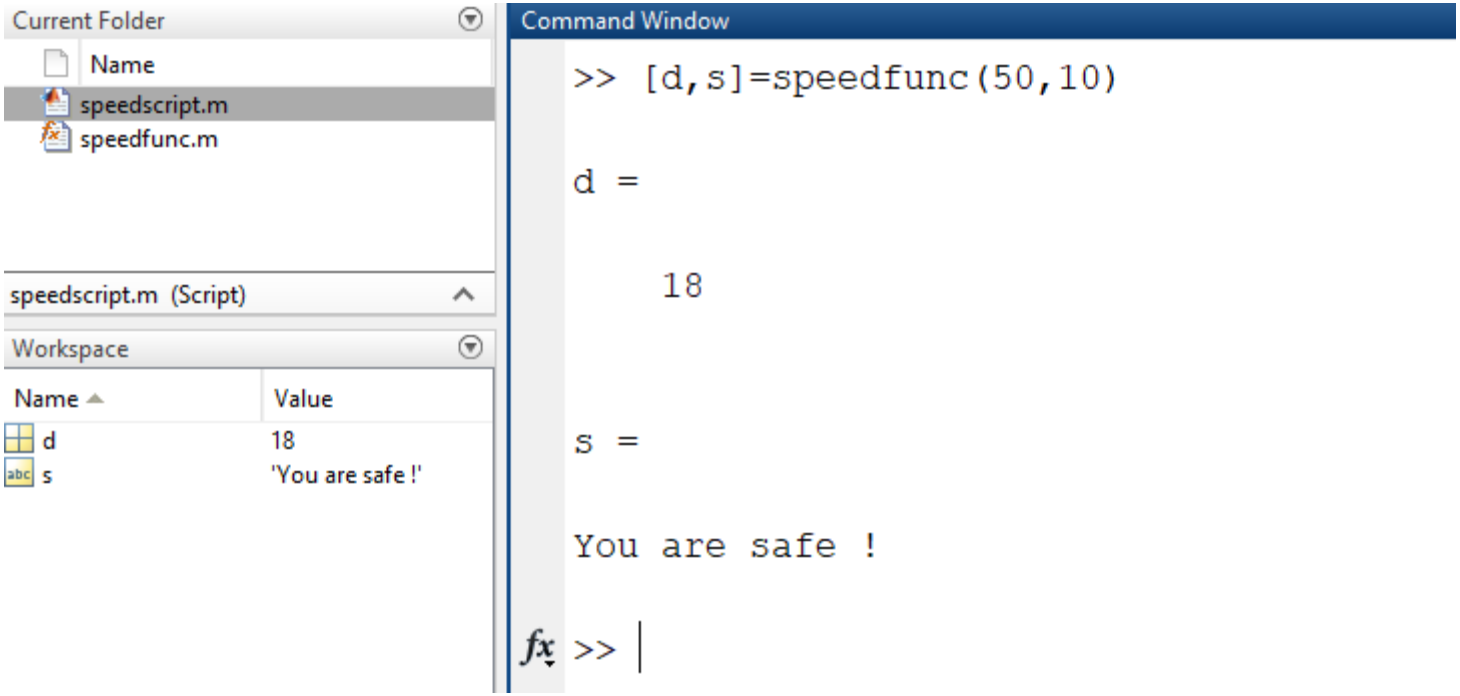
Running a function

- Variables defined in a function use a separate memory space.
- Internally created variables that are not in the output list will be destroyed automatically when the function call ends.
- As a general rule, variables can be accessed only in the memory space they were created.

The results

Distance = 50 m
Time= 10 seconds

Speed= 18 kph

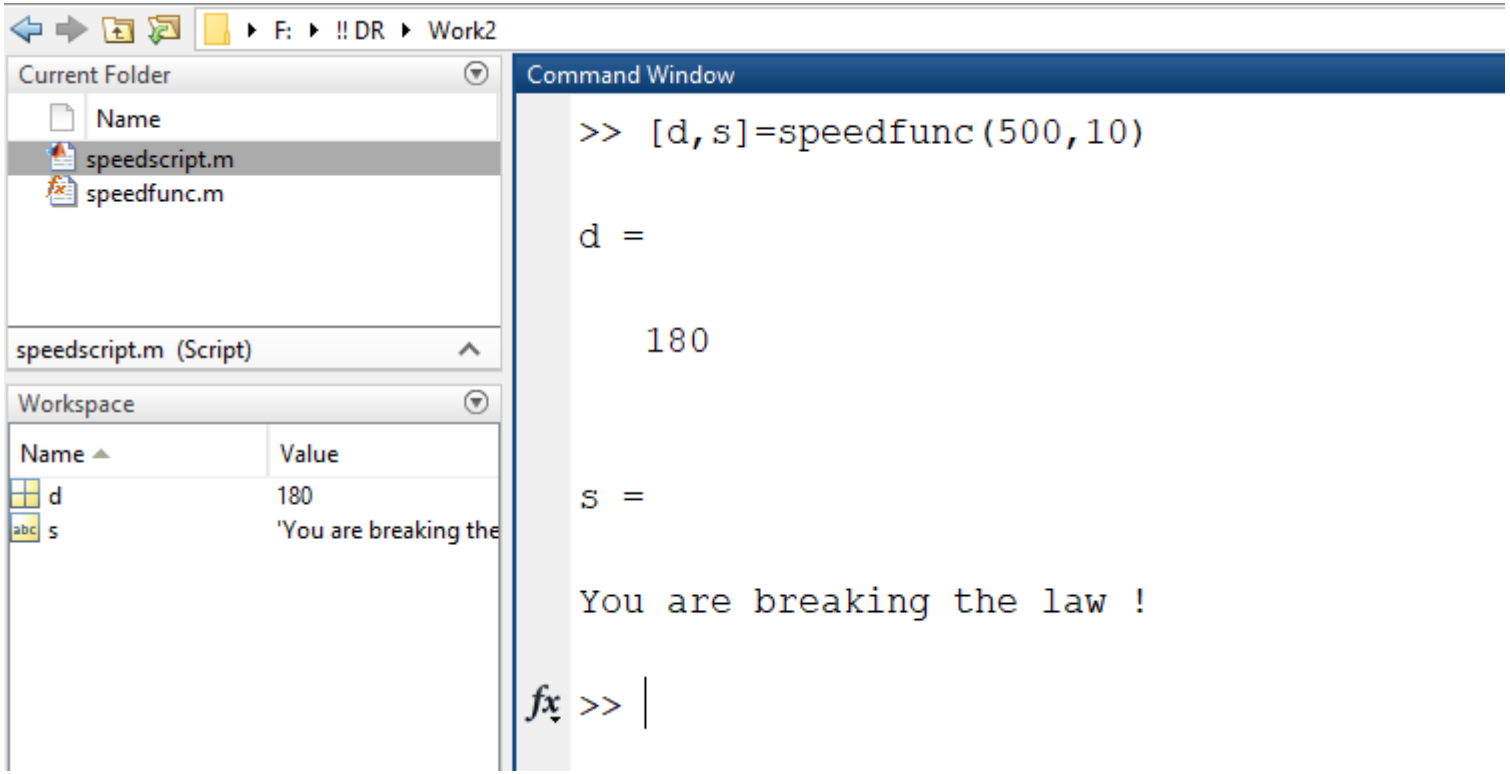


- **Only output variables are present in the Workspace !**

Breaking the law…

Distance = 500 m
Time= 10 seconds

Speed= 180 kph



The IF statement

Variant 1

```
if (condition is true)
    statements
end
```

Variant 2

```
if (condition is true)
    statements
else
    statements
end;
```

Variant 3

```
if (condition is true)
    statements
elseif (another condition is true)
    statements
elseif (a third condition is true)
else
    statements
end;
```

The IF statement

- IF statements can be nested
- Operators

== equality

>= greater or equal than

< less than

~= different than

- logical operators

&& for AND

II for OR

not or ~ for NOT

'exclude interval [5,12]'

```
if (a<5) || (a>12)
    statements
end
```

The SWITCH-CASE statement

- The variable var must be of an ordinal type

```
switch var
    case value1
        statements
    case value2
        statements
    case value3
        statements
    ……..
   otherwise
        statements
end
```

```
function [message]=options(value)

switch value
    case 1
        message='choice 1';
    case 2
        message='choice 2';
    case 3
        message='choice 3';
    otherwise
        mesaj='There are only three choices';
end;
```

The FOR loop

- **for known number of steps**

```
for counter = init_val : final_val
    statements
end;
```

```
for counter = init_val : step : final_val
    statements
end;
```

```
clc;
for i=3 : -0.5 : 1
    i
end;
```

The WHILE loop

- **for unknown number of steps**

```
while (condition is true)
    statements
end;
```

```
clc
i=1;
while i>0.5
    i=rand
end;
```

- Writing programs in the editor
  - Scripts
  - Functions

- The if, switch-case, for and while statements

End of Part Three - Review

- **Feel free to contact me with questions !**

Ovidiu Ivanov

Lecturer, PhD

The ‘Gheorghe Asachi’ Technical University

The Electrical Engineering Faculty

www.tuiasi.ro

ovidiuivanov@tuiasi.ro

THANK YOU !